305-CD-013-001

EOSDIS Core System Project

Release A CSMS Systems Management Subsystem Design Specification for the ECS Project

Final

July 1995

Hughes Information Technology Corporation
Landover, MD

Release A CSMS Systems Management Subsystem Design Specification for the ECS Project

Final

July 1995

Prepared Under Contract NAS5-60000
CDRL Item 046

SUBMITTED BY

__Parag N. Ambardekar /s/_____7/28/95_
Parag Ambardekar, Release A CCB Chairman Date
EOSDIS Core System Project

Hughes Information Technology Corporation
Landover, Maryland

This page intentionally left blank.

Preface

This document is one of sixteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release A of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-004, which provides an overview of the subsystems and 305-CD-018, the Data Dictionary, for those reviewing the object models in detail. A Release A SDPS and CSMS CDR Review Guide (510-TP-002) is also available.

The SDPS and CSMS subsystem design specification documents for Release A of the ECS Project include:

305-CD-004	Release A Overview of the SDPS and CSMS Segment System Design Specification
305-CD-005	Release A SDPS Client Subsystem Design Specification
305-CD-006	Release A SDPS Interoperability Subsystem Design Specification
305-CD-007	Release A SDPS Data Management Subsystem Design Specification
305-CD-008	Release A SDPS Data Server Subsystem Design Specification
305-CD-009	Release A SDPS Ingest Subsystem Design Specification
305-CD-010	Release A SDPS Planning Subsystem Design Specification
305-CD-011	Release A SDPS Data Processing Subsystem Design Specification
305-CD-012	Release A CSMS Segment Communications Subsystem DesignSpecification
305-CD-013	Release A CSMS Segment Systems Management Subsystem Design Specification
305-CD-014	Release A GSFC Distributed Active Archive Center Implementation
305-CD-015	Release A LaRC Distributed Active Archive Center Implementation
305-CD-016	Release A MSFC Distributed Active Archive Center Implementation
305-CD-017	Release A EROS Data Center Distributed Active Archive Center ?Implementation
305-CD-018	Release A Data Dictionary for Subsystem Design Specification
305-CD-019	Release A System Monitoring and Coordination Center Implementation

Object models presented in this document have been exported directly from CASE tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) at URL http://edhs1.gsfc.nasa.gov.

This document is a contract deliverable with an approval code 2. As such, it does not require formal Government approval, however, the Government reserves the right to request changes within 45 days of the initial submittal. Once approved, contractor changes to this document are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Corporation
1616 McCormick Drive
Landover, MD 20785

Abstract

The CSMS Systems Management Subsystem Design Specification defines the detailed design of the CSMS Systems Management Subsystem (MSS). It defines the MSS architectural design, as well as subsystem design based on Level 4 CSMS requirements. This document will be updated prior to the Release Readiness Review to incorporate the as-built configuration of the CSMS design.

Keywords: CSMS, system management, communications, networks, DCE, OODCE, OMT, Release A

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Final
iii through viii	Final
1-1 and 1-2	Final
2-1 through 2-4	Final
3-1 through 3-10	Final
4-1 through 4-32	Final
5-1 through 5-60	Final
6-1 through 6-150	Final
7-1 through 7-6	Final
A-1 through A-46	Final
B-1 through B-6	Final
AB-1 through AB-5	Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
305-CD-013-001	Final	July 1995	95-0476

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1 Identification 1-1
1.2 Scope 1-1
1.3 Status and Schedule 1-1
1.4 Organization 1-1

2. Related Documentation

2.1 Parent Documents 2-1
2.2 Applicable Documents 2-1
2.3 Information Documents 2-2
2.3.1 Information Documents Referenced 2-2
2.3.2 Information Documents Not Referenced 2-2

3. Systems Management Subsystem Overview

3.1 Introduction and Context 3-1
3.2 Systems Management Subsystem Overview 3-5
3.2.1 Systems Management Subsystem Structure 3-5
3.3 ECS Management Framework 3-7
3.3.1 Overview 3-7
3.3.2 Databases 3-8
3.3.3 MUI 3-8
3.3.4 Maps/Collections 3-9
3.3.5 Monitor/Control 3-9
3.3.6 Discovery 3-9

4. Management Agent CI

4.1 Management Agent Services 4-1
4.1.1 Management Agent Services Overview 4-1
4.1.2 Management Agent Services Context 4-4

4.1.3 Management Agent Services Object Model 4-5
4.1.4 Management Agent Services Dynamic Model 4-26
4.1.5 Management Agent Services Structure 4-28
4.1.6 Management Agent Services Management and Operation 4-32

5. MLCI - Management Logistics CSCI

5.1 Management Logistics Overview 5-1
5.2 Management Logistics Context 5-2
5.3 Management Logistics Object Model 5-3
5.3.1 Baseline Manager View 5-3
5.3.2 Software Change Manager View 5-20
5.3.3 Change Request Manager View 5-35
5.4 Software Change Management Dynamic Model 5-47
5.4.1 Ingest Algorithm 5-47
5.4.2 Record New Site Baseline 5-50
5.4.3 Evaluate System Enhancement Request 5-52
5.5 Management Logistics Structure 5-55
5.5.1 Baseline Manager CSC 5-55
5.5.2 Software Change Manager CSC 5-56
5.5.3 Change Request Manager CSC 5-58
5.6 MLCI Management and Operation 5-59
5.6.1 System Management Strategy 5-59
5.6.2 Operator Interfaces 5-59
5.6.3 Reports 5-60

6. MCI - Management Software CSCI

6.1 Fault Management 6-1
6.1.1 Fault Management Overview 6-1
6.1.2 Fault Management Context 6-4
6.1.3 Fault Management Object Model 6-5
6.1.4 Fault Management Dynamic Model 6-22
6.1.5 Fault Management Structure 6-24
6.1.6 Fault Management Management and Operation 6-26
6.2 Performance Management 6-27
6.2.1 Performance Management Overview 6-27
6.2.2 Performance Management Context 6-33
6.2.3 Performance Management Object Model 6-34
6.2.4 Performance Management Dynamic Model 6-44

6.2.5 Performance Management Structure 6-47
6.2.6 Performance Management Management and Operation 6-48
6.3 Security Management 6-49
6.3.1 Security Management Overview 6-49
6.3.2 Security Management Context 6-50
6.3.3 Security Management Object Model 6-50
6.3.4 Security Management Dynamic Model 6-63
6.3.5 Security Management Structure 6-66
6.3.6 Security Management Management and Operation 6-68
6.4 Accountability Management 6-68
6.4.1 Accountability Management Overview 6-68
6.4.2 Accountability Management Context 6-69
6.4.3 Accountability Management Object Model 6-70
6.4.4 Accountability Management Dynamic Model 6-81
6.4.5 Accountability Management Structure 6-82
6.4.6 Accountability Management Management and Operation 6-84
6.5 Physical Configuration Management Service 6-84
6.5.1 Physical Configuration Management Service Overview 6-84
6.5.2 Physical Configuration Management Service Context 6-85
6.5.3 Physical Configuration Management Service Object Model 6-86
6.5.4 Physical Configuration Management Service Dynamic Model 6-91
6.5.5 Physical Configuration Management Service Structure 6-94
6.5.6 Physical Configuration Management Service Management and Operation 6-95
6.6 Trouble Ticketing 6-96
6.6.1 Trouble Ticketing Overview 6-96
6.6.2 Trouble Ticketing Context 6-97
6.6.3 Trouble Ticketing Object Model 6-97
6.6.4 Trouble Ticketing Dynamic Model 6-109
6.6.5 Trouble Ticketing Structure 6-115
6.6.6 Trouble Ticketing Management and Operation 6-116
6.7 Management Data Access 6-117
6.7.1 Management Data Access Overview 6-117
6.7.2 Management Data Access Context 6-118
6.7.3 Management Data Access Object Model 6-118
6.7.4 Management Data Access Dynamic Model 6-138
6.7.5 Management Data Access Structure 6-140
6.7.6 Management Data Access Management and Operation 6-141
6.8 Ground Events Planning Service 6-141

6.8.1 Ground Events Planning Service Overview 6-141
6.8.2 Production Planning Workbench Context 6-142
6.8.3 Production Planning Workbench Object Model 6-142
6.8.4 Production Planning Workbench Dynamic Model 6-144
6.8.5 Production Planning Workbench Structure 6-148
6.8.6 Production Planning Management and Operation 6-149
6.9 Management DBMS 6-150
6.9.1 Overview 6-150
6.9.2 Implementation 6-150

7. MSS HWCI

7.1 MSS-HCI Design Drivers 7-1
7.1.1 Key Trade-off studies and Prototypes 7-1
7.1.2 Sizing and performance analysis 7-2
7.1.3 Scalability and Evolvability to Release B 7-4
7.2 HWCI Structure 7-4
7.2.1 Connectivity 7-5
7.2.2 HWCI Components 7-5
7.2.3 Failover and Recovery strategy 7-6

Figures

3.1-1. MSS High Level Context 3-4
3.3-1. HP OpenView Model 3-8
4-1. Management Agent CI Context 4-1
4.1-1. Management Agent Services 4-3
4.1-2. Management Agent Services Context Diagram 4-5
4.1-3. Management Agent Services Object Model 4-6
4.1-4. Get MIB Value Dynamic Model 4-26
4.1-5. SNMP Trap Generation Dynamic Model 4-27
5-1. Management Logistics CI Context 5-1
5.2-1. Management Logistics Context Diagram 5-3
5.3-1. Baseline Manager Object Model 5-4
5.3-2. Software Change Manager Object Model 5-22
5.3-3. Change Request Manager Object Model 5-36
5.4-1. Ingest Algorithm 5-48
5.4-2. Record New Site Baseline 5-51
5.4-3. Evaluate System Enhancement Request 5-53
6-1. Management Software CI Context 6-1

6.1-1. Fault Management Context Diagram 6-5
6.1-2. Fault Management Object Model 6-6
6.1-3. Fault Notification by an ECS Application Event Trace 6-23
6.2-1. Performance Management Context Diagram 6-34
6.2-2. Performance Management Object Model 6-35
6.2-3. Degradation of Performance Alert 6-44
6.2-4. Providing Performance Summary to SMC 6-45
6.3-1. Security Management Context Diagram 6-51
6.3-2. Security Management Object Model 6-52
6.3-3. Executing a Compliance Test 6-64
6.3-4. Reporting a Security Intrusion 6-65
6.4-1. Accountability Management Context Diagram 6-69
6.4-2. Accountability Management Object Model 6-70
6.4-3. Retrieving a User's Email Address 6-81
6.5-1. Physical Configuration Management Service Context Diagram 6-85
6.5-2. Physical Configuration Management Service Object Model 6-86
6.5-3. Add A New Node 6-91
6.5-4. Move An Existing Node 6-92
6.5-5. Delete An Existing Node 6-93
6.6-1. Trouble Ticketing Context Diagram 6-98
6.6-2. Trouble Ticketing Object Model 6-99
6.6-3. User Submits Trouble Ticket 6-109
6.6-4. A Trouble Ticket is Worked 6-111
6.6-5. A Trouble Ticket is Escalated 6-112
6.6-6. A Trouble Ticket is Forwarded 6-114
6.7-1. Management Data Access Context Diagram 6-118
6.7-2. Management Data Access Object Model 6-119
6.7-3. User Browses Log File Data 6-139
6.8-1. Production Planning Workbench Object Model 6-143
6.8-2. Ground Event Dynamic Model 6-145
6.8-3. Plan Creation Dynamic Model 6-147
7.2-1. MSS / CSS Redundant Server Configuration 7-6

Tables

3.1-1. Management Service Distribution 3-2
3.1-2. MSS Design Section Overview 3-3
3.2-1. MSS Implementation 3-5
4.1-1. Management Agent Services Components 4-29
5.5-1. Baseline Manager Components 5-55

5.5-2. Software Change Manager Components 5-56
5.5-3. Change Request Manager Components 5-58
6.1-1. Faults and Events Reported by ECS Managed Objects 6-3
6.1-2. Fault Management Components 6-24
6.2-1. Performance Manager by Managed Object 6-27
6.2-2. Host Performance Metrics 6-28
6.2-3. Router Performance Metrics 6-29
6.2-4. Hub Performance Metrics 6-29
6.2-5. Global System Performance Metrics 6-29
6.2-6. Application Performance Metrics 6-31
6.2-7. Process Performance Metrics 6-31
6.2-8. Disk Performance Metrics 6-32
6.2-9. Disk Performance Metrics 6-32
6.2-10. ECS-Specific Performance Metrics 6-32
6.2-11. Performance Management Components 6-47
6.3-1. Security Management Components 6-66
6.4-1. Accountability Management Components 6-82
6.5-1. Physical Configuration Management Service Components 6-94
6.6-1. Trouble Ticketing Components 6-115
6.7-1. Management Data Access Components 6-140
7.1-1. MSS Hardware Requirements Drivers 7-1
7.1-2. Key Trades and Prototypes 7-2

Appendix A. Requirements Trace

Appendix B. Application MIB Definition

Abbreviations and Acronyms

1. Introduction

1.1 Identification

This Release A Communications and System Management Segment (CSMS) Systems Management Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) item 046, with requirements specified in Data Item Description (DID) 305/DV3 is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

1.2 Scope

The Release A CSMS Systems Management Subsystem Design Specification defines the detailed design of the Systems Management Subsystem (MSS). It defines the Release A MSS architectural design, as well as subsystem design based on Level 4 MSS requirements. This document will be updated for the Release Readiness Review to incorporate the as-built design of the MSS.

This document reflects the June 21, 1995 Technical Baseline maintained by the contractor configuration control board in accordance with ECS Technical Direction No. 11, dated December 6, 1994.

1.3 Status and Schedule

This submittal of DID 305/DV3 meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. This version of the document is to be reviewed during the CSMS Critical Design Review, and subsequent comments will be incorporated into the final version.

1.4 Organization

The document is organized to describe the Systems Management Subsystem (MSS) design.

Section 1 provides information regarding the identification, scope, status, and organization of this document.

Section 2 provides a listing of the related documents which were used as source information for this document.

Section 3 provides an overview of the MSS, focusing on the MSS high-level design concept. This provides general background information to put MSS into context.

Section 4 contains the design of the Management Software CI (MCI). This contains the management applications, the management data access service, and the management DBMS.

Section 5 contains the design of the Management Logistics CI (MLCI). This includes the baseline manager, the change request manager, and the software change manager.

Section 6 contains the design of the Management Agents CI (MACI). This consists of the management agent services.

Section 7 contains the design of the Management Hardware CI (MHCI).

Appendix A contains the Traceability Matrix for Level 4 requirements to the design document.

The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this document.

2. Related Documentation

2.1 Parent Documents

The parent document is the document from which this CSMS Systems Management Subsystem Design Specification scope and content are derived.

194-207-SE1-001	System Design Specification for the ECS Project
304-CD-003-002	Communications and System Management Segment (CSMS) Requirements Specification for the ECS Project
423-41-01	Goddard Space Flight Center, EOSDIS Core System (ECS) Statement of Work
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System

2.2 Applicable Documents

The following documents are referenced within this CSMS Systems Management Subsystem Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

194-219-SE1-001	Interface Requirements Document Between EOSDIS Core System (ECS) and the NASA Science Internet (NSI)
219-CD-003-002	Interface Requirements Document Between EOSDIS Core System (ECS) and Landsat 7 System, Final
194-219-SE1-004	Interface Requirements Document Between EOSDIS Core System (ECS) and the Version 0 System
194-219-SE1-005	Interface Requirements Document Between EOSDIS Core System (ECS) and Science Computing Facilities
219-CD-006-003	Interface Requirements Document Between the EOSDIS Core System (ECS) and the National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC), Final
193-219-SE1-008	Interface Requirements Document Between EOSDIS Core System (ECS) and Program Support Communications Network, Draft
194-219-SE1-015	Interface Requirements Document Between EOSDIS Core System (ECS) and International Partners for Data Interoperability, Preliminary (formerly Interface Requirements Document Between EOSDIS Core System (ECS) and The European Space Agency)
194-219-SE1-018	Interface Requirements Document Between EOSDIS Core System (ECS) and Tropical Rainfall Measuring Mission (TRMM) Ground System

194-219-SE1-019	Interface Requirements Document Between EOSDIS Core System (ECS) and Earth Observing System (EOS) AM-1 Flight Operations
194-219-SE1-020	Interface Requirements Document Between EOSDIS Core System (ECS) and NASA Institutional Support Systems

2.3 Information Documents

2.3.1 Information Documents Referenced

The following documents are referenced herein and, amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS CSMS Systems Management Subsystem Design Specification.

194-102-MG1-001	Configuration Management Plan for the ECS Project
305-CD-004-001	Overview of Release A SDPS and CSMS System Design Specification
516-CD-001-003	Reliability Predictions for the ECS Project
518-CD-001-003	Maintainability Predictions for the ECS Project
193-00632TPW	DME Migration Study for the ECS Project
193-00561TPW	DCE Migration Study for the ECS Project
194-00605TPW	Use of ClearCase in the ECS Operational Environment
540-TP-001-001	Communications and System Management Segment (CSMS) Preliminary Design Review (PDR) Trade Studies for the ECS Project

2.3.2 Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of the CSMS Systems Management Subsystem Design Specification.

209-CD-002-001	Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System, Preliminary
209-CD-003-001	Interface Control Document Between EOSDIS Core System (ECS) and the EOS-AM Project for AM-1 Spacecraft Analysis Software, Preliminary
209-CD-004-001	Data Format Control Document for the ECS Flight Operations Segment AM-1 Project Data Base (PDB) Preliminary
220-CD-001-003	Communications Requirements for the ECS Project, Final
193-TP-561-001	Technical Paper: DCE Migration Study for the ECS Project
502-ICD-JPL/GSFC	Goddard Space Flight Center/MO&DSD, Interface Control Document Between the Jet Propulsion Laboratory and the Goddard Space Flight Center for GSFC Missions Using the Deep Space Network
530-DFCD-NCCDS/POCC	Goddard Space Flight Center/MO&DSD, Data Format Control Document Between the Goddard Space Flight Center Payload Operations Control Centers and the Network Control Center Data System

560-EDOS-0230.0001	Goddard Space Flight Center/MO&DSD, Earth Observing System (EOS) Data and Operations System (EDOS) Data Format Requirements Document (DFRD)
FIPS PUB 127-1	Federal Information Processing Standards Publication: Database Language SQL
RFC768	J. Postel; User Datagram Protocol
RFC791	J. Postel; Internet Protocol, 9/1/81 (obsolete/updated by RFC1060)
RFC792	J. Postel; Internet Control Message Protocol, 9/1/91
RFC793	J. Postel; Transmission Control Protocol, 9l/1/91
RFC826	D. Plummer; “Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, 11/1/82
RFC894	C. Hornig; Standard for the Transmission of IP Datagrams Over Ethernet Networks, 4/1/84
RFC895	J. Postel; Standard for the Transmission of IP Datagrams Over Experimental Ethernet Networks, 4/1/84
RFC903	R. Finlayson, et al; Reverse Address Resolution Protocol, 6/1/84
RFC1058	C. Hedrick; Routing Information Protocol, 6/1/88
RFC1060	J. Postel, J. Reynolds; ASSIGNED NUMBERS, 3/20/90 (obsolete/ updated by RFC1340)
RFC1105	K. Lougheed and Y. Rekhter; A Border Gateway Protocol, June 1989
RFC1157	M. Schoffstall, et al; A Simple Network Management Protocol (SNMP), 5/10/90
RFC1188	D. Katz; A Proposed Standard for the Transmission of IP Datagrams over FDDI Networks, 10/30/90 (obsolete/updated by RFC1390)
RFC1209	J. Lawrence, D. Piscitello; The Transmission of IP Datagrams Over the SMDS Service, 3/6/91
RFC1213	K. McCloghrie, M. Rose; Management Information Base for Network Management of TCP/IP-based Internets: MIB-II, 3/26/91
RFC1340	J. Reynolds, J. Postel; ASSIGNED NUMBERS, 7/10/92
RFC1374	J. Renwick, A. Nicholson; IP and ARP on HIPP, 11/2/92
RFC1390	D. Katz; Transmission of IP and ARP over FDDI Networks, 1/5/93
RFC1583	J. Moy; OSPF Version 2, 3/234/94
RFC1623	F. Kastenholz; Definitions of Managed Objects for the Ethernet-like Interface Types, 5/24/94
RFC1654	Y. Reckter and T. Li; A Border Gateway Protocol 4 (BPG-4), July 1994

This page intentionally left blank.

3. Systems Management Subsystem Overview

3.1 Introduction and Context

The MSS provides ECS Maintenance and Operations (M&O) Staff with the capability to manage the ECS enterprise, i.e., to perform network and system management services for all ECS resources, including all SDPS, FOS, and CSMS components. The MSS is composed of a combination of Commercial Off The Shelf (COTS) and custom management applications to provide a highly automated means for monitoring and managing the various ECS resources. At each ECS installation,[1] M&O Staff autonomously provide local management services associated with its ECS resources and, hence, are provided a local management view. At the System Monitoring and Coordination Center (SMC), M&O Staff provide enterprise monitoring and coordination services associated with the all ECS installations and are provided a system-wide management view. Extensive configurability is provided by the MSS applications to enable these views to be shared or controlled as necessary based on ECS management policy. Beside providing these views to M&O Staff for monitoring and control purposes, the management services make use of legacy CSS services such as electronic mail and bulletin board for coordination. The services provided by CSMS at the SMC, located at Goddard Space Flight Center (GSFC), are collectively referred to as Enterprise Monitoring and Coordination (EMC) throughout this document. In the same context, services provided by CSMS at DAACs and the EOS Operations Center (EOC) (sites) are collectively referred to as Local System Management (LSM).

The Management Subsystem (MSS) provides enterprise management (network and system management) for all ECS resources: commercial hardware (including computers, peripherals, and network routing devices), commercial software, and custom applications. Enterprise management reduces overall development and equipment costs, improves operational robustness, and promotes compatibility with evolving industry and government standards. Consistent with current trends in industry, the MSS thus manages both ECS's network resources per ESN requirements and ECS's host/application resources per SMC requirements. Additionally MSS also supports many requirements allocated to SDPS and FOS for management data collection and analysis/ distribution.

The MSS allocates services to both the system-wide and local levels. With few exceptions, the management services are fully decentralized, no single point of failure exists which would preclude user access. In principle every service is distributed unless there is an overriding reason for it to be centralized. MSS has two key specialization's: Enterprise Monitor and Coordination Services and Local System Management Services. The distribution of these services, shown in Table 3.1-1, provides maximum flexibility and policy neutrality in the design and implementation of MSS services.

---

1. Release A ECS installations include 4 Distributed Active Archive Centers (DAACs, at Goddard Space Flight Center, GSFC; Marshall Space Flight Center, MSFC; EROS Data Center, EDC; and Langley Research Center, LaRC), System Monitoring and Coordination Center (SMC, at GSFC), and EOS Operations Center (EOC, at GSFC).

Fully distributed client-server system management application services for distributed enterprises such as ECS are not commercially available today. For Release A, HP OpenView has been chosen as a framework for integration of multi-vendor network and system management products to support migration to a fully integrated management solution as such products become commercially available. This document covers the MSS services that will be fully implemented in Release A, other services will be supported through the use of Office Automation tools.

Table 3.1-1. Management Service Distribution (1 of 2)

Management Service	Enterprise Monitor and Coordination (EMC)	Local System Management (LSM)	Comments
Policies and Procedures	Provide Policy Decisions, Coordinate Policy, Policy Compliance Monitoring	Coordinate Policy, Site Level Policy Decisions, Policy Compliance Monitoring and reporting	Policy Management in Release A will be through the use of Office Automation Tools.
Fault Management	Receive Summary Reports from Sites, Monitor System Wide Resources (WANs), Perform Trend Analysis	Monitor, detect, isolate, diagnose, and recover from faults within domain	Largely COTS capabilities (HPOV), EMC maintains system-wide view from Site updates and monitoring
Performance	Trend analysis and system-wide view provided from Site updates	Collect server, hardware, and network performance data, analyze performance data, tune and report to SDPS/FOS/ EMC	Site performance is cooperative effort between LSM and SDPS/ FOS
Trends are through roll up of site reports	Trouble Ticketing	Summary Reports, View selected Site problems, support resolution	Document problem reports, track actions and closure. User and resource summaries
Remedy selected as TT package.	Physical Configuration Management	Same as DAAC	Maintain Physical location and configuration information
Commercial package to locate and record resources, detects changes to approved configuration	Security	Policy flowdown, system-wide monitoring and analysis	DCE Cell Management
Authentication, authorization, intrusion detection, DCE Cell Management	Largely public domain and COTS, HAL for cell management Policy flowdown and administration is through OA tools	Inventory	System-wide inventory creation and management
Site inventory data maintenance and management	Inventory Management in Release A is through OA tools	Logistics	System-wide monitoring of spares and consumable

Table 3.1-1. Management Service Distribution (2 of 2)

Management Service	Enterprise Monitor and Coordination (EMC)	Local System Management (LSM)	Comments
Site-level monitoring of spares and consumables including replenishment	Logistics Management in Release A is through OA tools	Maintenance	System-wide maintenance analysis
Establish and maintain PM schedules, monitor and coordinate off-site maintenance	Sites maintain schedules and records through OA tools	Configuration Management (CM)	(Software Change Manager)
Software CM of ECS baseline	Software CM for Site Baseline	Clearcase selected for Software CM	Baseline Manager
Consolidated baseline for system wide configuration and dependencies	Maintain site baseline for operational system configuration	COTS being evaluated, Selection expected before CDR	Change Request Manager
Maintain system wide status of change requests	Maintain record of configuration change requests, tracks status	COTS, DDTS has been selected based on evaluation and project experience with product.	Training
Coordinate training schedules, curricula, user feedback, and develop materials	Provide input on training schedules, curricula, local course development, and evaluation	Training Management in Release A is through OA tools used at Sites and SMC	Planning
System-wide schedule policy, priorities, performance assessment System wide ground event coordination	Schedule own resources based on system-wide priorities and policies, plan ground events and interface with FOS and PDPS	Release A provides for DAAC Resource planning with limited “Roll-up” capability for SMC	Reports
System-wide reporting based on “roll-up” of Site level data	Site-level reporting on performance, security, fault, and configuration information	Ad-hoc reporting from DBMS, other reports directly from COTS products	

The following sections describe the individual MSS Services in detail. An Overview, Object Model, Scenario, and Implementation Section is included for each service. Table 3.1-2 provides a roadmap to the MSS Design Section.

Table 3.1-2. MSS Design Section Overview (1 of 2)

Service	Section
Management Agent CI	4
Management Agents	4.1
Management Logistics CI	5
Configuration Management	5.1

Table 3.1-2. MSS Design Section Overview (2 of 2)

Service	Section
Management Software CI	6
Fault Management	6.1
Performance Management	6.2
Security Management	6.3
Accountability Management	6.4
Physical Configuration Management	6.5
Trouble Ticketing	6.6
Management Data Access	6.7
Common Management Services	6.8
Resource Planning	6.9

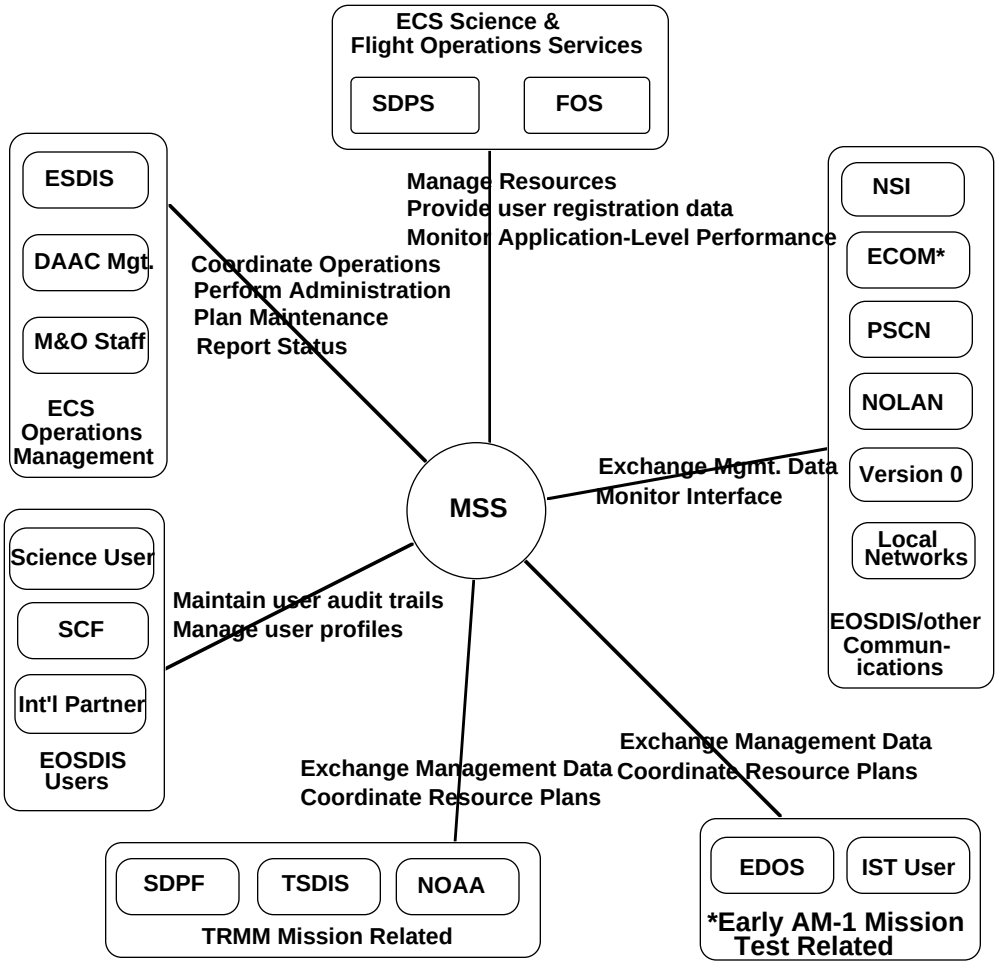


Figure 3.1-1. MSS High Level Context

3.2 Systems Management Subsystem Overview

3.2.1 Systems Management Subsystem Structure

A brief description of the MSS CSCI is provided in this section, a detailed description of each CSCI and components is presented Section 4 of this document.

The MSS is largely in the application domain, above the Open Systems Interconnection Reference Model (OSI-RM) application layer services. The management applications are supported by, and functionally dependent upon, other MSS and CSS services. The Management Agent Services are used to monitor and control managed objects (network, hardware, and software objects) within each management domain and provide the primary means of communicating status and control information between managed objects and management applications.

MSS is implemented mainly through the use of COTS, with some “wrapper” and custom code to support ECS unique requirements and to provide a layer of abstraction from COTS to minimize impact of migration to new technology or enhancements. The selection of each COTS package takes into consideration factors such as ECS requirements, commonality of the MMI, integration with HP OpenView, adherence to standards, vendor track record, and flexibility of operation.

The MSS implementation uses HP OpenView as the common framework for integration of the suite of common management functions, providing elements of fault management, configuration management, accountability management, performance management, and security management (FCAPS). This framework, and integrated applications, utilize the defacto industry standard Simple Network Management Protocol (SNMP) as the primary means of monitoring and controlling ECS management objects. Table 3.2-1 summarizes the MSS implementation for Release A.

Table 3.2-1. MSS Implementation (1 of 2)

Services	COTS (W/Glue) Custom Code	Notes
Management Software CI		
Fault	COTS+ Custom	Network provided through HPOV, Host, Peripheral and applications provided through “add-in” COTS + customization of COTS
Performance	COTS+ Custom	Network provided through HPOV, Host, peripheral, and software (applications) provided through Add-in
Security	COTS+ Public Domain + Custom	Implemented through HAL as DCE Cell manager, public domain for password checking, intrusion detection, Custom for accountability data
Accountability	COTS+ Custom	Significant part implemented through RDBMS, Agent provides for data collection, custom development for collection and processing of accountability data.

Table 3.2-1. MSS Implementation (2 of 2)

Services	COTS (W/Glue) Custom Code	Notes
Physical Configuration Management	COTS+ Scripts	COTS integrated with HP OpenView
Trouble Ticketing	COTS+ Scripts, Custom	COTS integrated with HP OpenView, Minimal custom and scripting
Management Data Access	Custom	All custom code. Combination of C++, Scripts, and SQL
Resource Planning	Heritage / Custom	Reuse of PDPS Planning Subsystem which includes MSS specific Methods
Management DBMS	COTS+ Custom	SQL to support report generation and Data filtering and import
Common Management Services	COTS	HP OpenView provides these services. Customization includes building maps, defining roles, scripting for operations resulting from events, configuration of HPOV parameters
Management Logistics CI		
Baseline Manager	COTS + Scripts	COTS with minimal scripting
Change Manager	COTS + Scripts	COTS with minimal scripting
Configuration Management	COTS + Scripts	COTS is ClearCase; scripts for rules and reports. RDBMS-based application for Baseline Manager
Management Agent CI		
Management Agents	COTS+ Custom	Management Agent and MIB, includes COTS extensible agent and extendible MIB to support ECS specific requirements.

As shown in Table 3.2-1, the MSS design depends heavily on COTS products, each of which will be configured to support ECS unique requirements. Of the MSS services, Management Data Access and Management Agents represent the largest portion of the custom development for Release A. Management Data Access is purely custom development and represents roughly one fourth of the custom developed code within MSS. Management Agents is a combination of COTS and custom development and also represents about one fourth of the MSS custom code. The COTS portion of the Management Agent Services includes a Master Agent which is configured for ECS, and a set of class libraries which are used to support the development of the sub-agent. Custom development includes the DCE Proxy Agent (provides for instrumentation of non-SNMP objects), sub-agents, and the ECS MIB which provides the controlling and reporting information for non-SNMP managed objects.

HP OpenView Network Node Manager (Common Management Services) provides the framework for integration of other management applications and an industry standard network management platform. OpenView itself must be configured to support the ECS specific design and

implementation. Part of this configuration is supported by the OpenView “Discovery” service which can automatically detect network devices which support the industry standard Simple Network Management Protocol (SNMP). Identification of other ECS managed objects is provided by custom development in the Management Agent to add these elements to the OpenView database. This information is used to build operations maps which depict the logical layout of network devices and other ECS managed objects. Part of the configuration then, entails the definition of operational “views” which support different operational roles within ECS. A second significant part of the configuration is developing scripts that define action routines for each event that is received for each managed object.

The other services depicted in Figure 3.2-1 will require varying amounts of configuration and customization. Integration into the OpenView framework is minimized through selection of COTS which have been integrated by the vendor whenever possible. Fault and Performance management represent the most complex configuration and customization development for Release A, while Trouble Ticketing and Physical Configuration Management represent the least complex development. A more detailed description of the custom development and scripting for each management service is provided in the individual service design descriptions in Section 4.

3.3 ECS Management Framework

3.3.1 Overview

As documented in 193-00632, DCE Migration Study for the ECS Project, and 193-00156, DME Migration Study for the ECS Project, the Open Software Foundation's (OSF) Distributed Management Environment (DME) is the selected distributed management architecture for ECS project. DME is an open architecture that is capable of evolving with new technologies and offers an integrated Distributed Enterprise System and Network Management Architecture for ECS.

Even though a full DME compliant implementation will not exist for Release A, most industry enterprise management players are adopting these technologies and migrating their existing products toward the DME architecture.

To mitigate risk, a DME precursor product (HP OpenView) has been selected as the ECS Management Framework for Release A. This selection provides an ECS migration path to management applications under the full DME architecture.

With this early selection prior to the preliminary design process, a detailed design of the Management Framework is not required. However, to support the design and object models of the management applications services and management agents services, a limited collection of DME object class interfaces, and their implementation in HP OpenView Network node Manager, are described in the following sections.

Since HP OpenView is a COTS product, its design will not be modeled. Further, its constituent components are not object-based. However, for purposes of clarity, the Figure 3.3-1 shows the components of HP OpenView as an aggregation of interface classes. The interface classes, as indicated before, are not real objects, but serve to clearly indicate the functionality provided by the set of interfaces abstracted by the representation. This functionality is available to management applications

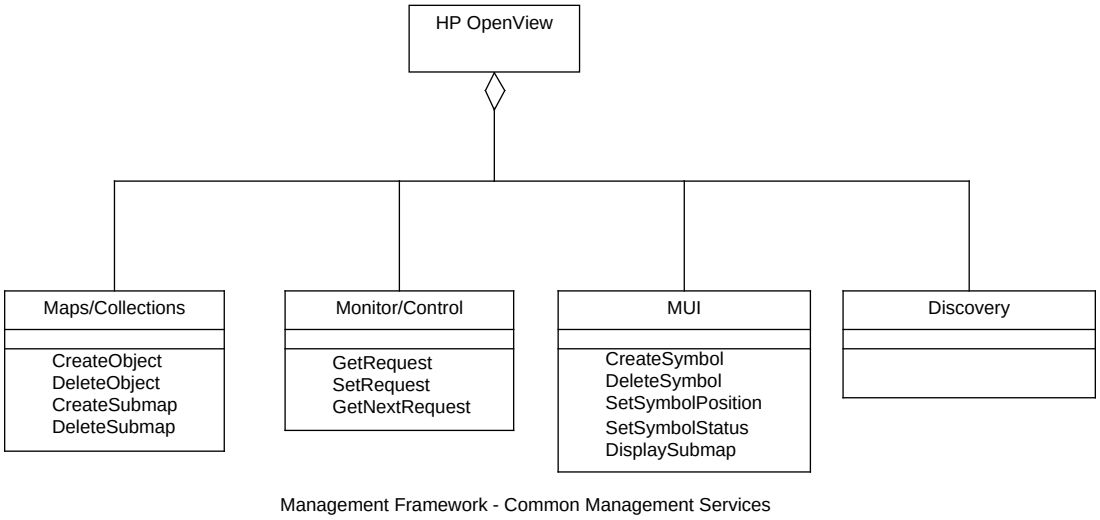


Figure 3.3-1. HP OpenView Model

3.3.2 Databases

Since the DME architecture does not define the architecture for the data store for the Management Framework, a short description of the data stores implemented in HP OpenView Network Node Manager is provided here. This provides the context for the description of the interfaces described in the following sections, and referenced in the sections on the Management Application Services. HP Openview Network Node Manager maintains its data in three major databases:

Object Database

The Object database contains all the objects (physical and logical) in the network that have been discovered. HP OpenView NNM automatically adds entries for objects (IP-addressable) that it discovers. Each object in this database has a set of fields (attributes) that describe the object. HP OpenView fills in values of all these objects as it discovers them. All developer-created objects (non-IP-addressable) need to be registered in this database. For each such object created, appropriate fields (attributes) must be created, and values must be assigned.

Map Database

The map database stores presentation information on each object stored in the object database. A map is a collection of objects from the Object database along with their relationships. A map contains a subset of all the objects in the Object database.

Topology Database

The topology database contains an electronic representation of the topology of the infrastructure of the network. This includes all entities that are IP-addressable.

3.3.3 MUI

HP OpenView has a graphical user interface which provides the capability to integrate the presentation of management applications. This provides a set of interfaces which provide the capability to support dialog interaction across the user interface.

The Management User Interface provides tools and services needed to build user interfaces for management applications in a technology independent way. Important features of the MUI are the ability to present management information maps, to support dialog interaction, and to separate presentation from interaction. For example, applications post the presentation information to the MUI which in turn maps this information to a specific toolkit like Motif. This allows the management application to remain independent of the underlying toolkit.

3.3.4 Maps/Collections

HP OpenView provides the interfaces that provide the capability to create and modify maps and submaps, as well as to retrieve information about maps, submaps and the objects on the maps and submaps.

The uniform management of the relationships between managed objects and the collection of objects in maps and sub-maps is provided by the Maps/Collections service. This service manages the collection-object and map-object relationships. The collection-object relationship provides the basic capability to manage unordered sets of references to other objects. The map-object relationship, in addition to maintaining the object references, maintains a graph to describes the connection between individual members of the map.

3.3.5 Monitor/Control

HP OpenView provides a set of interfaces which provide the capability for management applications to query remote management agents using the SNMP v1 protocol.

The Monitor/Control Service provides a uniform means for management applications to monitor and control managed objects, request and receive status on managed objects and process events/ traps from managed objects. This service provides the infrastructure via Simple Network Management Protocol (SNMP) for management applications to control the collection and monitoring of managed objects attributes that are defined in the Management Information Base (MIB); and the event management capabilities to support receiving, reporting, disseminating, logging of system event and for triggering actions in response to system events.

3.3.6 Discovery

HP OpenView provides the capability for automatic discovery of network devices, the capability for registering these newly discovered objects, and storing information about these managed objects in the databases. However, this service does not provide a callable interface for management applications to use.

The discovery service provides the basis for managed resource identification and detection (such as a repaired router returning to service). This service provides a capability to keep track of the system configuration by providing a common set of rules and interfaces to:

- register and unregistered new objects
- store information about them (e.g. in maps, collections etc.)
- notify M&O staff about discovery instances.

This page intentionally left blank.

4. Management Agent CI

The Management Agent CI consists of the Management Agent Services. The Management Agent Services context is provided in Figure 4-1.

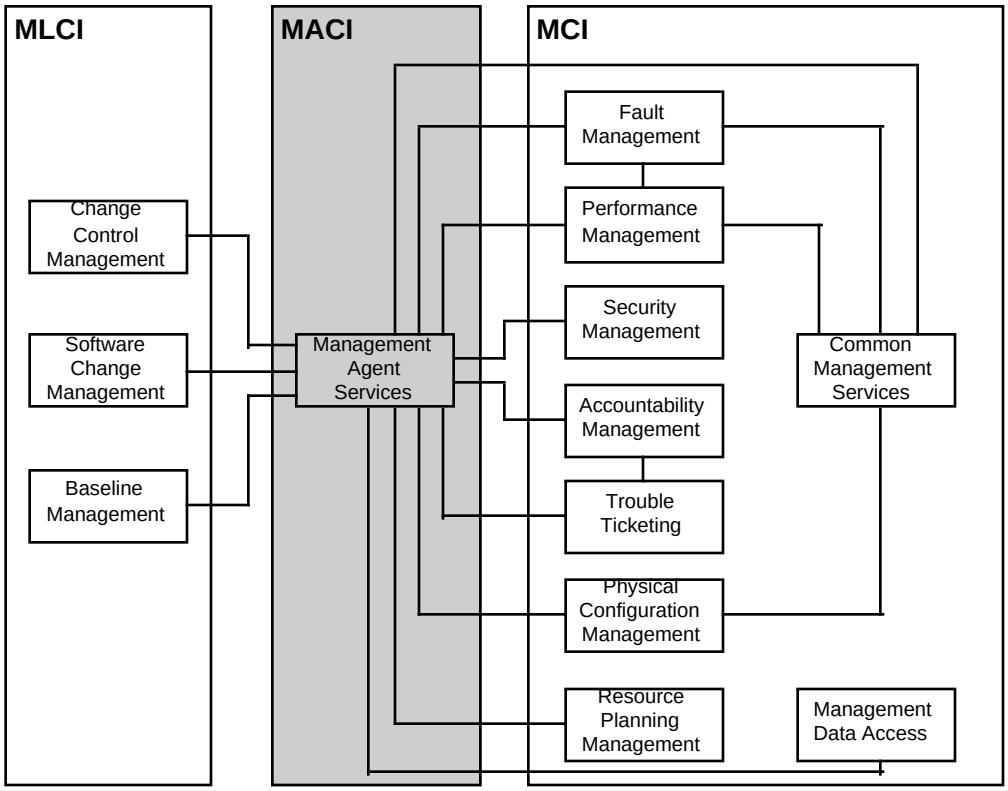


Figure 4-1. Management Agent CI Context

4.1 Management Agent Services

4.1.1 Management Agent Services Overview

The MSS provides ECS M&O Staff with the capability to manage the ECS enterprise, i.e., to perform network and system management services on all ECS resources, including all SDPS, FOS, and CSMS components.

The enterprise management system is based on the manager-agent model. It consists of management applications, a managed object model, and a management protocol. The management applications reside on managing system(s). They provide the interfaces for the human enterprise manager to perform management tasks. The managed object model consists of managed objects which are defined to represent the resources being managed. The underlying resources can be physical devices, system software or applications. The management agent is the implementation which substantiates the managed objects. It normally resides on each remote host performing

monitoring and control functions for the management applications which are on the managing system(s). The management applications communicate with agents through the management protocol.

The MSS is composed of a variety of management applications providing services such as fault, performance, security, and accountability management for ECS networks, hosts, as well as SDPS and FOS applications. The management applications reside on MSS Server. The management information of remote objects need to be conveyed to the management applications through the Management Agent Service which primarily resides on remote hosts.

The Management Agent Services can be broken down into several distinct parts (see figure 4.1-1). The first portion, a master agent responsible for communication of request and responses related to the retrieval of management information, will be supplied via procurement of a COTS product. The second portion, Deputy, handles secure delivery of requests for setting management information and generating traps securely. Deputy will be developed utilizing COTS supplied application programming interface calls to HP OpenView. The ECS subagent communicates both of the aforementioned types of management requests and responses to either an ECS developed application or proxy agent. This subagent will be a customization of COTS supplied functionality. Finally, Management Agent Services will deliver a generic proxy agent for the purposes of managing non-DCE applications (for example some COTS packages). This custom proxy agent will be needed to be customized using vendor provided API for each application.

The MSS Management Agent Service provides the following functions:

- Enables the retrieval of ECS managed object values in test or operational mode.
- Communicates via ECS management protocol with MSS Monitor/Control Service to respond to requests for managed object values.
- Communicates via ECS management protocol with MSS Monitor/Control Service to send ECS management event notifications/traps to the Monitor/Control Service.
- Communicates via ECS management protocol with MSS Monitor/Control Service to receive ECS management Set requests from the Monitor/Control Service.
- Provides an ECS management agent that is configurable to include community, location, log, etc.
- Provides the ECS management agent which can manage network devices.
- Provides the ECS management agent which is extensible for managing host systems.
- Provides the ECS management agent which is extensible for managing ECS applications.
- Provides proxy agent(s) for ECS network devices and applications that cannot be communicated through ECS protocols.
- Performs local polling on hosts to monitor the state of managed ECS resources.
- Provides instrumentation API to application developers to enable the manageability of ECS applications.
- Defines the managed object model to represent the management characteristics of ECS applications. The objects are defined in Management Information Base (MIB).

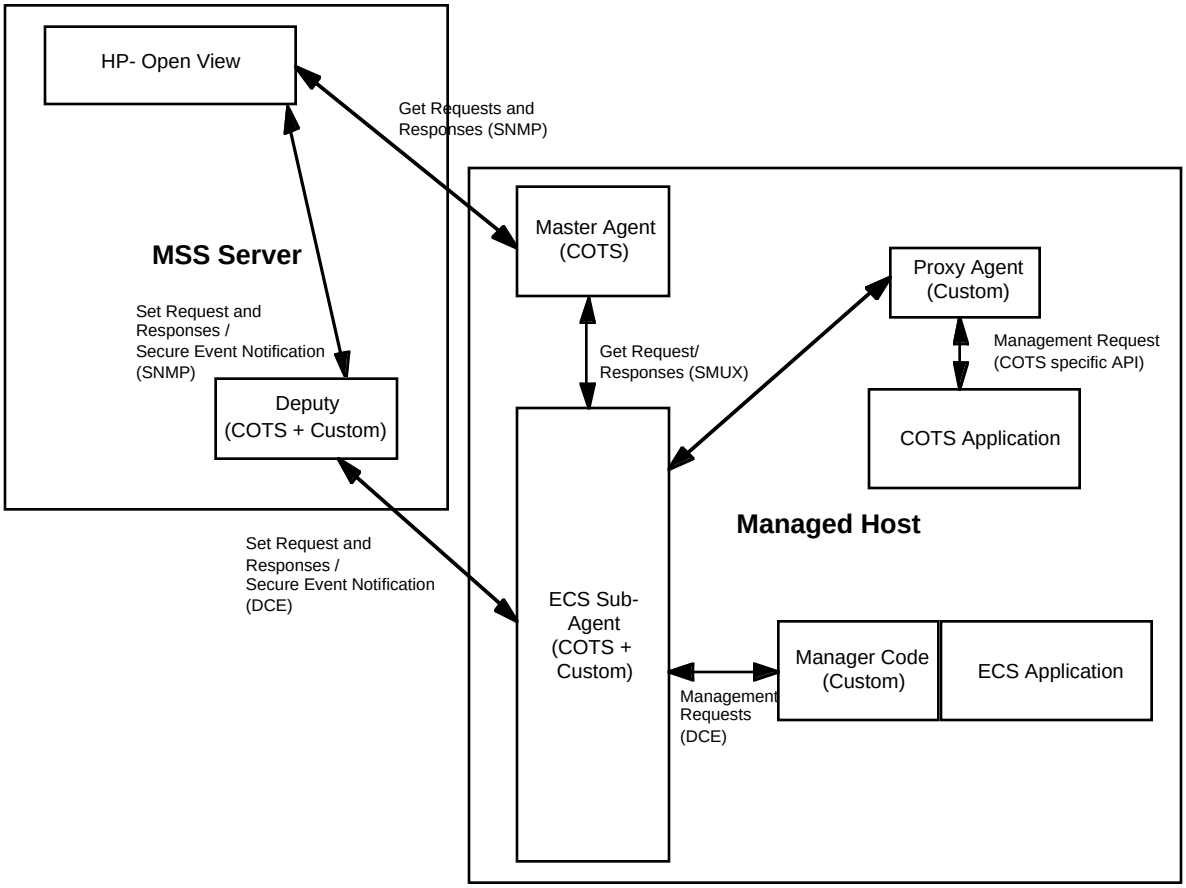


Figure 4.1-1. Management Agent Services

SNMP has been chosen as the management protocol since it is the defacto and Internet standard protocol for network management in TCP/IP environment. The MSS management applications pass SNMP requests to the agent to retrieve management information. For setting management information, it uses DCE RPC to send requests to remote agents for security reasons.

MSS management applications need to monitor the state of managed resources. It can be done by polling of remote resources. But, remote polling has certain impact on the network traffic. Therefore, the agents can perform local polling for the management applications to avoid the costly remote polling and to inform management applications about the state change of managed resources in a more timely manner.

Event handling will be provided by Management Agent Service to satisfy the need to dispatch events for orderly and prompt resolution to fix problems. All events will be logged locally on each host. Performance data will also be logged.

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

To manage ECS and COTS applications is a focus in Management Agent Service. A set of Instrumentation API will be provided to ECS application developers to use for the manageability of ECS applications. ECS applications can be categorized into two general types, OODCE-based or non-OODCE-based applications. Application developers can determine the performance metrics along with their threshold to monitor. Fault types can also be monitored and counted. For managing non-SNMP resources such as COTS, proxy agents will have to be used, and supplied by the resource provider. The front-end of the proxy agent uses the instrumentation API provided by MSS. The back-end of the proxy agent is the interface unique to each COTS resource.

MSS establishes an ECS file system on each managed host to store configuration-related files for applications. The root of this ECS file system is ECS. It has three subdirectories, FOS, SDPS, and CSMS. Within each one, it can be subdivided into subdirectories. Each application package should have its own subdirectory.

The full path name of the executable file needs to be stored in the application's directory. A file with suffix .inp will be used for the file containing the required input arguments. The .pwd will be used as a suffix for the keytab file. A .cfg file is created at installation time to contain configuration-related information. When an application starts, an .ins file is created for that particular application instance. It will be deleted when the application terminates.

It is assumed that the subagent responsible for the management of applications is always running. It is started at the boot time of the host. The request to start or to shutdown an application can be issued on the management application. This request will be passed securely to the agent and subagent on remote hosts. Then, the startup or shutdown actions will be performed.

The startup, shutdown, and other events will trigger event notifications to the management framework in SNMP traps.

Management applications can make SNMP requests to retrieve management information as MIB values. They can also set certain management information through the use of secure DCE RPC calls.

4.1.2 Management Agent Services Context

The Management Agent Service provides the means to communicate management requests and responses between management applications and managed resources. The managed resources include SDPS, FOS, CSMS applications as well as fault, performance, accountability, and security management services.

The management agent services context diagram is shown in Figure 4.1-2.

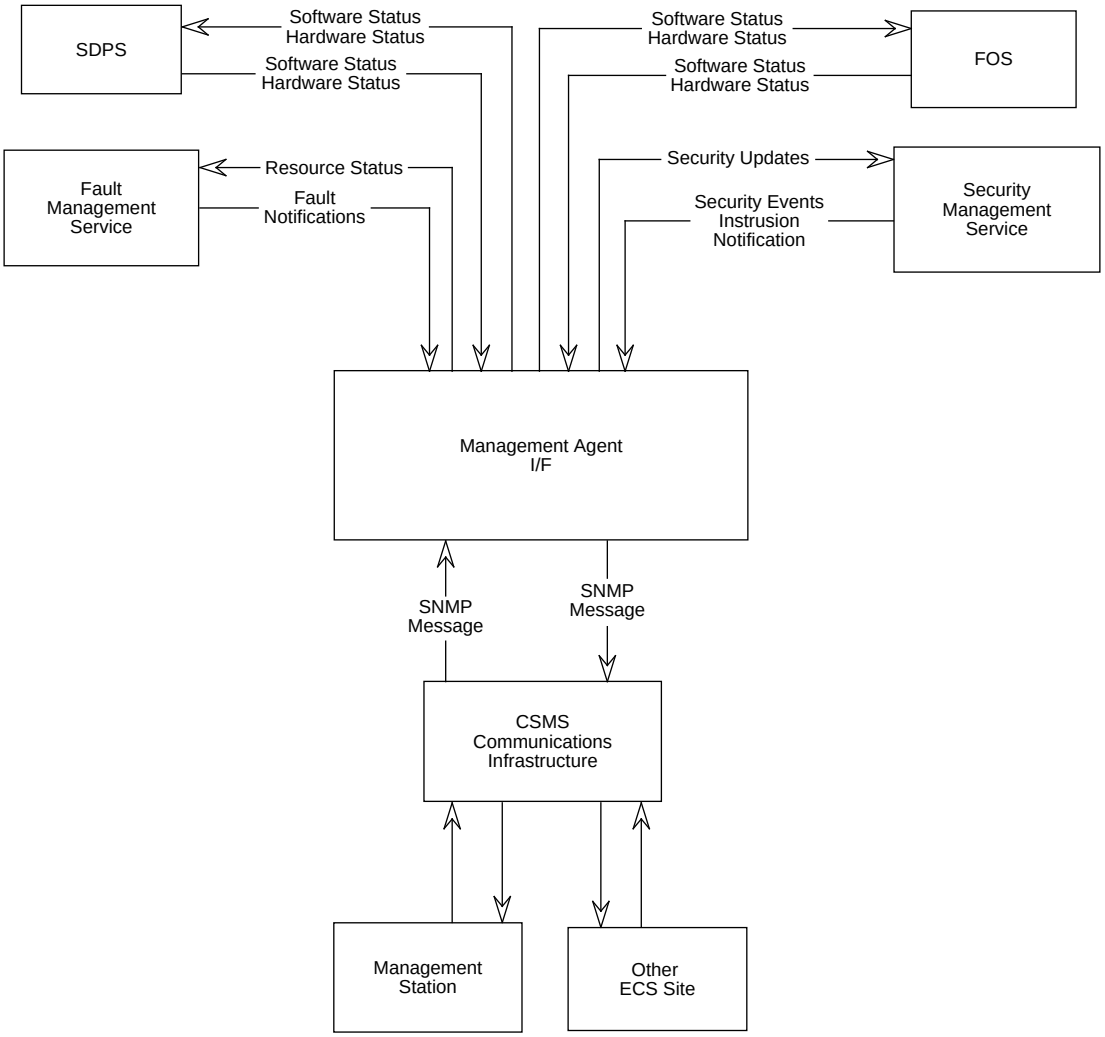


Figure 4.1-2. Management Agent Services Context Diagram

4.1.3 Management Agent Services Object Model

The Management Agent Services object model is shown in Figure 4.1-3.

The Management Agent Service object model shows the object classes in the Management Agent Service, the associations among them, their attributes and operations. The classes central to the service are:

- EcAgAgent - This the extensible SNMP master agent on the host. It receives SNMP requests from management applications through the management framework. It also communicates with subagents to support various MIB extensions. This is a COTS product. Current assumption is Peer Networks' agent and its tool kit to develop subagents.
- MsAgSubAgent - This is the subagent primarily for supporting the ECS application MIB. It receives requests from the master agent. Then, it retrieves MIB values through the calling of proper access methods.

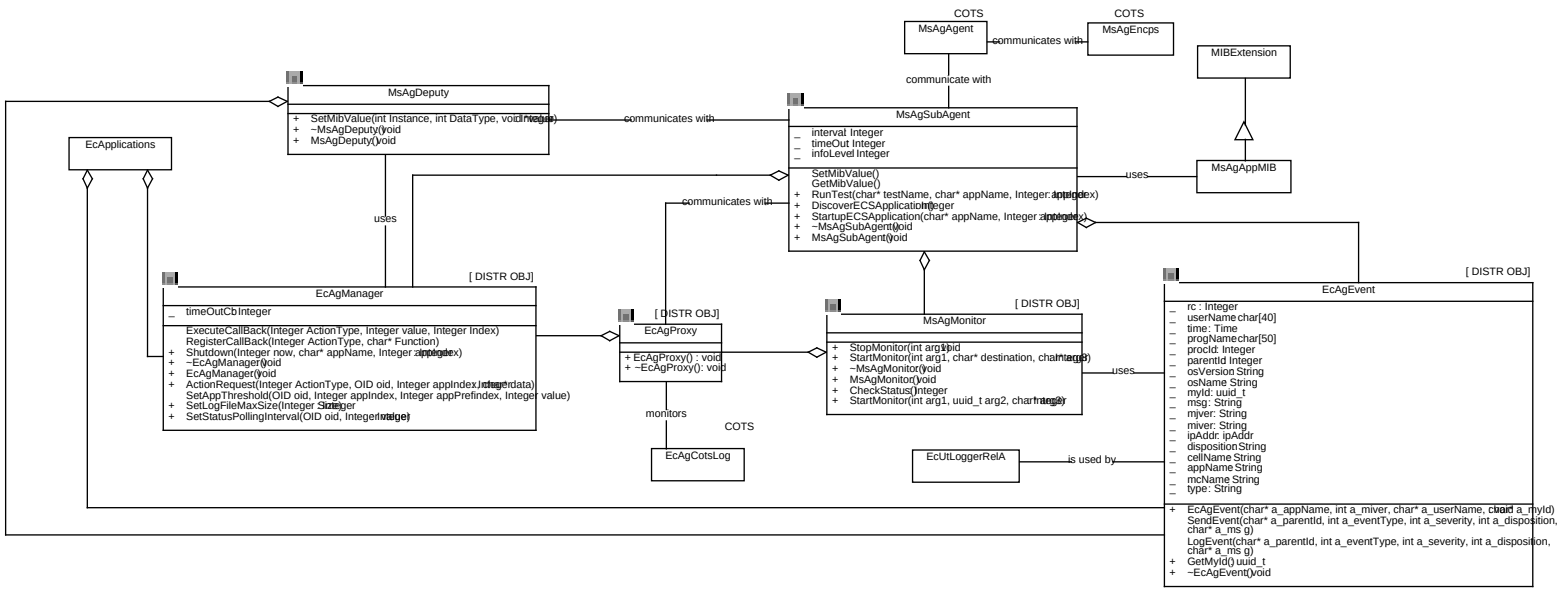


Figure 4.1-3. Management Agent Services Object Model

- MsAgMonitor - This is part of the subagent. It is used to perform local monitoring of the state of ECS applications.
- EcAgManager - This is part of the instrumentation code. It is used by the subagent or by the monitor to enable the manageability of ECS applications.
- EcAgEvent - This is part of the instrumentation code. It is used by ECS applications to send out event notifications for logging or eventually trigger the emission of SNMP traps. The event types include fault, performance, security, etc. The severity level includes fatal, error, warning, informational, etc.
- EcAgProxy - This is the proxy agent which enables the ECS subagent to communicate with non-OODCE COTS applications. Its front-end has the MSS instrumentation code to communicate with the subagent. Its back-end communicates with the COTS.
- MsAgDeputy - This is an entity on MSS Server for management applications to send Set requests to the agent, and to receive secure event notifications from the agent for trap generation. The reason for its existence is to use secure DCE RPC as the transport mechanism as opposed to use SNMP requests between MSS Server and managed hosts.
- MsAgEncps - This is an encapsulator which enables the master agent to communicate with non-Peer SNMP agents.
- MsAgAppMib - This is an SNMP MIB extension for managing applications. It includes the Agent group, the Application group, the Program group, the Process group, and the Traps group. In addition to general attributes, ECS application developers can input their own application-specific metrics or parameters to monitor. But, the access methods have to be provided as callback functions.

4.1.3.1 EcAgCotsLog Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class contains the log of the COTS programs.

Attributes:

None

Operations:

None

Associations:

The EcAgCotsLog class has associations with the following classes:

Class: EcAgProxy monitors

4.1.3.2 EcAgEvent Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

The EcAgEvent defines a distributed object. It provides the capability to dispatch events for orderly and prompt resolution should events occur. The SNMP protocol provides the capability to send traps from agent to SNMP manager. But, the traps are not secure and not reliable. The solution to these problems are using DCE RPC as the transport mechanism for security reasons and sending the traps from MSS Server to the management framework locally. The COTS HP OpenView guarantees the delivery of traps local on one host by using IPC as opposed to UDP. The ECS applications, the EcAgProxy agent, and the MsAgMonitor of the MsAgSubagent can send event notifications to the MsAgSubagent. The MsAgSubagent logs every event into MSS log file. Then, if the severity of the event equals to or is higher than the infoLevel variable, it sends this event notification further to the MsAgDeputy on the MSS Server which in turn convert the event to an SNMP trap and send it locally to the management framework.

Attributes:

appName - This attribute represents the name of the application.
Data Type: String
Privilege: Private
Default Value:

cellName - This attribute represents the name of the DCE cell to where this host belongs.
Data Type: String
Privilege: Private
Default Value:

disposition - This attribute represents the disposition of the application.
Data Type: String
Privilege: Private
Default Value:

ipAddr - This attribute represents the IP address of the host that the application runs on.
Data Type: ipAddr
Privilege: Private
Default Value:

mcName - This attribute represents the name of the host system.
Data Type: String
Privilege: Private
Default Value:

miver - This attribute represents the minor version of the application.

Data Type: String
Privilege: Private
Default Value:

mjver - This attribute represents the major version of the application.
Data Type: String
Privilege: Private
Default Value:

msg - This attribute represents the message from the application.
Data Type: String
Privilege: Private
Default Value:

myId - This attribute represents the UUID of this application.
Data Type: uuid_t
Privilege: Private
Default Value:

osName - This attribute the name of the operating system on this host.
Data Type: String
Privilege: Private
Default Value:

osVersion - This attribute represents the version of the operating system on this host.
Data Type: String
Privilege: Private
Default Value:

parentId - This attribute represents the UUID of the parent process of this application.
Data Type: Integer
Privilege: Private
Default Value:

procId - This attribute is the ID of the process which sends out the event.
Data Type: Integer
Privilege: Private
Default Value:

progName - This attribute represents the name of the program in the application package.
Data Type: char[50]
Privilege: Private
Default Value:

rc - This attribute represents the return code from the failed function call.

Data Type: Integer
Privilege: Private
Default Value:

time - This attribute represents the time that the event is sent.
Data Type: Time
Privilege: Private
Default Value:

type - This attribute represents the type of the event occurred.
Data Type: String
Privilege: Private
Default Value:

userName - This attribute represents the name of the user of the application.
Data Type: char[40]
Privilege: Private
Default Value:

Operations:

EcAgEvent - This represents the constructor for this class.
Arguments: char* a_appName, int a_miver, char* a_userName, char* a_myId
Return Type: void
Privilege: Public

GetMyId - This method is used to get the uuid of this application.
Arguments:
Return Type: uuid_t
Privilege: Public
PDL:PDL:uuid_t GetMyId ()
{
Call the uuid generator to get the uuid of this application
}

LogEvent - This method writes event as log entry in the log file.
Arguments: char* a_parentId, int a_eventType, int a_severity, int a_disposition, char* a_ms g
PDL:PDL: EcAgEvent::LogEvent (char* a_parentId, int a_eventType, int a_severity, int a_disposition, char* a_msg)
{
//if the severity exceeds the threshold for this particular kind of event
if (Severity> EventType.Threshold) {
//event must be forwarded to manager

```
EcAgEvent::SendEvent (EvenType, Severity)
   }
}
```

SendEvent - This method sends event notification out to requested applications through the Agent and SubAgent.
Arguments: char* a_parentId, int a_eventType, int a_severity, int a_disposition, char* a_ms g
PDL:PDL: MsAgEvent::SendEvent (int EventType, int Severity)

```
{
   //forward event to the manager
   MsAgDeputy::SendEvent (EventType, Severity, ...)
}
```

~EcAgEvent - This method specifies the destructor of the class.
Arguments:
Return Type: void
Privilege: Public

Associations:

The EcAgEvent class has associations with the following classes:
Class: EcUtLoggerRelA isusedby
Class: MsAgMonitor uses
EcApplications (Aggregation)
MsAgDeputy (Aggregation)
MsAgSubAgent (Aggregation)

4.1.3.3 EcAgManager Class

Parent Class: Not Applicable
Public: No Distributed Object: Yes
Purpose and Description:
This class defines a distributed object. It incorporates most of the manager control and instrumentation functionality for ECS applications and COTS. ECS application developers have to use the class library to instrument their code and to provide callback routines so that the ECS applications are manageable by MSS. The EcAgProxy developer has to do the same. The Application MIB provides table format for Fault, Performance, and Configuration parameters to be added by application developers. In order to monitor these parameters, application developers have to provide callback routines for the Subagent to retrieve the value of these parameters. This object provides the capability for the developers to register all the callback functions required. The callback mechanism provided by this object is a generic one. Application developer has to register the callback function first.

When the Get request comes in, it will trigger the execution of the callback function to retrieve the value. This object can be invoked by the MsAgDeputy to send Set requests to the MsAgSubagent. The MsAgSubagent can invoke the methods of this object to send Get, Set, or other requests to ECS applications or to the Proxy of COTS. The data structures that application developers have to use are based on the Application MIB definition. The following are examples: Action Type Data Structure Get_Performance typedef struct { char PerfType[n+1]; int PerfValue; int PerfThreshold; } Get_Fault typedef struct { char FaultType[n+1]; int FaultValue; } Get_Config typedef struct { char CfgParam[n+1]; char value[m+1]; } Shutdown typedef struct { int now; }

Attributes:

timeOutCb - This attribute contains the timeout value for the callback routine.
Data Type: Integer
Privilege: Private
Default Value:

Operations:

ActionRequest - This method passes the action request from the Deputy to the subAgent.
Arguments: Integer ActionType, OID oid, Integer appIndex, char* data
Return Type: Integer
Privilege: Public
PDL:PDL: int EcAgManager::ActionRequest(int ActionType, OID oid, int appIndex, char* data)

```
{
    validate the ActionType
    call access method based on the action type
}
```

EcAgManager - This method represents the constructor of the class.
Arguments:
Return Type: void
Privilege: Public

ExecuteCallBack - This method executes a callback routine.
Arguments: Integer ActionType, Integer value, Integer Index
PDL:PDL: int EcAgManager::ExecuteCallBack (int Action, void *value, int RowIndex)

```
{
   //determine if the application has registered an action for the requested callback
   if (CallBack function registered for action) {
return_val= Execute CallBack associated with the action passing the value and RowIndex
if (return_val==0) {
  //alert caller that now value can be associated with the request
```

```
  return NO_VALUE
}
   }
   else {
   //alert caller that no value can be associated with request
   return NO_VALUE
   }
   return VALUE_FOUND
}
```

RegisterCallBack - This method registers callback routines.
Arguments: Integer ActionType, char* Function
PDL:PDL: int EcAgManager::RegisterCallBack (int ActionType, ActionFunctionPtr *FncPtr)

```
{
   //make sure the specified action is a legal action. Legal actions are of the following type:
   //(GET_PERFORMANCE, GET_FAULT, GET_CONFIG, GET_PROCESS,
   // GET_ASSOCIATION, GET_PROGRAM, STARTUP, SHUTDOWN).
   if (Action in set (LEGAL_ACTIONS)) {
add FncPtr to callback table for specified action
   }
   else {
   //alert user of illegal action
   return ERROR
   }
}
```

SetAppThreshold - This method sets the threshold of performance metrics.
Arguments: OID oid, Integer appIndex, Integer appPrefindex, Integer value
PDL:PDL:int EcAgManager::SetAppThreshold (OID oid, int appIndex, int appPerfIndex, int value)

```
{
    Based on the oid, application index, and the index of
    application Performance table, set the "value" to the
    "threshold"
}
```

SetLogFileMaxSize - This method terminates the application.
Arguments: Integer Size
Return Type: Integer
Privilege: Public
PDL:PDL: int EcAgManager::SetLogFileMaxSize (int size)

```
{
```

```
   set the maximum size of the log file in memory and in configuration file
}
```

SetStatusPollingInterval - This method sets the status polling interval on a host.
Arguments: OID oid, Integer value
Return Type: Integer
Privilege: Public

```
PDL:PDL: int EcAgManager::SetStatusPollingInterval (OID oid, int value)
{
   set the status polling interval value in memory and in configuration file
}
```

Shutdown - This method terminates the application.
Arguments: Integer now, char* appName, Integer appIndex
Return Type: Integer
Privilege: Public

```
PDL:PDL: int EcAgManager::Shutdown (int now, char *appName, int appIndex)
{
   //if now==1, the application should immediately comply, otherwise the application can
comply with the request as soon as convenient.
   //if the user provides a callback for shutdown, then execute it first
   if (CallBack function registered for SHUTDOWN) {
     Execute CallBack associated with SHUTDOWN passing in value of now
   }
   //execute OODCE specific version of shutdown call
   OODCE::shutdown (now)
}
```

~EcAgManager - This method represents the destructor of the class.
Arguments:
Return Type: void
Privilege: Public

Associations:

The EcAgManager class has associations with the following classes:
Class: MsAgDeputy uses
EcAgProxy (Aggregation)
EcApplications (Aggregation)
MsAgSubAgent (Aggregation)

4.1.3.4 EcAgProxy Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

This object class is primarily for COTS' manageability. It includes the MSS instrumentation class library to enable the manageability of the COTS product. The front-end of this object is the MSS instrumentation code. The back-end of it is the interface to the COTS. It is unique to every COTS. In security management, the logs of COTS are monitored by this object. If an security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.

Attributes:

None

Operations:

EcAgProxy - This method specifies the constructor of the class.

Arguments:
Return Type: void
Privilege: Public

~EcAgProxy - This method specifies the destructor of the class.

Arguments:
Return Type: void
Privilege: Public

Associations:

The EcAgProxy class has associations with the following classes:

Class: MsAgSubAgent communicates with
Class: EcAgCotsLog monitors
MsAgMonitor (Aggregation)

4.1.3.5 EcApplications Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class represents ECS customized applications.

Attributes:

None

Operations:

None

Associations:

The EcApplications class has associations with the following classes:
None

4.1.3.6 EcUtLoggerRelA Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a public class that provides a logging capability used by the Management Agent Services logging function.

Attributes:

None

Operations:

None

Associations:

The EcUtLoggerRelA class has associations with the following classes:
Class: EcAgEvent isusedby

4.1.3.7 MIBExtension Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The SNMP MIB extension defined for resources being managed other than MIB II. This is an abstract class.

Attributes:

None

Operations:

None

Associations:

The MIBExtension class has associations with the following classes:
None

4.1.3.8 MsAgAgent Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This managed object class is the master (SNMP) agent on the host. It listens to port 161 to receive SNMP requests from management applications. It also sends SNMP traps to management applications when certain events occur. SNMPv1 is supported in Release A. MSS requires this master agent be extensible to support subagents. The agent performs authentication and authorization validations on incoming requests. If the requested MIB variables are in MIB II, it performs the functions requested. If the MIB variables are not in MIB II but in registered MIB extensions, it passes the request to the subagent which supports that particular MIB extension.

Attributes:

None

Operations:

None

Associations:

The MsAgAgent class has associations with the following classes:
Class: MsAgEncps communicates with
Class: MsAgSubAgent communicate with

4.1.3.9 MsAgAppMIB Class

Parent Class: MIBExtension
Public: No Distributed Object: No
Purpose and Description:

The SNMP MIB extension defined for applications, especially for ECS applications being managed. Three groups of objects are defined. Due to the number of variables, they are not all enumerated in this document. Each one has general attributes defined based on the functional areas of management applications such as configuration, performance, fault, and security, wherever applicable. The application-specific attributes can be defined by application developers in an extensible way using tables. Attributes: Application - the application to manage. Program - the program in the application package being managed. Process - the process of a program in the application being managed.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsAgAppMIB class has associations with the following classes:
Class: MsAgSubAgent uses

4.1.3.10 MsAgDeputy Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This object is used both by the management applications and by the subagent. The management applications can send Set requests to the subagent through this object. The subagent can send event notifications to this object so an SNMP trap can be emitted to management framework.

Attributes:

None

Operations:

MsAgDeputy - This is the constructor for this class.
Arguments:
Return Type: void
Privilege: Public

SetMibValue - This method passes the request to set the value(s) of MIB variables on remote hosts.
Arguments: int Instance, int DataType, void *value
Return Type: Integer
Privilege: Public

~MsAgDeputy - This is the destructor for this class.
Arguments:
Return Type: void
Privilege: Public

Associations:

The MsAgDeputy class has associations with the following classes:
Class: MsAgSubAgent communicateswith
Class: EcAgManager uses

4.1.3.11 MsAgEncps Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The encapsulator enables the MsAgAgent to communicate with non-Peer agents.

Attributes:

None

Operations:

None

Associations:

The MsAgEncps class has associations with the following classes:
Class: MsAgAgent communicateswith

4.1.3.12 MsAgMonitor Class

Parent Class: Not Applicable
Public: No Distributed Object: Yes
Purpose and Description:
MsAgMonitor is spawned by the MsAgSubagent. It provides the local polling capability to

monitor resources being managed. This can avoid the costly remote polling done by management applications while still being able to monitor the state of the resources. If error conditions occurred, it informs the agent to send a trap to the management application(s). The scope of this local polling is the host. The time interval applies to all the resources that are monitored by this object. This class also provides the capability for monitoring transient processes. Applications may specify that transient processes that they create be monitored for their presence until the monitoring is no longer required. Start and stop indications are needed from applications. Between the start and stop requests, the transient process is monitored. In the event that a monitored process fails, a notification is sent to the interested application.

Attributes:

None

Operations:

CheckStatus - This method performs local polling on all monitored applications on the host.

Arguments:
Return Type: Integer
Privilege: Public

```
PDL:MsAgMonitor::CheckStatus ( )
{
  // this method runs in its own thread, waking up every time quantum X
  while ( not done checking status ) {

    // for each application instance in the Object database
    for (each application instance in Object database) {

      // first, execute callbacks for all data that has a value and associated threshold
      for (each performance callback of current application) {
        execute callback
        if (returned data exceeds threshold) {
          // generate an event
          EcAgEvent::SendEvent ( EventType )
        }
      }

      // check the status of each application instance
      if ( MsAgManager::CheckStatus ( ) == REPORTABLE_STATUS ) {
        // application in a state that should be reported, so generate an event
        EcAgEvent::SendEvent ( EventType )
      }
    }
```

```
   sleep time quantum X
  }
}
```

MsAgMonitor - This method represents the constructor of the object.
Arguments:
Return Type: void
Privilege: Public

StartMonitor - This method monitors a non-OODCE server application.
Arguments: int arg1, char* destination, char* arg3
Return Type: Integer
Privilege: Public

```
PDL:PDL:MsAgMonitor::StartMonitor(char *ApplicationName, int PID, char *Arguments)
{
   //get application information from Object database
   ApplicationInfo=Get Object database information (ApplicationName)
   create file ("ApplicationInfo.ApplicationFileName.instance+ ApplicationInfo.NumInstancesRunning")
   populate file with application PID
   create submap(s) for the application instance
   increment the number of instance running the Object database
}
```

StartMonitor - This method monitors the OODCE server applications.
Arguments: int arg1, uuid_t arg2, char* arg3
Return Type: Integer
Privilege: Public

```
PDL:PDL: MsAgMonitor::StartMonitor (char *ApplicationName, DCEBindingHandle BindingHandle, char *Arguments)
{
   //get application information from Object database
   ApplicationiInfo=Get Object database information (ApplicationName)
   create ile ("ApplicationInfo.ApplicationFileName.instance+ ApplicationInfo.NumInstancesRunning")
   populate file with application instance BindingHandle
   create submap(s) for the application instance
   increment the number of instance running the Object database
}
```

StopMonitor - This method stops the monitoring of an application.
Arguments: int arg1
Return Type: void

Privilege: Public
PDL:MsAgMonitor::StopMonitor (int PID)
{
// get application information from Object database
ApplicationInfo = Get Object database information (PID)

// call the shutdown callback of the application
ApplicationInfo::ShutDown ( )

delete symbols and submaps of corresponding application instance

// delete the file signifying the application existence
delete the file ("ApplicationInfo.ApplicationFileName.instance + ApplicationInfo.InstanceNumber")
}

~MsAgMonitor - This method represents the constructor of the object.
Arguments:
Return Type: void
Privilege: Public

Associations:

The MsAgMonitor class has associations with the following classes:
Class: EcAgEvent uses
MsAgSubAgent (Aggregation)

4.1.3.13 MsAgSubAgent Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
This managed object class supports SNMP MIB extensions. It receives requests from the master agent. Based on Get or Set requests, it performs the retrieval or set functions onto resource or resource managers using available API. This object will instantiate another object MsAgMonitor to perform local polling on resources on the host.

Attributes:

infoLevel - This attribute specifies the level of information to send in the event notifications.
Data Type: Integer
Privilege: Private
Default Value:

interval - This attribute specifies the time interval for local polling.
Data Type: Integer
Privilege: Private
Default Value:

timeOut
Data Type: Integer
Privilege: Private
Default Value:

Operations:

DiscoverECSApplication - This method supports the discovery of ECS applications. This discovery may be initiated in order to initially discover all ECS applications, periodic discovery, or on demand such as after a crash of an ECS managed host or the MSS server itself. This discovery process may be initiated on either a specified host, or an all hosts. As each ECS application is discovered, the Management Agent Services issues a discovery trap to the Fault Management Application Service with information on the discovered application included as part of the variable bindings list so that the discovered application may be registered and presented visually on the appropriate submap. The discovery process may be initiated by an operator action at the user interface (customization of HP OpenView NNM user interface).
Arguments:
Return Type: Integer
Privilege: Public

```
PDL:MsAgMonitor::DiscoverECSApplication ( )
{
  // traverse the ECS file system, searching any ECS applications
  for ( each directory in ECS file system ) {

    // if a file exists with the extension .conf, then an application with the root
    // of that filename also exists.
    if ( file exists with extension ".conf" ) {

      // register the application
      create object for application in the Object database
      create submap(s) for the application

      // identify any instances of the process
      for ( each file that matches pattern "ApplicationName.instance*" ) {

        // register the instance of the application
```

```
      create object for application instance in the Object database
      create submap(s) for the application instance
    }
  }
 }
}
```

GetMibValue - This method sends request to resource manager to retrieve the value(s) of MIB variable(s).
Arguments:
PDL:EcAgSubAgent::GetMIBValue (OID oid, int instance, int dataType, void* value)

```
{
 // if the value to get is static
 if ( value is static value ) {
   // since static values are stored locally within the sub-agent, get data
   // from internal data structure
   retrieve value from internal data structure
 }
 else {
   // non static value must be retrieved through a callback within the application
   // get DCE binding to the application which contains the value to get
   get DCE binding
   bind to DCE binding

   // execute callback associated with the datatype
   value = MsAgManager_C::AgentRequestAndResponse ( RequestedValue )

 }

 // convert the value to that supported by the SNMP agent
 convert value
 return value to agent
}
```

MsAgSubAgent - This method represents the constructor of the object.
Arguments:
Return Type: void
Privilege: Public

RunTest - This method performs a test run on the host for management applications.
Arguments: char* testName, char* appName, Integer appIndex
Return Type: Integer
Privilege: Public

```
PDL:PDL: int EcAgSubAgent::RunTest (char *TestName, char *appName, int appIndex)
{
   //execute the specified test name
   return (::system (TestName))
}
```

SetMibValue - This method sets the value(s) of MIB variables on this subagent.
Arguments:

```
PDL:int MsAgSubAgent::SetValue ( int Instance, int DataType, void *value )
{
  // get DCE binding to the application instance which contains the value to be set
  get DCE binding
  bind to DCE binding

  // tell the application to set the value
  EcAgManager_S::ExecuteCallBack ( SET_DataType, value, Instance )
}
```

StartupECSApplication - This method is used to startup an ECS application.
Arguments: char* appName, Integer appIndex
Return Type: Integer
Privilege: Public

```
PDL:PDL:int MsAgMonitor::StartupECSApplication (char *appName,int appIndex)
{
Startup ECS Application
}
```

~MsAgSubAgent - This method represents the destructor of the object.
Arguments:
Return Type: void
Privilege: Public

Associations:

The MsAgSubAgent class has associations with the following classes:

Class: EcAgProxy communicateswith
Class: MsAgDeputy communicateswith
Class: MsAgAgent communicatewith
Class: MsAgAppMIB uses

4.1.4 Management Agent Services Dynamic Model

4.1.4.1 Get MIB Value

This scenario is depicted in Figure 4.1-4.

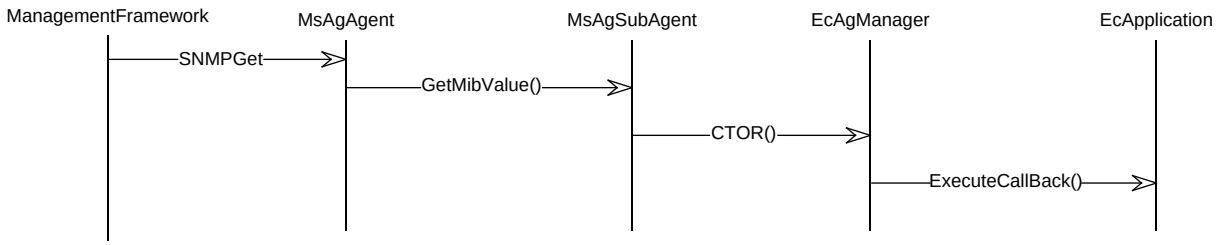


Figure 4.1-4. Get MIB Value Dynamic Model

4.1.4.1.1 Beginning Assumptions

The SNMP agent, ECS subagent, and the ECS application are all up and running.

4.1.4.1.2 Interfaces with Other Subsystems and Segments

Management Framework in MSS Fault Management

4.1.4.1.3 Stimulus

An MSS management application inquires the value of a MIB variable concerning an ECS application running on a remote host.

4.1.4.1.4 Participating Classes From the Object Model

EcAgAgent

MsAgSubAgent

EcAgManager

ECS application

4.1.4.1.5 Beginning System, Segment and Subsystem State(s)

The SNMP agent on the host is up and running. It listens to port 161 to wait for incoming SNMP requests.

The ECS subagent is functioning. It has established connections with the SNMP agent on the same host and waits for incoming requests through the agent-subagent protocol.

The ECS application is instrumented with the MSS-provided class library which includes the server part of the EcAgManager. This application is also up and running normally. The instrumented EcAgManager object has been instantiated.

4.1.4.1.6 Ending State

The SNMP agent is still listening to the port 161 for incoming SNMP requests.

The ECS subagent continuously await for incoming requests from the SNMP agent.

The ECS application is running to perform its own functions.

4.1.4.1.7 Scenario Description

A MSS management application needs to check the condition of a managed resource. That can be reflected by the value of a variable defined in ECS application MIB.

The management application issues an SNMP request to retrieve the value of that MIB variable. The request is passed from MSS Server to the SNMP agent on a particular remote host. The SNMP agent first does the authentication and authorization validations. If the request is allowed to access that MIB variable, then it checks the MIB registration tree to determine which agent or subagent is responsible for that MIB variable. For accessing the application MIB variables, it passes the request to ECS subagent through the agent-subagent protocol which is SNMP MUltipleXing protocol (SMUX).

The subagent is always waiting for incoming requests from the SNMP agent. When a request comes in, it determines which access method to use for retrieving the requested MIB variable and calls that access method. The retrieved value will be passed back to SNMP agent and then relayed back to the management application on the MSS Server.

4.1.4.2 SNMP Trap Generation

This scenario is depicted in Figure 4.1-5.

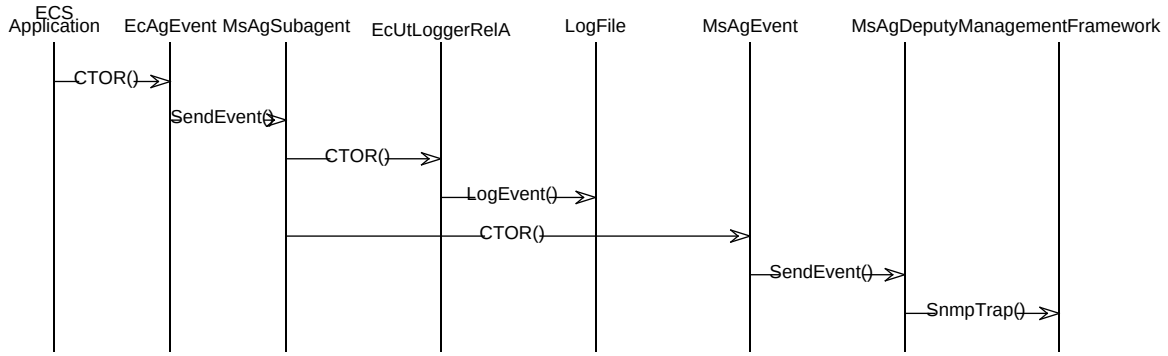


Figure 4.1-5. SNMP Trap Generation Dynamic Model

4.1.4.2.1 Beginning Assumptions

ECS applications are running on a managed ECS host. The SNMP agent and ECS subagent are all up and running. The management framework is running on MSS Server.

4.1.4.2.2 Interfaces with Other Subsystems and Segments

Management Framework in MSS Fault Management

EcUtLoggerRelA in CSS

4.1.4.2.3 Stimulus

An error condition occurs in an ECS application. The error is serious enough to inform the Fault Management application which runs on MSS Server.

4.1.4.2.4 Participating Classes From the Object Model

ECS application

EcAgEvent

MsAgMonitor

MsAgSubAgent

MsAgDeputy

4.1.4.2.5 Beginning System, Segment and Subsystem State(s)

The ECS application is instrumented with the MSS-provided class library. When the fault condition occurs, ECS application is able to send out event notifications through the sendEvent method of EcAgEvent object.The ECS subagent is up and running on the host. It is listening to receive DCE remote procedure calls.

The Deputy of SNMP manager (management framework) on MSS Server is up and running which is ready to receive DCE remote procedure calls.

4.1.4.2.6 Ending State

The ECS application may or may not be running caused by the error condition.

The ECS subagent continuous listening to receive DCE RPC calls.

The Deputy of SNMP manager continue to listen to receive DCE RPC calls.

An SNMP trap is generated to the management framework on MSS server. The fault management application detects the fault condition of that ECS application on that host.

4.1.4.2.7 Scenario Description

When a fault condition occurs in an ECS application, the application instantiates an object EcAgEvent. The application invokes the sendEvent method on that object sending an event notification to MsAgSubAgent. The MsAgSubAgent will log the event to MSS log which is managed by CSS EcUtLoggerRelA. Then, the MsAgsubAgent will check the severity of the event. If it is higher than the infoLevel, then this event notification will go further to the MSS Server. The MsAgSubAgent will instantiate an object EcAgEvent and invoke its sendEvent method to send this event to the MsAgDeputy on the MSS Server. The MsAgDeputy will then in turn convert the event to an SNMP trap and send it to the management framework which is HP OpenView.

4.1.5 Management Agent Services Structure

From a high-level point of view, the Management Agent Service includes five major parts which can be further decomposed as follows:

1. Agent
   - Extensible master agent (SNMP Agent)

- SNMP Subagent(s)
- Proxy Subagent(s)

2. Monitor
3. Event Handler
4. Agent External Interfaces
   - Manager-to-Agent interface
     - Standard management protocol (SNMP)
     - DCE RPC
   - Instrumentation API
5. Managed Object Model
   - Standard Management Information Base (MIB)
   - Private MIB extensions.

The Management Agent Service can be packaged as shown in Table 4.1-1.

Table 4.1-1. Management Agent Services Components

Component Name	COTS/Custom
MsAgAgent	COTS
MsAgSubAgent	C/C++ code and COTS
EcAgPrxy	C/C++ code
MsAgEncps	COTS
MsAgDpty	C/C++ code
EcAgInstrm	C++ code
MsAgApplMib	ASN.1

4.1.5.1 MsAgAgent - Extensible SNMP Master Agent CSC

Purpose and Description

In order to manage ECS applications, MSS requires the SNMP agent on each managed host be extensible. The extensible SNMP master agent receives SNMP requests from SNMP managers (management applications) which are on the MSS Server. The agent performs the tasks requested by the management applications. It accesses the managed resources to retrieve MIB values which are supported by this agent. For the MIB objects in extended MIBs, it passes the requests to the subagent(s) to perform the tasks.

The SNMP master agent chosen is Peer Network's agent, along with its toolkit to develop subagents and proxy agents. MSS has to customize the configuration file of the agent.

Mapping to objects implemented by this component

MsAgAgent - COTS

4.1.5.2 MsAgSubAgent - ECS Subagent CSC

Purpose and Description

This subagent supports the Application MIB which is defined by MSS. The functions it supports include:

- Retrieving Application MIB values for management applications.
- Performing local monitoring on applications to avoid excessive remote polling from management applications.
- Sending event notifications to the “deputy” of SNMP manager on MSS Server. That deputy will convert the events to SNMP traps to HP OpenView.
- Logging events onto the local MSS log.

The COTS provides libraries of API for agent-subagent interface and the development environment. The subagent code will be developed by MSS.

Mapping to objects implemented by this component

MsAgSubagt

- MsAgSubAgent - C/C++ code
- EcAgEvent_C - C++ code
- EcAgEvent_S - C++ code
- EcAgManager_C - C++ code
- EcAgManager_S - C++ code

MsAgMonitor

- CheckStatus - C++ code
- StartMonitor - C++ code
- StartMonitor - C++ code
- StopMonitor - C++ code
- SendEvent - C++ code

4.1.5.3 EcAgPrxy - DCE Proxy Agent CSC

Purpose and Description

This proxy agent is to provide the capability to manage devices or applications which don't support ECS protocols, namely non-OODCE COTS.

This proxy agent is composed of two parts. Its front-end is the instrumentation code which can communicate with the ECS subagent. Its back-end code uses COTS-specific mechanism to communicate with the COTS.

This code will be developed by MSS and COTS providers. MSS will provide the instrumentation code. The COTS-specific code has to be developed by application developers.

Mapping to objects implemented by this component

- MsAgProxy - C/C++ code
- MsAgMonitor - C++ code

4.1.5.4 MsAgEncps - Encapsulator for non-Peer Agent CSC

Purpose and Description

This Encapsulator is to enable the Peer master agent to communicate with non-Peer SNMP agent(s).

Some SNMP agent may not be extensible. In order for MSS to manage ECS applications, it is required to have an extensible SNMP agent. Therefore, the Peer's master agent will be used on each ECS managed host. This encapsulator can be used for Peer's master agent to communicate with the original SNMP agent so that the MIB it supports are still accessible.

This is COTS code. MSS only has to provide the customized configuration file.

Mapping to objects implemented by this component

- MsAgEncps - COTS

4.1.5.5 MsAgDpty - SNMP Manager's Deputy CSC

Purpose and Description

Since SNMP Set requests are not allowed. The Set functions will be done by DCE RPC calls through a “deputy” of SNMP manager on MSS Server. The SNMP manager (HPOV) sends the Set request to this deputy. It will pass the request to the remote application subagent. That subagent will decide what to do next. If it's necessary, it may invoke a call to the ECS application to perform the Set function.

The other function of this Deputy is to receive event notifications in a secure way through DCE RPC from remote hosts. It will convert the event notification to an SNMP trap and send it locally to HP OpenView.

This is a piece of custom code and will be developed by MSS.

Mapping to objects implemented by this component

MsAgDpty

- MsAgDeputy - C/C++ code
- EcAgManager_C - C++ code
- MsAgEvent_S - C++ code

4.1.5.6 EcAgInstrm - Instrumentation Class Library CSC

Purpose and Description

This class library is provided by MSS to enable the manageability of ECS applications. ECS application developers have to link this class library with their applications so that these applications can communicate with the application subagent.

This instrumentation code will be included by ECS applications and the proxy agent for COTS. It is custom code and will be developed by MSS.

Mapping to objects implemented by this component

EcAgManager

- Shutdown - C++ code

- ActionRequest - C++ code
- RegisterCallBack - C++ code
- ExecuteCallBack - C++ code

EcAgEvent

- LogEvent - C++ code
- SendEvent - C++ code

4.1.5.7 MsAgAppMib - Application MIB CSC

Purpose and Description

This class library is provided by MSS to enable the manageability of ECS applications. It includes the Agent group, the Application group, the Program group, the Process group, and the Trap group. The MIB variables of each group are in the spreadsheet format in the appendix of this document.

Mapping to objects implemented by this component

MsAgAppMib - ASN.1

4.1.6 Management Agent Services Management and Operation

4.1.6.1 System Management Strategy

The programs in Management Agent Service need to be monitored also.

Although the MsAgMonitor can monitor the MsAgSubAgent, if the subagent terminates abnormally, the monitor may not function. Therefore, it is important for fault management applications to poll the agent and subagents periodically to check the state of the subagents.

The potential performance impact will come from the following areas:

- The capability of processing incoming requests in the SNMP master agent
- The capability of processing incoming requests in the ECS subagent
- The capability of processing incoming requests in the Encapsulator
- The capability of processing outgoing event notifications in the ECS subagent
- The capability of processing outgoing event notifications in the Encapsulator
- The capability of processing outgoing traps in the SNMP master agent
- The number of threads that ECS DCE servers can have.

MSS is pursuing a task which could deliver material for this section with proper support from development teams.

4.1.6.2 Operator Interfaces

There are no GUIs for Management Agent Service.

4.1.6.3 Reports

There are no reports generated from Management Agent Service.

5. MLCI - Management Logistics CSCI

The Management Logistics CI consists of the Baseline Manager, Software Change Manager, and Change Request Manager Services. The Management Logistics context is provided in Figure 5-1.

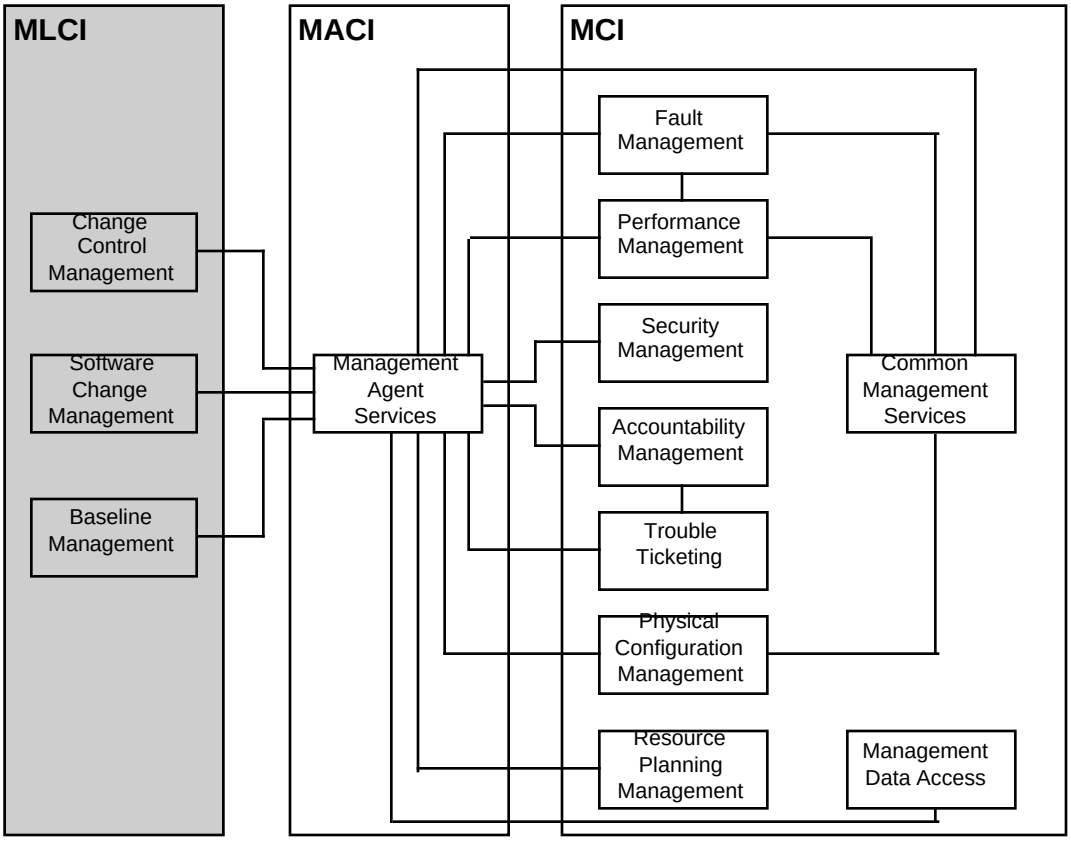


Figure 5-1. Management Logistics CI Context

5.1 Management Logistics Overview

For the TRMM release, the Management Logistics Configuration Item (MLCI) implements the Configuration Management Application Service (CMAS). It provides tools with which ECS staffs at the DAACs, EOC, and SMC track deployed ECS baselines and control changes to the hardware and software that comprise them.

CMAS maintains electronic stores of baseline data, software, and system change requests that enter the operational environment, making them and a variety of reports available for system maintenance and operations activities. It accepts ECS and algorithm software and non-real time configuration management data from formatted files or via operator interface. M&O staffs, sustaining engineers, and AIT teams rely on CMAS data stores to make, track and audit configuration changes and to help enforce ECS CM rules. They also use CMAS to produce

formatted files containing change requests, site baseline records, software, documentation, and reports that can be made available for distribution system-wide via CSS services such as e-mail, ftp, and the ECS bulletin board.

For this release, the MLCI design includes three service managers:

- Baseline Manager - COTS application that will help the DAACs, EOC, and SMC M&O staff manage ECS baseline records. The Baseline Manager was reported as being custom developed software at the PDR because a suitable COTS product had not been identified. However, since PDR, several potential products have been identified and a request for a proposal has been mailed to each of the owning vendors. Evaluation and selection of a Baseline Manager product is pending. We expect the Baseline Manager COTS application to be an interactive product that will facilitate the tracking of baselines but will require development of reports, tailoring of input forms, scripts to provide an interface to ClearCase and DDTS, and triggers for system management activities.
- Software Change Manager - A software application that will help the DAACs, EOC, and SMC M&O staffs organize and partition software, control software changes and versions and assemble sets of software for release purposes. ClearCase, a COTS product, has been selected to perform the Software Change Manager functions. Document 540-TP-001-001, Technical Paper: CSMS Preliminary Design Review Trade Studies for the ECS Project, contains a summary of the evaluation. Reference the white paper 194-00605 TPW, Use of ClearCase in the ECS Operational Environment for detailed information concerning the ClearCase evaluation. ClearCase provides extensive software library management facilities but requires development of reports, triggers to implement policies and provide an interface with DDTS and the Baseline Manager, scripts for frequently recurring operations, and views for providing access to related files.
- Change Request Manager - A software application that will enable the DAACs, EOC, and SMC staffs to register and keep track of configuration change requests (CCR), non-conformance (NCR), and deficiency reports (DR) electronically. The Distributed Defect Tracking System (DDTS), a COTS product, has been selected to perform the Change Request Manager functions. DDTS provides interactive functionality for tracking CCRs and NCRs but will require customization of forms and reports, tailoring of rules and event flow logic to the ECS environment, and scripts for interfacing with ClearCase and the Baseline Manager.

Each of these managers is described and discussed further in Section 5.3. A description the computer software units that will be developed for each manager is provided in Section 5.5.

5.2 Management Logistics Context

MLCI's sole interface to other ECS CSCIs for the TRMM release is to MSS' Management Agent CSCI (MACI). Events along this interface (see Figure 5.2-1) permit system management of configuration management applications through monitoring of faults and transfer of historical event data for audits and fault analysis.

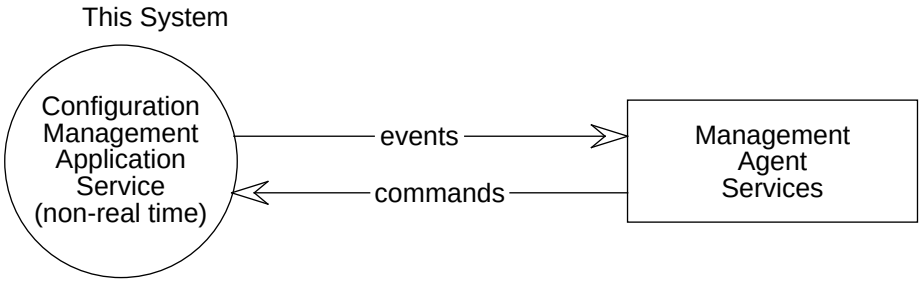


Figure 5.2-1. Management Logistics Context Diagram

5.3 Management Logistics Object Model

In the following sections, the CMAS object model is shown in three views, each capturing the distinct aspects of one service manager's capabilities. While some object classes appear in multiple views, for simplicity each view shows only those operations and attributes that are pertinent to the capabilities being described in the view.

5.3.1 Baseline Manager View

Baseline Manager software helps the DAACs, the EOC, and the SMC CM specialists maintain records that describe what comprises baselined operational system configurations (see figure 5.3-1). These records identify the versions of hardware and software items that baselines contain as well as item interdependencies. They also track the identity and location of devices and subsystems the items comprise, maintain chronological histories of the change requests and resource changes new baselines incorporate, and maintain traceability of baselines to their predecessors and their associated system releases. The Baseline Manager software at the DAACs and EOC is used to maintain data about baselines deployed to and established at the site, while the application at the SMC maintains consolidated data about baselines system-wide. The software also produces a variety of reports, including a master index of the resources comprising ECS baselines system-wide which can be accessed remotely or be posted to a configuration management folder on the ECS Bulletin Board for global viewing. The DAACs, EOC, and the SMC can exchange baseline records and reports electronically via formatted files.

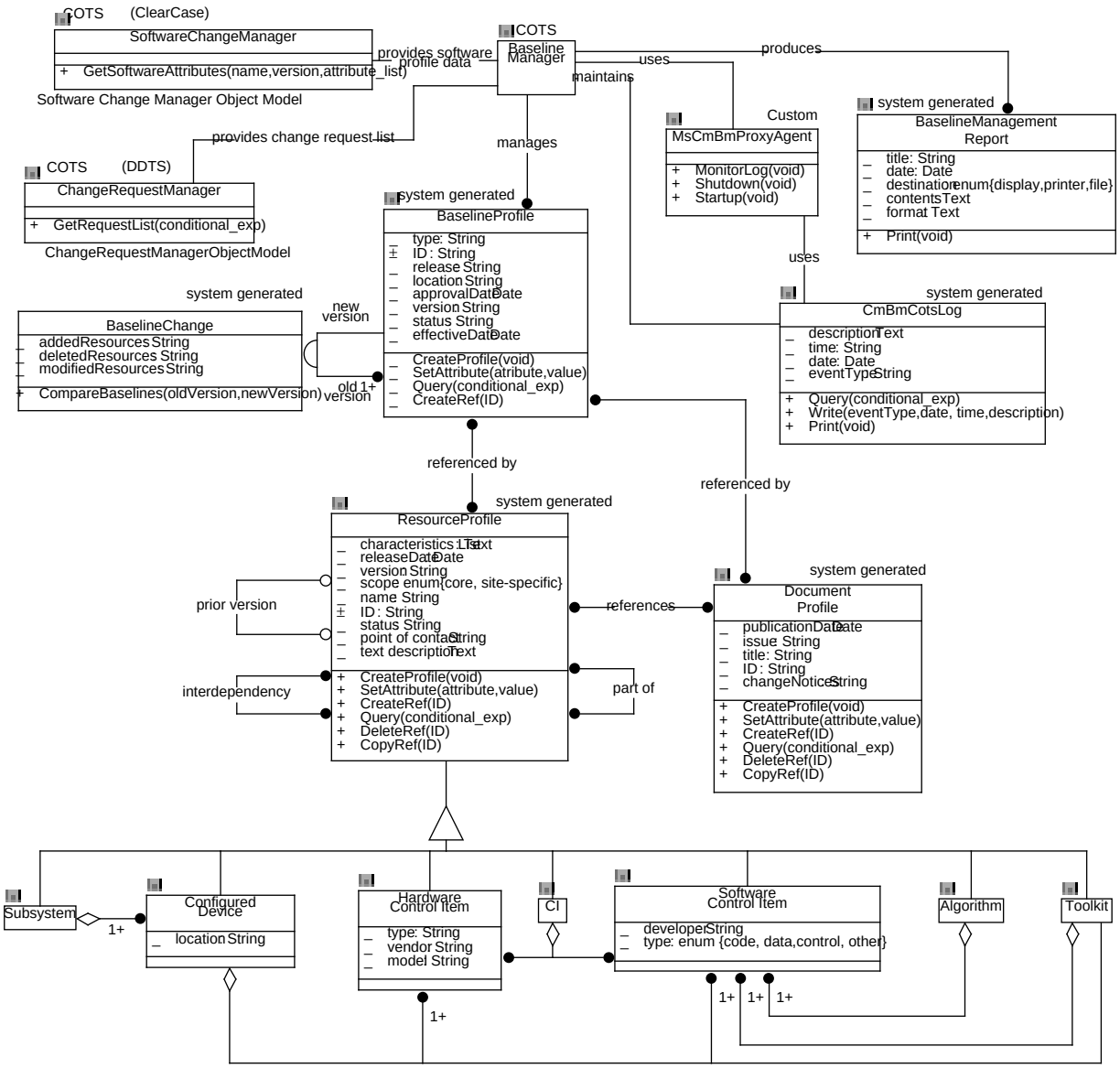


Figure 5.3-1. Baseline Manager Object Model

5.3.1.1 Algorithm Class

Parent Class: ResourceProfile

Public: No Distributed Object: No

Purpose and Description:

This class is a specialization of the ResourceProfile class. It provides a capability to identify and characterize operational algorithms.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The Algorithm class has associations with the following classes:
None

5.3.1.2 BaselineChange Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides a capability to compare two system baselines and identify what system resources changed.

Attributes:

addedResources - This attribute represents the list of the resources included in a baseline that were not in its predecessor.
Data Type: String
Privilege: Private
Default Value:

deletedResources - This attribute represents the list of the resources not in a baseline that were in its predecessor.
Data Type: String
Privilege: Private
Default Value:

modifiedResources - This attribute represents the list of the resources whose version in a baseline differs from that in the baseline's predecessor.
Data Type: String
Privilege: Private
Default Value:

Operations:

CompareBaselines - This operation builds a list of the difference in resources included in two specified baselines.
Arguments: oldVersion,newVersion
Return Type: Void
Privilege: Public

Associations:

The BaselineChange class has associations with the following classes:
None

5.3.1.3 BaselineManagementReport Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to retrieve and format historical system configuration details for display, storing, or printout.

Attributes:

contents - This attribute represents the data compiled and presented in a report.
Data Type: Text
Privilege: Private
Default Value:

date - This attribute represents the date a report is effective.
Data Type: Date
Privilege: Private
Default Value:

destination - This attribute represents the device to which a report is sent.
Data Type: enum{display,printer,file}
Privilege: Private
Default Value:

format - This attribute represents the data defining the structure of a report.
Data Type: Text
Privilege: Private
Default Value:

title - This attribute represents the name of a report.
Data Type: String
Privilege: Private
Default Value:

Operations:

Print - This operation produces a report at the set destination.
Arguments: void

Return Type: Void
Privilege: Public

Associations:

The BaselineManagementReport class has associations with the following classes:
Class: BaselineManager produces

5.3.1.4 BaselineManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a service manager that controls configuration profiles of deployed system resources and the hierarchical organizations of resources constituting operational system baselines.

Attributes:

None

Operations:

None

Associations:

The BaselineManager class has associations with the following classes:
Class: CmBmCotsLog maintains
Class: BaselineProfile manages
Class: BaselineManagementReport produces
Class: ChangeRequestManager provideschangerequestlist
Class: SoftwareChangeManager providessoftwareprofiledata
Class: MsCmBmProxyAgent uses

5.3.1.5 BaselineProfile Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides a capability to uniquely identify system baselines and the versions of deployed resources that comprise each baseline.

Attributes:

ID - This attribute represents the code used to name a type of baseline.
Data Type: String
Privilege: Protected
Default Value:

approvalDate - This attribute represents the date a baseline is formally sanctioned by an approval authority.
Data Type: Date
Privilege: Private
Default Value:

effectiveDate - This attribute represents the date a baseline is placed in production.
Data Type: Date
Privilege: Private
Default Value:

location - This attribute represents a site at which a baseline is deployed.
Data Type: String
Privilege: Private
Default Value:

release - This attribute represents the identity of the ECS system release with which the baseline is associated.
Data Type: String
Privilege: Private
Default Value:

status - This attribute represents the code that identifies whether a baseline is in test, in production, or inactive.
Data Type: String
Privilege: Private
Default Value:

type - This attribute represents the classification that distinguishes among baselines according to user-specifiable designations.
Data Type: String
Privilege: Private
Default Value:

version - This attribute represents the current revision to an established baseline.
Data Type: String
Privilege: Private
Default Value:

Operations:

CreateProfile - This operation builds a single record of baseline data and stores it in the baseline manager database.
Arguments: void
Return Type: Void
Privilege: Private

CreateRef - This operation creates and stores a reference between two objects in accordance with database referential integrity rules.
Arguments: ID
Return Type: Void
Privilege: Private

Query - This operation retrieves specified objects and attribute values from the baseline manager database.
Arguments: conditional_exp
Return Type: Void
Privilege: Private

SetAttribute - This operation assigns a named attribute a specified value.
Arguments: attribute,value
Return Type: Void
Privilege: Private

Associations:

The BaselineProfile class has associations with the following classes:
Class: BaselineManager manages
Class: DocumentProfile referencedby
Class: ResourceProfile referencedby

5.3.1.6 CI Class

Parent Class: ResourceProfile
Public: No Distributed Object: No
Purpose and Description:
This class is a specialization of the ResourceProfile class. It provides a capability to identify ECS Configuration Items to which deployed hardware and software belong.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The CI class has associations with the following classes:
None

5.3.1.7 ChangeRequestManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a service manager that controls change request data and manages operator-initiated actions associated with it.

Attributes:

None

Operations:

GetRequestList - This operation obtains values for specified attributes of specified resource change requests.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

Associations:

The ChangeRequestManager class has associations with the following classes:
Class: BaselineManager provideschangerequestlist

5.3.1.8 CmBmCotsLog Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to maintain a record of error and library modification events.

Attributes:

date - This attribute represents the date on which an event occurred.
Data Type: Date
Privilege: Private
Default Value:

description - This attribute represents the narrative that explains the nature of an event.
Data Type: Text
Privilege: Private
Default Value:

eventType - This attribute represents the classification of a system event according to source and required action.
Data Type: String
Privilege: Private
Default Value:

time - This attribute represents the time at which an event occurred.
Data Type: String
Privilege: Private
Default Value:

Operations:

Print - This operation sends an entry from the Baseline Manager log to an output device.
Arguments: void
Return Type: Void
Privilege: Public

Query - This operation reads an entry from the Baseline Manager log.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

Write - This operation adds an entry to the Baseline Manager log.
Arguments: eventType,date, time,description
Return Type: Void
Privilege: Public

Associations:

The CmBmCotsLog class has associations with the following classes:
Class: BaselineManager maintains
Class: MsCmBmProxyAgent uses

5.3.1.9 ConfiguredDevice Class

Parent Class: ResourceProfile

Public: No Distributed Object: No

Purpose and Description:

This class is a specialization of the ResourceProfile class. It provides a capability to identify and characterize individual, deployed configured devices, and it identifies the hardware and software items to comprise each.

Attributes:

location - This attribute represents the place at a site where an integrated operational ECS device can be found.

Data Type: String

Privilege: Private

Default Value:

Operations:

All Operations inherited from parent class

Associations:

The ConfiguredDevice class has associations with the following classes:

Subsystem (Aggregation)

5.3.1.10 DocumentProfile Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides a capability to maintain key, technical information about deployed system documentation. It tracks issues of system documents and relates them to specific system baselines and resources.

Attributes:

ID - This attribute represents the code that uniquely identifies system documentation.

Data Type: String

Privilege: Private

Default Value:

changeNotices - This attribute represents the list of document change notices included in the current edition of a document.

Data Type: String
Privilege: Private
Default Value:

issue - This attribute represents the nomenclature used to distinguish among versions of a single edition of a document.
Data Type: String
Privilege: Private
Default Value:

publicationDate - This attribute represents the date associated with a document.
Data Type: Date
Privilege: Private
Default Value:

title - This attribute represents the nomenclature that distinguishes document volumes from one another.
Data Type: String
Privilege: Private
Default Value:

Operations:

CopyRef - This operation creates and stores a set of references between two objects identical to the references between one of the objects and a third.
Arguments: ID
Return Type: Void
Privilege: Public

CreateProfile - This operation builds a single record of baseline data and stores it in the baseline manager database.
Arguments: void
Return Type: Void
Privilege: Public

CreateRef - This operation creates and stores a reference between two objects in accordance with database referential integrity rules.
Arguments: ID
Return Type: Void
Privilege: Public

DeleteRef - This operation removes a reference between two objects in accordance with database referential integrity rules.
Arguments: ID

Return Type: Void
Privilege: Public

Query - This operation retrieves specified objects and attribute values from the baseline manager database.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

SetAttribute - This operation assigns a named attribute a specified value.
Arguments: attribute,value
Return Type: Void
Privilege: Public

Associations:

The DocumentProfile class has associations with the following classes:
Class: BaselineProfile referencedby
Class: ResourceProfile references

5.3.1.11 HardwareControlItem Class

Parent Class: ResourceProfile
Public: No Distributed Object: No
Purpose and Description:
This class is a specialization of the ResourceProfile class. It provides a capability to identify and characterize individual, deployed hardware items.

Attributes:

model - This attribute represents the identifier for a single type of item of a manufacturer's product line.
Data Type: String
Privilege: Private
Default Value:

type - This attribute represents the classification that distinguishes hardware components according to function.
Data Type: String
Privilege: Private
Default Value:

vendor - This attribute represents the name of a hardware component's manufacturer.

Data Type: String
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The HardwareControlItem class has associations with the following classes:
CI (Aggregation)
ConfiguredDevice (Aggregation)

5.3.1.12 MsCmBmProxyAgent Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides a system management interface for lifecycle services, event reporting, and instrumentation.

Attributes:

None

Operations:

MonitorLog - This operation provides event data for the application manager.
Arguments: void
Return Type: Void
Privilege: Public

Shutdown - This operation results in termination of Baseline Manager server processes.
Arguments: void
Return Type: Void
Privilege: Public

Startup - This operation results in the start of Baseline Manager servers and associated file system setup tasks.
Arguments: void
Return Type: Void

Privilege: Public

Associations:

The MsCmBmProxyAgent class has associations with the following classes:
Class: BaselineManager uses
Class: CmBmCotsLog uses

5.3.1.13 ResourceProfile Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides a capability to maintain key, technical information about deployed hardware and software resources that both identifies and characterizes them.

Attributes:

ID - This attribute represents the code that uniquely identifies a resource profile.
Data Type: String
Privilege: Protected
Default Value:

characteristics List - This attribute represents the collection of information describing key aspects of a profiled resource's configuration.
Data Type: Text
Privilege: Private
Default Value:

name - This attribute represents the nomenclature used to identify a profiled resource.
Data Type: String
Privilege: Private
Default Value:

point of contact - This attribute represents the name of a designated individual associated with the profiled resource.
Data Type: String
Privilege: Private
Default Value:

releaseDate - This attribute represents the date the profiled resource was released to production.
Data Type: Date
Privilege: Private

Default Value:

scope - This attribute represents the classification of a profiled resource according to the number of sites where it is deployed.
Data Type: enum{core, site-specific}
Privilege: Private
Default Value:

status - This attribute represents the code that identifies the stage to which a profiled resource has reached in its lifecycle.
Data Type: String
Privilege: Private
Default Value:

text description - This attribute represents the narrative that describes a profiled resource.
Data Type: Text
Privilege: Private
Default Value:

version - This attribute represents the current revision to a profiled resource.
Data Type: String
Privilege: Private
Default Value:

Operations:

CopyRef - This operation creates and stores a set of references between two objects identical to the references between one of the objects and a third.
Arguments: ID
Return Type: Void
Privilege: Public

CreateProfile - This operation builds a single record of baseline data and stores it in the baseline manager database.
Arguments: void
Return Type: Void
Privilege: Public

CreateRef - This operation creates and stores a reference between two objects in accordance with database referential integrity rules.
Arguments: ID
Return Type: Void
Privilege: Public

DeleteRef - This operation removes a reference between two objects in accordance with database referential integrity rules.
Arguments: ID
Return Type: Void
Privilege: Public

Query - This operation retrieves specified objects and attribute values from the Baseline Manager database.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

SetAttribute - This operation assigns a named attribute a specified value.
Arguments: attribute,value
Return Type: Void
Privilege: Public

Associations:

The ResourceProfile class has associations with the following classes:
Class: BaselineProfile referencedby
Class: DocumentProfile references

5.3.1.14 SoftwareChangeManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a server manager that controls a software library and manages the execution of operator initiated events associated with it.

Attributes:

None

Operations:

GetSoftwareAttributes - This operation obtains values for specified attributes of specified library files.
Arguments: name,version,attribute_list
Return Type: Void
Privilege: Public

Associations:

The SoftwareChangeManager class has associations with the following classes:
Class: BaselineManager providessoftwareprofiledata

5.3.1.15 SoftwareControlItem Class

Parent Class: ResourceProfile
Public: No Distributed Object: No
Purpose and Description:
This class is a specialization of the ResourceProfile class. It provides a capability to identify and characterize individual, deployed software items.

Attributes:

developer - This attribute represents the organization that developed the profiled resource.
Data Type: String
Privilege: Private
Default Value:

type - This attribute represents the classification that distinguishes among the type of information contained in a profiled, software component resource.
Data Type: enum {code, data,control, other}
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The SoftwareControlItem class has associations with the following classes:
Algorithm (Aggregation)
CI (Aggregation)
ConfiguredDevice (Aggregation)
Toolkit (Aggregation)

5.3.1.16 Subsystem Class

Parent Class: ResourceProfile
Public: No Distributed Object: No
Purpose and Description:
This class is a specialization of the ResourceProfile class. It provides a capability to identify and characterize individual, deployed subsystems and the configured devices that comprise

each.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The Subsystem class has associations with the following classes:
None

5.3.1.17 Toolkit Class

Parent Class: ResourceProfile
Public: No Distributed Object: No
Purpose and Description:
This class is a specialization of the ResourceProfile class. It provides a capability to identify and characterize versions of ECS toolkits. It also identifies the ECS devices on which each toolkit resides.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The Toolkit class has associations with the following classes:
ConfiguredDevice (Aggregation)

5.3.2 Software Change Manager View

ClearCase, the Software Change Manager, provides version control for custom software, data, and documentation files (see Figure 5.3-2). It stores these files in its software library. At the SMC the ClearCase library will contain the master copy of all software that is deployed to the sites. ClearCase libraries at the DAACs will contain the software deployed to the site as well as files and

data created when tailoring the software for the local operating environment.

ClearCase's library organizes the files and automatically assigns new file version identifiers whenever a change is checked in. Associations can be created so related files and their attributes can be collectively read, archived, checked in and out, and reported on.

Views are the means by which operators select the file versions on which to work. Operations on an object in a view are regulated through use of access profiles that map userids against operations the userids are authorized to perform. (Restrictions can be incrementally tightened as a version is promoted in stages to production.) ClearCase checks out (signs out) files to be modified to reduce the risk of losing changes and helps detect conflicting changes when people work on a version in parallel. ClearCase also constructs builds and generates build records so that builds can be re-created.

ClearCase interfaces with the Baseline Manager and DDTS. In response to a GetSoftwareAttributes request, it exports to the Baseline Manager attributes that describe software, data, and documentation in its libraries. The interface with DDTS consists of a GetVerification request to evaluate a conditional expression for verifying that change to a file version has been approved.

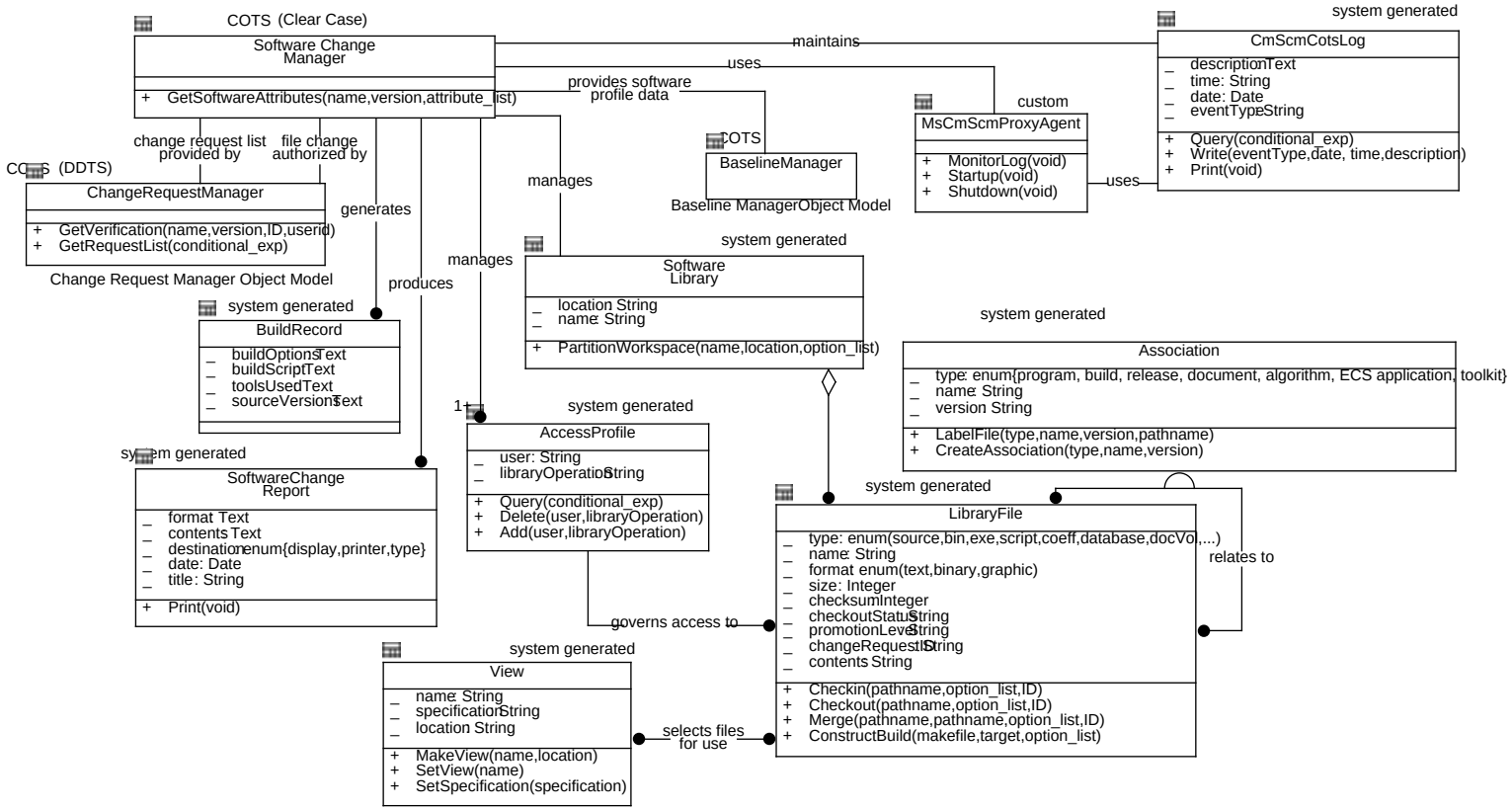


Figure 5.3-2. Software Change Manager Object Model

5.3.2.1 AccessProfile Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides the capability to maintain permissions that authorize individuals and groups to perform specific library operations.

Attributes:

libraryOperation - This attribute represents the action performed on library files.

Data Type: String
Privilege: Private
Default Value:

user - This attribute represents the UNIX logon identifier or group name to which a set of permissions can be assigned.

Data Type: String
Privilege: Private
Default Value:

Operations:

Add - This operation creates a record describing a specified operation that a specified individual or group can perform on a specified system entity.

Arguments: user,libraryOperation
Return Type: Void
Privilege: Public

Delete - This operation removes a record describing a specified operation that a specified individual or group can perform on a specified system entity.

Arguments: user,libraryOperation
Return Type: Void
Privilege: Public

Query - This operation retrieves a record describing a specified operation that a specified individual or group can perform on a specified system entity.

Arguments: conditional_exp
Return Type: Void
Privilege: Public

Associations:

The AccessProfile class has associations with the following classes:

Class: LibraryFile governsaccessto

Class: SoftwareChangeManager manages

5.3.2.2 Association Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides the capability to tag related library files in order to facilitate group operations.

Attributes:

name - This attribute represents the nomenclature for an individual collection of system data.

Data Type: String

Privilege: Private

Default Value:

type - This attribute represents the classification that distinguishes among groups of related library files on which operations can be performed collectively.

Data Type: enum{program, build, release, document, algorithm, ECS application, toolkit}

Privilege: Private

Default Value:

version - This attribute represents the rev level of a group of related resources.

Data Type: String

Privilege: Private

Default Value:

Operations:

CreateAssociation - This operation creates and stores the specification for a named association type.

Arguments: type,name,version

Return Type: Void

Privilege: Public

LabelFile - This operation establishes an association between two or more library files.

Arguments: type,name,version,pathname

Return Type: Void

Privilege: Public

Associations:

The Association class has associations with the following classes:
None

5.3.2.3 BaselineManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a service manager that controls configuration profiles of deployed system resources and the hierarchical organizations of resources constituting operational system baselines.

Attributes:

None

Operations:

None

Associations:

The BaselineManager class has associations with the following classes:
Class: SoftwareChangeManager providessoftwareprofiledata

5.3.2.4 BuildRecord Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to audit and record the files, tools, text and options used in constructing a build.

Attributes:

buildOptions - This attribute represents the collection of information about management options invoked in executing the build.
Data Type: Text
Privilege: Private
Default Value:

buildScript - This attribute represents the listing of the text in a build script.

Data Type: Text
Privilege: Private
Default Value:

sourceVersions - This attribute represents the collection of information identifying the version of source files used in constructing a build.
Data Type: Text
Privilege: Private
Default Value:

toolsUsed - This attribute represents the collection of information identifying the version of tools used in constructing a build.
Data Type: Text
Privilege: Private
Default Value:

Operations:

None

Associations:

The BuildRecord class has associations with the following classes:
Class: SoftwareChangeManager generates

5.3.2.5 ChangeRequestManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a service manager that controls change request data and manages operator-initiated actions associated with it.

Attributes:

None

Operations:

GetRequestList - This operation obtains values for specified attributes of specified resource change requests.
Arguments: conditional_exp

Return Type: Void
Privilege: Public

GetVerification - This operation evaluates a specified conditional expression on resource change request records and returns a status code.
Arguments: name,version,ID,userid
Return Type: Void
Privilege: Public

Associations:

The ChangeRequestManager class has associations with the following classes:
Class: SoftwareChangeManager changerequestlistprovidedby
Class: SoftwareChangeManager filechangeauthorizedby

5.3.2.6 CmScmCotsLog Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to maintain a record of error and library modification events.

Attributes:

date - This attribute represents the date on which an event occurred.
Data Type: Date
Privilege: Private
Default Value:

description - This attribute represents the narrative that explains the nature of an event.
Data Type: Text
Privilege: Private
Default Value:

eventType - This attribute represents the classification of a system event according to source and required action.
Data Type: String
Privilege: Private
Default Value:

time - This attribute represents the time at which an event occurred.
Data Type: String

Privilege: Private
Default Value:

Operations:

Print - This operation sends an entry from the Software Change Manager log to an output device.
Arguments: void
Return Type: Void
Privilege: Public

Query - This operation reads an entry from the Software Change Manager log.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

Write - This operation adds an entry to the Software Change Manager log.
Arguments: eventType,date, time,description
Return Type: Void
Privilege: Public

Associations:

The CmScmCotsLog class has associations with the following classes:
Class: SoftwareChangeManager maintains
Class: MsCmScmProxyAgent uses

5.3.2.7 LibraryFile Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to hold and control versions of software, documentation, control data and test data.

Attributes:

changeRequestID - This attribute represents the collection of resource change request identifiers used in authorizing checkin of a new library file.
Data Type: String
Privilege: Private
Default Value:

checkoutStatus - This attribute represents the code that identifies whether or not a library

file is in the process of being modified.
Data Type: String
Privilege: Private
Default Value:

checksum - This attribute represents the sum of the number of bits in a library file.
Data Type: Integer
Privilege: Private
Default Value:

contents - This attribute represents the collection of data stored in a library file.
Data Type: String
Privilege: Private
Default Value:

format - This attribute represents the classification that distinguishes among library files based on the way its contents are encoded
Data Type: enum(text,binary,graphic)
Privilege: Private
Default Value:

name - This attribute represents the nomenclature for an individual collection of system data.
Data Type: String
Privilege: Private
Default Value:

promotionLevel - This attribute code identifies the lifecycle stage that the contents of a library file has reached.
Data Type: String
Privilege: Private
Default Value:

size - This attribute represents the number of bytes a file contains.
Data Type: Integer
Privilege: Private
Default Value:

type - This attribute represents the classification of a library file according to kind of contents.
Data Type: enum(source,bin,exe,script,coeff,database,docVol,...)
Privilege: Private
Default Value:

Operations:

Checkin - This operation stores a new version of a file in the library.
Arguments: pathname,option_list,ID
Return Type: Void
Privilege: Public

Checkout - This operation makes a copy of a library file available for modification.
Arguments: pathname,option_list,ID
Return Type: Void
Privilege: Public

ConstructBuild - This operation executes a build script, performing one or more target rebuilds, and creates an audit record of the build.
Arguments: makefile,target,option_list
Return Type: Void
Privilege: Public

Merge - This operation forms new contents of a library file by combining two or more sets of changes that had been made to that file in parallel.
Arguments: pathname,pathname,option_list,ID
Return Type: Void
Privilege: Public

Associations:

The LibraryFile class has associations with the following classes:
Class: AccessProfile governsaccessto
Class: View selectsfilesforuse
SoftwareLibrary (Aggregation)

5.3.2.8 MsCmScmProxyAgent Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides a system management interface for lifecycle services, event reporting, and instrumentation.

Attributes:

None

Operations:

MonitorLog - This operation provides event data for the application manager.
Arguments: void
Return Type: Void
Privilege: Public

Shutdown - This operation results in termination of Software Change Manager server processes.
Arguments: void
Return Type: Void
Privilege: Public

Startup - This operation results in the start of Software Change Manager servers and associated file system setup tasks.
Arguments: void
Return Type: Void
Privilege: Public

Associations:

The MsCmScmProxyAgent class has associations with the following classes:
Class: CmScmCotsLog uses
Class: SoftwareChangeManager uses

5.3.2.9 SoftwareChangeManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a server manager that controls a software library and manages the execution of operator initiated events associated with it.

Attributes:

None

Operations:

GetSoftwareAttributes - This operation obtains values for specified attributes of specified library files.
Arguments: name,version,attribute_list
Return Type: Void

Privilege: Public

Associations:

The SoftwareChangeManager class has associations with the following classes:
Class: ChangeRequestManager changerequestlistprovidedby
Class: ChangeRequestManager filechangeauthorizedby
Class: BuildRecord generates
Class: CmScmCotsLog maintains
Class: AccessProfile manages
Class: SoftwareLibrary manages
Class: SoftwareChangeReport produces
Class: BaselineManager providessoftwareprofiledata
Class: MsCmScmProxyAgent uses

5.3.2.10 SoftwareChangeReport Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to format and order selected, software library data for display, storing, or printout.

Attributes:

contents - This attribute represents the data compiled and presented in a report.
Data Type: Text
Privilege: Private
Default Value:

date - This attribute represents the date a report is effective.
Data Type: Date
Privilege: Private
Default Value:

destination - This attribute represents the device to which a report is sent.
Data Type: enum{display,printer,type}
Privilege: Private
Default Value:

format - This attribute represents the data defining the structure of a report.
Data Type: Text
Privilege: Private

Default Value:

title - This attribute represents the name of a report.
Data Type: String
Privilege: Private
Default Value:

Operations:

Print - This operation produces a report at the set destination.
Arguments: void
Return Type: Void
Privilege: Public

Associations:

The SoftwareChangeReport class has associations with the following classes:
Class: SoftwareChangeManager produces

5.3.2.11 SoftwareLibrary Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
This class provides the capability to organize and manage the physical placement of collections of files containing ECS custom software, scientific software, and associated test data, control data, and documentation.

Attributes:

location - This attribute represents the site at which the software library is located.
Data Type: String
Privilege: Private
Default Value:

name - This attribute represents the nomenclature for an individual collection of system data.
Data Type: String
Privilege: Private
Default Value:

Operations:

PartitionWorkspace - This operation configures and administers the software library's structure.
Arguments: name,location,option_list
Return Type: Void
Privilege: Public

Associations:

The SoftwareLibrary class has associations with the following classes:
Class: SoftwareChangeManager manages

5.3.2.12 View Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class maintains sets of specifications operators use to select the version of library objects on which to work.

Attributes:

location - This attribute represents the network region, host, and local pathname at which the view is stored.
Data Type: String
Privilege: Private
Default Value:

name - This attribute represents the nomenclature for an individual collection of system data.
Data Type: String
Privilege: Private
Default Value:

specification - This attribute represents a conditional expression for qualifying library objects for inclusion in a view.
Data Type: String
Privilege: Private
Default Value:

Operations:

MakeView - This operation creates a default view, assigning it a specified name.
Arguments: name,location
Return Type: Void

Privilege: Public

SetSpecification - This operation updates the specification contained in a named view.
Arguments: specification
Return Type: Void
Privilege: Public

SetView - This operation selects the library files that satisfy the named view's specification and sets them as the context for library operations.
Arguments: name
Return Type: Void
Privilege: Public

Associations:

The View class has associations with the following classes:
Class: LibraryFile selectsfilesforuse

5.3.3 Change Request Manager View

DDTS, the Change Request Manager, maintains proposals for changing the configuration of the ECS and tracks their implementation (see Figure 5.3-3). CM specialists interact with the application to compose, register, and track the status of Resource Change Requests electronically. Several types of change requests exist, such as configuration change requests, non-conformance reports, deviations, and waivers. Amendments to change requests are versions of the initial request, and each request may include electronic attachments such as supporting data and impact assessments. Multiple data stores can exist for change requests, and change requests may be mailed among them electronically. Originators are notified when events occur, and reports containing lists of change requests, their descriptions, and their status are produced such that they can be posted to the ECS Bulletin Board.

DDTS interacts with the Baseline Manager and ClearCase. It responds to GetRequestList with resource change request data that satisfy requester-specified criteria, and it answers GetVerification requests by returning a status reflecting the result of evaluating a requester-supplied conditional expression.

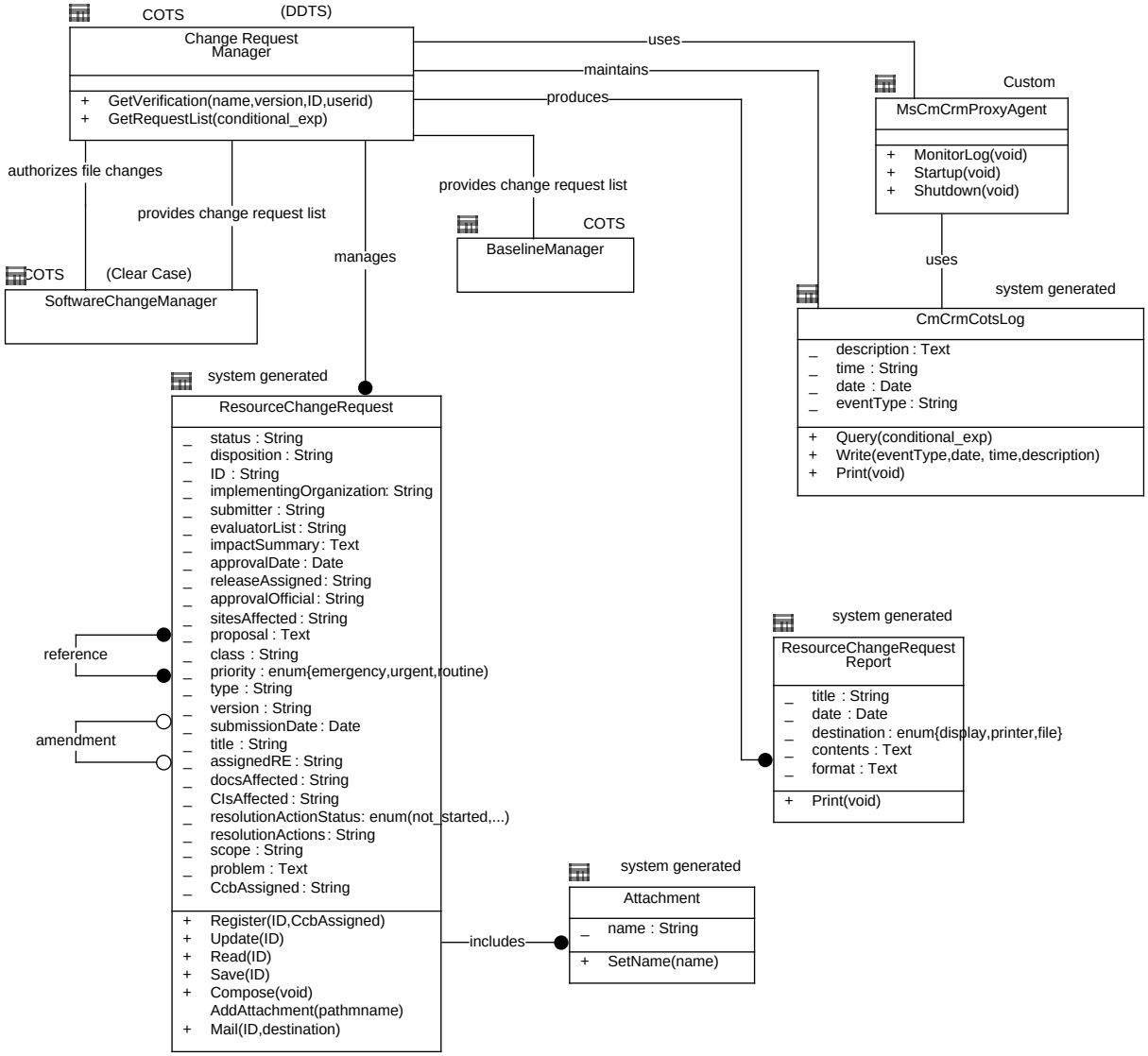


Figure 5.3-3. Change Request Manager Object Model

5.3.3.1 Attachment Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides a capability to maintain lists of files that contain support information associated with specific change requests.

Attributes:

name - This attribute represents the nomenclature used to identify a profiled resource.
Data Type: String
Privilege: Private
Default Value:

Operations:

SetName - This operation records a specified name for a specified attachment.
Arguments: name
Return Type: Void
Privilege: Public

Associations:

The Attachment class has associations with the following classes:
Class: ResourceChangeRequest includes

5.3.3.2 BaselineManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a service manager that controls configuration profiles of deployed system resources and the hierarchical organizations of resources constituting operational system baselines.

Attributes:

None

Operations:

None

Associations:

The BaselineManager class has associations with the following classes:
Class: ChangeRequestManager provideschangerequestlist

5.3.3.3 ChangeRequestManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No

Purpose and Description:
This class is a service manager that controls change request data and manages operator-initiated actions associated with it.

Attributes:

None

Operations:

GetRequestList - This operation obtains values for specified attributes of specified resource change requests.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

GetVerification - This operation evaluates a specified conditional expression on resource change request records and returns a status code.
Arguments: name,version,ID,userid
Return Type: Void
Privilege: Public

Associations:

The ChangeRequestManager class has associations with the following classes:
Class: SoftwareChangeManager authorizesfilechanges
Class: CmCrmCotsLog maintains
Class: ResourceChangeRequest manages
Class: ResourceChangeRequestReport produces
Class: BaselineManager provideschangerequestlist
Class: SoftwareChangeManager provideschangerequestlist
Class: MsCmCrmProxyAgent uses

5.3.3.4 CmCrmCotsLog Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to maintain a record of error and library modification events.

Attributes:

date - This attribute represents the date on which an event occurred.
Data Type: Date
Privilege: Private
Default Value:

description - This attribute represents the narrative that explains the nature of an event.
Data Type: Text
Privilege: Private
Default Value:

eventType - This attribute represents the classification of a system event according to source and required action.
Data Type: String
Privilege: Private
Default Value:

time - This attribute represents the time at which an event occurred.
Data Type: String
Privilege: Private
Default Value:

Operations:

Print - This operation sends an entry from the Change Request Manager log to an output device.
Arguments: void
Return Type: Void
Privilege: Public

Query - This operation reads an entry from the Change Request Manager log.
Arguments: conditional_exp
Return Type: Void
Privilege: Public

Write - This operation adds an entry to the Change Request Manager log.
Arguments: eventType,date, time,description
Return Type: Void
Privilege: Public

Associations:

The CmCrmCotsLog class has associations with the following classes:
Class: ChangeRequestManager maintains
Class: MsCmCrmProxyAgent uses

5.3.3.5 MsCmCrmProxyAgent Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides a system management interface for lifecycle services, event reporting, and instrumentation.

Attributes:

None

Operations:

MonitorLog - This operation provides event data for the application manager.

Arguments: void
Return Type: Void
Privilege: Public

Shutdown - This operation results in termination of server processes.

Arguments: void
Return Type: Void
Privilege: Public

Startup - This operation results in the start of Change Request Manager servers and associated file system setup tasks.

Arguments: void
Return Type: Void
Privilege: Public

Associations:

The MsCmCrmProxyAgent class has associations with the following classes:

Class: ChangeRequestManager uses
Class: CmCrmCotsLog uses

5.3.3.6 ResourceChangeRequest Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides a capability to maintain records that describe and status proposed changes to ECS resources.

Attributes:

CIsAffected - This attribute represents the list of the configuration items whose configuration is affected by a proposed system change.
Data Type: String
Privilege: Private
Default Value:

CcbAssigned - This attribute represents the name of a configuration control board having authority to approve the resource change request.
Data Type: String
Privilege: Private
Default Value:

ID - This attribute represents a unique identifier for a resource change request.
Data Type: String
Privilege: Private
Default Value:

approvalDate - This attribute represents the date a final decision was made concerning a proposed system change.
Data Type: Date
Privilege: Private
Default Value:

approvalOfficial - This attribute is the name of the individual whose decision is reflected in the proposed change's disposition.
Data Type: String
Privilege: Private
Default Value:

assignedRE - This attribute represents the responsible engineer designated to analyze and/ or implement a proposed system change.
Data Type: String
Privilege: Private
Default Value:

class - This attribute is the classification that distinguishes change requests according to management level needed for approval.
Data Type: String
Privilege: Private
Default Value:

disposition - This attribute represents the final decision made by a designated approval

official concerning a proposed system change.
Data Type: String
Privilege: Private
Default Value:

docsAffected - This attribute represents the list of the system documents affected by a proposed system change.
Data Type: String
Privilege: Private
Default Value:

evaluatorList - This attribute is collection of names of organizations designated to assess the impact of a proposed system change.
Data Type: String
Privilege: Private
Default Value:

impactSummary - This attribute is the information that summarizes assessments of the impact of a proposed change.
Data Type: Text
Privilege: Private
Default Value:

implementingOrganization - This attribute is the name of the organization assigned to implement a proposed change.
Data Type: String
Privilege: Private
Default Value:

priority - This attribute is the urgency with which a proposed change is needed.
Data Type: enum{emergency,urgent,routine)
Privilege: Private
Default Value:

problem - This attribute represents the narrative description of the circumstances justifying a configuration change.
Data Type: Text
Privilege: Private
Default Value:

proposal - This attribute is the narrative description of the proposed change.
Data Type: Text
Privilege: Private
Default Value:

releaseAssigned - This attribute represents the code for the ECS release in which a proposed change is targeted for implementation.
Data Type: String
Privilege: Private
Default Value:

resolutionActionStatus - This attribute represents the stage reached in accomplishing an action associated with resolving a system non-conformance.
Data Type: enum(not_started,...)
Privilege: Private
Default Value:

resolutionActions - This attribute represents the list of actions associated with resolving a system non-conformance.
Data Type: String
Privilege: Private
Default Value:

scope - This attribute represents the range of system baselines affected by a proposed change.
Data Type: String
Privilege: Private
Default Value:

sitesAffected - This attribute is the collection of names of ECS sites affected by a proposed change.
Data Type: String
Privilege: Private
Default Value:

status - This attribute identifies the stage a proposed change has reached in its lifecycle.
Data Type: String
Privilege: Private
Default Value:

submissionDate - This attribute represents the date a proposed change was first registered.
Data Type: Date
Privilege: Private
Default Value:

submitter - This attribute is a name of the individual who registers a proposed change.
Data Type: String
Privilege: Private
Default Value:

title - This attribute is the nomenclature used to identify the proposed change.
Data Type: String
Privilege: Private
Default Value:

type - This attribute is the classification that distinguishes among change requests according to form used.
Data Type: String
Privilege: Private
Default Value:

version - This attribute is the current revision/amendment to a proposed change.
Data Type: String
Privilege: Private
Default Value:

Operations:

AddAttachment
Arguments: pathname

Compose - This operation obtains and compiles information to be stored in a change request record.
Arguments: void
Return Type: Void
Privilege: Public

Mail - This operation uses Unix mail utility to send a notification of a CCR/NCR/DR to a specified address.
Arguments: ID,destination
Return Type: Void
Privilege: Public

Read - This operation gets the contents of a stored change request record for display.
Arguments: ID
Return Type: Void
Privilege: Public

Register - This operation forwards a change request record to the SMC.
Arguments: ID,CcbAssigned
Return Type: Void
Privilege: Public

Save - This operation stores a change request record in a sequential file.

Arguments: ID
Return Type: Void
Privilege: Public

Update - This operation stores a new or modified change request record in the change request database.
Arguments: ID
Return Type: Void
Privilege: Public

Associations:

The ResourceChangeRequest class has associations with the following classes:
Class: Attachment includes
Class: ChangeRequestManager manages

5.3.3.7 ResourceChangeRequestReport Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability to retrieve and format change request details and metrics for display, storing, or printout.

Attributes:

contents - This attribute represents the data compiled and presented in a report.
Data Type: Text
Privilege: Private
Default Value:

date - This attribute represents the date a report is effective.
Data Type: Date
Privilege: Private
Default Value:

destination - This attribute represents a device to which a report is sent.
Data Type: enum{display,printer,file}
Privilege: Private
Default Value:

format - This attribute represents the data defining the structure of a report.
Data Type: Text
Privilege: Private

Default Value:

title - This attribute represents a name of a report.
Data Type: String
Privilege: Private
Default Value:

Operations:

Print - This operation produces a report at the set destination.
Arguments: void
Return Type: Void
Privilege: Public

Associations:

The ResourceChangeRequestReport class has associations with the following classes:
Class: ChangeRequestManager produces

5.3.3.8 SoftwareChangeManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is a server manager that controls a software library and manages the execution of operator initiated events associated with it.

Attributes:

None

Operations:

None

Associations:

The SoftwareChangeManager class has associations with the following classes:
Class: ChangeRequestManager authorizesfilechanges
Class: ChangeRequestManager provideschangerequestlist

5.4 Software Change Management Dynamic Model

(The scenarios that follow conform to the policies and methodologies contained in 194-102-MG1-001, Configuration Management Plan for the ECS Project.)

5.4.1 Ingest Algorithm

This scenario traces the actions associated with receipt and tracking of scientific software at the sites. The scenario is depicted in Figure 5.4-1.

5.4.1.1 Beginning Assumptions

The algorithm is new.

5.4.1.2 Interfaces with Other Subsystems and Segments

None.

5.4.1.3 Stimulus

An algorithm delivery package is received by file transfer at a DAAC from an SCF and is ready to be stored in the DAAC's EOSDIS software library.

5.4.1.4 Participating Classes From the Object Model

SoftwareChangeManager

View

SoftwareLibrary

ChangeRequestManager

LibraryFile

CmScmCotsLog

BuildRecord

AccessProfile

Association

BaselineManager

BaselineProfile

ResourceProfile

5.4.1.5 Beginning System, Segment and Subsystem State(s)

SDP Toolkit exists in site software library.

Delivery package contains files to be stored and controlled.

A Resource Change Request (of type CCR) has been prepared by the site, staffed, approved for implementation at the DAAC, and registered in the SMC's Configuration Change Request (CCR) database.

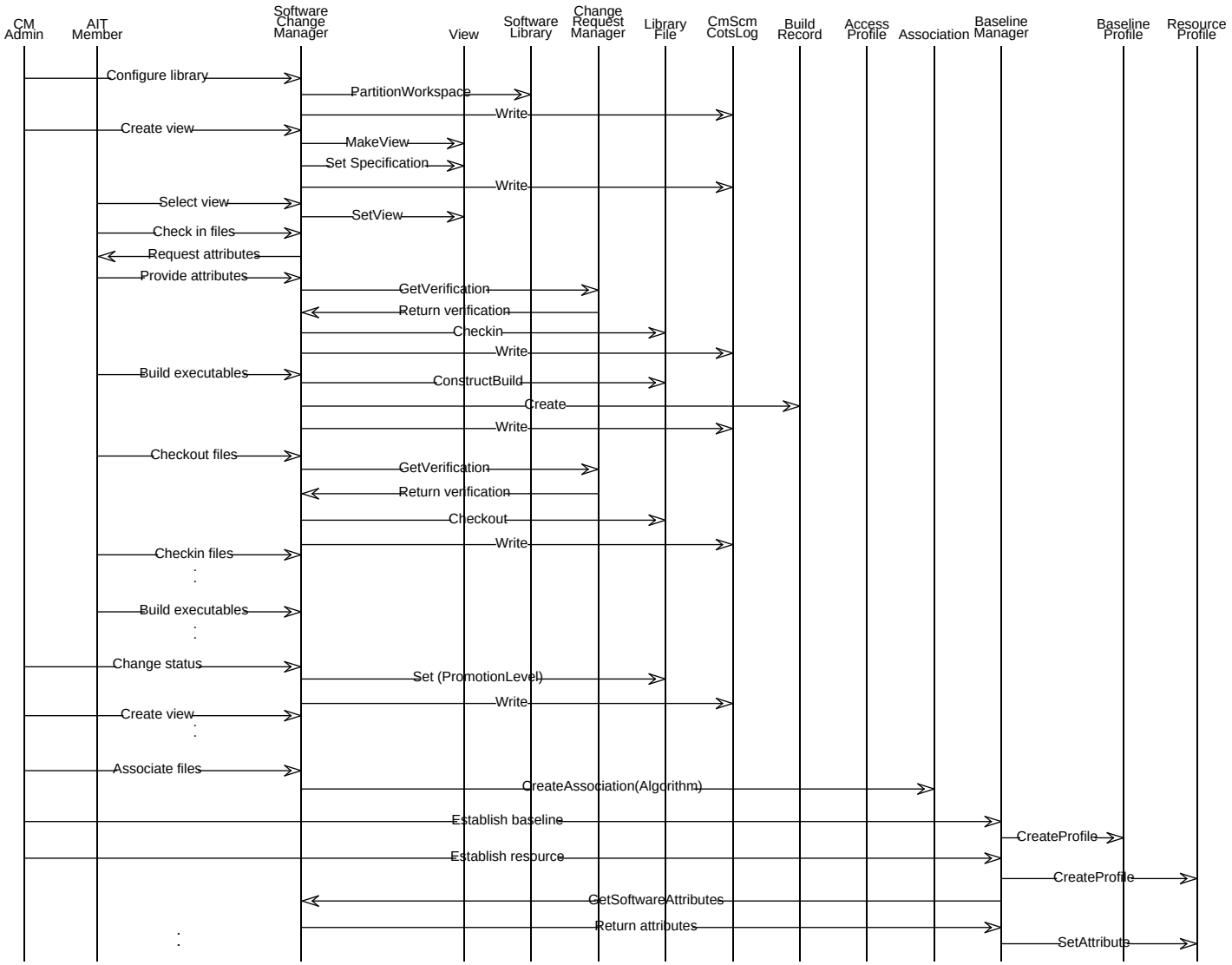


Figure 5.4-1. Ingest Algorithm

5.4.1.6 Ending State

Algorithm files are stored in library and secured from change by other than CM staff.

Records exist with which to re-create builds.

Site and SMC baseline data reflect the new algorithm.

Configuration change request is closed.

5.4.1.7 Scenario Description

The CM specialist interacts with ClearCase, the Software Change Manager, to configure the site library, establishing branches, elements, and a view for the algorithm files and access permissions for algorithm integration team (AIT) members. The library change events are logged.

AIT members use ClearCase to select the view and check algorithm files (e.g., code, calibration coefficients, controls) into the site library, supplying attribute and change request information as requested.

ClearCase sends DDTS, the Change Request Manager, a request to verify change request information and receives the verification.

The library file is checked in, and the event is logged.

AIT members direct ClearCase to build executables configured with the SCF version of the SDP Toolkit. ClearCase creates build records and stores them for reference in re-creating the builds. Library change events are logged.

Integration testing reveals that software changes are needed. AIT members, using their view, have ClearCase check out specific source files. After first verifying access, the files are made available for modification.

The modified file is checked in (described earlier), except the CCR number supplied is only checked against the DDTS database if ClearCase finds it differs from what was used during check-in of the previous version of the code. The checkin operation assigns new version identifiers to changed files. Library change events are logged.

AIT members again construct builds, this time configured with the DAAC version of the SDP Toolkit.

As the algorithm progresses in this way through stages to production, authorized persons change the life cycle status for algorithm files to reflect their promotion.

Upon approval of algorithm turnover to production, the CM specialist uses the Baseline Manager to establish a new, operational baseline record in the DAAC's Baseline Manager database and requests that a new algorithm resource profile be established using data in the software library. The Baseline Manager requests attributes from ClearCase.

The CM specialist interacts with ClearCase to create a release partition in the library for the frozen algorithm files, copy them there, lock them and label them with an algorithm identifier. A view is created with which the files can be accessed.

The CM specialist directs DDTS to update the resource change requests to reflect completed actions.

5.4.2 Record New Site Baseline

This scenario traces actions associated with recording baseline changes that introduce new configured devices at the sites that implement expanded system functionality. The scenario is depicted in Figure 5.4-2.

5.4.2.1 Beginning Assumptions

None.

5.4.2.2 Interfaces with Other Subsystems and Segments

None.

5.4.2.3 Stimulus

ECS sites' hardware and software have to undergo change due to hardware upgrade and software revision.

5.4.2.4 Participating Classes From the Object Model

Baseline Manager

HardwareControlItem

SoftwareControlItem

ConfiguredDevice

DocumentProfile

BaselineProfile

BaselineChange

Change Request Manager

BaselineManagementReport

5.4.2.5 Beginning System, Segment and Subsystem State(s)

Two resource change requests (of type CCR) that propose changes to the operational baselines at all sites have been recorded in the DDTS database and approved for implementation by appropriate CCBs. One CCR covers replacing a platform of the ECS hardware suite with a more capable model. The other CCR covers a new COTS application for use with the hardware.

Formatted files containing descriptions of the new resources and associated documentation has been received at the SMC.

Views exist that provide access to operations baselined versions of software in the library.

5.4.2.6 Ending State

Site baseline data has been updated to reflect the implemented change.

SMC baseline data has been updated to reflect the new baselines at the sites.

Resource change requests have been closed out.

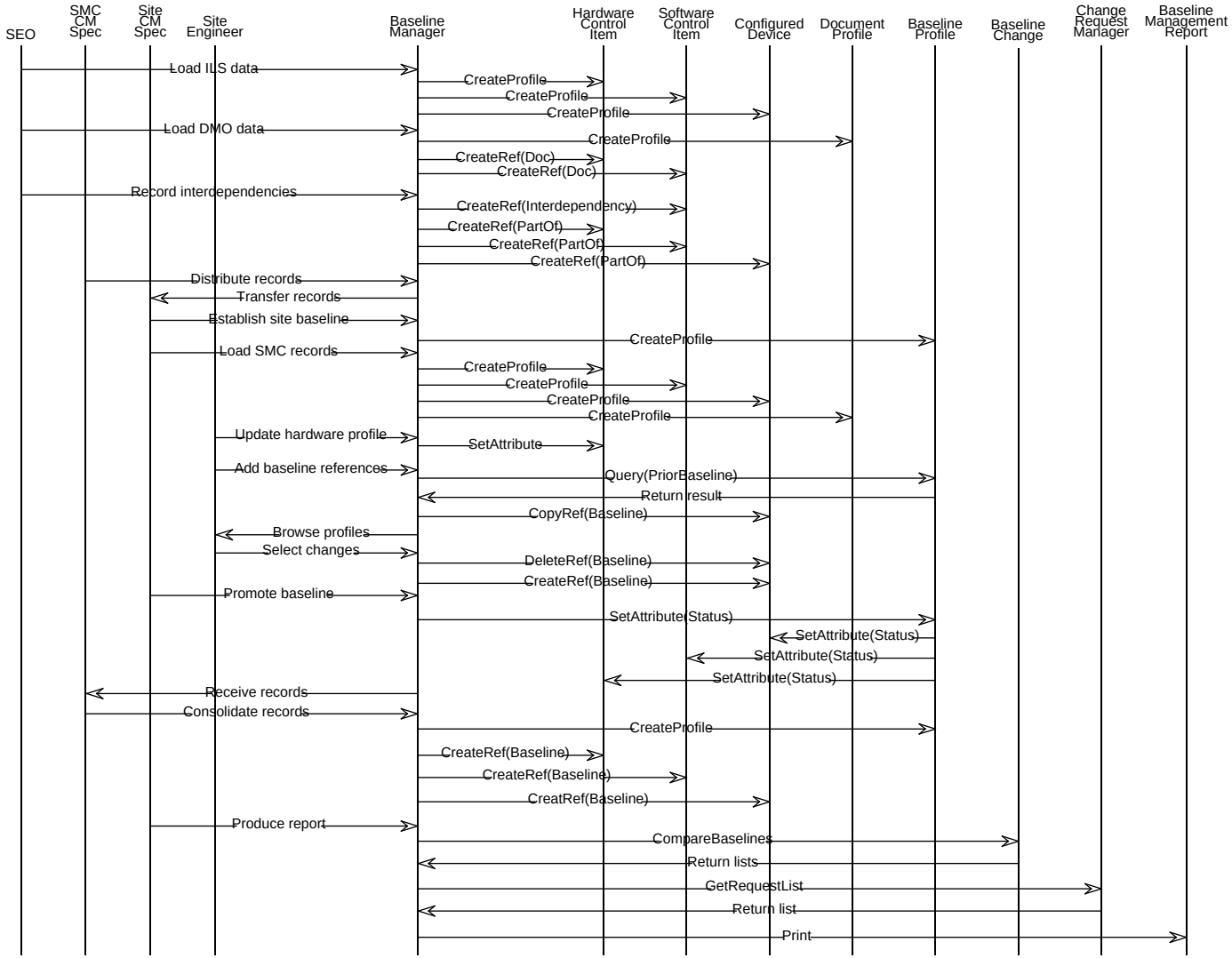


Figure 5.4-2. Record New Site Baseline

5.4.2.7 Scenario Description

SEO staff member directs Baseline Manager to load new hardware and software control item and configured device profile data from formatted files provided by the ILS Office.

SEO staff member loads new document profiles from formatted files provided by DMO.

Based on information available in the resource and document profiles, Baseline Manager automatically establishes references that associate the new resources with appropriate system documents.

SEO staff member interacts with Baseline Manager to adjust resource interdependencies stored in the database.

Based on the identifiers for the new devices, the Baseline Manager automatically establishes references that specify where they fit in the hierarchical organization of the ECS.

An SMC CM specialist distributes Baseline Manager data changes to the sites. (Method contingent on tool selection.)

Site CM specialist interacts with the Baseline Manager to establish a profile record for a new site baseline.

Site CM specialist interacts with the Baseline Manager to merge profile data from the SMC.

Once sites' engineers receive the new resources, they use the Baseline Manager to add any previously unknown information (perhaps serial numbers) to the resource profiles.

Each site's engineer establishes references between the new site baseline profile and the resources it includes. Baseline Manager automatically selects prior baseline's resources and lets the engineer browse resource profiles when selecting changes.

When appropriate, the site CM specialist changes the baseline profile's status to reflect production. Baseline Manager automatically updates the status of corresponding resources the baseline includes and freezes the baseline-related records.

Sites' CM specialists send their new baseline records to the SMC. (Method contingent on tool selection.)

SMC CM specialist generates consolidated reports suitable for posting to the ECS Bulletin Board. Baseline Manager calls on DDTS to provide the relevant CCR list.

5.4.3 Evaluate System Enhancement Request

This scenario traces the actions associated with registering, assessing, and recording status of proposed changes to ECS resources. The scenario is depicted in Figure 5.4-3.

5.4.3.1 Beginning Assumptions

None.

5.4.3.2 Interfaces with Other Subsystems and Segments

CSS

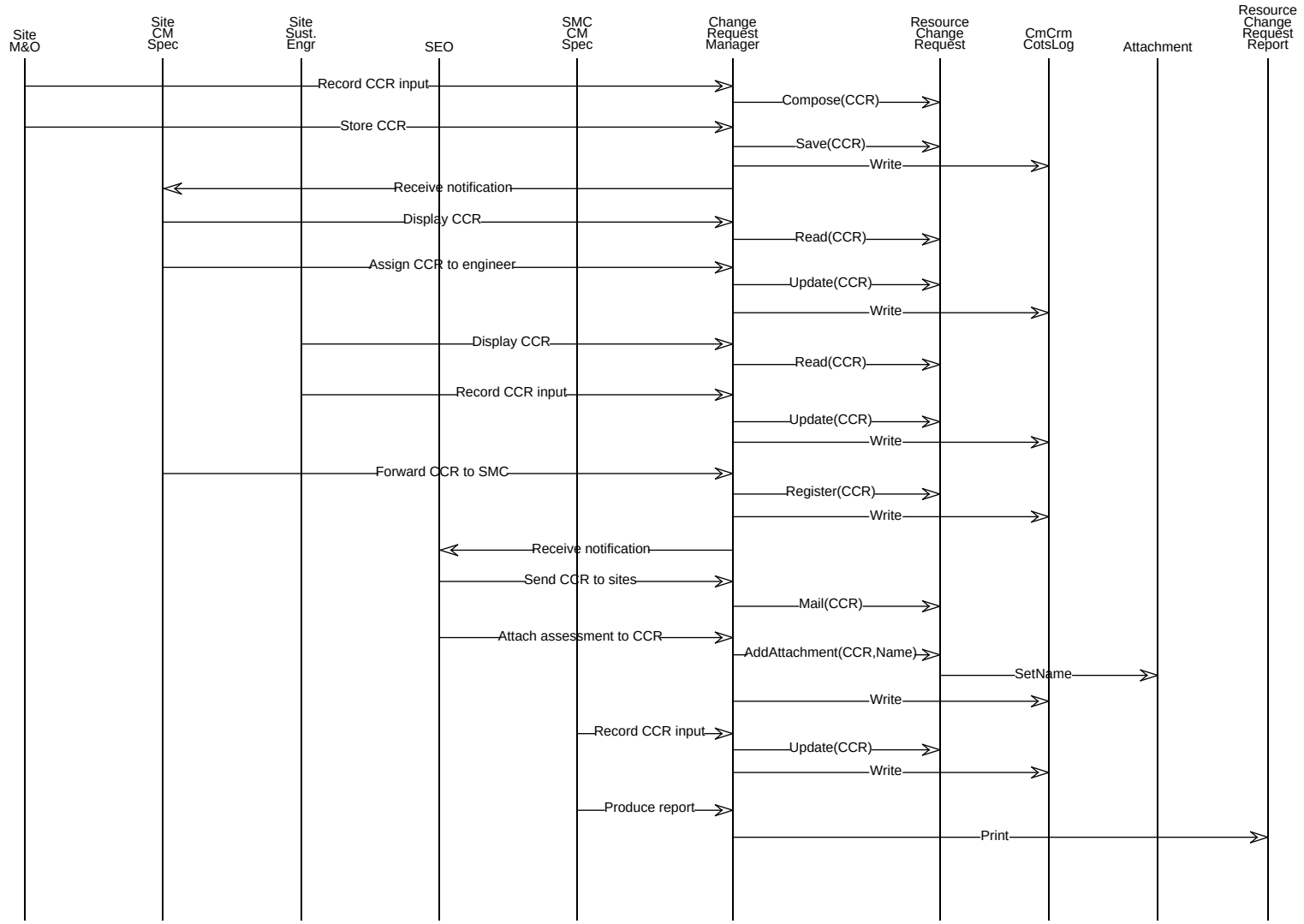


Figure 5.4-3. Evaluate System Enhancement Request

5.4.3.3 Stimulus

A member of the user community recommends to a site that the ECS might be modified in some specific way.

5.4.3.4 Participating Classes From the Object Model

Change Request Manager

ResourceChangeRequest

CmCrmCotsLog

Attachment

ResourceChangeRequest Report

5.4.3.5 Beginning System, Segment and Subsystem State(s)

None.

5.4.3.6 Ending State

A change request record exists in change request database and contains the request's description, its impact summary, and its disposition by the CCB.

The change request record references associated assessments and assessment requests.

The change request and its current status are made available for posting to the ECS bulletin board.

5.4.3.7 Scenario Description

An M&O staff member interacts with DDTS to compose an electronic resource change request (in this case, a configuration change request (CCR)). The staff member directs DDTS to save the request. DDTS saves the CCR and notifies the site CM administrator of the presence of the CCR.

Site CM Administrator reviews the CCR for completeness, and assigns it to the site sustaining engineer for evaluation.

Sust. Engr. evaluates CCR, determines that the CCR has system-wide impact, and enters his/her analysis information into the CCR record. Sustaining Engineer recommends site CCB approves sending CCR to the sustaining engineer organization (SEO) at the SMC for assessment and to ESDIS CCB for approval action.

At site CCB's direction, the site CM specialist directs DDTS to send the CCR to the SEO at the SMC. DDTS mails the request to the SMC CM specialist and the SEO is notified.

An SEO staff member directs DDTS to mail the CCR to all site sustaining engineers for site assessment. As impact assessments are received, the SEO staff member directs DDTS to attach them to the change request.

SEO reviews assessments and sends copy of CCR and its recommendation to ESDIS for approval.

At CCB direction, the SMC's CM specialist uses DDTS to record CCR disposition, identity of the organization assigned responsibility for implementation, and to update the change request's status.

The SMC's CM specialist directs DDTS to produce a report that lists all of the active CCRs and make it available for posting to the ECS Bulletin Board for system-wide viewing.

5.5 Management Logistics Structure

5.5.1 Baseline Manager CSC

This CSC provides all the functionality necessary to record and track what resources constitute ECS operational baselines system-wide. The CMAS object classes it includes are shown in Table 5.5-1.

Table 5.5-1. Baseline Manager Components

CMAS Object Class	Implementation
BaselineManager	COTS
BaselineProfile	System generated
BaselineChange	System generated
ResourceProfile	System generated
DocumentProfile	System generated
Subsystem	System generated
ConfiguredDevice	System generated
HardwareControlItem	System generated
CI	System generated
SoftwareControlItem	System generated
Algorithm	System generated
Toolkit	System generated
CmBmCotsLog	System generated
BaselineManagementReport	System generated
MsCmBmProxyAgent	Custom

The Baseline Manager CSC is a transaction-oriented, data-intensive application that will likely perform many more queries than updates. Data store operations are relatively infrequent and accomplished largely interactively by a CM specialist using a keyboard alone for direct data entry or together with files containing formatted data from other applications such as ClearCase or a Baseline Manager at a different site.

The CSC consists of COTS clients and servers for which the following CSUs will be developed:

- Profile Definitions - customization of COTS-provided specifications for baseline manager's records, for baseline profiles, document profiles, and resource profiles
- Profile Input Forms - customization of COTS-provided screens and triggers to accommodate customized profile records
- Profile Import Script - custom script to facilitate adding formatted file data into the baseline manager database

- Profile Export Script - custom script to facilitate producing formatted file data from the database
- Profile Update Triggers - custom coding in native COTS language that implements procedural rules and efficiencies such as automatic carry forward of resource profile links when a new baseline profile is created; specific triggers with identified when a COTS product is selected
- Baseline Manager Pre-defined Reports - custom coding in native COTS language or customization of COTS-provided specifications for the following menu-selectable, pre-defined reports:
  - Configured Articles List
  - Baselined Documents Report
  - As-Built Resource List
  - Baseline Manager Profiles Report
  - Baseline Change Report

The Release A Overview Design Specification (305-CD-004-001) describes these reports.

- System Management Triggers - custom coding in native COTS language to invoke MACI-provided objects that report application events and handle system management callbacks.
- Baseline Manager Proxy Agent - customization of Management Agent CSCI's proxy agent. C++ code for interacting with application management agents.

5.5.2 Software Change Manager CSC

This CSC provides all the functionality necessary to version control files containing custom software, data, and documentation. The CMAS object classes it includes are shown in Table 5.5-2.

Table 5.5-2. Software Change Manager Components

CMAS Object Class	Implementation
SoftwareChangeManager	COTS (ClearCase)
SoftwareLibrary	System generated
LibraryFile	System generated
Association	System generated
View	System generated
AccessProfile	System generated
BuildRecord	System generated
CmScmCotsLog	System generated
SoftwareChangeReport	System generated
MsCmScmProxyAgent	Custom

As indicated previously, ClearCase, a COTS application, will perform the Software Change Manager function. This client/server application provides extensive library management facilities, but requires development of the following CSUs:

- Library Installers - customization of COTS-provided scripts and specifications to initially configure a software library, including establishing physical file storage locations, creating storage directories, and defining standard attributes that will be maintained for files stored in the library.
- Library View Files - customization of COTS-provided specifications for initial and default views of library objects. This include defining commands to set up a view (s) for group use, naming the view (s), specifying the location for the view storage directory, creating view storage directory, and setting up view for a default set of files.
- File Checkin Trigger - custom coding in native COTS language to enforce file change policies. This trigger monitors the “check-in of files” oriented operations. The trigger executes a script which ask for a CCR/NCR/DR number that's associated with the file being checked in, calls DDTS to verify the number (if not previously verified), evaluates the code returned from DDTS, allows check-in of the file if the return code indicates number is valid, rejects check-in of the file if number is invalid and provides reason for the rejection.
- File Checkout Trigger - custom coding in native COTS language to enforce file change policies. This trigger monitors the “check-out of files” oriented operations. The trigger executes a script which ask for a CCR/NCR/DR number that's associated with the file being checked out, calls DDTS to verify the number and its association with the requested file, evaluates the code returned from DDTS, allows check-out of the file if the return code indicates number is valid, rejects check-out of the file if number is invalid and provides reason for the rejection.
- Baseline Manager Interface Script - custom coding in native COTS language to service calls from the Baseline Manager. This script uses the parameters passed from the Baseline Manager to locate specified software file(s) in the software library, gets the file(s) attributes values and place values in a formatted file.
- ClearCase Pre-defined Reports - custom coding in native COTS language or customization of COTS-provided specifications for the following menu-selectable, pre-defined reports:
  - Software Library Objects Report
  - Software Library Builds Report
  - Software Library Version Tree List
  - Software Library Registered View List
  - Software Library View Specification List
  - Software Library Checkouts List
  - Software Library Event History List

The Release A Overview Design Specification (305-CD-004-001) describes these reports.

- System Management Triggers/Scripts - custom coding in native COTS language to invoke MACI-provided objects that report application events and handle system management callbacks.
- ClearCase Proxy Agent - customization of Management Agent CSCI's proxy agent. C++ code for interacting with application management agents.

5.5.3 Change Request Manager CSC

This CSC provides all the functionality necessary to compose, submit, coordinate, and track status of proposals to change ECS resources. The CMAS object classes it includes are shown in Table 5.5-3.

Table 5.5-3. Change Request Manager Components

CMAS Object Class	Implementation
ChangeRequestManager	COTS (Distributed Defect Tracking System, (DDTS))
ResourceChangeRequest	System generated
Attachment	System generated
CmCrmCotsLog	System generated
ResourceChangeRequestReport	System generated
MsCmCrmProxyAgent	Custom

Like the Baseline Manager, DDTS, the Change Request Manager, is an interactive, transaction-oriented application. This client/server application requires development of the following CSUs:

- Resource Change Request Definitions - customization of COTS-provided specifications for resource change request records.
- Resource Change Request Input Forms - customization of COTS-provided screens and triggers to accommodate customized resource change requests.
- GetVerification Script - custom coding in native COTS language to service calls for evaluations of specified conditional expressions on resource change request records.
- GetRequestList Script - custom coding in native COTS language to service calls for values for specified attributes of specified resource change requests.
- Mail-related Triggers - custom coding in native COTS language that implement notification policies.
- DDTS Pre-defined Reports - Custom coding in native COTS language or customization of COTS-provided specifications for the following menu-selectable, pre-defined reports:
  - Change Request Report
  - Change Request List
  - Change Request Metrics Report

The Release A Overview Design Specification (305-CD-004-001) describes these reports.

- System Management Triggers - Custom coding in native COTS language to invoke MACI-provided objects that report application events and handle system management callbacks.
- DDTS Proxy Agent - customization of Management Agent CSCI's proxy agent. C++ code for interacting with application management agents.

5.6 MLCI Management and Operation

5.6.1 System Management Strategy

MLCI's configuration management CSCs require administration common to transaction-oriented applications that maintain repositories of data. Administrative operations include startup, shutdown, backup, performance tuning, and data maintenance. Scripts that facilitate performing these operations are available to local operators or system administrators via the CSCs, UNIX shell, and -- in the case of startup and shutdown - via the Fault Manger.

MLCI's servers operate continuously under normal circumstances. Each MLCI CSC uses an event log, maintained internally by the COTS application, that identifies and timestamps errors and changes to management data when they occur. These logs can be read, printed, and archived in a manner that can be tailored to local management policies. Selected events are also automatically reported to other, local MSS applications via MSS' management agents.

In addition to the COTS applications, MLCI contains metadata, configuration parameters, as well as mechanisms that implement system management policies. Installation will provide an initial configuration for distribution of databases, database structures, and default specifications for system resources, access controls, reports and data views. Generally, these can be customized to enhance performance and to accommodate growth. Data stores will require periodic maintenance to remove obsolete or unwanted records, references, and work files and to archive historical data. As usage expands, specifications for data viewing and control will need adjusting and storage may need reconfiguring. The latter might include establishing new data partitions, adjusting and moving data stores across devices, and resizing buffer caches where possible in response to performance indicators.

5.6.2 Operator Interfaces

All MLCI CSCs will offer operators both graphical and command line interfaces as provided by their respective COTS vendors and described in their user manuals. In some cases, custom menus and scripts will be added to facilitate accomplishing repetitive or complex actions. These enhancements will be identified once COTS products are selected and procured.

In addition, operators can invoke select management options (such as startup and shutdown) via a common management user interface. This is provided by the Fault Management Application Service and is described in Section 4.1.

5.6.3 Reports

MCI provides a variety of standard MSS fault, configuration, accounting, performance, and security reports needed to support application management ECS-wide. These are described in the Release A Overview Design Specification (305-CD-004-001). In addition to these, each MLCI COTS application will produce:

a) event history reports for diagnosing problems and auditing database changes;

b) access profile reports for controlling use of the application.

6. MCI - Management Software CSCI

The Management Software CI consists of the Fault Management Service, the Performance Management Service, the Security Management Service, the Accountability Management Service, the Physical Configuration Management Service, the Trouble Ticketing Service, the Management Data Access Service, the Ground Events Planning Service, and the Management Database. The Management Software CI context is provided in Figure 6-1.

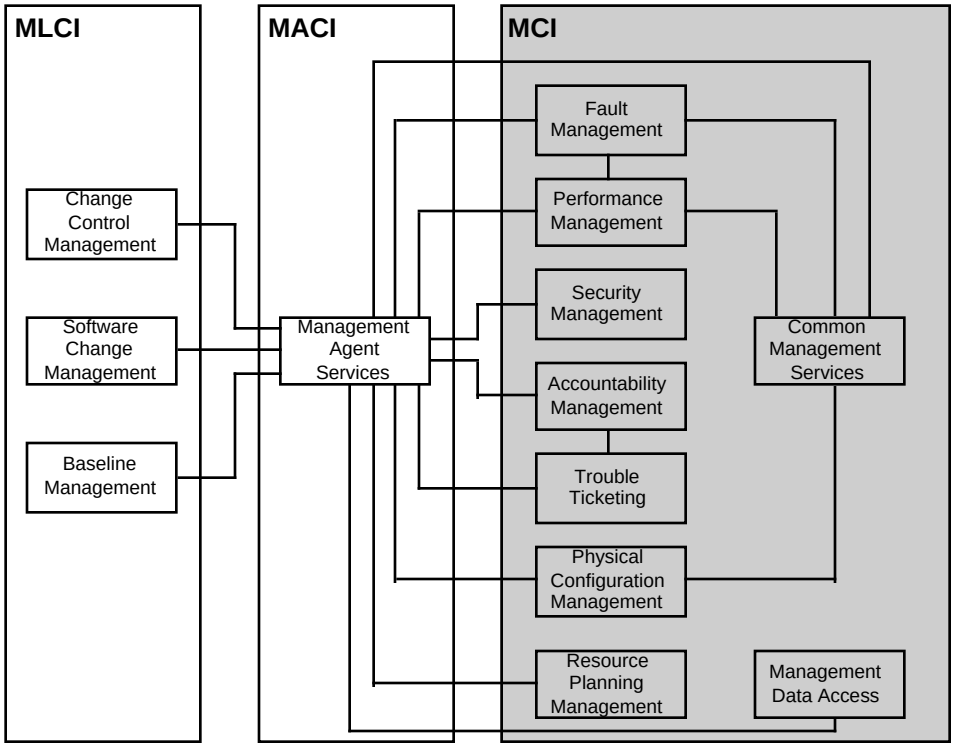


Figure 6-1. Management Software CI Context

6.1 Fault Management

6.1.1 Fault Management Overview

The Fault Management Application Service provides the capability to detect, diagnose, isolate and recover from faults that occur in the managed objects within ECS. The entities or managed objects in ECS that need to be monitored for faults include network devices (such as hosts, hubs and routers), systems software (databases and middleware such as DCE) and applications (such as the Planning Subsystem and the Data Server Subsystem). Fault Management encompasses activities such as the ability to trace faults through the system, to execute diagnostic tests, and to initiate corrective or recovery actions upon the isolation of errors in order to correct the faults. The

detection of faults involves the identification of an unacceptable change in the state of a managed object. The diagnosis and isolation of a fault involves the determination of the cause of the fault from the recorded symptom, through the use of diagnostic tests, where necessary. The recovery from a fault condition involves the initiation of a corrective action in order to restore the system to normal operational status.

The Fault Management Application Service has two instances: at each of the DAACs and at the SMC. The Fault Management Application Service resident at each DAAC collects and operates on fault data local to the site. Summaries of this data are sent periodically to the SMC. The SMC Fault Management Application Service operates on these summaries of fault data collected system-wide by Fault Management Application Service at the various DAACs in order to perform system-wide fault trends analysis.

The Fault Management Application Service at each DAAC provides the capability to generate notifications of fault conditions and alert indicators in the event of defined thresholds being exceeded. It provides diagnostic information and the diagnostic tests that facilitate the isolation, location and the identification of the cause of the faults local to the DAAC. It further provides the mechanisms for the generation of notifications upon the detection of faults, and the mechanisms for the definition of automated actions to be executed in response to the occurrence of well-defined faults or events. The DAAC Fault Management Application Service, provides the mechanism to generate reports based on information in its database. The Fault Management Application Service at each site, sends summary data periodically to the SMC for trends analysis.

Since a fault is an unacceptable change in the state of a managed object, it follows that the Fault Management Application Service provides for the detection of changes in the state of managed objects in order to be able to distinguish the unacceptable changes that constitute faults from acceptable changes. The Fault Management Application Service, therefore, provides the capabilities for real-time configuration management to include the startup, shutdown and discovery of ECS applications. Further, since the service maintains the status of resources, it provides the capability to provide the status of these resources, such as processors and associated disks, upon requests from subsystems such as the Planning Subsystem.

The SMC Fault Management Application Service provides the mechanism to receive notifications of fault conditions from the Fault Management Services at the DAACs. This is expected to facilitate the coordination of the isolation, diagnosis and the resolution of multi-site and system-wide faults, disruptions, and security events such as break-in attempts. This may, in some cases, include coordination with external providers for the analysis and recovery from fault conditions. The SMC Fault Management Application Service provides the mechanism to generate reports based on the information it collects and receives from the various Fault Management Application Services at the DAACs.

Faults in hardware devices are detected and reported through the use of a combination of the industry standard Simple Network Management Protocol traps and IP status polling. Faults in software are reported by the Management Agent Services. ECS applications may report faults to the Fault Management Application Service through the use of a public class exported by the Management Agent Service. This class is described in detail in the section on Management Agent Services. The section on Errors, Exceptions and Fault Handling in the Release A Overview Design Specification (305-CD-004-001) provides the context and the criteria for usage of this mechanism

for ECS applications to report their faults. Table 6.1-1 provides a representative sample of the faults and events detected and reported by the Fault Management Application Service for different managed objects in the system. The list of faults/events and the managed objects is not all-inclusive. The Fault Management Application Service provides for the notification of any type of event associated with a managed object through the public classes exported by the Management Agent Services.

Table 6.1-1. Faults and Events Reported by ECS Managed Objects

Managed Object	Fault/Event
Standard SNMP Traps	Cold Startup
Warm Startup	Link Up
Link Down	Authentication Failure
Network Device	Node Added
Node Deleted	Node Down
Node Marginal	Node Unknown
Node Up	Interface Card
Interface Added	Interface Deleted
Interface Disconnected	Interface Down
Interface Marginal	Interface Unknown
Interface Up	Interface Unmanaged
Disk drive	Disk drive on-line
Disk drive off-line	Disk drive warning
Disk drive unknown state	Printer
Printer printing	Printer idle
Printer warming	Tape drive
Tape drive on-line	Tape drive off-line
ECS Application	Application failed
Application missing	Application startup
Application shutdown	Application discovered
ECS Database	Database up
Database down	GTWAY
Database update error	GTWAY
Database access error	GTWAY
Failure to allocate memory	GTWAY
Database lock table full	GTWAY
Database media failure	PRONG
Data staging/destaging error	INGST
Host not connected	STMGT
Configuration file corrupted or deleted	STMGT
User pull area full	STMGT
Queue full	STMGT
Network failure errors	

6.1.1.1 Fault Management COTS

HP OpenView Network Node Manager has been selected as the Enterprise Management Framework. This COTS product inherently provides the capabilities for fault and configuration management of TCP/IP networks (SNMP devices). In the Object Model, Section 4.1.3, HP OpenView Network Node Manager is represented by the object labeled ManagementFramework. This product provides capabilities and features to allow customization for fault and configuration management of the network. This customization, represented as MsFlConfig in the Object Model, includes the following tasks:

- discovery of IP-addressable devices on the network
- creation maps and submaps
- add discovered managed objects to the appropriate submaps to graphically represent the topology of the network
- change and propagate status of managed object based on faults/events
- definition faults to detect
- definition of monitoring criteria
- definition of thresholds on attribute values
- definition of notification mechanisms
- definition of forwarding criteria
- definition of automatic actions to be executed in response to specific faults
- association recovery actions with faults
- configuring the event log browser for browsing of fault/events

The product also provides application programming interfaces (APIs) and an extensible graphical user interface to allow its capabilities to be extended, through custom development, for the fault and configuration management of non-SNMP entities such as ECS applications. This custom development, with PDL, is discussed in the appropriate sections of the object model.

6.1.2 Fault Management Context

The Fault Management Application Service, as shown in the context diagram, interfaces with ECS managed objects (via Management Agent Services), and with systems external to ECS, namely EBNET, SDPF, TRMM, NOLAN, NOAA, NSI, PSCN and local campus networks. The managed objects, as described in the previous section, comprise hardware resources (such as routers, hosts and hubs), systems software (including databases and middleware) and ECS applications (such as the Data Server, the Planning subsystem). The information exchanged across these interfaces, as shown in the diagram, is described here.

The Management Agents co-resident with the managed object classes provide notifications of faults with diagnostic information to the Fault Management Application Service. The Performance Management Application Service sends notifications of conditions of degradation of performance to the Fault Management Application Service. The Security Management Application Service sends notifications of security events to the Fault Management Application Service.

The external systems provide fault notifications, fault status, estimated down of resources due to the fault, results of fault analysis, fault resolution information, and summary fault and performance information.

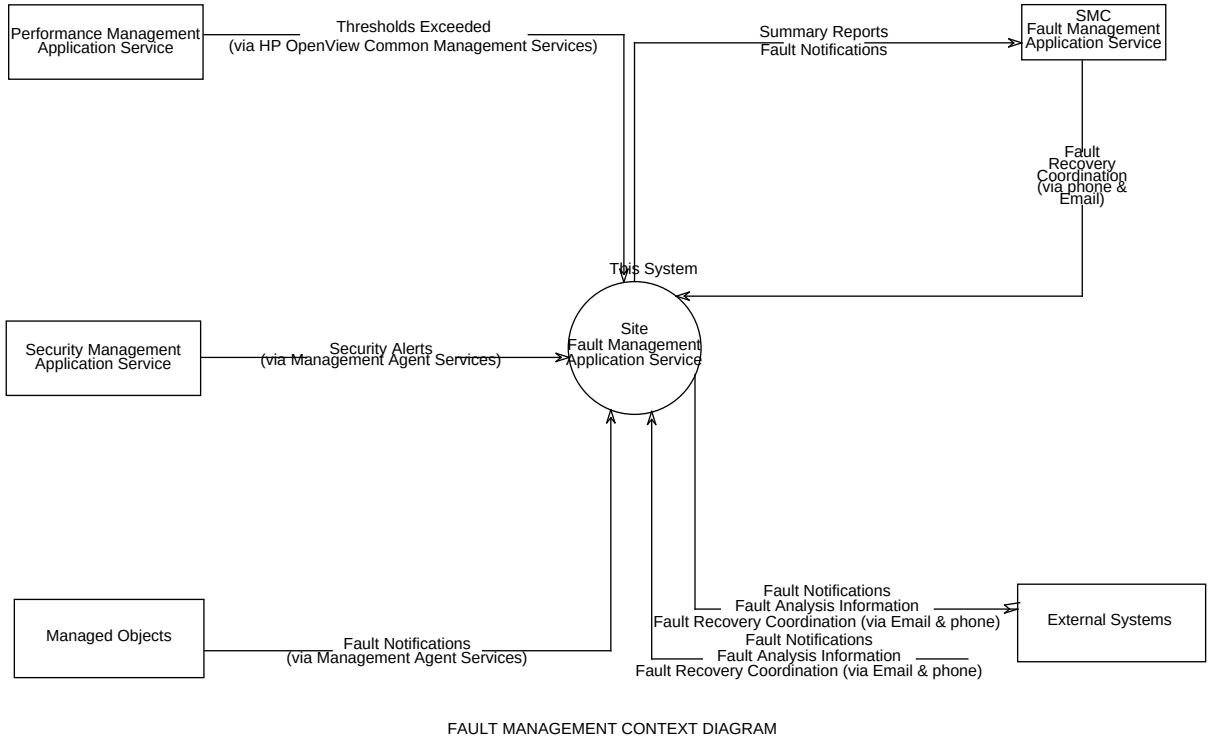


Figure 6.1-1. Fault Management Context Diagram

6.1.3 Fault Management Object Model

Figure 6.1-2, the Object Model for Fault Management depicts the major classes and their associations with one another. These are described below:

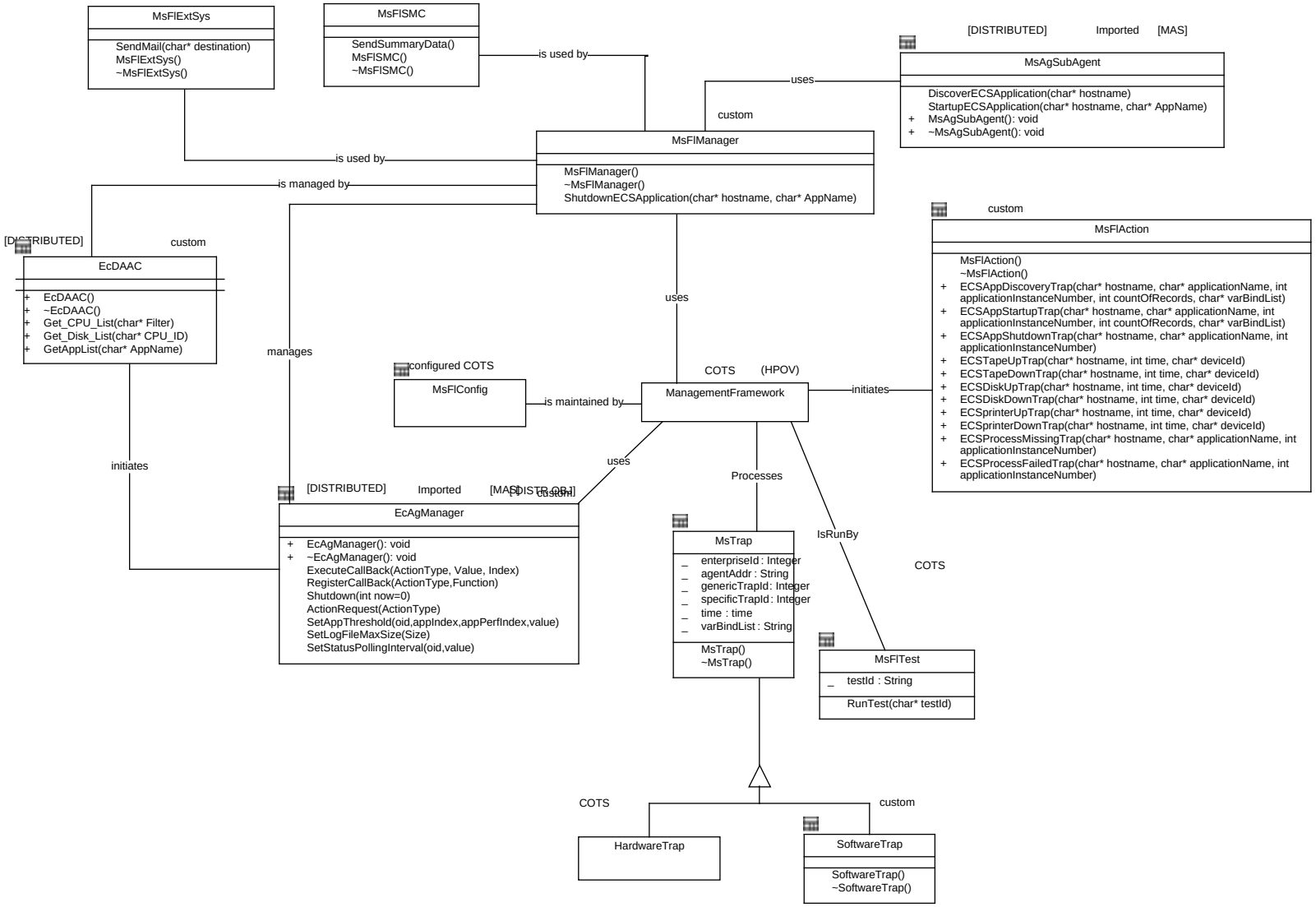


Figure 6.1-2. Fault Management Object Model

6.1.3.1 EcAgManager Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

This class defines a distributed object. It incorporates most of the manager control and instrumentation functionality for ECS applications and COTS. ECS application developers have to use the class library to instrument their code and to provide callback routines so that the ECS applications are manageable by MSS. The EcAgProxy developer has to do the same. The Application MIB provides table format for Fault, Performance, and Configuration parameters to be added by application developers. In order to monitor these parameters, application developers have to provide callback routines for the Subagent to retrieve the value of these parameters. This object provides the capability for the developers to register all the callback functions required. The callback mechanism provided by this object is a generic one. Application developer has to register the callback function first. When the Get request comes in, it will trigger the execution of the callback function to retrieve the value. This object can be invoked by the MsAgDeputy to send Set requests to the MsAgSubagent. The MsAgSubagent can invoke the methods of this object to send Get, Set, or other requests to ECS applications or to the Proxy of COTS. The data structures that application developers have to use are based on the Application MIB definition. The following are examples: Action Type Data Structure Get_Performance typedef struct { char PerfType[n+1]; int PerfValue; int PerfThreshold; } Get_Fault typedef struct { char FaultType[n+1]; int FaultValue; } Get_Config typedef struct { char CfgParam[n+1]; char value[m+1]; } Shutdown typedef struct { int now; }

Attributes:

None

Operations:

ActionRequest - This method passes the action request from the Deputy to the subagent.

Arguments: ActionType

EcAgManager - This method represents the constructor of the class.

Arguments:

Return Type: void

Privilege: Public

ExecuteCallBack - This method executes a callback routine.

Arguments: ActionType, Value, Index

```
PDL:int EcAgManager::ExecuteCallBack ( int Action, void *Value, int RowIndex )
{
  // determine if the application has registered an action for the requested callback
  if ( CallBack function registered for action ) {
```

```
      return_val = Execute CallBack associated with the action passing the value and RowIndex
    if ( return_val == 0 ) {
      // alert caller that now value can be associated with the request
      return NO_VALUE
    }
  }
  else {
    // alert caller that no value can be associated with request
    return NO_VALUE
  }

  return VALUE_FOUND
}
```

RegisterCallBack - This method registers callback routines.
Arguments: ActionType,Function
PDL:int EcAgManager::RegisterCallBack (int Action, ActionFunctionPtr *FncPtr)

```
{
  // make sure the specified action is a legal action.  Legal actions are of
    // the following type ( GET_PERFORMANCE, GET_FAULT, GET_CONFIG, GET_PROCESS,
  // GET_ASSOCIATION, GET_PROGRAM, STARTUP, SHUTDOWN).

  if ( Action in set ( LEGAL_ACTIONS ) ) {
    add FncPtr to callback table for specified action
  }
  else {
    // alert user of illegal action
    return ERROR
  }
}
```

SetAppThreshold - This method sets the threshold of performance metrics.
Arguments: oid,appIndex,appPerfIndex,value

SetLogFileMaxSize - This method sets the maximum size of the MSS log file on a host.
Arguments: Size

SetStatusPollingInterval - This method set the status polling interval on a host.
Arguments: oid,value

Shutdown - This method terminates the application.
Arguments: int now=0
PDL:int EcAgManager::Shutdown (int now = 0)

```
{
  // if now == 1, the application should immediately comply, otherwise the application
  // can comply with the request as soon as convenient.

  // if the user provides a callback for shutdown, then execute it first
  if ( CallBack function registered for SHUTDOWN ) {
    Execute CallBack associated with SHUTDOWN passing in value of now
  }

  // execute OODCE specific version of shutdown call
  OODCE::shutdown ( now )
}
```

~EcAgManager - This method represents the destructor of the class.
Arguments:
Return Type: void
Privilege: Public

Associations:

The EcAgManager class has associations with the following classes:
Class: EcDAAC initiates
Class: MsFlManager manages
Class: ManagementFramework uses

6.1.3.2 EcDAAC Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:

Attributes:

None

Operations:

EcDAAC - This method represents the constructor of the class.
Arguments:
Return Type: Void
Privilege: Public

GetAppList - This method returns a list of application classes (dependent) given an

application class name as input.
Arguments: char* AppName
Return Type: Void
Privilege: Public

Get_CPU_List - This method takes a filter as an argument, and returns a list of hosts matching the filter criteria.
Arguments: char* Filter
Return Type: Void
Privilege: Public

Get_Disk_List - This method returns a list of disks attached to a specified processor. The processor is specified by the argument CPU_ID.
Arguments: char* CPU_ID
Return Type: Void
Privilege: Public

~EcDAAC - This method represents the destructor of the class.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The EcDAAC class has associations with the following classes:
Class: EcAgManager initiates
Class: MsFlManager ismanagedby

6.1.3.3 HardwareTrap Class

Parent Class: MsTrap
Public: No Distributed Object: No
Purpose and Description:
This class represents traps received from hardware devices. There are 5 standard traps defined, in addition to which there are enterprise traps defined by the vendor of the routers and hubs that will be deployed in ECS. These are COTS provided.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The HardwareTrap class has associations with the following classes:
None

6.1.3.4 ManagementFramework Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is HP OpenView Network Node Manager, a COTS product. This product provides the management framework with the underlying management services for the management of SNMP-based network devices. It also provides the necessary integration points and services for the integration of management applications. Since this class is all COTS, it will not be described in detail here. The reader is referred to the documentation set of HP OpenView Network Node Manager for further details on the product.

Attributes:

None

Operations:

None

Associations:

The ManagementFramework class has associations with the following classes:
Class: MsFlTest IsRunBy
Class: MsTrap Processes
Class: MsFlAction initiates
Class: MsFlConfig ismaintainedby
Class: EcAgManager uses
Class: MsFlManager uses

6.1.3.5 MsAgSubAgent Class

Parent Class: Not Applicable
Public: Yes Distributed Object: No
Purpose and Description:
This managed object class supports SNMP MIB extensions. It receives requests from the master agent. Based on Get or Set requests, it performs the retrieval or set functions onto

resource or resource managers using available API. This object will instantiate another object MsAgMonitor to perform local polling on resources on the host.

Attributes:

None

Operations:

DiscoverECSApplication - This method supports the discovery of ECS applications. This discovery may be initiated in order to initially discover all ECS applications, periodic discovery, or on demand such as after a crash of an ECS managed host or the MSS server itself. This discovery process may be initiated on either a specified host, or an all hosts. As each ECS application is discovered, the Management Agent Services issues a discovery trap to the Fault Management Application Service with information on the discovered application included as part of the variable bindings list so that the discovered application may be registered and presented visually on the appropriate submap. The discovery process may be initiated by an operator action at the user interface (customization of HP OpenView NNM user interface).

Arguments: char* hostname

```
PDL:MsAgMonitor::DiscoverECSApplication ( )
{
  // traverse the ECS file system, searching any ECS applications
  for ( each directory in ECS file system ) {

    // if a file exists with the extension .conf, then an application with the root
    // of that filename also exists.
    if ( file exists with extension ".conf" ) {

      // register the application
      create object for application in the Object database
      create submap(s) for the application

      // identify any instances of the process
      for ( each file that matches pattern "ApplicationName.instance*" ) {

        // register the instance of the application
        create object for application instance in the Object database
        create submap(s) for the application instance
      }
    }
  }
}
```

MsAgSubAgent - This method represents the constructor of the object.
Arguments:
Return Type: void
Privilege: Public

StartupECSApplication - This method issues a directive to the Management Agent Services on the specified host to perform the startup of the specified ECS application. Each ECS application is found in a well known place in the standard ECS filesystem (described in the section on Management Agent Services). As the application completes its startup, it issues a startup trap to the Fault Management Application Service so that the Service may update the status of the registered application on the graphical user interface.
Arguments: char* hostname, char* AppName

~MsAgSubAgent - This method represents the destructor of the object.
Arguments:
Return Type: void
Privilege: Public

Associations:

The MsAgSubAgent class has associations with the following classes:
Class: MsFlManager uses

6.1.3.6 MsFlAction Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The class MsFlAction provides the capabilities to process events generated from external stimuli. This processing may include the issuing of notifications to an operator via the Graphical User Interface using the services of HP OpenView NNM, or the launching of automated actions in response to the received notifications, based on the configuration information set up. The methods in this class are executed in response to receiving traps. The traps received and processed are listed below against with the managed objects they correspond to: Network devices (provided by HP OpenView) OV_node_up OV_node_down Communication links (provided by HP OpenView) OV_link_up OV_link_down Interface cards (provided by HP OpenView) OV_IF_up OV_IF_down Tape drives (ECS-specific) ECS_tape_up ECS_tape_down) Disk drives (ECS-specific) ECS_disk_up ECS_disk_down Printers (ECS-specific) ECS_printer_up ECS_printer_down ECS applications (ECS-specific) ECS_application_startu ECS_application_shutdown ECS_application_discovery ECS_application_missing ECS_application_failed

Attributes:

None

Operations:

ECSAppDiscoveryTrap - This method is executed when a discovery trap is received. A discovery trap is received when the agent sends a trap after discovering an application on an ECS managed host. The method does the following: Based on the parameters received, a selection name is constructed using the hostname, application name, and the instance number. The submap for the host is located Case: Instance number = 0 This indicates that the software exists on the host, but there are no instances running An entry is created in the Object database, and a symbol (icon) is created on the submap corresponding to the host, based on the selection name The color of the icon is put into a state to indicate that it is not running Case: Instance number > 0 This indicates the discovery of an instance of the application The Object database is searched for an occurrence of instance If not found, a submap for the new application instance is created For each process in the var bind list: creates an object for each process in the var bind list sets the shutdown UUID for the process assigns the object a selection name based on the naming convention creates a symbol corresponding to the process object put the symbol on the appropriate submap sets the process symbol to a normal state
Arguments: char* hostname, char* applicationName, int applicationInstanceNumber, int countOfRecords, char* varBindList
Return Type: Void
Privilege: Public

ECSAppShutdownTrap - This method, executed in response to receiving a ShutdownTrap, processes the variable bindings of the Trap, updates the attributes of the appropriate managed object in the Object database, uses the methods provided by the ManagementFramework (HPOV) in order to remove the iconic representation of the physical ECS Application on the graphical user interface. This method does the following: A selection name is constructed from the hostname, the application name, and the instance number The Object database is searched to locate the Object and the submap corresponding to this application The symbols corresponding to each component process are located and deleted the objects corresponding to each component process are located and deleted If this is the only instance of the application, the instance number is set to zero, else the object and symbol corresponding to the application are deleted
Arguments: char* hostname, char* applicationName, int applicationInstanceNumber
Return Type: Void
Privilege: Public

ECSAppStartupTrap - This method, executed in response to receiving a StartupTrap, processes the variable bindings of the Trap, updates the attributes of the appropriate managed object in the HPOV Object and Map databases, uses the methods provided by the

HPOV in order to display an iconic representation of the physical ECS Application on the graphical user interface. A startup trap is sent to the HP OpenView when an application completes the startup process. The varbinds included as part of the trap include the application name, application instance, count of the processes info (include the pid, the process name, and the UUID of the shutdown method). This method performs the following: - constructs the selection name for the application instance by concatenating the hostname, application name, and the instance number. - locates the submap for the host specified by the hostname - parses the var bind list for the components of the application - (the var bind list contains sequences of: the process name, process id, and the UUID of the shutdown interface for each process belonging to the application) - for each component process in the var BindList: it creates a selection name creates an object for the process in the Object database stores its selection name, shutdown UUID in the Object database creates a sybmol for the process on the parent application submap sets the state of the symbol to normal
Arguments: char* hostname, char* applicationName, int applicationInstanceNumber, int countOfRecords, char* varBindList
Return Type: Void
Privilege: Public

ECSDiskDownTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, int time, char* deviceId
Return Type: Void
Privilege: Public

ECSDiskUpTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, int time, char* deviceId
Return Type: Void
Privilege: Public

ECSProcessFailedTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, char* applicationName, int applicationInstanceNumber
Return Type: Void
Privilege: Public

ECSProcessMissingTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, char* applicationName, int applicationInstanceNumber
Return Type: Void

Privilege: Public

ECSTapeDownTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, int time, char* deviceId
Return Type: Void
Privilege: Public

ECSTapeUpTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, int time, char* deviceId
Return Type: Void
Privilege: Public

ECSprinterDownTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, int time, char* deviceId
Return Type: Void
Privilege: Public

ECSprinterUpTrap - This method constructs a selection name for the managed object, locates it in the Object database, updates its status, and writes a record for RMA purposes via LogEvent in MsEvent_c
Arguments: char* hostname, int time, char* deviceId
Return Type: Void
Privilege: Public

MsFlAction - This method represents the constructor of the class.
Arguments:

~MsFlAction - This method represents the destructor of the class.
Arguments:

Associations:

The MsFlAction class has associations with the following classes:
Class: ManagementFramework initiates

6.1.3.7 MsFlConfig Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class represents the capability of the ManagementFramework (HPOV NNM) to maintain a mapping between traps (or events) and the Actions to be executed in response to their occurrence. The capability for the definition of the configured information is made available by the ManagementFramework via the graphical user interface to the operator. The reader is referred to the documentation set of HP OpenView Network Node Manager for further details on this capability.

Attributes:

None

Operations:

None

Associations:

The MsFlConfig class has associations with the following classes:

Class: ManagementFramework ismaintainedby

6.1.3.8 MsFlExtSys Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class represents the interface to external systems such as NSI.

Attributes:

None

Operations:

MsFlExtSys - This is the default constructor for this class.

Arguments:

SendMail - This method sends a mail message to the external system as specified by the destination field.

Arguments: char* destination

~MsFlExtSys - This is the destructor for this class.
Arguments:

Associations:

The MsFlExtSys class has associations with the following classes:
Class: MsFlManager isusedby

6.1.3.9 MsFlManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the necessary functionality to perform the real-time configuration management functions of the discovery, startup and shutdown of ECS applications (such as the Science Data Server or the Data Processing Service). It also provides the functionality to dispatch real-time notifications to ECS Applications via the Management Agent Services. These functions are initiated by external stimuli (operator actions) at the user interface, which is provided by HP OpenView NNM.

Attributes:

None

Operations:

MsFlManager - This method represents the constructor of the class.
Arguments:

ShutdownECSApplication - This method creates an instance of EcAgManager (the class exported by the Management Agent Services, and is a proxy to the application) and invokes the Shutdown method. This effects a graceful shutdown of the application. As an application is shut down gracefully, the application emits a shutdown trap to the Fault Management Application Service in response to which the application instance is deregistered.
Arguments: char* hostname, char* AppName

~MsFlManager - This method represents the destructor of the class.
Arguments:

Associations:

The MsFlManager class has associations with the following classes:
Class: EcDAAC ismanagedby
Class: MsFlExtSys isusedby
Class: MsFlSMC isusedby
Class: EcAgManager manages
Class: ManagementFramework uses
Class: MsAgSubAgent uses

6.1.3.10 MsFlSMC Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the interface between the Fault Management Application Service at a site and the Fault Management Application Service at the SMC.

Attributes:

None

Operations:

MsFlSMC - This is the default constructor for this class.
Arguments:

SendSummaryData - This method sends summary data from the site to the SMC.
Arguments:

~MsFlSMC - This is the destructor for this class.
Arguments:

Associations:

The MsFlSMC class has associations with the following classes:
Class: MsFlManager isusedby

6.1.3.11 MsFlTest Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents Diagnostic Tests, as available from vendors.

Attributes:

testId - This identifies the test to be executed.
Data Type: String
Privilege: Private
Default Value:

Operations:

RunTest - This method runs the specified test.
Arguments: char* testId

Associations:

The MsFlTest class has associations with the following classes:
Class: ManagementFramework IsRunBy

6.1.3.12 MsTrap Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
MsTrap represent changes in the state of managed objects (COTS and custom). These changes may be acceptable changes in state (normal events), or they may represent unacceptable changes in state (faults) of managed objects. Traps are generated by Management Agent Services and are received by the ManagementFramework (HPOV NNM), which determines the appropriate MsFlAction to be executed based on the configuration information represented by MsFlConfig. Traps for COTS products are defined by the vendors of the COTS products (such as routers), whereas Traps unique to ECS (for ECS Applications) are defined in the ECS Application MIB. These include special Traps such as DiscoveryTraps, StartupTraps and ShutdownTraps. These are discussed in the section on the MsFlAction class.

Attributes:

agentAddr - This attribute specifies the IP address of the managed object where the trap originated.
Data Type: String
Privilege: Private
Default Value:

enterpriseId - This attribute specifies the ECS enterprise identification ID. This is represented in dot notation.
Data Type: Integer
Privilege: Private
Default Value:

genericTrapId - This attribute specifies the generic trap id (0-4 for standard traps, 6 for enterprise specific traps)
Data Type: Integer
Privilege: Private
Default Value:

specificTrapId - This attribute specifies the specific Id of the enterprise-specific traps (discussed in the MsFlAction Class)
Data Type: Integer
Privilege: Private
Default Value:

time - his attribute specifies the time, in seconds, since a reference data in the past at the managed object when the trap was generated.
Data Type: time
Privilege: Private
Default Value:

varBindList - This attribute specifies a list of the variable bindings sent with the trap.
Data Type: String
Privilege: Private
Default Value:

Operations:

MsTrap - This is the default constructor for this class.
Arguments:

~MsTrap - This is the destructor for this class.
Arguments:

Associations:

The MsTrap class has associations with the following classes:
Class: ManagementFramework Processes

6.1.3.13 SoftwareTrap Class

Parent Class: MsTrap
Public: No Distributed Object: No
Purpose and Description:

This class represents traps generated by the Management Agent Services for faults events detected in ECS applications, or faults reported by ECS applications.

Attributes:

All Attributes inherited from parent class

Operations:

SoftwareTrap - This method represents the constructor of the class.
Arguments:

~SoftwareTrap - This method represents the destructor of the class.
Arguments:

Associations:

The SoftwareTrap class has associations with the following classes:
None

6.1.4 Fault Management Dynamic Model

6.1.4.1 Fault Notification by an ECS Application

This scenario traces the events associated with the Data Server reporting a fault as a result of calling another application. As a result of receiving this fault report, a notification is then sent to the Ingest Server which is dependent on the Data Server for its operations. The scenario is depicted in Figure 6.1-3.

6.1.4.1.1 Beginning Assumptions

None.

6.1.4.1.2 Interfaces with Other Subsystems and Segments

Management Agent Services

6.1.4.1.3 Stimulus

An ECS application (the Data Server) generates a Fault notification as the result of a call to another application which ends with a fatal error status.

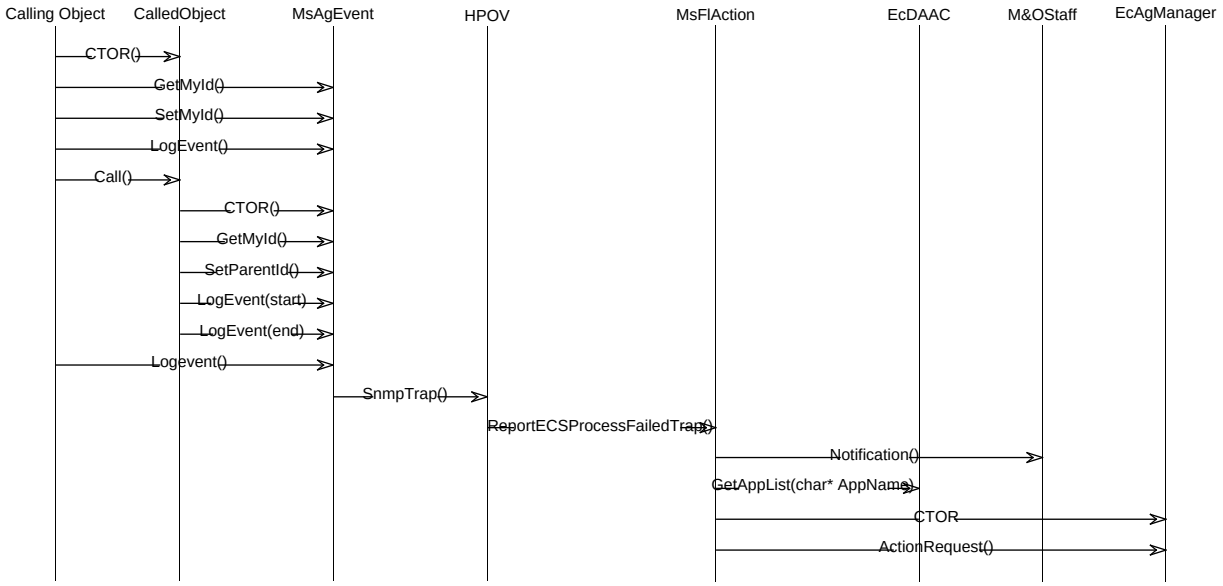


Figure 6.1-3. Fault Notification by an ECS Application Event Trace

6.1.4.1.4 Participating Classes From the Object Model

CallingObject (An ECS Application - Data Server)

CalledObject (Another ECS Application that the Ingest Server calls)

EcAgEvent

HPOV (HP OpenView Network Node Manager)

MsFlAction

EcDAAC

M&O Staff

EcAgManager

6.1.4.1.5 Beginning System, Segment and Subsystem State(s)

The configuration is set up to execute a specified action based upon the Ingest Server reporting the specific type of fault used in this scenario. This action locates the Ingest Server and sends it a notification of the fault since the Ingest Server depends on the well-being of the Data Server.

6.1.4.1.6 Ending State

The M&O Staff are notified via the icons corresponding to the Data Server changing color, and the Ingest Server is sent a notification of the fault. The state of the object corresponding to the Data Server has changed to critical as a result of the fault.

6.1.4.1.7 Scenario Description

This description describes the accompanying event trace. Some of the objects on the event trace are included for purposes of The Data Server (CallingObject) calls another application (CalledObject). The call returns and reports a fatal error. The Data Server instantiates EcAgEvent, sets the attributes and logs the event via LogEvent. EcAgEvent logs the event to the MSS History Logfile, and generates a fault notification via an SNMP trap. HP OpenView, upon receipt of the trap, based on configuration information established, invokes the appropriate method MsFlAction. This method retrieves the Application instance (EcAgManager - the managed object proxy of the Ingest Server) and changes its state. This causes a notification to be sent to the M&O Staff. The MsFlAction method then invokes method GetAppList on EcDAAC, which retrieves the list of its dependents. From this list, it determines which application it needs to send a notification to (Ingest Server in this case), instantiates a proxy to the Ingest Server application (EcAgManager - a public class exported by the Management Agent Services), and invokes ActionRequest on the proxy. This method sends the notification to the Ingest Server application.

6.1.5 Fault Management Structure

Table 6.1-2 lists the components of the Fault Management Application Service.

Table 6.1-2. Fault Management Components

Component Name	COTS/Custom
MsFlManager	Custom (C++ code)
HPOV	COTS
MsFlConfig	Configuration of COTS (HPOV)
MsFlAction	Custom (C++ code and scripts)
MsManager	C++ code imported from Management Agent Services
EcDAAC	External C++ Class category
MsFlTest	Vendor-provided diagnostic tests

6.1.5.1 Management Framework CSC

Purpose and Description

The Management Framework provides the framework for Enterprise Management (network and systems management). It provides the integration points for management applications. In order to provide network and system management solutions using the framework, some amount of customization and configuration is necessary. This will involve the loading of MIBs, discovering and customizing the user interface (the visual displays), the association of faults with automated actions, where necessary.

Mapping to objects implemented by this component

ManagementFramework

MsFlConfig

Candidate products

HP OpenView Network Node Manager

6.1.5.2 Diagnostic Tests CSC

Purpose and Description

These diagnostic tests, meant for fault isolation and diagnosis will be tests as provided by vendors.

Mapping to objects implemented by this component

MsFlTest

6.1.5.3 Application Management CSC

Purpose and Description

The Application Management component of real-time configuration management has to do with the discovery, startup or shutdown of ECS applications.

Mapping to objects implemented by this component

MsFlManager

DiscoverECSApplications - C++ code

StartupECS Applications - C++ code

ShutdownECSApplications - C++ code

6.1.5.4 Automatic Actions CSC

Purpose and Description

Automatic actions are actions that are initiated in response to a well known event. These well known event include the discovery of an ECS Application, the notification of an application starting up or shutting down gracefully.

Mapping to objects implemented by this component

MsFlAction

ECSAppDiscoveryTrap - C++ code

ECSAppStartupTrap - C++ code

ECSAppShutdownTrap - C++ code

Scripts:

ECSTapeUpTrap

ECSTapeDownTrap

ECSDiskUpTrap

ECSDiskDownTrap

ECSPrinterUpTrap

ECSPrinterDownTrap

ECSProcessMissingTrap

ECSProcessFailedTrap

6.1.5.5 Resource Class Category CSC

Purpose and Description

The Resource Class Category encapsulates the proxy objects that the Management Framework maintains. This class category provides services such as the status of a processor, or the status of its associated disks.

Mapping to objects implemented by this component

EcDAAC

GetCPUList - C++ code

GetDiskList(CPUID) - C++ code

6.1.6 Fault Management Management and Operation

6.1.6.1 System Management Strategy

The Fault Management Application Service is based on HP OpenView NNM, which generates notifications when it detects partial failures of its components. Components of HP OpenView may be individually restarted. In the case of a total failure, it may be restarted. All error messages are logged to the local log file. In the case of a hardware failure of the MSS server, a hot standby with dual-attached disks for a quick failover will be provided.

6.1.6.2 Operator Interfaces

The Operator Interface to Fault Management is the graphical user interface provided by HP OpenView Network Node Manager.

6.1.6.3 Reports

The following predefined Fault Management reports will be provided:

FDDI Interface utilization report - graphical depiction of the utilization of an FDDI interface on operator-selected interface.

Ethernet Errors report -- graphical depiction of real-time ethernet errors on operator-selected node(s).

SNMP Protocol Errors report - graphical depiction of real-time SNMP errors on operator-selected node(s).

SNMP Authentication Failures report - tabular list of management systems that have caused an SNMP authentication failure on the selected node(s).

SNMP Events Log - tabular listing of all SNMP events reported to HP OpenView for the selected node(s).

Site Host Errors report - tabular listing of the count of various host errors over an operator-specified period of time for each host at the site.

EMC Host Errors report - tabular listing of the count of various host errors over an operator-specified period of time for each site.

Other fault management statistics will be reportable via ad hoc reports provided by HP OpenView and the report generation capability associated with the management RDBMS.

Further information on fault management reports is available in the Release A Overview Design Specification (305-CD-004-001).

6.2 Performance Management

6.2.1 Performance Management Overview

The Performance Management Application Service provides the capability to continuously gather statistical and historical data on the operational states of applications, operating system resources and network components, to analyze the data collected by comparing with established criteria, adjust measurement criteria or initiate other corrective actions as necessary in order to ensure an optimal utilization of resources. The service allows for the benchmarking and trends analysis of network component performance, in addition to collecting performance data on scientific algorithms. The Performance Management Application Service has two instances: one at each of the DAACs and one at the SMC. The site Performance Management Application Service collects and processes performance data local to the site.

Site performance management data is periodically summarized and sent to the SMC for analysis by the SMC Performance Management Application. The SMC Performance Management Application Service, which has capabilities similar to those of the site Performance Application Services, operates on performance data collected system-wide by the various site Performance Management Application Services in order to evaluate system-level performance and system-wide trends. In addition, the SMC Performance Management Application Service is also capable of connecting directly to each of the DAACs as required to monitor the performance of site elements.

For Release A, the Performance Management Application Service will consist mainly of two COTS applications, HP OpenView and a separate performance management COTS application (hereafter referred to as the COTS Performance Application). Table 6.2-1 provides an indication of responsibility between HP OpenView and the COTS Performance Application for providing performance management of the various ECS managed objects.

Table 6.2-1. Performance Manager by Managed Object Table

Managed Object	Performance Manager
Hosts	HP OpenView
Routers	HP OpenView
Hubs	HP OpenView
Gateways	HP OpenView
FDDI links	HP OpenView
Ethernet links	HP OpenView
ECS Applications	COTS Performance Application
Operating systems	COTS Performance Application
File systems	COTS Performance Application

HP OpenView provides operators to specify, for each managed object, the following information:

- Performance attributes to be collected. Management information bases (MIBs) are used to define attributes that can be collected from various managed objects. Each performance attribute that can be measured has an associated object identifier (oid) specified in a MIB.

HP OpenView collects data for each oid that has been specified for a particular managed object.

- Frequency of performance attribute data collection. This value can be set differently for each attribute associated with a managed object.
- Threshold(s) which indicate degraded performance condition(s) (one or more can be set for each oid on each managed object). For each threshold set, a corresponding rearm value can also be set. Once the threshold is exceeded, the performance attribute must then fall below the rearm value before the performance degradation is cleared. This prevents the generation of multiple degradation alerts in the case where the performance attribute value is fluctuating around the threshold value.
- Performance attributes to be logged. HP OpenView logs only that data specified by the operator. For each oid on each managed object, performance management can take one of three forms:
  1. attribute not monitored
  2. attribute monitored but not logged
  3. attribute monitored and logged

HP OpenView can monitor any performance management attributes that are included in MIBs supported by ECS management agents. The following tables, Table 6.2-2 through 6.2-4, provide a representative sample of the performance attributes that are monitored using HP OpenView for different types of managed objects. These attributes are not inclusive. The operator always has the capability to collect information on additional supported attributes or to stop collecting information on listed attributes.

Table 6.2-2. Host Performance Metrics Table

Attribute type	Attribute Description
Interfaces	Status of each interface
	No. of octets received on each interface
	No. of octets sent out on each interface
SNMP	No. of SNMP messages received
	No. of SNMP messages sent
Devices	Status of each host device (unknown, running, warning, testing, or down)
Processor	Avg. percentage of time over the last minute that the processor was not idle
SW Run	No. of total centi-seconds of CPU allocated to each running process
	Total amount of real system memory allocated to each running process

Table 6.2-3. Router Performance Metrics Table

Attribute type	Attribute Description
Interfaces	Status of each interface
	No. of octets received on each interface
	No. of octets sent out on each interface
SNMP	No. of SNMP messages received
	No. of SNMP messages sent
Local system group	CPU busy percentage in the last 5 second period
	Average CPU busy percentage over the last minute
Local interface group	Five minute average of input bits per second on each interface
	Five minute average of output bits per second on each interface

Table 6.2-4. Hub Performance Metrics Table

Attribute type	Attribute Description
Interfaces	Status of each interface
	No. of octets received on each interface
	No. of octets sent out on each interface
SNMP	No. of SNMP messages received
	No. of SNMP messages sent

The COTS Performance Application provides similar monitoring capabilities for applications and operating systems. The following tables, Table 6.2-5 through 6.2-9, provide a representative sample of the performance attributes that will be monitored using the COTS performance application. These attributes are not inclusive. The operator will have the capability to collect information on additional supported attributes or to stop collecting information on listed attributes.

Table 6.2-5. Global System Performance Metrics Table (1 of 2)

Attribute type	Attribute Description
Summary metrics	CPU use during interval (percentage of total and seconds
	Number and rate of physical disk I/Os
	Maximum percent full of all disk filesets
CPU metrics	System CPU use during interval (percent of total and seconds)
	User CPU use during interval (percent of total and seconds)
	CPU idle time during interval (percent of total and seconds)
	Rate of system procedure calls during interval
Disk metrics	Number of disk drives configured on the system
	Average utilization of busiest disk during interval
	Number and rate of physical disk reads during interval
	Number and rate of physical disk writes during interval
	Number and rate of physical disk transfers during interval
	Number and rate of disk reads by file system during interval

Table 6.2-5. Global System Performance Metrics Table (2 of 2)

Attribute type	Attribute Description
	Number and rate of disk writes by file system during interval
	Number and rate of disk reads for memory management during interval
	Number and rate of disk writes for memory management during interval
Networking	Number of configured LAN interfaces
	Number and rate of network file system requests during interval
	Rate of LAN errors per minute
	Rate of LAN collisions per minute
Memory use	Main memory use (percent of total)
	Swap space use on disk (percentage of total)
	Number and rate memory page faults during interval
	Number of process swaps during interval
	Percent of virtual memory currently in active use
Process queue depths (load factors)	Number of processes in run queue during interval
	Number of processes waiting for disk during interval
	Number of processes waiting for memory during interval
	Number of processes currently in sleep state during interval
	Number of processes waiting for some other reason during interval
User/process metrics	Number of user sessions during interval
	Number of processes alive during interval
	Number of processes active during interval
	Number of processes started during interval
	Number of processes that completed during interval
	Average run time of completing process during interval (in secs)
LAN metrics (per LAN intfc)	Number and rate of arriving LAN packets during interval
	Number and rate of outbound LAN packets during interval
	Number and rate of LAN errors during interval
	Number and rate of LAN collisions during interval
Individual disk drive metrics	Number and rate of file system reads during interval
	Number and rate of file system writes during interval
	Disk utilization during interval (percent of total)

Table 6.2-6. Application Performance Metrics Table

Attribute type	Attribute Description
Summary metrics	Application's CPU use during interval (percent of total and in secs)
	Number and rate of physical disk transfers during interval
Process count metrics	Average number of processes in application
	Average number of active processes in application
	Number of application processes that completed during interval
	Run time of completing application processes (in secs)
	Average process priority in application
	Standard deviation of process priorities
CPU metrics	CPU use for user processes during interval (percent of total and secs)
	CPU use for system processing (percent of total and secs)
Disk metrics	Number and rate of logical disk reads during interval
	Number and rate of logical disk writes during interval
	Number and rate of physical disk reads during interval
	Number and rate of physical disk writes during interval
Memory metrics	Main memory use (percent of total)
	Swap space use on disk (percent of total)

Table 6.2-7. Process Performance Metrics Table (1 of 2)

Attribute type	Attribute Description
Process identification metrics	Process identification number
	Application number
	Program name
	Logon user name
	Logon device name or number
	Parent and group identification numbers
	Execution priority/scheduling queue
	Last reason for stopping execution
Summary metrics	CPU use during interval (percent of total and secs)
	Number and rate of physical disk transfers during interval
	Total time process ran
CPU metrics	CPU use for system processing (percent of total and secs)
	CPU use for user processing (percent of total and secs)
Disk metrics	Number and rate of logical disk reads during interval
	Number and rate of logical disk writes during interval
Memory metrics	Resident set size (Kbytes)
	Size of test+data+stack memory (Kbytes)
	Number of page faults to memory
	Number of page faults to disk

Table 6.2-7. Process Performance Metrics Table (2 of 2)

Attribute type	Attribute Description
Overall process lifetime metrics	Number and rate of logical disk transfers by process
	Number of terminal transactions by process
	Average terminal think time overall (in secs)
	Average terminal response to prompt time overall (in secs)
	Number of user transactions by process
	Average user think time overall (in secs)

Table 6.2-8. Disk Performance Metrics Table

Attribute type	Attribute Description
Disk metrics	Number of disk devices configured
	Disk utilization during interval
	Rate and number of file system reads
	Rate and number of file system writes

Table 6.2-9. Disk Performance Metrics Table

Attribute type	Attribute Description
System configuration metrics	Operating system version
	Number of processors in the system
	Number of disk devices and their device IDs
	Number of LAN devices
	Main memory size (total and available to users)
	Swapping space allocated

In addition, the performance management application service must also monitor ECS-specific metrics for ECS applications. This information will be collected by management agent services and stored in a history log. The operator will be given an interface to the management agent for performance management reasons to set polling intervals and application thresholds. The operator can access this ECS-specific data by browsing the logs or by generating reports from the management RDBMS. A sample of the ECS-specific performance metrics is provided in Table 6.2-10. These attributes are not inclusive. The operator will have the capability to collect information on additional supported attributes or to stop collecting information on listed attributes.

Table 6.2-10. ECS-Specific Performance Metrics Table

Attribute Type	Attribute Description
Processing performance	CPU utilization for each PGE
	memory utilization for each PGE
	disk space utilization for each PGE
Storage management performance	total staged data volume (in GB)

6.2.2 Performance Management Context

The performance management context diagram is shown in Figure 6.2-1. The Performance Management Application Service has interfaces with the following other services:

- Management Agent Services - The Performance Management Application Service has two basic interfaces with the Management Agent Services. The first allows the performance management application to query the management agent for performance data on a managed object and receive the data from the agent. The second interface allows the agent to monitor and control the COTS Performance Application. This allows the Fault Management Application Service to monitor the COTS Performance Application and send it startup and shutdown commands.
- Management RDBMS - The Performance Management Application Service has an interface with the management RDBMS to generate performance reports.
- External Systems - The Performance Management Application Service has an interface with external systems to exchange performance data. This data will be sent as processed data via an e-mail interface.
- Fault Management Application Service - The Performance Management Application Service has an interface with the Fault Management Application Service to send event notifications whenever thresholds are exceeded.
- SMC Performance Management Application Service - The site Performance Management Application Service has an interface with the SMC Performance Management Application service to provide site performance information in several ways. First, the site will periodically generate summary reports on a prearranged basis. These reports will be in the form of database records that can be combined with other site reports to form an ECS-wide summary of performance. Second, the SMC can send a request for a site performance data. This can be in the form of a log file or a report. If it is a log file, the site will simply send a copy of the requested log file. If it is a report, the site will then generate the report and send it to the SMC, again in database format. Third, the SMC can remotely log onto the site HP OpenView application to gather specific real-time performance information.

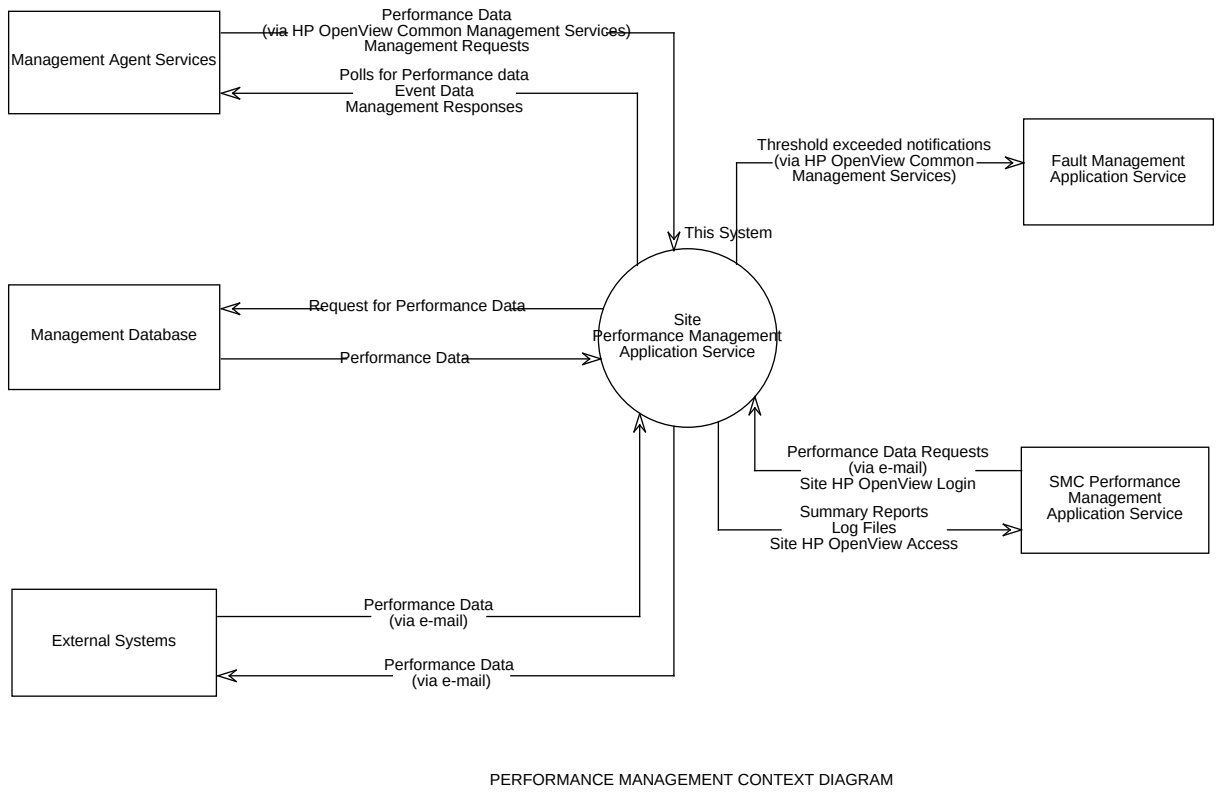


Figure 6.2-1. Performance Management Context Diagram

6.2.3 Performance Management Object Model

The performance management object model is shown in Figure 6.2-2.

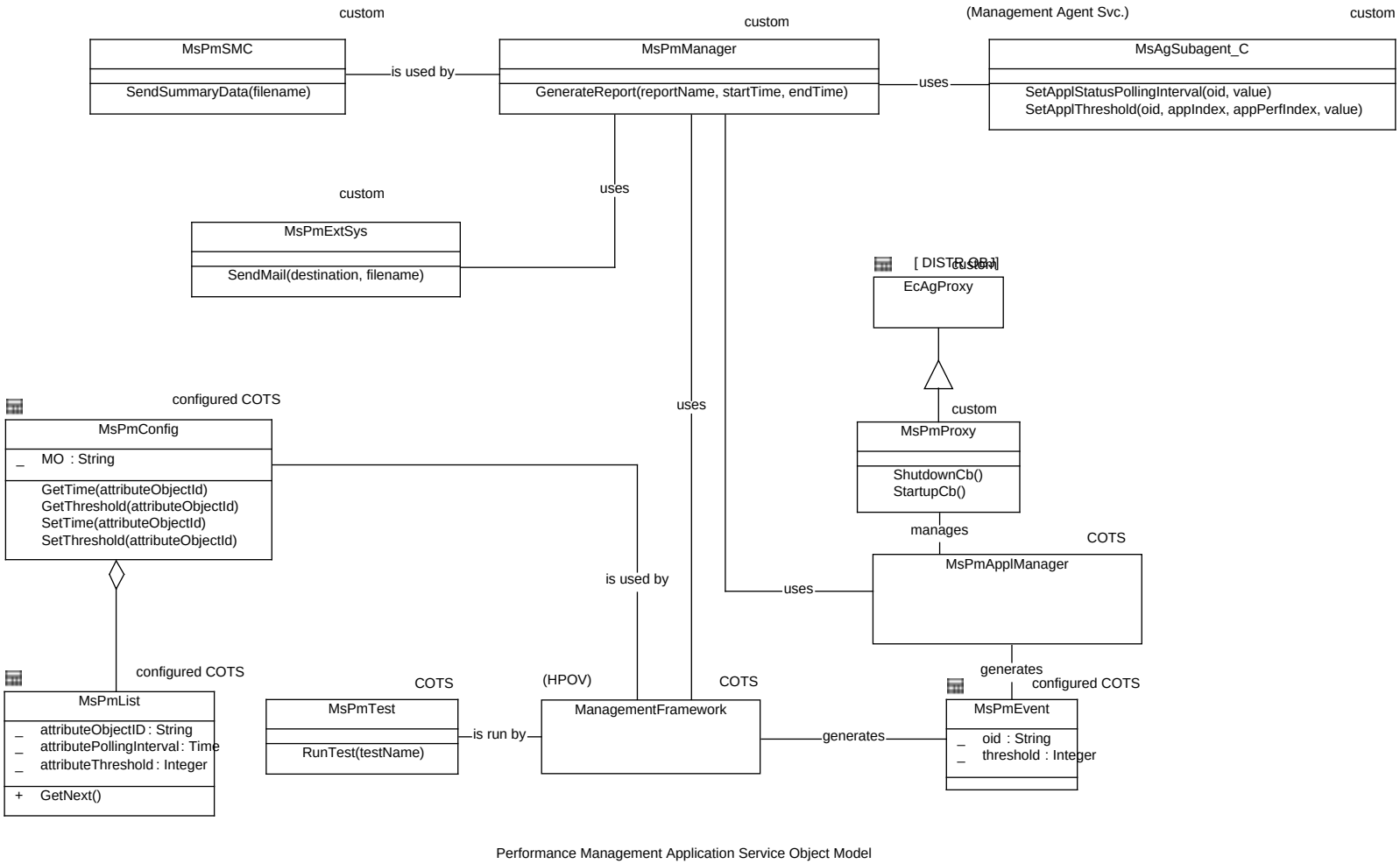
6.2.3.1 EcAgProxy Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

This object class is primarily for COTS' manageability. It includes the MSS instrumentation class library to enable the manageability of the COTS product. The front-end of this object is the MSS instrumentation code. The back-end of it is the interface to the COTS. It is unique to every COTS. In security management, the logs of COTS are monitored by this object. If an security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.



Performance Management Application Service Object Model

Figure 6.2-2. Performance Management Object Model

Attributes:

None

Operations:

None

Associations:

The EcAgProxy class has associations with the following classes:
None

6.2.3.2 ManagementFramework Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is HP OpenView Network Node Manager, a COTS product. This product provides the management framework with the underlying management services for the management of SNMP-based network devices. It also provides the necessary integration points and services for the integration of management applications. Since this class is all COTS, it will not be described in detail here. The reader is referred to the documentation set of HP OpenView Network Node Manager for further details on the product.

Attributes:

None

Operations:

None

Associations:

The ManagementFramework class has associations with the following classes:
Class: MsPmEvent generates
Class: MsPmTest isrunby
Class: MsPmConfig isusedby
Class: MsPmManager uses

6.2.3.3 MsAgSubagent_C Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class is imported from Management Agent Services. It provides the capability for the MsPmManager to configure the collection of application performance data by the management agent located on the application host platform. This class is implemented via C++ code.

Attributes:

None

Operations:

SetApplStatusPollingInterval - This method configures the polling interval for the identified management agent that is monitoring applications on a host. This specifies the time period over which the management agent will poll all applications currently running on its host for performance data.
Arguments: oid, value

SetApplThreshold - This method specifies the threshold value (value) against which the identified performance parameter (specified by appPerfIndex) is to be compared for the application (specified by appIndex).
Arguments: oid, appIndex, appPerfIndex, value

Associations:

The MsAgSubagent_C class has associations with the following classes:

Class: MsPmManager uses

6.2.3.4 MsPmApplManager Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class is a COTS product that monitors ECS system performance. It provides the capability to monitor performance parameters covering areas such as memory utilization, CPU utilization, disk i/o, and queue lengths. Since this class is all COTS, it will not be described in detail here. Additional information is available in the MsPmApplManager documentation.

Attributes:

None

Operations:

None

Associations:

The MsPmApplManager class has associations with the following classes:
Class: MsPmEvent generates
Class: MsPmProxy manages
Class: MsPmManager uses

6.2.3.5 MsPmConfig Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides configuration information to the Management Framework. It is used by the ManagementFramework to store thresholds and performance measurement intervals for performance metrics. This class is implemented by configuring the ManagementFramework COTS package.

Attributes:

MO - This is a string that identifies the managed object.
Data Type: String
Privilege: Private
Default Value:

Operations:

GetThreshold - This method retrieves the threshold value(s) that have been set for the specified attribute for this managed object. This method is implemented by HP OpenView.
Arguments: attributeObjectId

GetTime - This method retrieves the time interval at which the specified attribute is to be polled for this managed object. This method is provided by HP OpenView.
Arguments: attributeObjectId

SetThreshold - This method sets the threshold value(s) for the specified attribute of this managed object. Whenever the ManagementFramework retrieves managed object attribute values, it will compare those values against these thresholds to determine whether a performance degradation has occurred. This method is implemented by HP OpenView to allow operator input of threshold value(s).
Arguments: attributeObjectId

SetTime - This method sets the time interval at which the ManagementFramework will poll this managed object to obtain the value of the specified attribute. The method is provided by HP OpenView to allow the operator to set or modify the time interval.
Arguments: attributeObjectId

Associations:

The MsPmConfig class has associations with the following classes:
Class: ManagementFramework isusedby

6.2.3.6 MsPmEvent Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class defines the event that is generated by the ManagementFramework or the MsPmApplManager whenever a measured attribute value exceeds a configured threshold. The generated event is forwarded to the fault management element of the ManagementFramework. This class is implemented by configuring the ManagementFramework and MsPmApplManager COTS packages.

Attributes:

oid - This attribute specifies the object identification of the attribute for which a threshold value has been exceeded.
Data Type: String
Privilege: Private
Default Value:

threshold - This attribute specifies an integer representing the threshold level (severity) that has been exceeded.
Data Type: Integer
Privilege: Private
Default Value:

Operations:

None

Associations:

The MsPmEvent class has associations with the following classes:
Class: ManagementFramework generates
Class: MsPmApplManager generates

6.2.3.7 MsPmExtSys Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the interface for the MsPmManager to send performance management information, reports, and notifications to external systems via e-mail.

Attributes:

None

Operations:

SendMail - This operation generates an electronic mail message to send the performance information stored in filename to the specified destination.
Arguments: destination, filename

Associations:

The MsPmExtSys class has associations with the following classes:
Class: MsPmManager uses

6.2.3.8 MsPmList Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class stores configuration information. It is implemented by configuring the ManagementFramework COTS package.

Attributes:

attributeObjectID - This represents a specific performance parameter as a sequence of integers to correspond to the location of the parameter in the MIB tree structure.
Data Type: String
Privilege: Private
Default Value:

attributePollingInterval - This attribute specifies the time interval with which the ManagementFramework should poll the managed object for this attribute.
Data Type: Time
Privilege: Private
Default Value:

attributeThreshold - This attribute specifies the value which, if exceeded, should result in an alert being generated.
Data Type: Integer
Privilege: Private
Default Value:

Operations:

GetNext - This operation gets the next attribute in the list for the same MO.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsPmList class has associations with the following classes:
MsPmConfig (Aggregation)

6.2.3.9 MsPmManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the capability for generating and sending performance management reports to external systems and the SMC. This class is implemented via scripts.

Attributes:

None

Operations:

GenerateReport - This method generates the specified predefined report from the management database from data collected on the time periods indicated.
Arguments: reportName, startTime, endTime

Associations:

The MsPmManager class has associations with the following classes:
Class: MsPmSMC isusedby
Class: ManagementFramework uses
Class: MsAgSubagent_C uses
Class: MsPmApplManager uses
Class: MsPmExtSys uses

6.2.3.10 MsPmProxy Class

Parent Class: EcAgProxy
Public: No Distributed Object: No
Purpose and Description:
This class is a specialization of the EcAgProxy class. It provides the capability for the monitoring and management of MsPmApplManager. This class is implemented by customizing C++ code developed under management agent services.

Attributes:

All Attributes inherited from parent class

Operations:

ShutdownCb - This method shuts down the performance management COTS product by executing a vendor-provided shutdown script. It is a specialization of the class provided by management agent services.
Arguments:

StartupCb - This method starts the performance management COTS product using a vendor-provided script. It is a specialization of the class provided by management agent services.
Arguments:

Associations:

The MsPmProxy class has associations with the following classes:
Class: MsPmApplManager manages

6.2.3.11 MsPmSMC Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class sends summary performance data to the SMC via e-mail. This data is in the form of a standard summary report generated by the MsPmManager from information logged in the management database.

Attributes:

None

Operations:

SendSummaryData - This operation sends the specified file to the SMC.

Arguments: filename

Associations:

The MsPmSMC class has associations with the following classes:

Class: MsPmManager isusedby

6.2.3.12 MsPmTest Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class represents the tests that may be run to gather information on the performance of managed objects. These tests are COTS products. This class is implemented by the ManagementFramework COTS package.

Attributes:

None

Operations:

RunTest - This method runs the specified performance test. For Release A, these tests are as provided by COTS vendors.

Arguments: testName

Associations:

The MsPmTest class has associations with the following classes:

Class: ManagementFramework isrunby

6.2.4 Performance Management Dynamic Model

6.2.4.1 Degradation of Performance Alert

The degradation of performance alert scenario is depicted in Figure 6.2-3.

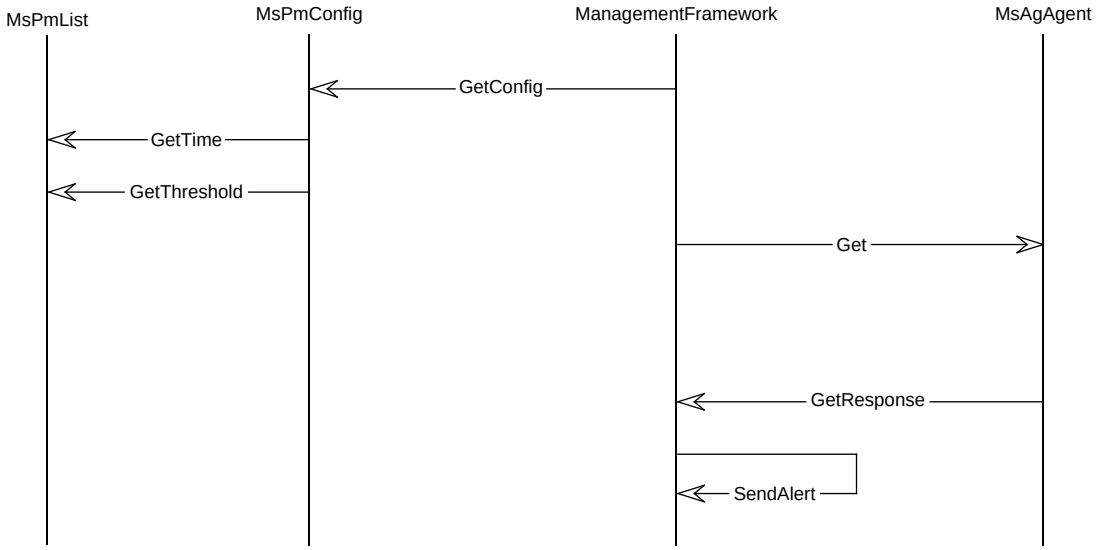


Figure 6.2-3. Degradation of Performance Alert

6.2.4.1.1 Beginning Assumptions

Performance thresholds have been stored for system managed objects.

6.2.4.1.2 Interfaces with Other Subsystems and Segments

An event is reported to the fault management element of the ManagementFramework to signal that the measured attribute has exceeded the set threshold.

6.2.4.1.3 Stimulus

A measured managed object attribute exceeds the set threshold.

6.2.4.1.4 Participating Classes From the Object Model

ManagementFramework

MsPmConfig

MsPmList

6.2.4.1.5 Beginning System, Segment and Subsystem State(s)

The performance management system is in the normal operational state.

6.2.4.1.6 Ending State

A performance degradation event is sent to the fault management element of the ManagementFramework and the performance management system is in the normal operational state.

6.2.4.1.7 Scenario Description

The ManagementFramework sends a GetConfig command to MsPmConfig for a specified managed object. For the attributes which are to be measured for the specified managed object, MsPmConfig sends commands to MsPmList to retrieve the time intervals at which the specified managed object is to be polled for the performance attributes, and the threshold values for which, if the values are exceeded, the ManagementFramework shall send an alert to the fault management component of the ManagementFramework. The time interval and threshold information is then sent to the ManagementFramework. When the time interval has elapsed, the ManagementFramework sends a Get command to the management agent responsible for managing the managed object. The Agent sends an AgentRequestResponse to the MsManager_S to extract the data from the managed object. The MsManager_S provides the Agent with the attribute value in a GetResponse, which the Agent then forwards to the ManagementFramework. The ManagementFramework compares the attribute value against the threshold value that was obtained from MsPmConfig. The ManagementFramework identifies that a threshold value has been exceeded, the constructs an alert to be sent to the fault management element of the ManagementFramework, sets the appropriate parameters to identify the cause of the fault (managed object name, attributeObjectId, measured attribute value), and sends the fault to the Fault management element.

6.2.4.2 Providing Performance Summary to SMC

In this scenario, the performance manager generates a daily summary of the site performance and forwards the summary to the SMC and to an external user. The providing performance summary to SMC scenario is shown in Figure 6.2-4.

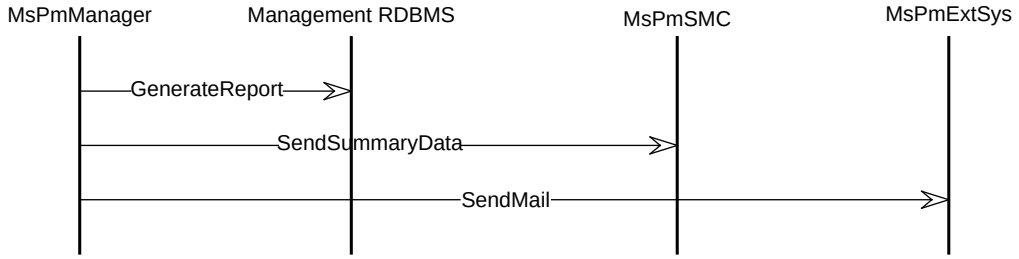


Figure 6.2-4. Providing Performance Summary to SMC

6.2.4.2.1 Beginning Assumptions

None.

6.2.4.2.2 Interfaces with Other Subsystems and Segments

Interface with management data access service to retrieve performance data. Interface with the SMC performance management component. Interface with an external system.

6.2.4.2.3 Stimulus

A daily site performance report is required to be delivered to the SMC and to an external system.

6.2.4.2.4 Participating Classes From the Object Model

MsPmExtSys

MsPmManager

MsPmSMC

6.2.4.2.5 Beginning System, Segment and Subsystem State(s)

The system is operating in the normal operational mode.

6.2.4.2.6 Ending State

The system is operating in the normal operational mode with the performance data sent to the SMC and the external system.

6.2.4.2.7 Scenario Description

The MsPmManager contains a script that uses MDA to create a standard daily report. A GenerateReport command is issued from the MsPmManager, specifying the name of the standard daily report script. The script extracts the specified data from MDA and arranges it in the report format. The report can then be saved to a file by the MsPmManager. The MsPmManager then issues a SendSummaryData command, specifying the name of the recently saved report file, causing the summary data report to be sent electronically to the SMC. The MsPmManager then issues a SendMail command, specifying the electronic address of the external system and the file name of the recently saved report file. This causes the summary data report to be sent to the external system.

6.2.5 Performance Management Structure

Table 6.2-11 identifies the components of the Performance Management Application Service.

Table 6.2-11. Performance Management Components

Element	Implementation (COTS/Custom)
ManagementFramework	COTS
MsPmApplManager	COTS
MsPmConfig	Configuration
MsPmEvent	Configuration
MsPmExtSys	Script
MsPmList	Configuration
MsPmManager	Script
MsPmSMC	Custom
MsPmTest	COTS
MsPmCallbacks	Custom
MsPmProxy	Custom

6.2.5.1 Performance Manager CSC

Purpose and Description

This CSC provides the basis for Enterprise Management (network and systems management). It provides the integration points for management applications. In order to provide network and system management solutions using the framework, some amount of customization and configuration is necessary. This will involve the loading of MIBs, discovering and customizing the user interface (the visual displays), the setting of thresholds and polling intervals for managed object attributes, where necessary.

Mapping to objects implemented by this component

ManagementFramework

MsPmApplManager

MsPmConfig

MsPmEvent

MsPmList

6.2.5.2 Report Generation and Distribution CSC

Purpose and Description

The application performance manager provides the performance data gathering and analysis functions.

Mapping to objects implemented by this component

ManagementFramework

MsPmExtSys

SendMail

MsPmManager

GenerateReport - scripts

GenerateAdHocReport

MsPmSMC

SendSummaryData

6.2.5.3 Performance Test CSC

Purpose and Description

The Test CSC provides the capability for benchmark testing of ECS managed objects. Tests implemented in Release A will be as provided by COTS.

Mapping to objects implemented by this component

ManagementFramework

MsPmTest

6.2.5.4 Performance Management Proxy CSC

Purpose and Description

This CSC provides the interface for the management of the MsPmApplManager.

Mapping to objects implemented by this component

MsPmCallBacks

MsPmProxy

Candidate Products

Custom - C++ code

6.2.6 Performance Management Management and Operation

6.2.6.1 System Management Strategy

The MsPmApplManager utilizes the MSS Management Agent Services for its management. The class MsPmProxy provides for the startup, shutdown, and monitoring of MsPmApplManager from a management application (Fault Management Service).

Since the ManagementFramework Performance Management service is integrated with the Fault Management service under HP OpenView, there will be no management service to receive an event message if the performance management service fails. Therefore, the ManagementFramework will log all reportable detected errors to a file for post processing.

6.2.6.2 Operator Interfaces

All operator interfaces will be through HP OpenView or through a COTS performance management package.

6.2.6.3 Reports

The following predefined performance management reports are available:

Interface Traffic Statistics report -- graphical representation of packet statistics (in real time) for operator-specified node(s)

SNMP Traffic report -- graphical representation of incoming and outgoing SNMP packets (in real time) for operator-specified node(s).

SNMP Operations report -- graphical representation (in real time) of the number of SNMP operations requested of and performed by the SNMP agent on the selected node(s).

Site Host Resource Utilization report -- tabular listing of statistics (minimum, average, and maximum) for various host performance metrics on each host at the site.

EMC Host Resource Utilization report -- tabular listing of statistics (minimum, average, and maximum) for various host performance metrics on each host at each specified site

Disk Space report -- Text-based report that lists the available file system space on the operator-specified node.

Additional performance reports will be provided on an ad hoc basis.

Further information on performance management reports is available in the Release A Overview Design Specification (305-CD-004-001).

6.3 Security Management

6.3.1 Security Management Overview

The Security Management Application Service provides for the management of the security mechanisms that are used to protect and control access to ECS resources. It provides the rules and the implementation for authentication procedures, the maintenance of authorization facilities, the maintenance of security logs, intrusion detection and recovery procedures. The mechanisms used to provide security in ECS comprise three distinct parts: network security, distributed communications security, and host-based security.

Network security management involves the management of routing tables used for address-based filtering (network authorization). This is implemented through router COTS configuration files through which access control rules are specified.

Distributed communications security addresses communications between software entities such as clients and servers employing mechanisms such as Kerberos/DCE for real-time authentication exchange. The management of distributed communications security involves the management of the authentication database (the DCE registry database), the authorization database (DCE Access Control List Managers). This is implemented through the use of HAL DCE Cell Manager. The DCE Cell Manager is a COTS product that provides a Motif-based capability to administer the DCE security registry (authentication database), and the access controls on cell resources (authorization database).

Host-based security management addresses the control of access to and the protection of these mechanisms, in addition to the management of compliance to established security policy (e.g. password usage guidelines), and intrusion detection (e.g. break-ins). Access control to network

services is implemented through TCP wrappers, a public domain tool. Compliance management is implemented through public domain products npasswd, crack, and SATAN. Intrusion detection is implemented through the public domain product Tripwire, and custom development.

The Security Management Application Service has two instances, the site and the SMC Security Management Application Services. The site Security Management Application Service manages security databases local to it, manages compliance to security directives and guidelines established and disseminated by the SMC, performs intrusion detection checks in order to maintain the integrity of ECS resources, provide the capability to analyze security audit trails, and provide the mechanisms to generate reports for such these activities. The SMC Security Management Application Service is responsible for establishing and disseminating security guidelines to the sites, disseminating security advisories received from external systems (security agencies such as CERT and NIST) to the sites, receive security reports from the sites, and to receive notifications of and coordinate the recovery from detected security breaches at the sites and external systems.

6.3.2 Security Management Context

The Security Management Application Service, as shown in the context diagram, Figure 6.3-1, interfaces with the SMC, the Fault Management Application Service, the Management Database and with systems external to ECS, namely V0, NSI, IP and NCC. The information exchanged across these interfaces, as shown in the diagram, is described here.

Notifications of security events and summary data are forwarded by the Security Management Application Service to the SMC, while coordination for recovery and security advisories are received from the SMC. The interface to the Fault Management Application Service (via the Management Agent Service) allows for the Security Management Application Service to send security event notifications, fault events, and receive startup and shutdown commands. The interface to the Management Database provides access to the management data for the purpose of report generation.

The external systems and the Security Management Application Service exchange notifications of security breaches and recovery coordination information.

6.3.3 Security Management Object Model

The AuthenticationDB and the AuthorizationDB represent the authentication and the authorization databases respectively. The authentication database contains records for Principals. Principals may be users, clients, or servers. Each principal has an identification and a password, and may belong to one or more groups. The M&O staff may create, modify, and delete principal information. These are capabilities provided by COTS products (operating system-based authentication, DCE authentication database (rgy_edit)). HAL DCE Cell Manager is a COTS product that provides the capability for DCE Cell Management, to include the management of the authentication database (Registry database) for the management of DCE principals.

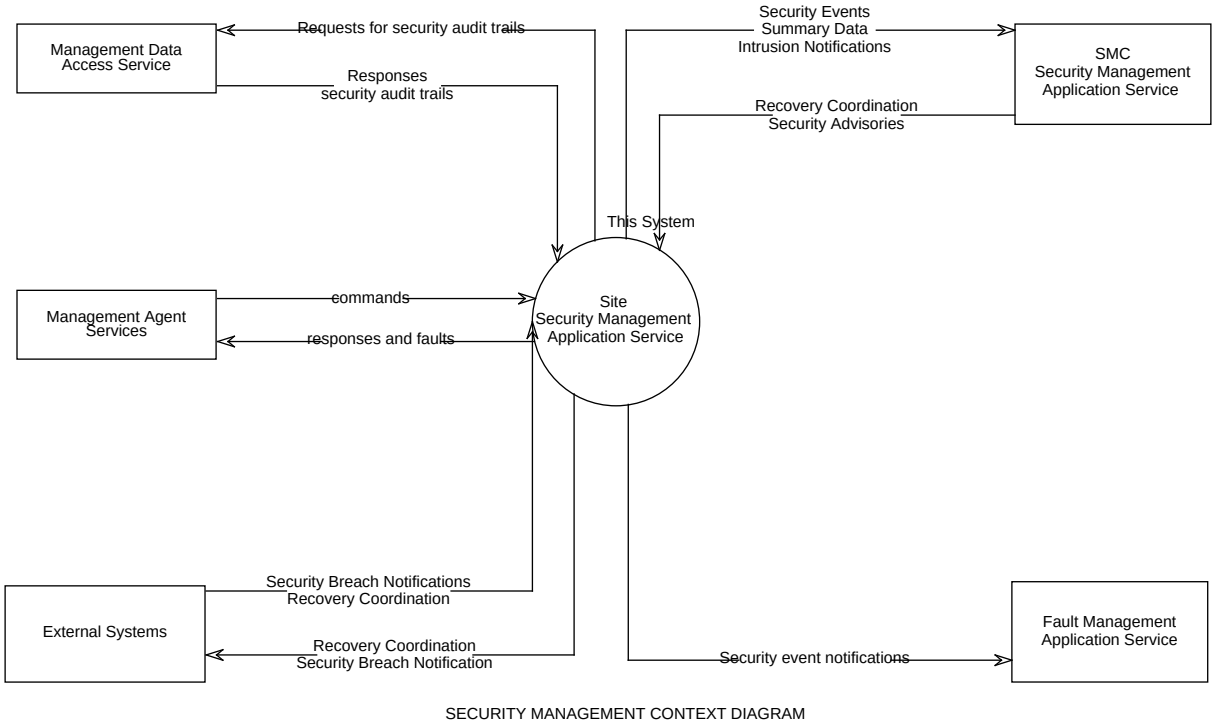


Figure 6.3-1. Security Management Context Diagram

Principals attempt to access security-managed resources. These resources comprise data and services. Access to these resources is controlled by entries in the authorization database. The Authorization databases control access to the access control lists and allows M&O staff to update these access control lists in order to control access by Principals of security-managed resources. These authorization capabilities are provided by the operating system access control lists, router configuration databases, configuration files of TCP wrappers, and DCE acl_edit (all of which are COTS products). HAL DCE Cell Manager, as mentioned above, is a COTS product that provides the capability for the management of Access Control Lists associated with cell resources.

Security Tests are run on a scheduled basis, and on-demand at the request of the site M&O staff, in order to audit the implementation of the security mechanisms. MsScManager is the controller class for the Security Management Application Service. It provides the capability to run a test (MsScTest) on demand. There are two kinds of security tests: ComplianceTests and IntrusionDetectionTests. ComplianceTests comprise tests that audit passwords for criteria such as being easily-guessable or the incorrect length, tests that check for file system integrity, the presence of world-readable and world-writable directories. IntrusionDetectionTests comprise tests that check for evidence of break-ins and break-in attempts.

Notifications of security events, security data and security reports are sent to the SMC. The security data forwarded from the various sites allows the Security Management Service to correlate events at different sites. Notifications of security breaches are sent to the SMC (and external systems such as NSI, CERT and the NASIRC), while security advisories are received from them in addition to the coordination of recovery from security events.

The security management object model is shown in Figure 6.3-2.

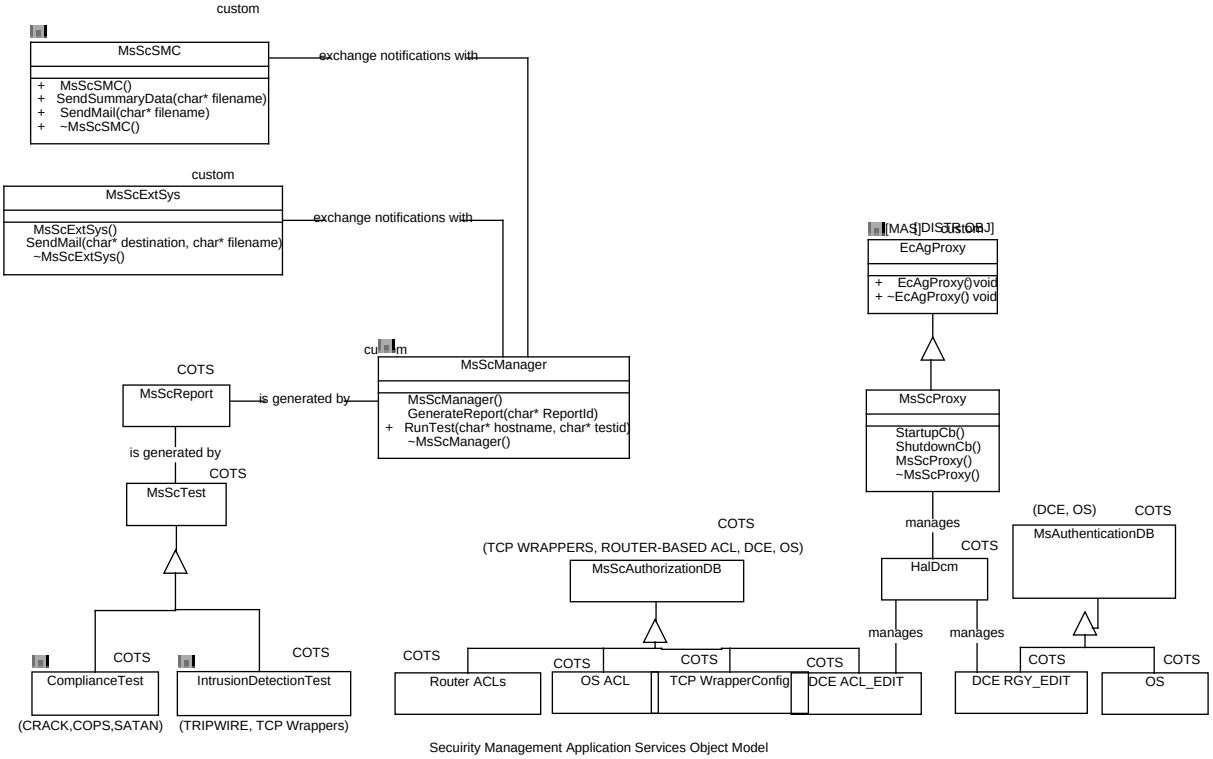


Figure 6.3-2. Security Management Object Model

6.3.3.1 ComplianceTest Class

Parent Class: MsScTest

Public: No Distributed Object: No

Purpose and Description:

This class represents a security test that checks for the compliance to established security policy. These tests are implemented through public domain products crack, COPS and SATAN. Crack checks for the adherence to established policy for passwords by attempting to guess passwords. COPS and SATAN generate analysis of the security mechanisms of specified hosts. Since these represent COTS products, these will not be described in detail here. The reader is referred to the appropriate COTS documentation.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The ComplianceTest class has associations with the following classes:
None

6.3.3.2 DCEACL_EDIT Class

Parent Class: MsScAuthorizationDB
Public: No Distributed Object: No
Purpose and Description:
This class represents the ACL_EDIT utility provided by DCE for the purpose of managing ACLs associated with DCE servers. Since this is a COTS product (DCE), it will not be described in detail here. The reader is referred to the DCE documentation set for details.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DCEACL_EDIT class has associations with the following classes:
Class: HalDcm manages

6.3.3.3 DCERGY_EDIT Class

Parent Class: MsAuthenticationDB
Public: No Distributed Object: No
Purpose and Description:
This class represents the DCE utility RGY_EDIT used for the management of DCE principals in the DCE Registry database (authentication database). Since this is a COTS product, it will not be described in detail here. The reader is referred to the DCE documentation set for details.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The DCERGY_EDIT class has associations with the following classes:
Class: HalDcm manages

6.3.3.4 EcAgProxy Class

Parent Class: Not Applicable
Public: No Distributed Object: Yes
Purpose and Description:
This object class is primarily for COTS' manageability. It includes the MSS instrumentation class library to enable the manageability of the COTS product. The front-end of this object is the MSS instrumentation code. The back-end of it is the interface to the COTS. It is unique to every COTS. In security management, the logs of COTS are monitored by this object. If an security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.

Attributes:

None

Operations:

EcAgProxy - This method specifies the constructor of the class.
Arguments:
Return Type: void
Privilege: Public

~EcAgProxy - This method specifies the destructor of the class.
Arguments:
Return Type: void
Privilege: Public

Associations:

The EcAgProxy class has associations with the following classes:
None

6.3.3.5 HalDcm Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class represents the COTS product HAL DCE Cell Manager. This product provides the capability to manage DCE services, the DCE authentication database (the Registry), access control for cell resources (authorization). Since this is a COTS product, it will not be described in detail here. The reader is referred to the documentation set of the product.

Attributes:

None

Operations:

None

Associations:

The HalDcm class has associations with the following classes:

Class: DCEACL_EDIT manages

Class: DCERGY_EDIT manages

Class: MsScProxy manages

6.3.3.6 IntrusionDetectionTest Class

Parent Class: MsScTest

Public: No Distributed Object: No

Purpose and Description:

This class represents a security test that checks for intrusions. Tripwire is a public domain product that tests for the integrity of a file system by generating checksums of files and comparing them with a previously generated database of checksums. The configuration of this product involves establishing the database of file signatures, and establishing a schedule for the execution of the tests, and the capability for the execution of the tests on-demand. TCP wrappers is a public domain product that monitors and controls access to network services on a host. The configuration of this product involves the specification of access rules for network services, configuring the logging of access attempts. Since these represent COTS products, they will not be described in detail here. The reader is referred to the appropriate COTS documentation set.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The IntrusionDetectionTest class has associations with the following classes:
None

6.3.3.7 MsAuthenticationDB Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the authentication databases that provide authentication for principals. This functionality is provided by COTS products.

Attributes:

None

Operations:

None

Associations:

The MsAuthenticationDB class has associations with the following classes:
None

6.3.3.8 MsScAuthorizationDB Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the authorization databases that provide access control for resources. This functionality is provided by COTS products (router configuration files, operating system access control lists, TCP wrapper configuration files, and DCE ACLs). Since these are provided by COTS, they will not be described in detail here. The reader is referred to the appropriate documentation set for details.

Attributes:

None

Operations:

None

Associations:

The MsScAuthorizationDB class has associations with the following classes:
None

6.3.3.9 MsScExtSys Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the interface for MsScManager to send security data or electronic mail to external systems.

Attributes:

None

Operations:

MsScExtSys - This is the default constructor for this class.
Arguments:

SendMail - This method sends a notification or report via electronic mail to the external system specified by the destination.
Arguments: char* destination, char* filename

~MsScExtSys - This is the default destructor for this class.
Arguments:

Associations:

The MsScExtSys class has associations with the following classes:
Class: MsScManager exchangenotificationswith

6.3.3.10 MsScManager Class

Parent Class: Not Applicable

Public: No Distributed Object: No

Purpose and Description:

This class provides the capability for the M&O Staff to generate security reports, and to run initiate the execution of security tests.

Attributes:

None

Operations:

GenerateReport

Arguments: char* ReportId

MsScManager - This is the default constructor for this class.

Arguments:

RunTest - This method runs the specified security test on the specified host.

Arguments: char* hostname, char* testid

Return Type: Void

Privilege: Public

~MsScManager - This is the destructor for this class.

Arguments:

Associations:

The MsScManager class has associations with the following classes:

Class: MsScExtSys exchangenotificationswith

Class: MsScSMC exchangenotificationswith

Class: MsScReport isgeneratedby

6.3.3.11 MsScProxy Class

Parent Class: EcAgProxy

Public: No Distributed Object: No

Purpose and Description:

This class is a specialization of the EcAgProxy class. It provides the capability for the monitoring and management of the Security Management Service (startup and shutdown).

Attributes:

All Attributes inherited from parent class

Operations:

MsScProxy - This is the default constructor for this class.
Arguments:

ShutdownCb - This method is used to shutdown the COTS product. It invokes the vendor supplied shutdown script.
Arguments:

StartupCb - This method is used for the startup of the COTS product. It invokes the vendor supplied startup script.
Arguments:

~MsScProxy - This is the destructor for this class.
Arguments:

Associations:

The MsScProxy class has associations with the following classes:
Class: HalDcm manages

6.3.3.12 MsScReport Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the reports generated by the COTS and from the security data stored in the management database.

Attributes:

None

Operations:

None

Associations:

The MsScReport class has associations with the following classes:
Class: MsScManager isgeneratedby
Class: MsScTest isgeneratedby

6.3.3.13 MsScSMC Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the interface between the site Security Management Service and the SMC. It provides the capability to send a report, or a an electronic mail message to the SMC.

Attributes:

None

Operations:

MsScSMC - This is the default constructor for this class.
Arguments:
Return Type: Void
Privilege: Public

SendMail - This method sends a file containing security data, specified by filename, to the SMC.
Arguments: char* filename
Return Type: Void
Privilege: Public

SendSummaryData - This method sends a report containing security data, specified by filename, to the SMC.
Arguments: char* filename
Return Type: Void
Privilege: Public

~MsScSMC - This is the destructor for this class.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsScSMC class has associations with the following classes:
Class: MsScManager exchangenotificationswith

6.3.3.14 MsScTest Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the Compliance Management and Intrusion Detection tests that may be run. These tests are COTS products, and will not be described in detail here. The reader is referred to the documentation set of the COTS.

Attributes:

None

Operations:

None

Associations:

The MsScTest class has associations with the following classes:
Class: MsScReport isgeneratedby

6.3.3.15 OS Class

Parent Class: MsAuthenticationDB
Public: No Distributed Object: No
Purpose and Description:
This class represents the authentication database provided by the operating system. Since this is provided by COTS, it will not be described in detail here. The reader is referred to the appropriate documentation for details.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The OS class has associations with the following classes:
None

6.3.3.16 OSACL Class

Parent Class: MsScAuthorizationDB
Public: No Distributed Object: No
Purpose and Description:
This class represents access controls provided by an operating system for host resources. Since this represents COTS, it will not be described in detail here. The reader is referred to the appropriate documentation for details.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The OSACL class has associations with the following classes:
None

6.3.3.17 RouterACLs Class

Parent Class: MsScAuthorizationDB
Public: No Distributed Object: No
Purpose and Description:
This class represents router access control lists, used to filter incoming and outgoing packets based on the access control rules specified. This is a COTS product, and will not be described in detail here. The reader is referred to the appropriate documentation for details.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The RouterACLs class has associations with the following classes:
None

6.3.3.18 TCPWrapperConfig Class

Parent Class: MsScAuthorizationDB
Public: No Distributed Object: No
Purpose and Description:
This class represents the configuration files of TCP Wrappers used to control access to network services on a host. This control is established through access rules specified for the various network services. This is the customization of the COTS required. Since this functionality is provided by COTS, it will not be described in detail here. The reader is referred to the appropriate COTS documentation for details.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The TCPWrapperConfig class has associations with the following classes:
None

6.3.4 Security Management Dynamic Model

6.3.4.1 Executing a Compliance Test

This scenario is depicted in Figure 6.3-3.

6.3.4.1.1 Beginning Assumptions

None.

6.3.4.1.2 Interfaces with Other Subsystems and Segments

Management Agent Services

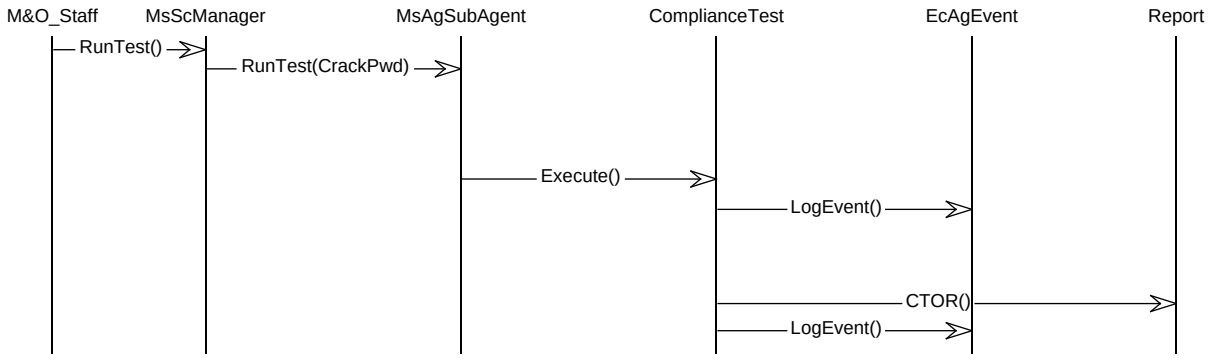


Figure 6.3-3. Executing a Compliance Test

6.3.4.1.3 Stimulus

An operator initiates the execution of a password guessing test (a compliance test).

6.3.4.1.4 Participating Classes From the Object Model

MsScManager

ComplianceTest

EcAgEvent

Report

6.3.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and subsystem are in a steady state.

6.3.4.1.6 Ending State

The system, segment and subsystem are in a steady state.

6.3.4.1.7 Scenario Description

In response to operator-provided stimulus, the RunTest method is invoked for the run of a test to determine the compliance of passwords on the host to established policy. This causes the compliance test to be instantiated on the remote host specified, by way of the Management Agent Services. Upon instantiation, a message is logged to indicate the start of the test to the MSS History Log (via EcAgEvent) on the remote host. The test generates a report which is written to a well-known directory, and another event is written to the MSS History Log to indicate the completion of the test.

6.3.4.2 Reporting a Security Intrusion

This scenario is depicted in Figure 6.3-4.

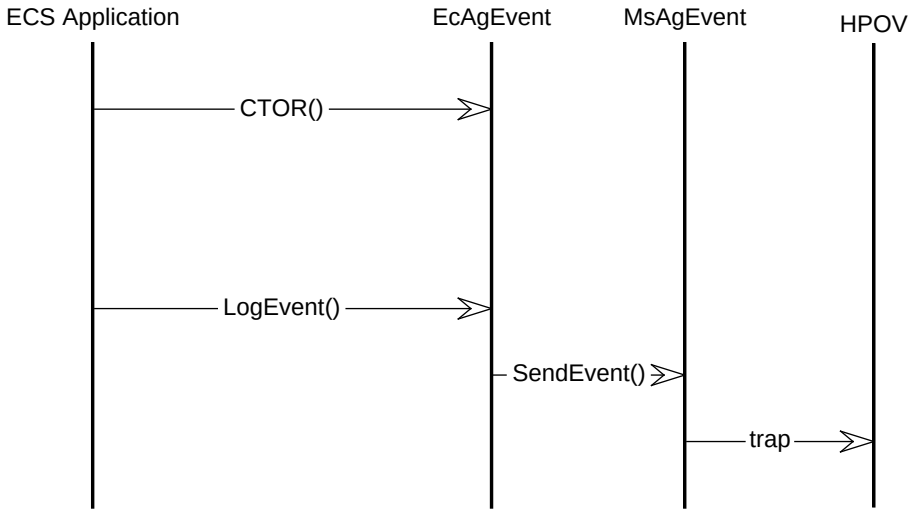


Figure 6.3-4. Reporting a Security Intrusion

6.3.4.2.1 Beginning Assumptions

None.

6.3.4.2.2 Interfaces with Other Subsystems and Segments

Management Agent Services

6.3.4.2.3 Stimulus

An application receives a request from an unauthorized client.

6.3.4.2.4 Participating Classes From the Object Model

An ECS application

EcAgEvent

MsAgEvent

UUID

ManagementFramework (HPOV)

6.3.4.2.5 Beginning System, Segment and Subsystem State(s)

The system, segment and subsystem are in a steady state.

6.3.4.2.6 Ending State

The system, segment and subsystem are in a steady state.

6.3.4.2.7 Scenario Description

An ECS application receives a request from a client for a service. The authorization check performed by the application fails since the client is not authorized to access the requested service. This unauthorized access is a security violation that needs to be reported. The ECS Application sends an alert via EcAgEvent. EcAgEvent logs the event to the MSS History Log, and forwards a real-time notification to MsAgEvent which generates an SNMP trap and sends it to the Management Framework (HP OpenView NNM).

6.3.5 Security Management Structure

Table 6.3-1 lists the components of the Security Management Application Service.

Table 6.3-1. Security Management Components

Component Name	COTS/Custom
MsScManager	Custom(C++ code)
MsScSMC	Custom
MsScExtSys	Custom
MsScReport	Custom
MsScProxy	Custom
Authentication Databases	COTS (Operating system, DCE Registry database) HAL DCE Cell Manager selected for the management of the DCE Registry database.
Authorization Databases	COTS (Operating system, DCE Access Control Lists, router configuration files, TCP wrapper access control configuration files). HAL DCE Cell Manager selected for the management of DCE ACLs.
ComplianceTest	COTS (SATAN, Crack, COPS)
IntrusionDetectionTest	COTS (Tripwire, TCP Wrappers)

6.3.5.1 Security Manager CSC

Purpose and Description

The security manager provides the capability for the M&O staff to initiate security tests and the generation of reports.

Mapping to objects implemented by this component

MsScManager

RunTest - C++ code

Report - scripts

Candidate products

Custom

6.3.5.2 Security Databases CSC (COTS)

Purpose and Description

The security databases include the Authentication and the Authorization databases, which provide for the management of user accounts and their access privileges respectively. These are all COTS products.

Mapping to objects implemented by this component

AuthenticationDB

AuthorizationDB

Candidate products

Operating System Password Files

DCE Registry Database

Router Configuration Files

TCP Wrappers configuration files

Operating System Access Control Lists

DCE Access Control Lists

6.3.5.3 Tests CSC (COTS)

Purpose and Description

These tests comprise compliance management tests and intrusion detection tests. Compliance tests permit the verification of adherence to an established security policy. These tests are all COTS.

Mapping to objects implemented by this component

ComplianceTest

IntrusionDetectionTest

Candidate products

CRACK, COPS, SATAN, TRIPWIRE

6.3.5.4 DCE Cell Management CSC

Purpose and Description

This class provides the capability to manage DCE services including security services.

Mapping to objects implemented by this component

HalDcm

Candidate products

HAL DCE Cell Manager

6.3.5.5 Security Management Proxy CSC

Purpose and Description

This CSC provides the interface for the management of the Security Service.

Mapping to objects implemented by this component

MsScProxy

MsScCallBacks

Candidate products

Custom - C++ code

6.3.6 Security Management Management and Operation

6.3.6.1 System Management Strategy

The Security Management Service Management Strategy utilizes the MSS Management Agent Services for its management. The class MsScProxy provides for the startup, shutdown, and monitoring of HalDcm from a management application (Fault Management Service).

6.3.6.2 Operator Interfaces

The public domain security products have command line interfaces. Some products have a graphical user interface as well. These interfaces will be available to the operator. The HAL DCE Cell Manager has both interfaces.

6.3.6.3 Reports

The following predefined security management reports will be available:

Security Compromise report -- table listing details for all detected security violations or attempted intrusions.

Security Compromise Statistics report -- table of statistics summarizing security violations or attempted intrusions over an operator-specified time period.

Other security management reports will be generated on an ad hoc basis.

Further information on security management reports is available in the Release A Overview Design Specification (305-CD-004-001).

6.4 Accountability Management

6.4.1 Accountability Management Overview

The Accountability Management Service provides the capabilities of User Registration and the generation of reports from audit trails.

ECS provides for two generic classes of users: guest users and registered users. Guest users are users that have not formally registered to become registered users. Registered users are those guest users that have submitted requests for a registered user account, and have had accounts created for them, based on an approval process. Registered users are allowed access to services and products beyond those available to guest users.

Guest users are provided the capability to submit a request for a registered user account, which is captured into a database of pending requests. Operators may access this database of pending requests in the process of user registration, in order to create a registered user account from a list of pending requests.

User registration provides the operators the capability to create accounts against requests submitted by guest users wishing to become authorized ECS users. The registration service provides the capabilities for the creation, modification and maintenance of accounts with user profiles.

The user profile contains information about the user. This includes the name of the user, a user identification code, the user's primary DAAC, the organizational affiliation, investigating group (such as an instrument team) affiliation (if any), the project the user is affiliated with, the name of

the PI of the project, the mailing address of the user, the shipping address to which data needs to be sent, media preferences for orders, the user's telephone number and the user's electronic mail address (if any).

The Accountability Management Service makes the user profile available to the various subsystems, such as the Data Server subsystem, information such as the user's electronic mail address and the shipping address, which are used for the distribution of data products ordered.

The Audit Trail capability provides the means to verify the integrity of the system. This comprises the generation of a user audit trail and a security audit trail with data collected from a variety of sources.

6.4.2 Accountability Management Context

The Accountability Management Service, as shown in the external interface context diagram, interfaces with other subsystems in order to provide access to User Profiles. The interface to the SMC provides the capability for the Accountability Application Service to send summary accountability data and reports to the SMC. The Management Database provides access to the management data for the purpose of generating reports. The Management Agent Service interfaces with the Accountability Management Service for the purpose to sending management commands, such as a shutdown command, and for events and faults that are reported by this service.

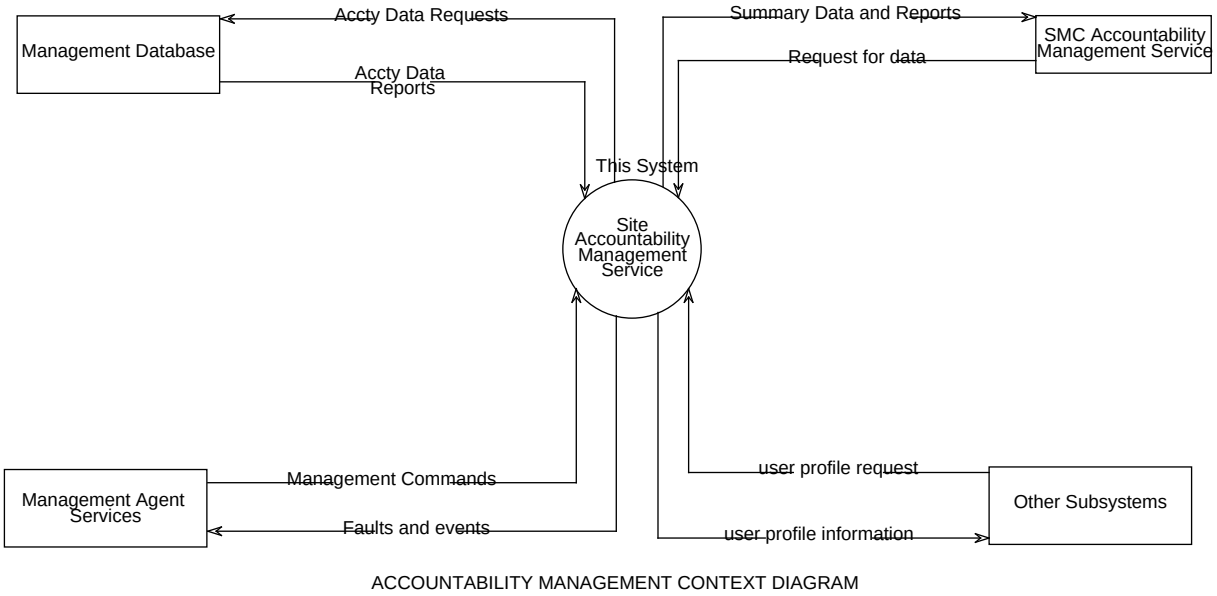


Figure 6.4-1. Accountability Management Context Diagram

6.4.3 Accountability Management Object Model

The accountability management object model is shown in Figure 6.4-2.

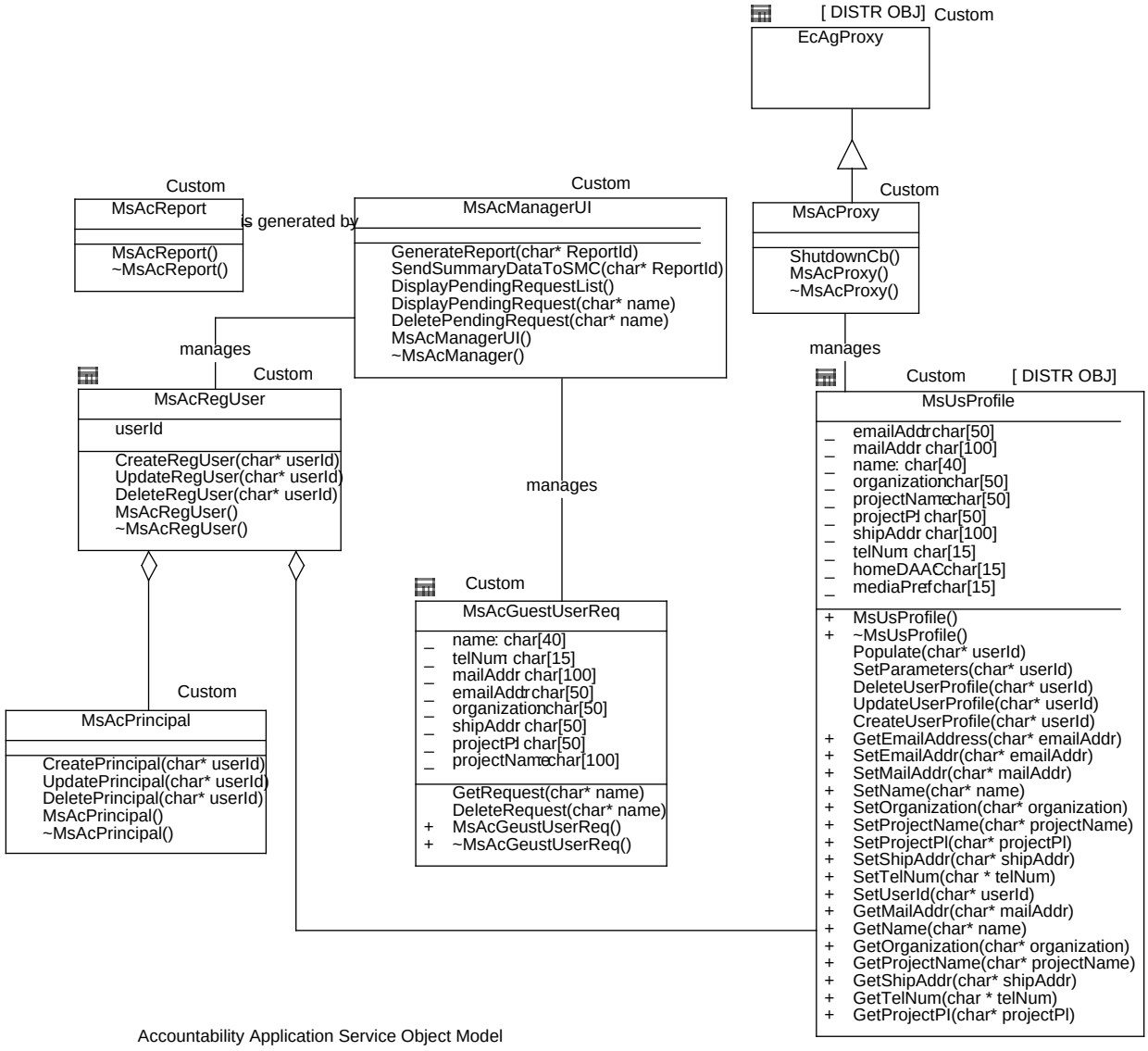


Figure 6.4-2. Accountability Management Object Model

6.4.3.1 EcAgProxy Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

This object class is primarily for COTS' manageability. It includes the MSS instrumentation class library to enable the manageability of the COTS product. The front-end of this object is the MSS instrumentation code. The back-end of it is the interface to the

COTS. It is unique to every COTS. In security management, the logs of COTS are monitored by this object. If an security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.

Attributes:

None

Operations:

None

Associations:

The EcAgProxy class has associations with the following classes:
None

6.4.3.2 MsAcGuestUserReq Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides access to the pending requests for registered user accounts.

Attributes:

emailAddr - This attribute represents the electronic mail address of the user.
Data Type: char[50]
Privilege: Private
Default Value:

mailAddr - This attribute represents the mailing address of the guest user.
Data Type: char[100]
Privilege: Private
Default Value:

name - This attribute represents the name of the guest user.
Data Type: char[40]
Privilege: Private
Default Value:

organization - This attribute represents the organization the user is affiliated to.
Data Type: char[50]

Privilege: Private
Default Value:

projectName - This attribute represents the name of the organization the user is affiliated with.
Data Type: char[100]
Privilege: Private
Default Value:

projectPl - This attribute represents the name of the PI for the project.
Data Type: char[50]
Privilege: Private
Default Value:

shipAddr - This attribute represents the address that products ordered by the user are shipped to.
Data Type: char[50]
Privilege: Private
Default Value:

telNum - This attribute represents the telephone number of the guest user.
Data Type: char[15]
Privilege: Private
Default Value:

Operations:

DeleteRequest
Arguments: char* name

GetRequest
Arguments: char* name

MsAcGeustUserReq - This method represents the constructor of the class.
Arguments:
Return Type: Void
Privilege: Public

~MsAcGeustUserReq - This method represents the destructor of the class.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsAcGuestUserReq class has associations with the following classes:
Class: MsAcManagerUI manages

6.4.3.3 MsAcManagerUI Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class provides the user interface to allow an operator to view pending requests for registered accounts, create a registered user account from an entry in the pending requests list.

Attributes:

None

Operations:

DeletePendingRequest
Arguments: char* name

DisplayPendingRequest - This method retrieves from the pending request database, and displays all fields of a specific pending request selected from a list of all pending requests.
Arguments: char* name

DisplayPendingRequestList - This method deletes the specified pending request from the pending request database.
Arguments:

GenerateReport - This method generates a specified report from the data consolidated from the history log files.
Arguments: char* ReportId

MsAcManagerUI - This method represents the constructor of the class.
Arguments:

SendSummaryDataToSMC
Arguments: char* ReportId

~MsAcManager - This method represents the destructor of the class.
Arguments:

Associations:

The MsAcManagerUI class has associations with the following classes:
Class: MsAcReport isgeneratedby
Class: MsAcGuestUserReq manages
Class: MsAcRegUser manages

6.4.3.4 MsAcPrincipal Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents a DCE principal which is accessed by means of CSS provided APIs.

Attributes:

None

Operations:

CreatePrincipal
Arguments: char* userId

DeletePrincipal
Arguments: char* userId

MsAcPrincipal - This method represents the constructor of the class.
Arguments:

UpdatePrincipal
Arguments: char* userId

~MsAcPrincipal - This method represents the destructor of the class.
Arguments:

Associations:

The MsAcPrincipal class has associations with the following classes:
MsAcRegUser (Aggregation)

6.4.3.5 MsAcProxy Class

Parent Class: EcAgProxy
Public: No Distributed Object: No
Purpose and Description:

This class is a specialization of the EcAgProxy class. It provides the capability for the monitoring and management of the Accountability Management Service.

Attributes:

All Attributes inherited from parent class

Operations:

MsAcProxy - This method represents the constructor of the class.
Arguments:

ShutdownCb - This method initiates a shutdown of the Accountability Management Service.
Arguments:

~MsAcProxy - This method represents the destructor of the class.
Arguments:

Associations:

The MsAcProxy class has associations with the following classes:
Class: MsUsProfile manages

6.4.3.6 MsAcRegUser Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents a registered user in the system. This class is an aggregation of a principal and a user profile. The class MsAcPrincipal represents a DCE principal, and is accessed by means of CSS provided APIs.

Attributes:

userId - This attribute represents a unique user identification.

Operations:

CreateRegUser
Arguments: char* userId

DeleteRegUser

Arguments: char* userId

MsAcRegUser - This method represents the constructor of the class.
Arguments:

UpdateRegUser
Arguments: char* userId

~MsAcRegUser - This method represents the destructor of the class.
Arguments:

Associations:

The MsAcRegUser class has associations with the following classes:
Class: MsAcManagerUI manages

6.4.3.7 MsAcReport Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents accountability reports that are generated by this service. These reports are generated from the data in the management database.

Attributes:

None

Operations:

MsAcReport - This method represents the constructor of the class.
Arguments:

~MsAcReport - This method represents the destructor of the class.
Arguments:

Associations:

The MsAcReport class has associations with the following classes:
Class: MsAcManagerUI isgeneratedby

6.4.3.8 MsUsProfile Class

Parent Class: Not Applicable

Public: Yes Distributed Object: Yes

Purpose and Description:

This is a public class (and a distributed object) that is exported by this service. Other subsystems use this class in order to retrieve the user profile for a specified user.

Attributes:

emailAddr - This attribute represents the electronic mail address of the user.
Data Type: char[50]
Privilege: Private
Default Value:

homeDAAC - The home DAAC or the primary DAAC for the user.
Data Type: char[15]
Privilege: Private
Default Value:

mailAddr - This attribute represents the mailing address of the user.
Data Type: char[100]
Privilege: Private
Default Value:

mediaPref - The media the user prefers data to be delivered on.
Data Type: char[15]
Privilege: Private
Default Value:

name - This attribute represents the name of the user.
Data Type: char[40]
Privilege: Private
Default Value:

organization - This attribute represents the organization the user is affiliated to.
Data Type: char[50]
Privilege: Private
Default Value:

projectName - This attribute represents the name of the organization the user is affiliated with.
Data Type: char[50]
Privilege: Private
Default Value:

projectPl - This attribute represents the name of the PI for the project.
Data Type: char[50]
Privilege: Private
Default Value:

shipAddr - This attribute represents the address that products for this user are shipped to.
Data Type: char[100]
Privilege: Private
Default Value:

telNum - This attribute represents the telephone number of the user.
Data Type: char[15]
Privilege: Private
Default Value:

Operations:

CreateUserProfile
Arguments: char* userId

DeleteUserProfile
Arguments: char* userId

GetEmailAddress - This method retrieves the Email address of the user from the user profile.
Arguments: char* emailAddr
Return Type: Void
Privilege: Public

GetMailAddr - This method retrieves the mailing address from the current user profile.
Arguments: char* mailAddr
Return Type: Void
Privilege: Public

GetName - This method retrieves the name of the user from the current user profile.
Arguments: char* name
Return Type: Void
Privilege: Public

GetOrganization - This method retrieves the organization name from the current user profile.
Arguments: char* organization
Return Type: Void

Privilege: Public

GetProjectName - This method retrieves the name of the project from the current user profile.
Arguments: char* projectName
Return Type: Void
Privilege: Public

GetProjectPI - This method retrieves the name of the project PI from the current user profile.
Arguments: char* projectPl
Return Type: Void
Privilege: Public

GetShipAddr - This method retrieves the shipping address from the current user profile.
Arguments: char* shipAddr
Return Type: Void
Privilege: Public

GetTelNum - This method retrieves the telephone number from the current user profile.
Arguments: char * telNum
Return Type: Void
Privilege: Public

MsUsProfile - This method specifies the constructor for this class.
Arguments:
Return Type: Void
Privilege: Public

Populate - This method retrieves the user profile for the specified userId.
Arguments: char* userId

SetEmailAddr - This method updates the Email address for the current user profile.
Arguments: char* emailAddr
Return Type: Void
Privilege: Public

SetMailAddr - This method sets the mailing address for the current user profile.
Arguments: char* mailAddr
Return Type: Void
Privilege: Public

SetName - This method updates the name for the current user profile.
Arguments: char* name
Return Type: Void

Privilege: Public

SetOrganization - This method updates the organization name for the current user profile. It does not, as the other Set methods, update the user profile database. It updates the attribute in the MsUsProfile object.
Arguments: char* organization
Return Type: Void
Privilege: Public

SetParameters - This method sets the value for each attribute from the MsUsProfile object in the user profile database.
Arguments: char* userId

SetProjectName - This method sets the specified value for the project name for the current user profile (MsUsProfile)
Arguments: char* projectName
Return Type: Void
Privilege: Public

SetProjectPl - This method sets the specified value for the Project PI for the current user profile (MsUsProfile).
Arguments: char* projectPl
Return Type: Void
Privilege: Public

SetShipAddr - This method sets the specified shipping address for the current user profile (MsUsProfile).
Arguments: char* shipAddr
Return Type: Void
Privilege: Public

SetTelNum - This method sets the telephone number for the current user profile (MsUsProfile).
Arguments: char * telNum
Return Type: Void
Privilege: Public

SetUserId - This method sets the specified userId in the current user profile (MsUsProfile).
Arguments: char* userId
Return Type: Void
Privilege: Public

UpdateUserProfile
Arguments: char* userId

~MsUsProfile - This method specifies the destructor for this class.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsUsProfile class has associations with the following classes:
Class: MsAcProxy manages
MsAcRegUser (Aggregation)

6.4.4 Accountability Management Dynamic Model

6.4.4.1 Retrieving a User's Email Address

This scenario traces the events associated with an ECS Application retrieving a User Profile. It is depicted in Figure 6.4-3.

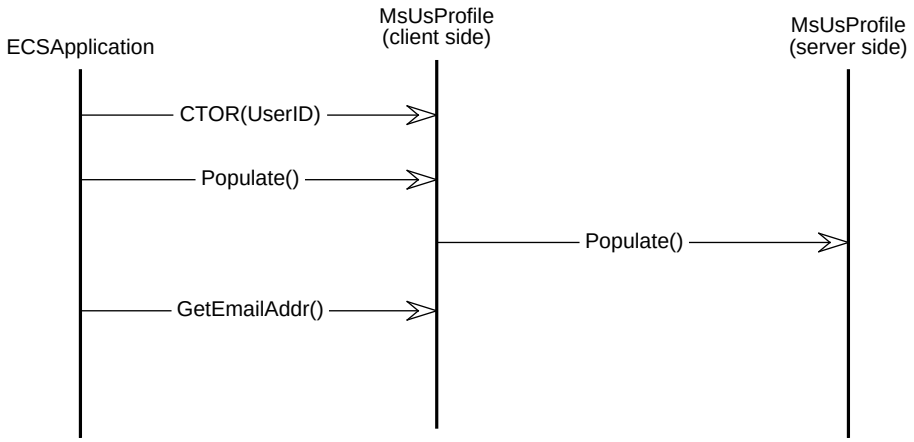


Figure 6.4-3. Retrieving a User's Email Address

6.4.4.1.1 Beginning Assumptions

None.

6.4.4.1.2 Interfaces with Other Subsystems and Segments

An ECS application that needs to retrieve a User Profile for the Email Address

6.4.4.1.3 Stimulus

An ECS application initiates a call to retrieve the User Profile for a registered user.

6.4.4.1.4 Participating Classes From the Object Model

MsUsProfile

MsAcRegUser

6.4.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.4.4.1.6 Ending State

The calling ECS application retrieves the Email address, and the system, segment and subsystem are in a normal steady state.

6.4.4.1.7 Scenario Description

An ECS application creates an instance of MsUsProfile (a public class exported by the Accountability Application Service) and invokes the method Populate in order to retrieve a User Profile. The method Populate in turn invokes the method Populate in the MsUsProfile in the server side object, which retrieves the attributes of the User Profile object. The ECS application then retrieves the user's Electronic Mail address through the use of the method GetemailAddr.

6.4.5 Accountability Management Structure

Table 6.4-1 lists the components of the Accountability Management Service.

Table 6.4-1. Accountability Management Components

Component Name	COTS/Custom
Accountability Manager	Custom (C/C++ code, HTML)
User Profile Server	Custom (C++ code)
Accountability Management Proxy	Custom

6.4.5.1 Accountability Manager CSC

Purpose and Description

The Accountability Manager CSC provides the Operator Interface which allows operators to view and delete pending requests, and to create, modify and delete registered accounts.

Mapping to objects implemented by this component

MsAcManagerUI

Generate Report

SendSummaryDataToSMC

DisplayPendingRequestList - HTML

C code

DisplayPendingRequest - HTML

C code

DeletePendingRequest - HTML

C code

MsAcGuestUserReq

GetRequest - C++ code

DeleteRequest - C++ code

MsAcRegUser

Principal

CreatePrincipal - C++ code

UpdatePrincipal - C++ code

DeletePrincipal - C++ code

6.4.5.2 User Profile Server CSC

Purpose and Description

This CSC provides the server side functionality of the public class exported by Accountability Management Service. This class responds to requests from the client (imported into ECS applications) and provides the user profile requested in response to the call.

Mapping to objects implemented by this component

MsUsProfile

C++ code

Populate

SetEmailAddr

SetMailAddr

SetName

SetOrganization

SetProjectName

SetProjectPI

Set ShipAddr

SetTelNum

SetParameters

GetEmailAddr

GetMailAddr

GetName

GetOrganization

GetProjectName

GetProjectPI

Get ShipAddr

GetTelNum

CreateUserProfile

UpdateUserProfile

DeleteUserProfile

6.4.5.3 Management Proxy CSC

Purpose and Description

This proxy (exported by the Management Agent Services) facilitates the management (startup and shutdown) of Accountability Management Service, and the logging of events.

Mapping to objects implemented by this component

C++ code

MsAcProxy

ShutdownCb

6.4.6 Accountability Management Management and Operation

6.4.6.1 System Management Strategy

The Accountability Management Service utilizes the public class exported by the Management Agent Services, represented by MsAcProxy. These classes facilitate the management of the service.

6.4.6.2 Operator Interfaces

The Accountability Management Service provides a graphical user interface. This user interface provides the operator the capability to view and delete pending user requests; and to create, modify and delete registered user accounts from (approved) pending requests.

6.4.6.3 Reports

The Accountability Management Service provides the following predefined reports:

User Characterization report -- provides a summary of the types and number of ECS users

System Access Profile report -- provides a summary of the types and number of ECS accesses

Utilization of User Services Personnel Accountability report -- provides a summary of the types and number of user services contacts with ECS users

Other accountability management reports, such as user audits and data audits, will be created on an ad hoc basis.

Further information on accountability management reports is available in the Release A Overview Design Specification (305-CD-004-001).

6.5 Physical Configuration Management Service

6.5.1 Physical Configuration Management Service Overview

The Physical Configuration Management Service (PCMS) provides the capability to track, manage, and control all the physical elements in the network. It integrates graphics with data to create a complete electronic model of the physical infrastructure of the network. It provides tools to locate physical proximity of down nodes, place newly discovered nodes, and manage circuit changes. It supports a variety of network administration applications including inventory, billing,

and troubleshooting. It has mechanisms to track everything from maintenance data, network protocol data to software registration. In addition, it provides integration support to several Trouble Ticket applications.

The Physical Configuration Management Service is provided by the COTS package MountainView by Accugraph. The MountainView package is responsible for transforming the logical network management environment into a physical one. It has a graphics module to create new drawings or to import existing facility drawings to set up a physical layout of the network. It has a database module that can link to a Structured Query Language (SQL) relational database to perform advanced graphical database functions. The database can be used to associate non-graphical data with individual graphical elements in the network drawings. For example, a distressed device can be located in the database and the associated drawing automatically loaded. Once loaded, the device will appear on the physical map with its current status. The MountainView package is capable of attaching symbols in the drawings with the database. Once the attachment is created, changes made to the drawings will be updated to the database by executing a drawing-to-database update function. The reverse is also true.

The MountainView package has a standard link definition to the SNMP MIB standard. This link provides direct population of physical network component information into the database. The acquisition of network component data can be accomplished in two general manners. The first involves application extensions that link the MountainView package directly into the logical network management platform. This allows dynamic network data to be loaded into the database automatically. The second is through the MountainView user interface. This allows the upload of information on request by executing a command function. This is useful in the area of problem resolution by requesting information on a troubled device.

6.5.2 Physical Configuration Management Service Context

The physical configuration management service context diagram is shown in Figure 6.5-1.

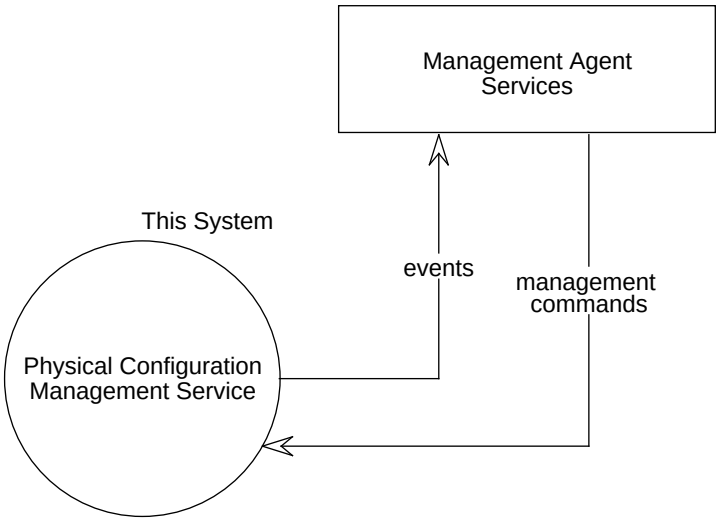


Figure 6.5-1. Physical Configuration Management Service Context Diagram

6.5.3 Physical Configuration Management Service Object Model

The Physical Configuration Management Object Model, Figure 6.5-2, indicates the classes in the Physical Configuration Management Service, the associations among them, their attributes and their operations. The classes central to the service are the PhysicalConfigurationManager, NetworkManager, and MsPcProxy.

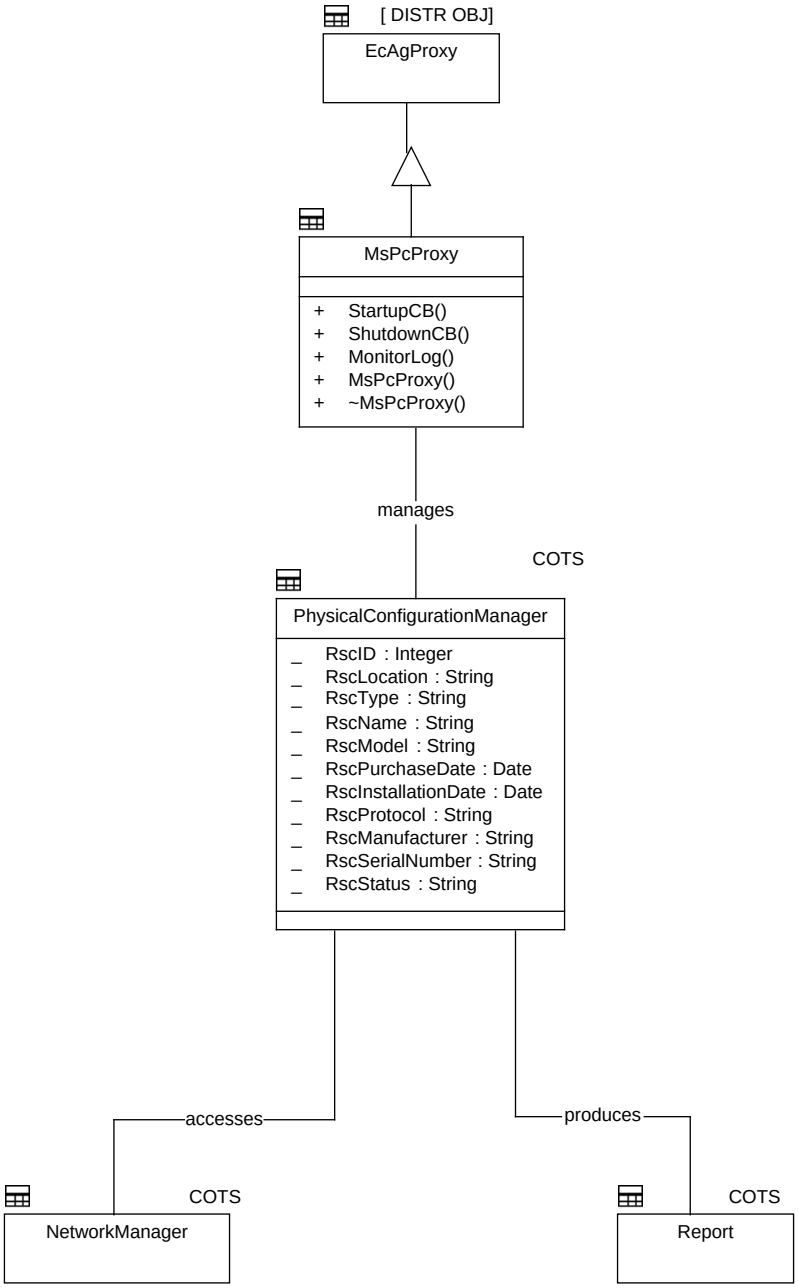


Figure 6.5-2. Physical Configuration Management Service Object Model

6.5.3.1 EcAgProxy Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

This object class is primarily for COTS' manageability. It includes the MSS instrumentation class library to enable the manageability of the COTS product. The front-end of this object is the MSS instrumentation code. The back-end of it is the interface to the COTS. It is unique to every COTS. In security management, the logs of COTS are monitored by this object. If an security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.

Attributes:

None

Operations:

None

Associations:

The EcAgProxy class has associations with the following classes:

None

6.5.3.2 MsPcProxy Class

Parent Class: EcAgProxy

Public: No Distributed Object: No

Purpose and Description:

The Physical Configuration Proxy Agent class provides the interface to the Management Agent Services. It allows the Physical Configuration Manager software to be remotely monitored and managed.

Attributes:

All Attributes inherited from parent class

Operations:

MonitorLog - This monitors the log file for errors and sends appropriate information to the MSS event page and event log file. This is the mechanism employed by the Physical Configuration Manager Services for managing errors.

Arguments:
Return Type: Void
Privilege: Public

MsPcProxy - This method represents the constructor for the object.
Arguments:
Return Type: Void
Privilege: Public

ShutdownCB - This method initiates the shutdown of the Physical Configuration Manager software.
Arguments:
Return Type: Void
Privilege: Public

StartupCB - This method initiates the startup of the Physical Configuration Manager software.
Arguments:
Return Type: Void
Privilege: Public

~MsPcProxy - This method represents the destructor for the object.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsPcProxy class has associations with the following classes:
Class: PhysicalConfigurationManager manages

6.5.3.3 NetworkManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The NetworkManager class is responsible for the logical management of the network. The Physical Configuration Manager has application extensions to read network component information directly from the Network Manager class and load it into the database, or a command can be executed at the request of the user to extract the information. This allows dynamic network data to be acquired and processed.

Attributes:

None

Operations:

None

Associations:

The NetworkManager class has associations with the following classes:
Class: PhysicalConfigurationManager accesses

6.5.3.4 PhysicalConfigurationManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The Physical Configuration Manager class is responsible for transforming logical network management into a physical one. It provides a variety of tools and mechanisms to collect, maintain, and control information concerning the physical components of the network.

Attributes:

RscID - the unique ID assigned to each resource
Data Type: Integer
Privilege: Private
Default Value:

RscInstallationDate - the installation date of the resource
Data Type: Date
Privilege: Private
Default Value:

RscLocation - the location of the resource
Data Type: String
Privilege: Private
Default Value:

RscManufacturer - the manufacturer of the resource
Data Type: String
Privilege: Private
Default Value:

RscModel - the model of the resource
Data Type: String

Privilege: Private
Default Value:

RscName - the name of the resource
Data Type: String
Privilege: Private
Default Value:

RscProtocol - the protocol of the resource
Data Type: String
Privilege: Private
Default Value:

RscPurchaseDate - the purchase date of the resource
Data Type: Date
Privilege: Private
Default Value:

RscSerialNumber - the serial number of the resource
Data Type: String
Privilege: Private
Default Value:

RscStatus - the status of the resource. This will be either be up, down, marginal, managed or unmanaged
Data Type: String
Privilege: Private
Default Value:

RscType - the type of the resource
Data Type: String
Privilege: Private
Default Value:

Operations:

None

Associations:

The PhysicalConfigurationManager class has associations with the following classes:
Class: NetworkManager accesses
Class: MsPcProxy manages
Class: Report produces

6.5.3.5 Report Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
To format and produce a variety of ad-hoc and canned reports.

Attributes:

None

Operations:

None

Associations:

The Report class has associations with the following classes:
Class: PhysicalConfigurationManager produces

6.5.4 Physical Configuration Management Service Dynamic Model

The following scenarios demonstrate typical Physical Configuration Management Service functions relating to the physical configuration management of the network.

6.5.4.1 Add A New Node Scenario

This scenario is depicted in Figure 6.5-3.

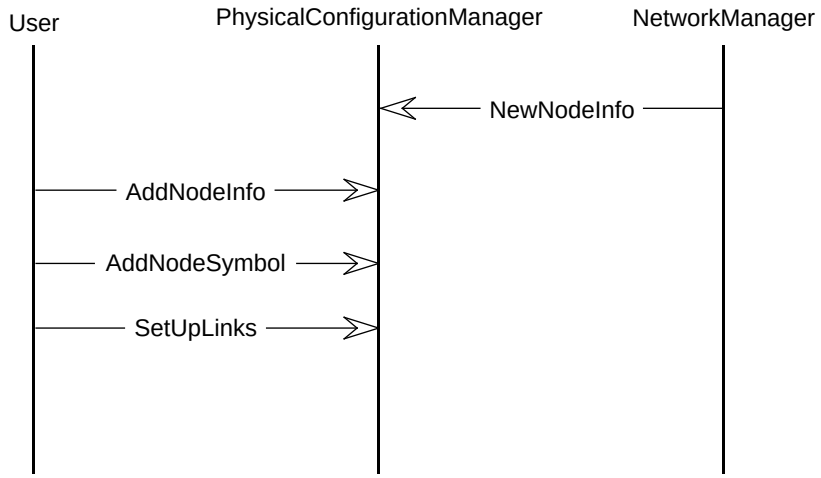


Figure 6.5-3. Add A New Node

6.5.4.1.1 Beginning Assumptions

The network is operating normally.

6.5.4.1.2 Interfaces with Other Subsystems and Segments

None.

6.5.4.1.3 Stimulus

A new node is added to the network.

6.5.4.1.4 Participating Classes From the Object Model

NetworkManager

6.5.4.1.5 Beginning System, Segment and Subsystem State(s)

The Physical Configuration Manager and the Network Manager are operating normally.

6.5.4.1.6 Ending State

All the relevant information concerning the new node becomes part of the system.

6.5.4.1.7 Scenario Description

After a new node is added to the network, certain attribute data associated with the node is gathered automatically by the network manager software and added to the database. Any additional information is entered manually and the network physical map is updated to include the new node symbol. Links are set up to attach the new node symbol with the appropriate information fields in the database.

6.5.4.2 Move An Existing Node Scenario

This scenario is depicted in Figure 6.5-4.

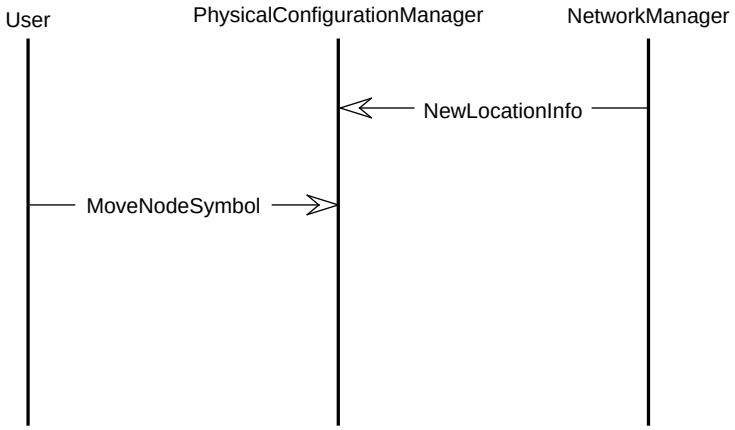


Figure 6.5-4. Move An Existing Node

6.5.4.2.1 Beginning Assumptions

The network is operating normally.

6.5.4.2.2 Interfaces with Other Subsystems and Segments

None.

6.5.4.2.3 Stimulus

An existing node is moved to a new location in the network.

6.5.4.2.4 Participating Classes From the Object Model

NetworkManager

6.5.4.2.5 Beginning System, Segment and Subsystem State(s)

The Physical Configuration Manager and the Network Manager are operating normally.

6.5.4.2.6 Ending State

The information concerning the new location is configured into the system.

6.5.4.2.7 Scenario Description

After an existing node is moved to a new location in the network, the attribute data associated with the node is reassigned automatically by the network management software and reflected in the database. The node symbol on the network map is moved to the new location with all its database links preserved.

6.5.4.3 Delete An Existing Node Scenario

This scenario is depicted in Figure 6.5-5.

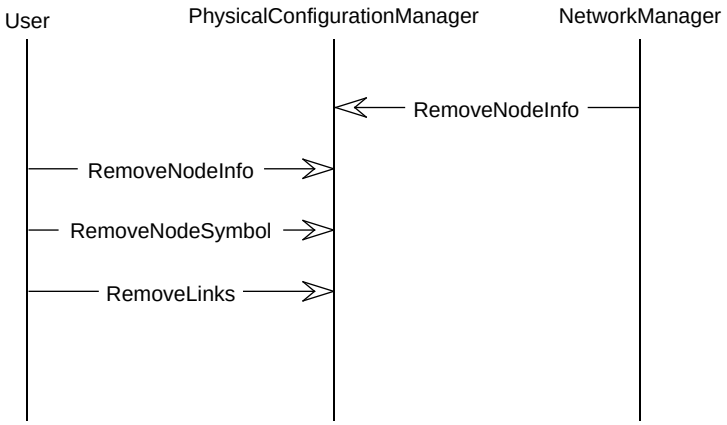


Figure 6.5-5. Delete An Existing Node

6.5.4.3.1 Beginning Assumptions

The network is operating normally.

6.5.4.3.2 Interfaces with Other Subsystems and Segments

None.

6.5.4.3.3 Stimulus

An existing node is deleted from the network.

6.5.4.3.4 Participating Classes From the Object Model

NetworkManager

6.5.4.3.5 Beginning System, Segment and Subsystem State(s)

The Physical Configuration Manager and the Network Manager are operating normally.

6.5.4.3.6 Ending State

The node symbol and information related to the deleted node are removed from the system.

6.5.4.3.7 Scenario Description

After an existing node is deleted from the network, the attribute data associated with the node is removed automatically by the network management software and reflected in the database. The node symbol on the network map is removed with all its database links deleted.

6.5.5 Physical Configuration Management Service Structure

Table 6.5-1 lists the components of the Physical Configuration Management Service.

Table 6.5-1. Physical Configuration Management Service Components

Component Name	COTS/Custom
Physical Configuration Manager	COTS
Network Manager	COTS
Physical Configuration Proxy Agent	Custom

6.5.5.1 Physical Configuration Manager CSC

Purpose and Description

The Physical Configuration Manager CSC is the COTS product MountainView. It is the configuration management tool that integrates logical network management with physical network management to provide a complete automated network management platform. It provides a variety of services related to the physical infrastructure of the network and means for displaying, tracking, and reporting on the location and status of the various physical components of the system.

Mapping to objects implemented by this component

None.

Candidate products

MountainView

6.5.5.2 Network Manager CSC

Purpose and Description

The Network Manager CSC is the COTS product HP OpenView. This product provides the management framework with the underlying management services for the management of SNMP-based network devices. It also provides the necessary integration points and services for the integration of the MountainView application. Since this is a COTS product, it will not be described in detail here. The reader is referred to the documentation set of HP OpenView for further details on the product.

Mapping to objects implemented by this component

None.

Candidate products

HP OpenView

6.5.5.3 Physical Configuration Proxy Agent CSC

Purpose and Description

The Physical Configuration Proxy Agent CSC provides the interface with the Management Agent Services. It allows for remote startup and shutdown of the Physical Configuration Manager software. It also monitors the log file for errors and sends appropriate messages to the MSS agent event page and event log file. This is the mechanism employed by the Physical Configuration Management Service for managing errors.

Mapping to objects implemented by this component

None.

CSU: Customization of Proxy Agent

6.5.6 Physical Configuration Management Service Management and Operation

6.5.6.1 System Management Strategy

The Physical Configuration Management utilizes the MSS Management Agent Services for its administration. The Physical Configuration Proxy Agent allows the remote startup, shutdown, and administration of the Physical Configuration Manager software via the Management Framework.

6.5.6.2 Operator Interfaces

The operator interface is a Motif-like GUI multi-windowing environment. It includes user definable function buttons and a hierarchical command menu. It supports user definable window and settings tailored to context sensitive working environment.

6.5.6.3 Reports

Reports are available for the following types of data:

- inventory data
- network protocol data
- software registration data
- SNMP profile data
- network trap data
- maintenance data
- connectivity data

The above reports are examples of stock reports provided by the Physical Configuration Management Service. Addition reports can be generated by combining or selecting from the different types of data available. Examples:

Hardware Inventory Report - A report containing information such as type, model, serial number, etc., for all the network components.

Software Inventory Report - A report containing information such as title, version number, installation date, etc., for all the software packages installed on different network devices.

6.6 Trouble Ticketing

6.6.1 Trouble Ticketing Overview

The Trouble Ticketing Service (TTS) provides the DAACs a common environment and means of classifying, tracking, and reporting problem occurrence and resolution to both ECS users and support staff members. TTS's core functionality is provided by the Remedy Action Request System, a COTS product. Through the configuration of this product, TTS will

- provide a graphical user interface for support staff members to access all TTS services
- include a definition of the common trouble ticket entry format
- store trouble tickets
- retrieve trouble tickets through a wide variety of criteria (ad-hoc queries)
- provide the ability to “forward” problems from one DAAC to another (or DAAC to SMC)
- produce stock and common reports
- interface with the common e-mail environment to provide automatic notification to users and support staff members
- offer an application programming interface through which applications could submit trouble tickets
- provide summary information to the SMC from each DAAC to allow trend reports regarding trouble tickets.
- define a consistent “life-cycle” for trouble tickets (through a set of standard status codes and escalation and action rule definition)
- allow each DAAC a degree a customization through definition of further escalation and action rules

Escalation rules are simply time activated events which execute on trouble tickets which meet a set of specified criteria. Actions which can be taken include notification (of either a user or support staff member), writing to a log file, setting a field value on the trouble ticket, or even running a custom written process. Qualifications can be expressed on any trouble ticket data TTS tracks. Examples of custom escalation rules might include...

- if a “High” priority trouble ticket stays in “Assigned” for more than 48 hours without being moved to “In-Progress”, re-notify the assigned support staff member
- if a “Low” priority trouble ticket is not moved to “Closed” within 14 days, raise the priority to “Medium” and re-notify the assigned support staff member

Active links are similar to escalation rules with the exception that they are defined to take place on a specified action rather than at a given time. Examples of custom active links which can be defined by a particular DAAC include...

- if a high priority trouble ticket is closed with a particular resolution code, notify a specified member of the support staff (perhaps a manager)

In addition to the functionality provided by Remedy, TTS will utilize a set of custom HTML documents to provide users with the ability to submit new trouble tickets and query the current status of any of their previous entries. Access to TTS through this technique will provide users an easy method for reporting problems in an environment with which most are already familiar. Additionally, as another means of trouble ticket entry, the TTS will provide a textual e-mail template through which automated entry of trouble tickets is also possible. Finally, support staff members are able to enter trouble tickets through the Remedy provided interface for problem received via other methods (e.g. phone calls).

6.6.2 Trouble Ticketing Context

The Trouble Ticketing Service context diagram is shown in Figure 6.6-1. TTS receives management requests, (e.g. start up, shutdown) from the Management Agent. Once the requests are completed an event is logged through the Management Agent.

6.6.3 Trouble Ticketing Object Model

Figure 6.6-2 represents the classes which model the Trouble Ticketing Service.

6.6.3.1 EcAgProxy Class

Parent Class: Not Applicable

Public: No Distributed Object: Yes

Purpose and Description:

This object class is primarily for COTS' manageability. It includes the MSS instrumentation class library to enable the manageability of the COTS product. The front-end of this object is the MSS instrumentation code. The back-end of it is the interface to the COTS. It is unique to every COTS. In security management, the logs of COTS are monitored by this object. If an security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.

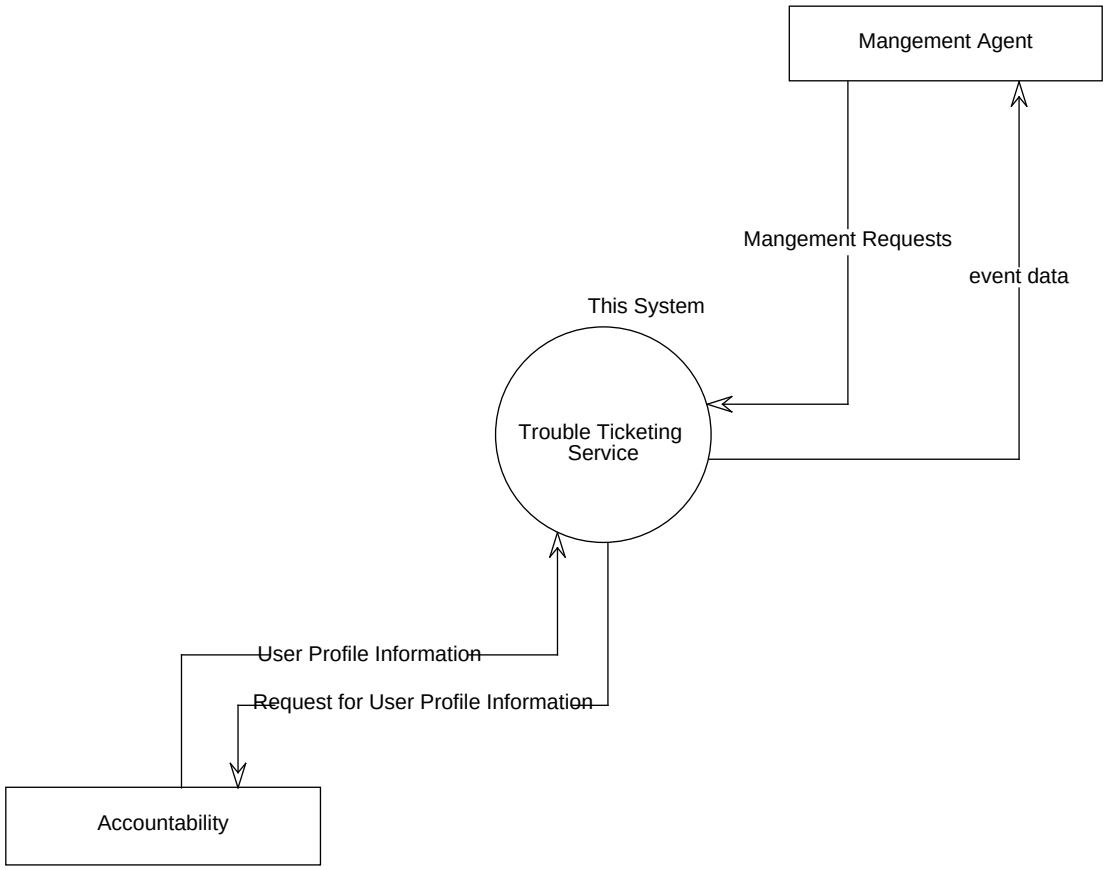


Figure 6.6-1. Trouble Ticketing Context Diagram

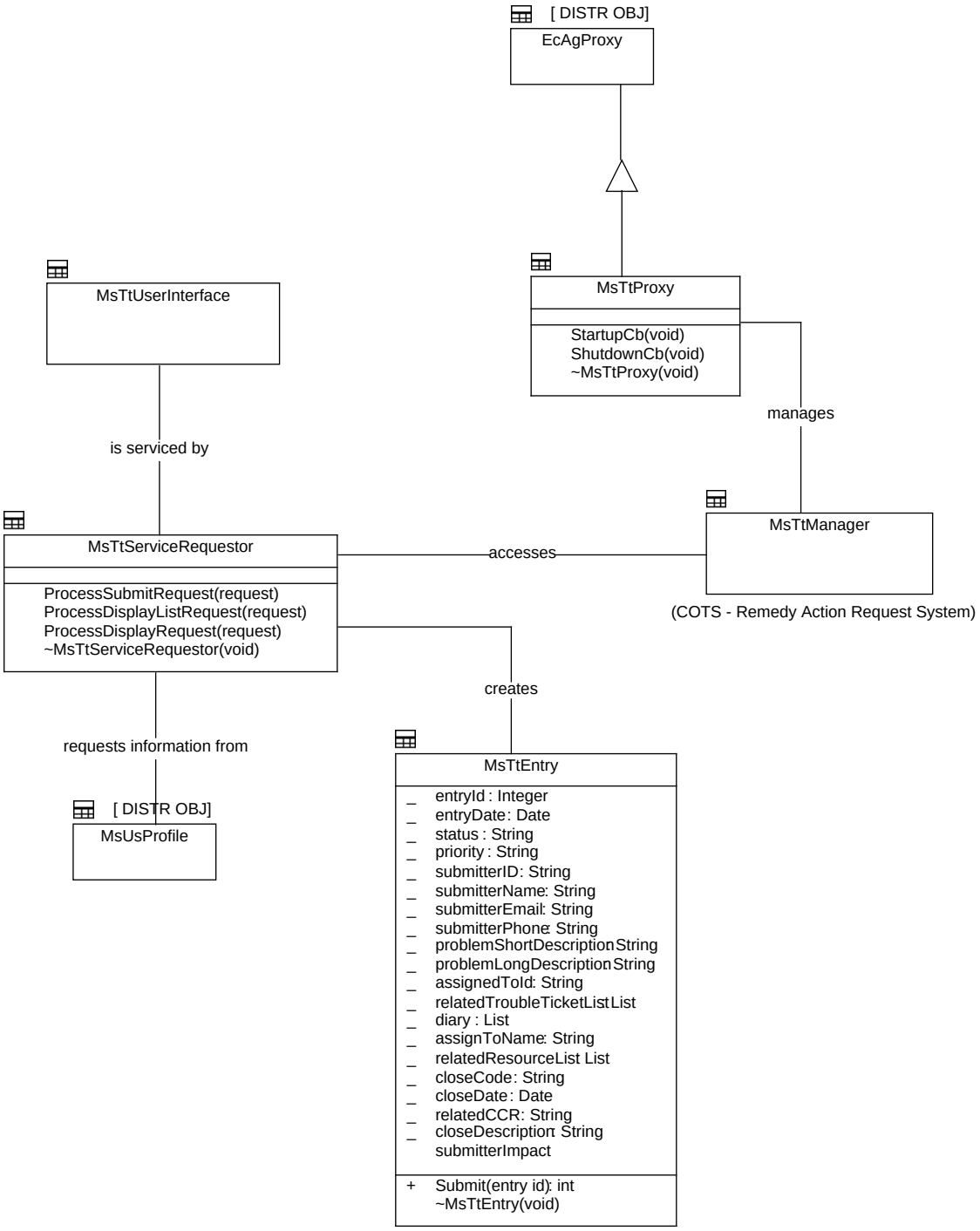


Figure 6.6-2. Trouble Ticketing Object Model

Attributes:

None

Operations:

None

Associations:

The EcAgProxy class has associations with the following classes:
None

6.6.3.2 MsTtEntry Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The MsTtEntry class models a request for action on a particular problem and the subsequent actions performed on it. This class encapsulates the common definition of a trouble ticket configured in the ECS implementation of the Remedy Action Request System

Attributes:

assignToName - the name of the support staff member working this problem
Data Type: String
Privilege: Private
Default Value:

assignedToId - the user id of the support staff member working this problem
Data Type: String
Privilege: Private
Default Value:

closeCode - a code indicating the type of resolution, valid values include but are not limited to: Software Bug Fix Hardware Replaced No Problem Found Sent to CCR Duplicate Trouble Ticket
Data Type: String
Privilege: Private
Default Value:

closeDate - the date the trouble ticket was closed
Data Type: Date

Privilege: Private
Default Value:

closeDescription - a detailed description of the resolution of this problem
Data Type: String
Privilege: Private
Default Value:

diary - entries made by the support staff members which describe the steps taken through the lifecycle of the trouble ticket
Data Type: List
Privilege: Private
Default Value:

entryDate - the entry date into system (submit date)
Data Type: Date
Privilege: Private
Default Value:

entryId - the unique identifier (generated by MsTtManager, unique prefix for each DAAC)
Data Type: Integer
Privilege: Private
Default Value:

priority - the priority assigned by the support staff, valid values are: Low: the system's capabilities are not seriously compromised. Medium: a portion of the system's capabilities have are affected or are non-functional High: a large degree of the system's capabilities are affected or are non-functional
Data Type: String
Privilege: Private
Default Value:

problemLongDescription - a detailed description of the problem
Data Type: String
Privilege: Private
Default Value:

problemShortDescription - a brief (1 line) description of the problem
Data Type: String
Privilege: Private
Default Value:

relatedCCR - a related CCR if it is determined that a CCR is necessary
Data Type: String
Privilege: Private

Default Value:

relatedResourceList - the resource(s) to which this problem is related (hardware and software)
Data Type: List
Privilege: Private
Default Value:

relatedTroubleTicketList - a list of trouble tickets related to this instance
Data Type: List
Privilege: Private
Default Value:

status - the current status, valid values are: New: indicates the trouble ticket has just been submitted by a user Assigned: indicates the trouble ticket has been assigned to a member of the support staff In-Process: indicates a support staff member has received the trouble ticket as in the process of determining or implementing the solution. Closed: indicates the trouble ticket has been closed Reachieved: indicates the trouble ticket has reached archival age (i.e. should no longer be reported on, and is eligible for purge from the system)
Data Type: String
Privilege: Private
Default Value:

submitterEmail - the e-mail address of the submitter
Data Type: String
Privilege: Private
Default Value:

submitterID - the id of the submitter of the trouble ticket
Data Type: String
Privilege: Private
Default Value:

submitterImpact - indicates the impact as seen by the submitter

submitterName - the name of the submitter
Data Type: String
Privilege: Private
Default Value:

submitterPhone - the phone number of the submitter
Data Type: String
Privilege: Private
Default Value:

Operations:

Submit - The method submits a trouble ticket to the MsTtManager (Remedy Action Request System)
Arguments: entry id
Return Type: int
Privilege: Public

```
PDL:MsTtEntry::Submit(entry_id)
{
// initialize the connection with the MsTtManager

ARInitialization()

// submit the trouble ticket

rc = ARCreateEntry(trouble ticket structure, entry_id)

if (rc == 0)
return 0
else
return error

// terminate the connection with the AR System

ARTermination()
}
```

~MsTtEntry - This is the destructor for the MsTtEntry class.
Arguments: void

Associations:

The MsTtEntry class has associations with the following classes:
Class: MsTtServiceRequestor creates

6.6.3.3 MsTtManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The MsTtManager class represents the Remedy Action Request System, a COTS product. This product provides the core functionality for tracking, classifying, and reporting problem occurrence and resolution. Since this class is purely COTS, it will not be described in detail here. For ECS specific configuration information of the product, the reader is

directed to DID 305, section 4.6.5.1, Trouble Ticketing Management Services CSC. For detailed product information, the reader is directed to the Remedy Action Request System documentation set.

Attributes:

None

Operations:

None

Associations:

The MsTtManager class has associations with the following classes:
Class: MsTtServiceRequestor accesses
Class: MsTtProxy manages

6.6.3.4 MsTtProxy Class

Parent Class: EcAgProxy
Public: No Distributed Object: No
Purpose and Description:
The MsTtProxy class provides the interface to the Management Agent Services. It allows the MsTtManager (Remedy) software to be remotely monitored and managed. The methods on this class are the callbacks provided as specific implementation of the MSS lifecycle calls.

Attributes:

All Attributes inherited from parent class

Operations:

ShutdownCb
Arguments: void

StartupCb
Arguments: void

~MsTtProxy - This is the destructor for the MsTtProxy class.
Arguments: void

Associations:

The MsTtProxy class has associations with the following classes:
Class: MsTtManager manages

6.6.3.5 MsTtServiceRequestor Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The MsTtServiceRequestor class is responsible for processing requests from the MsTtUserInterface and fulfilling them using the functionality provided by the MsTtManager (Remedy).

Attributes:

None

Operations:

ProcessDisplayListRequest - This method translates the request for a particular trouble ticket list and submits the request to the manager.
Arguments: request

```
PDL:MsTtServiceRequestor::ProcessDisplayListRequest(request)
{

// translate the request from standard HTML variable/value pairs

do
get first pair
convert plus signs to spaces
convert control characters
look after the equal sign for the value
populate AR request structure
while (not end of stdin request)

// initialize MsTtManager (Remedy) connection -

ARInitialization()

if (rc == 0)

// request the trouble ticket list
```

```
ARCreateGetEntryList()

if (rc == 0)
send list to stdout
else
send error to stdout

// terminate our connection with MsTtManager (Remedy)

ARTermination()

else
send error to stdout
return failure
}
```

ProcessDisplayRequest - This method translates the submit request from the MsTtUserInterface and submits the request to the manager.
Arguments: request

```
PDL:MsTtServiceRequestor::ProcessDisplayRequest(request)
{

// translate the request from standard HTML variable/value pairs

do
get first pair
convert plus signs to spaces
convert control characters
look after the equal sign for the value
populate AR request structure
while (not end of stdin request)

// initialize MsTtManager (Remedy) connection -

ARInitialization()

if (rc == 0)

// request the trouble ticket list

ARCreateGetEntry()

if (rc == 0)
send entry to stdout
```

```
else
send error to stdout

// terminate our connection with MsTtManager (Remedy)

ARTermination()
else

send error to stdout
return failure
}
```

ProcessSubmitRequest - This method processes a request to submit a trouble tickets by translating the request, instantiating a new trouble ticket, and submitting it.
Arguments: request

```
PDL:MsTtServiceRequestor::ProcessSubmitRequest(request)
{

// instantiate a new trouble ticket entry

new TroubleTicket = MsTtEntry

// translate the request from standard HTML variable/value pairs

do
get first pair
convert plus signs to spaces
convert control characters
look after the equal sign for the value

// now set the value in our trouble ticket

newTroubleTicket.SetX = obtained value;

while (not end of stdin request)

rc = newTroubleTicket.Submit(entry_id);

if (rc == 0)
send entry id to stdout
return success
else
send error to stdout
return failure
```

}

~MsTtServiceRequestor - This is the destructor for the MsTtServiceRequestor class.
Arguments: void

Associations:

The MsTtServiceRequestor class has associations with the following classes:
Class: MsTtManager accesses
Class: MsTtEntry creates
Class: MsTtUserInterface isservicedby
Class: MsUsProfile requestsinformationfrom

6.6.3.6 MsTtUserInterface Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
The MsTtUserInterface class represents the set of HTML documents which define the common interface provided to the users in order to create or query the status their trouble tickets.

Attributes:

None

Operations:

None

Associations:

The MsTtUserInterface class has associations with the following classes:
Class: MsTtServiceRequestor isservicedby

6.6.3.7 MsUsProfile Class

Parent Class: Not Applicable
Public: Yes Distributed Object: Yes
Purpose and Description:
This is a public class (and a distributed object) that is exported by this service. Other subsystems use this class in order to retrieve the user profile for a specified user.

Attributes:

None

Operations:

None

Associations:

The MsUsProfile class has associations with the following classes:
Class: MsTtServiceRequestor requestsinformationfrom

6.6.4 Trouble Ticketing Dynamic Model

The following scenarios demonstrate typical TTS interaction as it relates to both users and support staff members.

6.6.4.1 User Submits a Trouble Ticket

This scenario represents the typical sequence of events and interactions for a user submitting a new trouble ticket into the system. The scenario is shown in Figure 6.6-3.

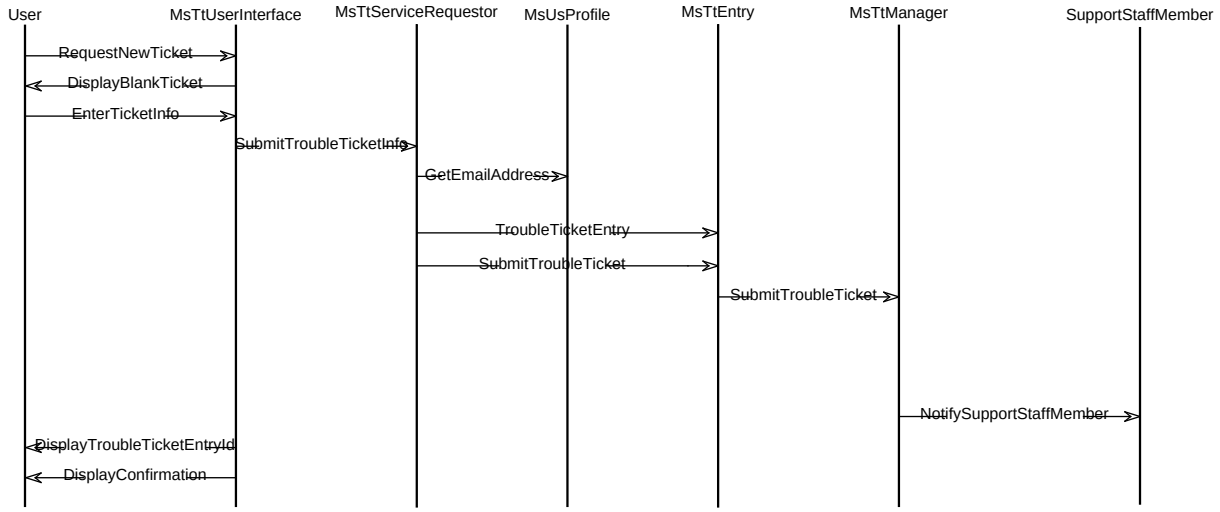


Figure 6.6-3. User Submits Trouble Ticket

6.6.4.1.1 Beginning Assumptions

none

6.6.4.1.2 Interfaces with Other Subsystems and Segments

none

6.6.4.1.3 Stimulus

A user encounters a problem.

6.6.4.1.4 Participating Classes From the Object Model

MsTtUserInterface

MsTtServiceRequestor

MsTtManager

MsTtEntry

6.6.4.1.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is in normal operational state.

6.6.4.1.6 Ending State

A new trouble ticket is created. The MsTtManager is in normal operational state.

6.6.4.1.7 Scenario Description

A user of the system encounters a problem. This problem may be with either hardware or software. The user then invokes the MsTtUserInterface. The user indicates that a new trouble ticket is to be created. The MsTtUserInterface displays a trouble ticket form. Next, the user proceeds to enter into the form, the information pertaining to the particular problem. When complete, the user indicates that the trouble ticket is to be submitted.

The MsTtUserInterface submits the trouble ticket to MsTtServiceRequestor. Retrieving the user detail information, the MsTtServiceRequestor instantiates a trouble ticket and submits it to the MsTtManager for entry into the Remedy System. The manager creates a new trouble ticket with a unique entry id. Additionally, the MsTtManager notifies the support staff member responsible for assigning trouble tickets that a new one has entered the system.

The entry id and confirmation of the successful transaction are returned to the MsTtEntry, MsTtServiceRequestor and finally the MsTtUserInterface. The MsTtUserInterface indicates a successful trouble ticket submission to user.

6.6.4.2 A Trouble Ticket is Worked

This scenario represents the typical sequence of events and interactions for a trouble ticket to be worked after it has entered the system. This scenario is shown in Figure 6.6-4.

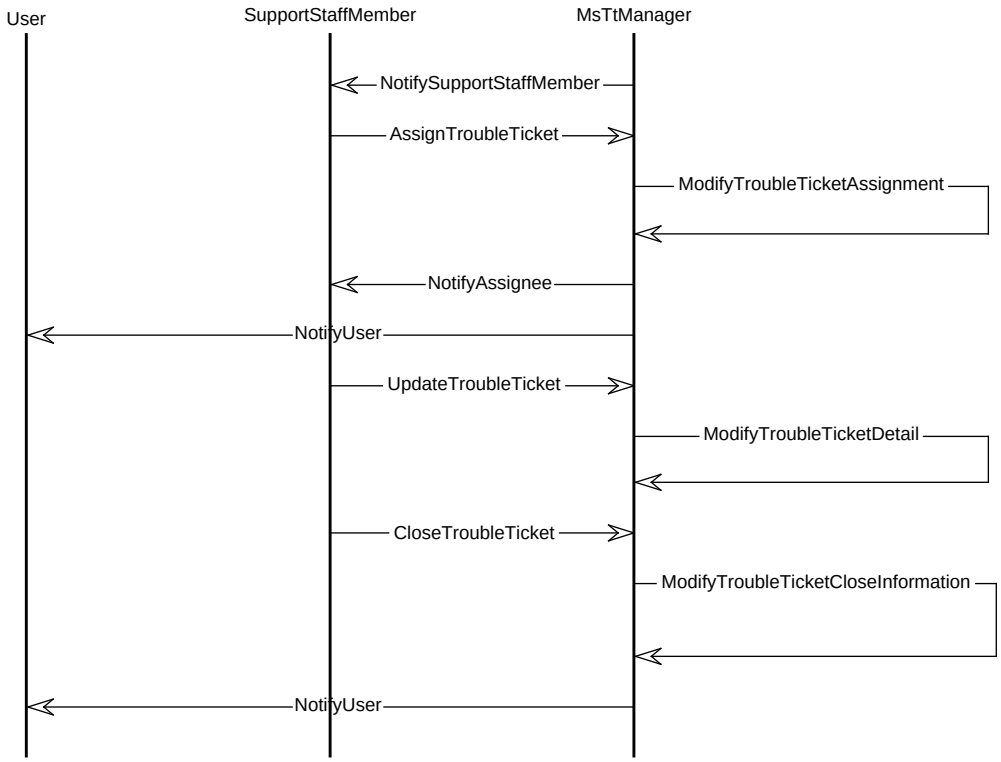


Figure 6.6-4. A Trouble Ticket is Worked

6.6.4.2.1 Beginning Assumptions

none

6.6.4.2.2 Interfaces with Other Subsystems and Segments

none

6.6.4.2.3 Stimulus

A new trouble ticket enters the system.

6.6.4.2.4 Participating Classes From the Object Model

MsTtUserInterface

MsTtManager

6.6.4.2.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is in normal operational state.

6.6.4.2.6 Ending State

A trouble ticket is closed. The MsTtManager is in normal operational state.

6.6.4.2.7 Scenario Description

The support staff member responsible for assigning trouble tickets receives notification that a new trouble ticket has entered the system. On examining the detail information, the support staff member assigns the trouble ticket. The trouble ticket status is then updated to reflect the assignment and the assignee is notified (via e-mail). Additionally, the submitter of the trouble ticket is sent an e-mail indicating that their trouble ticket has been assigned.

Assessing the trouble ticket, the assignee forms a plan to resolve the issue. Adding this information to the trouble ticket, it is then updated, and the status is moved to “In-Progress”. Once again the submitter is sent an e-mail regarding the change in status.

When the issue has been resolved, the assignee updates the trouble ticket with any subsequent information regarding the solution and assigns it a closing code (e.g. Bug-Fix, Hardware-Replaced, etc.). The trouble ticket is modified and the user sent a final e-mail message.

6.6.4.3 A Trouble Ticket is Escalated

This scenario represents the typical sequence of events which occur when a trouble ticket is escalated. The scenario is shown in Figure 6.6-5.

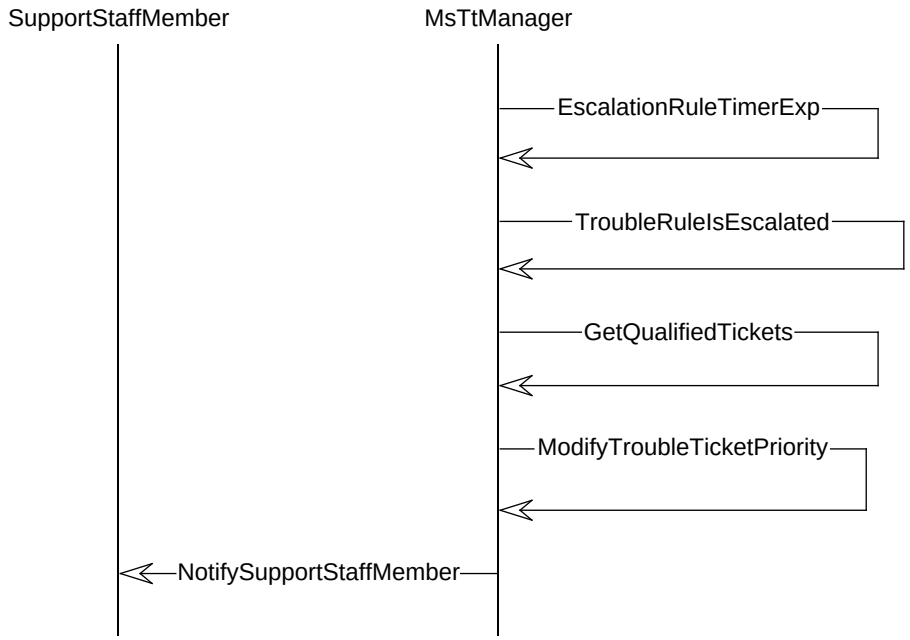


Figure 6.6-5. A Trouble Ticket is Escalated

6.6.4.3.1 Beginning Assumptions

none

6.6.4.3.2 Interfaces with Other Subsystems and Segments

none

6.6.4.3.3 Stimulus

A defined escalation rule is hit for a trouble ticket.

6.6.4.3.4 Participating Classes From the Object Model

MsTtManager

SupportStaffMember

6.6.4.3.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is in normal operational state.

6.6.4.3.6 Ending State

A trouble ticket is escalated. The MsTtManager is in normal operational state.

6.6.4.3.7 Scenario Description

At a specified time interval, an escalation rule is executed. This escalation rule defines a time based rule for taking some action on trouble tickets which meets a particular criteria. An example which would fit this scenario would be a rule to notify a support staff member if a trouble ticket has gone unassigned for more than 24 hours.

For those trouble tickets which qualify, the specified action is taken. This action may include any combination of notifications or updates to the trouble ticket itself. An example could include raising the priority of a trouble ticket if it has not been resolved within a given time period.

6.6.4.4 A Trouble Ticket is Forwarded

This scenario represents the typical sequence of events which occur when a trouble ticket is forwarded from a DAAC to the SMC. The scenario is shown in Figure 6.6-6.

6.6.4.4.1 Beginning Assumptions

none

6.6.4.4.2 Interfaces with Other Subsystems and Segments

none

6.6.4.4.3 Stimulus

A user reports a problem to a DAAC via a trouble ticket.

6.6.4.4.4 Participating Classes From the Object Model

MsTtManager

6.6.4.4.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager at both the DAAC and the SMC is normal operational state.

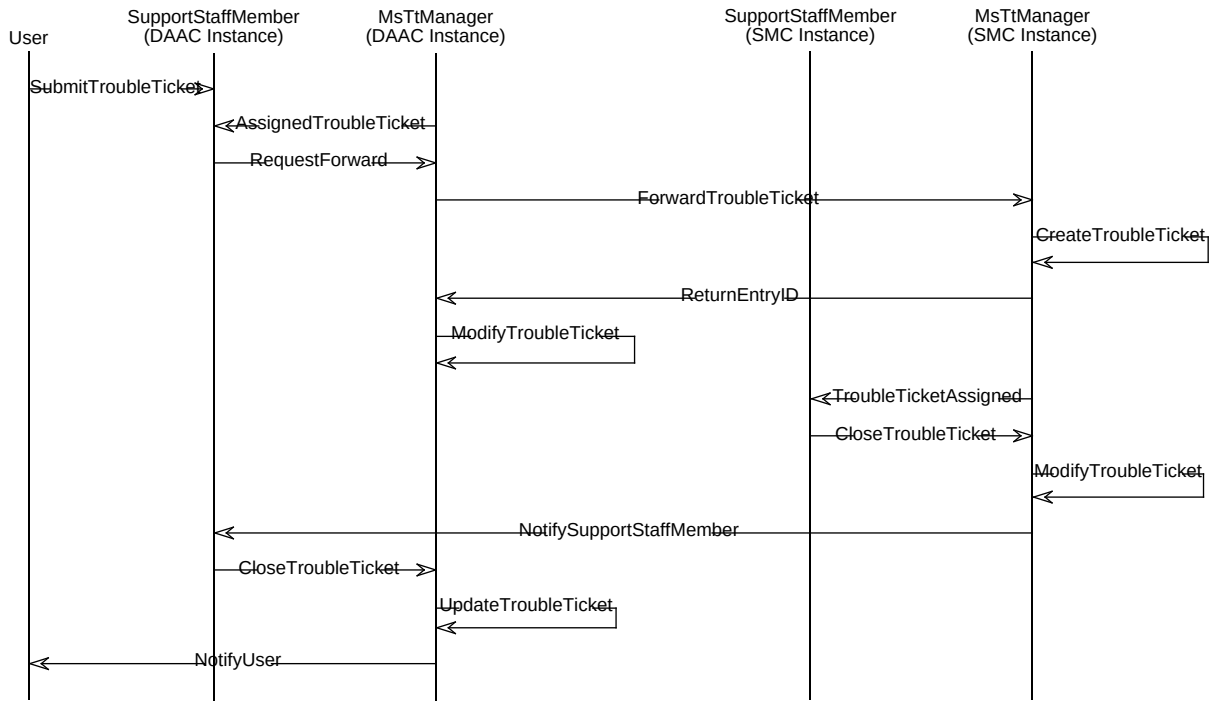


Figure 6.6-6. A Trouble Ticket is Forwarded

6.6.4.4.6 Ending State

The user is notified of closure of the trouble ticket. The MsTtManager is in normal operational state.

6.6.4.4.7 Scenario Description

A user at a DAAC reports a problem. When the support staff member assigned to this trouble ticket examines the detail of the problem, it is traced back to a problem at the SMC (for example a malfunctioning router). The support staff member at the DAAC indicates that the trouble ticket is to be forwarded to the SMC for work.

From here, the MsTtManager will submit a new trouble ticket with the identical problem detail information to the SMC. The submitter information on this trouble ticket will correspond to the support staff member forwarding the trouble ticket. Additionally, the trouble ticket will indicate that it originated from the particular DAAC. After the successful entry of forwarded trouble ticket, the original will be updated to reflect the new, related, trouble ticket entry id. This information allows the support staff member at the originating DAAC to check the status of the new trouble ticket at any time.

To the SMC, the new trouble ticket is assigned, worked, and eventually closed as would any other. When the trouble ticket is closed, the support staff member at the originating DAAC will receive notification, via e-mail, of that event. At this point, the original trouble ticket can be updated and closed. When this ticket is closed, the originating user will be notified of the resolution of their problem.

6.6.5 Trouble Ticketing Structure

Table 6.6-1. Trouble Ticketing Components

Component Name	COTS/Custom
Trouble Ticketing Management Services	COTS (Remedy Action Request System)
Trouble Ticketing User Interface	Custom
Trouble Ticketing Service Requestor	Custom
Trouble Ticketing Management Agent	Custom

6.6.5.1 Trouble Ticketing Management Services CSC

Purpose and Description

This CSC provides the core functionality of the trouble ticketing services. It is implemented by the Remedy Action Request System software package. It allows for the entry, modification, administration, and reporting of trouble tickets.

Mapping to objects implemented by this component

MsTtManager

CSU's

The items listed below are configuration tasks within the Remedy Action Request System.

Define trouble ticket format - product interaction

Define trouble ticket summary information format - product interaction

Define trouble ticket standard forms (e.g. entry, list request, etc.) - product interaction

Define standard reports - product interaction

Define standard escalation rules - product interaction

Define standard active links - product interaction

6.6.5.2 Trouble Ticketing User Interface CSC

Purpose and Description

This CSC provides the interface between the users (those who submit trouble tickets) and the MsTtServiceRequestor. It provides a common interface to submit and query the status of a user's trouble tickets. The functionality listed will be implemented as HTML documents.

Mapping to objects implemented by this component

MsTtUserInterface

CSU's:

Submit Trouble Ticket Form (HTML Document)

Display Trouble Ticket List Form (HTML Document)

Display Trouble Ticket Form (HTML Document)

6.6.5.3 Trouble Ticketing Service Requestor CSC

Purpose and Description

This CSC provides the interface between the MsTtManager (Remedy) and the MsTtUserInterface. It translates requests from the HTTPD server and fulfills them using the MsTtManager API functions.

Mapping to objects implemented by this component

MsTtServiceRequestor

MsTtEntry

CSU's:

ProcessSubmitRequest (C++ code)

ProcessDisplayListRequest (C++ code)

ProcessDisplayRequest (C++ code)

Submit (C++ code)

6.6.5.4 Trouble Ticketing Proxy Agent CSC

Purpose and Description

This CSC provides the interface with the Management Agent Services. It allows for remote administration and monitoring of the Remedy software package.

Mapping to objects implemented by this component

MsTtProxy

CSU's:

Customization of Proxy Agent (C++ code)

6.6.6 Trouble Ticketing Management and Operation

6.6.6.1 System Management Strategy

The Trouble Ticketing Service Management Strategy utilizes the MSS Management Agent model for its administration. The MsTtProxy will allow the remote startup and shutdown of the Trouble Ticketing Service via the Management Framework.

6.6.6.2 Operator Interfaces

The Trouble Ticketing Service provides two user interfaces:

MsTtManager (Remedy Action Request System) - This Motif GUI provides access to all of the functionality supplied by the software package. This functionality includes trouble ticket entry, modification, rule definition, administration and reporting. It is through this interface that support staff members will access TTS.

MsTtUserInterface - This set of HTML documents provide the primary interface for users to submit and query trouble tickets.

6.6.6.3 Reports

Trouble Ticket Status Report - This reports indicates the status of a set of trouble tickets based on a particular criteria (e.g. by date range, assigned-user, status...).

Trouble Ticket Resource Report - This report indicates by resource the number and type of problems encountered by affected resource.

Trouble Ticket User Report - This report indicates by submitter the number and type of trouble tickets in the system.

Trouble Ticket Statistics Report - This report indicates for a a particular criteria statistical information such as mean time to close.

Trouble Ticket Status Report (SMC) - This report provide a summary of the number of tickets by status and priority across all DAACs.

The above reports are meant to be examples of stock reports provided by the Trouble Ticketing Service. TTS allows for both extensive customization of the above reports and creation of new ones. The reporting capabilities include the ability to display not only data contained in the database but also statistical and correlation functions on that data. These custom reports can be defined and saved by individual support staff members or made available globally.

6.7 Management Data Access

6.7.1 Management Data Access Overview

The Management Data Access (MDA) Service is responsible for centralizing, processing and providing access to the information which is logged into the management data log file on each managed host from various sources via the MSS Management Agent Services. This log data includes performance, security, fault, accountability, and other ECS application event information.

One of MDA's primary functions is to centralize the log file data at each DAAC. It accomplishes this by transferring the individual management data log files from each managed host, to the MSS server, on a scheduled basis. The log files are later processed and retained for a period of time before being transferred by MDA to the ECS Data Archives. The schedule which defines when all these events occur is configurable. For each of the ECS hosts' log file, time intervals, absolute time specifications, and or size threshold parameters may be set. Any of these parameters may trigger the transfer of a file to the MSS server. MDA's graphical user interface allows the viewing and updating of all of its schedule configuration parameters.

MDA's graphical user interface also provides the user access to the contents of this log data. Provided a host, time period, and selection filter information, MDA will retrieve the requested data and display it using its log file browser. Once displayed, options are given for sorting, additional filtering, and saving of the data. It should be noted that while this access is typically used to access DAAC (or SMC/EOC) local log files, it also provides the capability to browse the log data located across sites.

For the purposes of longer term analysis and reporting, MDA will process the management data log files, loading selected information to the Management RDBMS. The data MDA loads will be configurable and shall include:

- Counter Metrics - maintains a count of the number of times a particular event occurred over the time period
- Duration Metrics - locates the start and ending time for a particular “transaction” and calculates the duration
- Summation/Average Metrics - maintains a sum or average of a value for a particular event type over the time period.
- Detail Events - for a particular event type, loads the detail of each of these events to the database.

6.7.2 Management Data Access Context

The Management Data Access context diagram is shown in Figure 6.7-1. MDA receives management requests, (e.g. start up, shutdown) from the Management Agent. Once the requests are completed an event is logged through the Management Agent. Additionally, MDA and the Management Agent communicate in order to set the maximum file size for the management log file. MDA utilizes the CSS file transfer mechanism and ECS Data Archives for log file archival.

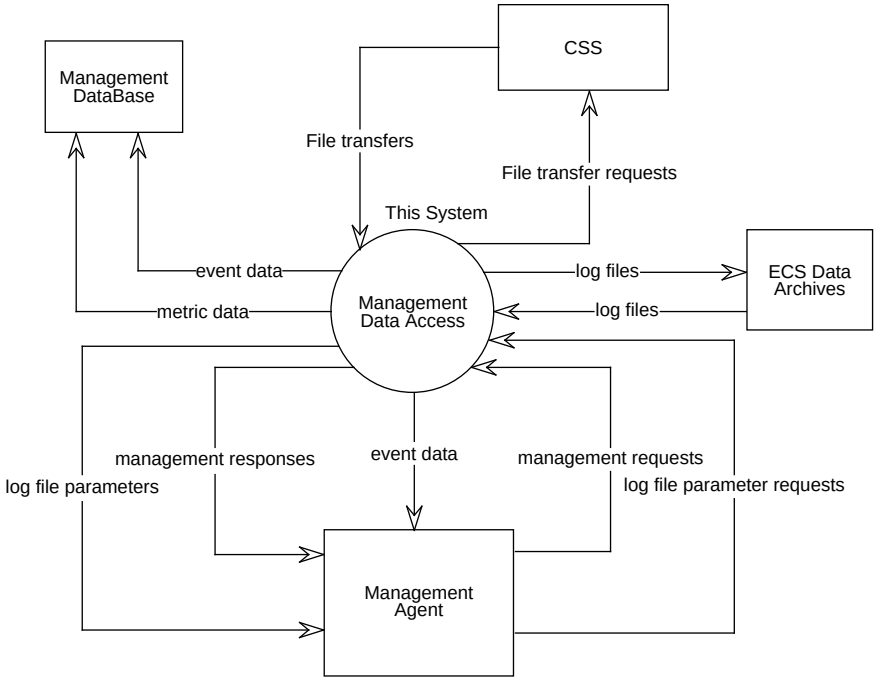


Figure 6.7-1. Management Data Access Context Diagram

6.7.3 Management Data Access Object Model

The Object Model for the Management Data Access Service is shown in Figure 6.7-2.

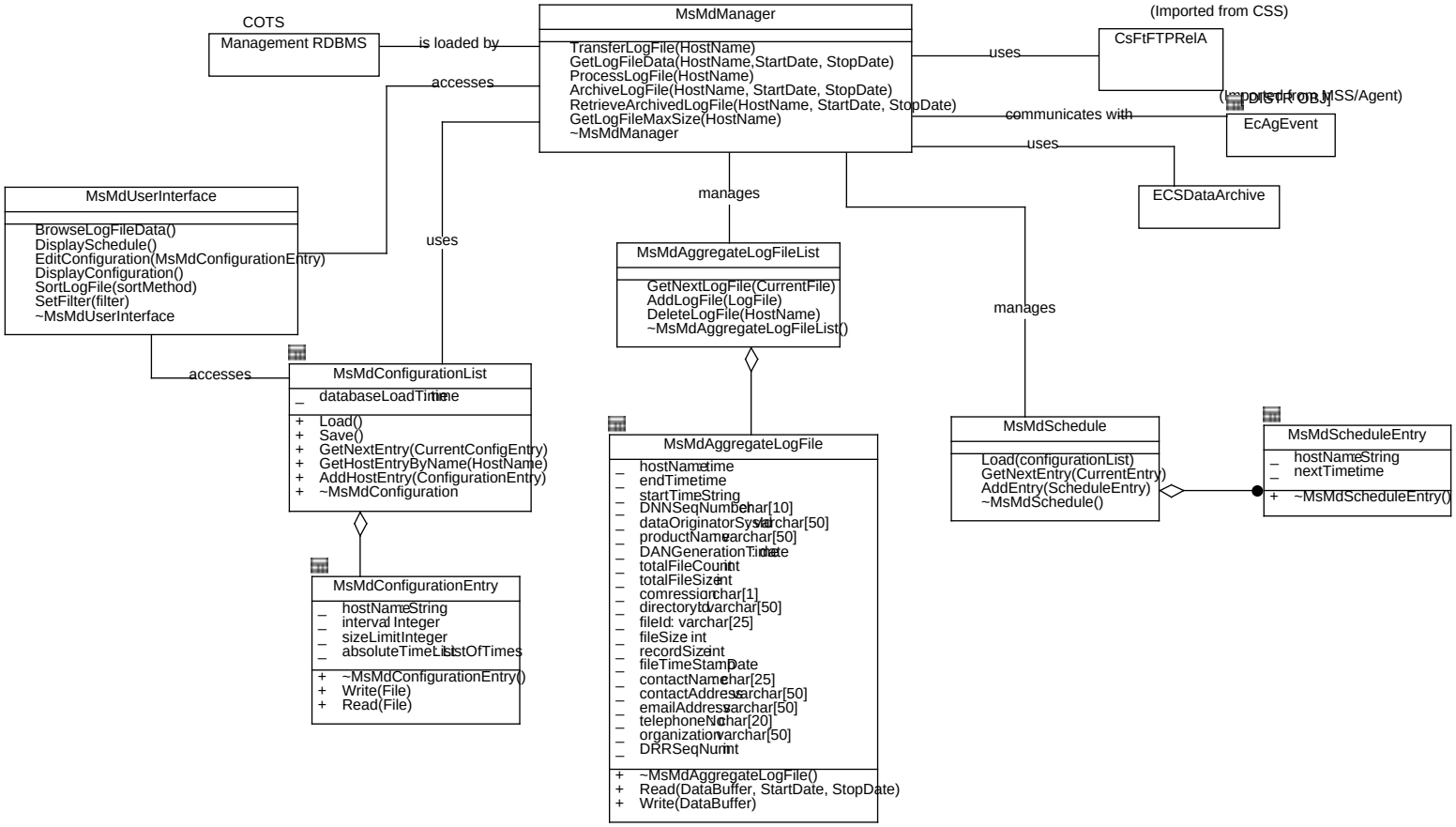


Figure 6.7-2. Management Data Access Object Model

6.7.3.1 CsFtFTPRelA Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class is imported from CSS

Attributes:

None

Operations:

None

Associations:

The CsFtFTPRelA class has associations with the following classes:
Class: MsMdManager uses

6.7.3.2 ECSDataArchive Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the ECS Data Archive facility.

Attributes:

None

Operations:

None

Associations:

The ECSDataArchive class has associations with the following classes:
Class: MsMdManager uses

6.7.3.3 EcAgEvent Class

Parent Class: Not Applicable
Public: No Distributed Object: Yes

Purpose and Description:
The EcAgEvent defines a distributed object. It provides the capability to dispatch events for orderly and prompt resolution should events occur. The SNMP protocol provides the capability to send traps from agent to SNMP manager. But, the traps are not secure and not reliable. The solution to these problems are using DCE RPC as the transport mechanism for security reasons and sending the traps from MSS Server to the management framework locally. The COTS HP OpenView guarantees the delivery of traps local on one host by using IPC as opposed to UDP. The ECS applications, the EcAgProxy agent, and the MsAgMonitor of the MsAgSubagent can send event notifications to the MsAgSubagent. The MsAgSubagent logs every event into MSS log file. Then, if the severity of the event equals to or is higher than the infoLevel variable, it sends this event notification further to the MsAgDeputy on the MSS Server which in turn convert the event to an SNMP trap and send it locally to the management framework.

Attributes:

None

Operations:

None

Associations:

The EcAgEvent class has associations with the following classes:
Class: MsMdManager communicateswith

6.7.3.4 ManagementRDBMS Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the Management Data Relational Database (a COTS package).

Attributes:

None

Operations:

None

Associations:

The ManagementRDBMS class has associations with the following classes:
Class: MsMdManager isloadedby

6.7.3.5 MsMdAggregateLogFile Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents an aggregate of the log file data for a single managed host over a given time period. The data is held in these files until it is process for the Management RDBMS and archived.

Attributes:

DANGenerationTime - The ISO time of DAN generation
Data Type: date
Privilege: Private
Default Value:

DNNSeqNumber - The sequence number of DAN assigned by Data Server and subsequently used by MSS for data retrieval requests.
Data Type: char[10]
Privilege: Private
Default Value:

DRRSeqNum - The Data Retrieval Request number sent to data server when requesting data retrieval from the archives
Data Type: int
Privilege: Private
Default Value:

comression - THe flag to indicate if files have been compressed prior to archival
Data Type: char[1]
Privilege: Private
Default Value:

contactAddress - The address of the contact person
Data Type: varchar[50]
Privilege: Private
Default Value:

contactName - The name of the contact person requesting data archiving
Data Type: char[25]

Privilege: Private
Default Value:

dataOriginatorSysId - Id of Data Originator
Data Type: varchar[50]
Privilege: Private
Default Value:

directoryId - The path and address of the directory where the files are located
Data Type: varchar[50]
Privilege: Private
Default Value:

emailAddress - The e-mail address of the contact person
Data Type: varchar[50]
Privilege: Private
Default Value:

endTime - the ending time of the period covered by the aggregate
Data Type: time
Privilege: Private
Default Value:

fileId - The name of the files that are to be archived (Repeat for each file to be archived)
Data Type: varchar[25]
Privilege: Private
Default Value:

fileSize - The size of the individual file in bytes to be archived
Data Type: int
Privilege: Private
Default Value:

fileTimeStamp - the date and time this file was created
Data Type: Date
Privilege: Private
Default Value:

hostName - the host name for this aggregate
Data Type: time
Privilege: Private
Default Value:

organization - The organization of the contact person
Data Type: varchar[50]

Privilege: Private
Default Value:

productName - Generic names of data files
Data Type: varchar[50]
Privilege: Private
Default Value:

recordSize - The size of record in this file
Data Type: int
Privilege: Private
Default Value:

startTime - the starting time of the period covered by the aggregate
Data Type: String
Privilege: Private
Default Value:

telephoneNo - The telephone number of the contact person
Data Type: char[20]
Privilege: Private
Default Value:

totalFileCount - Total number of files to be archived
Data Type: int
Privilege: Private
Default Value:

totalFileSize - Sum of file lengths in bytes
Data Type: int
Privilege: Private
Default Value:

Operations:

Read - This operation read the aggregate, retrieving data based on a start and stop date
Arguments: DataBuffer, StartDate, StopDate
Return Type: Void
Privilege: Public

```
PDL:MsMdAggregateLogFile::Read ( DataBuffer, StartDate, StopDate )
{
  // skip all entries older than StartDate
  while ( entry = read(file) AND ( entry::Date < StartData ) ) {}
```

```
  // keep reading values until entries are newer than StopDate
  while ( entry = read(file) AND ( entry::Date < StopDate ) )
    store entry in DataBuffer

  return DataBuffer
}
```

Write - This operation writes a data buffer to the aggregate
Arguments: DataBuffer
Return Type: Void
Privilege: Public

```
PDL:MsMdAggregateLogFile::Write ( DataBuffer )
{
  // write value to file
  write DataBuffer to end of file
  update header
}
```

~MsMdAggregateLogFile
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsMdAggregateLogFile class has associations with the following classes:
MsMdAggregateLogFileList (Aggregation)

6.7.3.6 MsMdAggregateLogFileList Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents a list aggregate log files for the hosts at a single DAAC.

Attributes:

None

Operations:

AddLogFile - This operation adds an aggregate log file to the list.
Arguments: LogFile
PDL:MsMdAggregateLogFileList::AddLogFile (LogFile)

```
{
  // add log file to the end of the list
  add at end of list LogFile
}
```

DeleteLogFile - The operation removes and destroys a log file from the list.
Arguments: HostName

```
PDL:MsMdAggregateLogFileList::DeleteLogFile ( LogFile )
{
  // search the list for the log file
  CurrentFile = MsMdAggregateLogFileList::GetNextLogFile ( NULL )
  while ( CurrentFile is legal entry ) {

    // if current log file is the one to be deleted
    if ( CurrentFile == LogFile ) {

 // remove it from the list
 remove CurrentFile from list
 return OK

    }
    CurrentFile = MsMdAggregateLogFileList::GetNextLogFile ( CurrentFile )
  }

  return LOG_FILE_NOT_FOUND
}
```

GetNextLogFile - This operation returns the next aggregate log file in the list.
Arguments: CurrentFile

```
PDL:MsMdAggregateLogFileList::GetNextLogFile ( CurrentFile )
{
  // if CurrentFile is equal to null, then return the first log file in the list
  if ( CurrentFile == NULL )
    return FirstLogFile

  if ( CurrentFile has a next file )
    return CurrentFiles next file
  else
    return NO_MORE_LOG_FILES
}
```

~MsMdAggregateLogFileList
Arguments:

Associations:

The MsMdAggregateLogFileList class has associations with the following classes:
Class: MsMdManager manages

6.7.3.7 MsMdConfigurationEntry Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the configuration parameters for a single managed MSS host logfile.

Attributes:

absoluteTimeList
Data Type: ListOfTimes
Privilege: Private
Default Value:

hostName - the name of the host to which this entry applies
Data Type: String
Privilege: Private
Default Value:

interval - the interval in minutes the log file is to be transferred
Data Type: Integer
Privilege: Private
Default Value:

sizeLimit - the size threshold at which the log file is to be transferred
Data Type: Integer
Privilege: Private
Default Value:

Operations:

Read - This operation reads a single configuration entry to permanent storage.
Arguments: File
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationEntry::Read ( File )
{
  // read values from the file
  HostName = GetToken ( file )
```

```
  Interval = GetToken ( file )
  SizeLimit = GetToken ( file )
  AbsoluteTimeLimit = GetToken ( file )
}
```

Write - This operation writes a single configuration entry to permanent storage.
Arguments: File
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationEntry::Write ( File )
{
  // Write values from the file
  PutToken ( HostName, file )
  Putoken  ( Inteval, file )
  PutToken ( SizeLimit, file )
  PutToken ( AbsoluteTimeLimit, file )
}
```

~MsMdConfigurationEntry
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsMdConfigurationEntry class has associations with the following classes:
MsMdConfigurationList (Aggregation)

6.7.3.8 MsMdConfigurationList Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the configuration of the MsMdManager.

Attributes:

databaseLoadTime - the time of day the aggregate log files should be processed and loaded to the Management
Data Type: time
Privilege: Private

Default Value:

Operations:

AddHostEntry - This operation adds an MsMdHostConfigurationEntry to the current configuration.

Arguments: ConfigurationEntry
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationList::AddHostEntry ( Entry )
{
  // add entry to the linked list
  attach Entry to end of list
}
```

GetHostEntryByName - This operation gets the MsMdHostConfigurationEntry for a give host name.

Arguments: HostName
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationList::GetHostEntryByName ( HostName )
{
  // search each ConfigurationEntry for HostName
  CurrentEntry = MsMdConfigurationList::GetNextEntry ( NULL )
  while ( CurrentEntry is legal entry ) {

    if ( CurrentEntry::HostName == HostName )
 return CurrentEntry

    CurrentEntry = MsMdConfigurationList::GetNextEntry ( CurrentEntry )
  }

  return NO_ENTRY_FOR_SPECIFIED_HOSTNAME
}
```

GetNextEntry - This operation gets the next MsMdHostConfigurationEntry

Arguments: CurrentConfigEntry
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationList::GetNextEntry ( CurrentEntry )
{
  // if CurrentEntry is equal to NULL, then return the first log file in the list
  if ( CurrentEntry == NULL )
```

```
    return first ConfigurationEntry

  if ( CurrentEntry has a next file )
    return CurrentEntries next file
  else
    return NO_MORE_LOG_FILES
}
```

Load - This operation loads the configuration from permanent storage.
Arguments:
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationList::Load ( )
{
  // open configuration file
  file = open ( configuration_file )

  // read all the entries in the configuration file
  while ( NewEntry = new MsMdConfigurationEntry::Read ( file ) ) {

    // add the configuration entry to the list
    MsMdConfigurationList::AddHostEntry ( NewEntry )
  }

  // close the configuration file
  close ( configuration_file )
}
```

Save - This operation writes the configuration to permanent storage.
Arguments:
Return Type: Void
Privilege: Public

```
PDL:MsMdConfigurationList::Save ( )
{
  // open the configuration file
  file = open ( configuration_file )

  // write each configuration entry to disk
  CurrentEntry = MsMdConfigurationList::GetNextEntry ( NULL )
  while ( CurrentEntry is a legal entry ) {

    // write the entry to disk
    MsMdConfigurationEntry::Write ( file )

    // get the next configuration entry
```

```
    CurrentEntry = MsMdConfigurationList::GetNextEntry ( CurrentEntry )
  }

  // close the configuration file
  close ( configuration_file )
}
```

~MsMdConfiguration
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsMdConfigurationList class has associations with the following classes:
Class: MsMdUserInterface accesses
Class: MsMdManager uses

6.7.3.9 MsMdManager Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class encapsulates the functionality required to centralize and process the ECS log files.

Attributes:

None

Operations:

ArchiveLogFile - This operation utilizes the ECS Archival Services to archive an individual MsMdAggregateLogFile.
Arguments: HostName, StartDate, StopDate

```
PDL:MsMdManager::ArchiveLogFile ( HostName, StartDate, StopDate )
{
  // get log file to archive
  LogFile = MsMdConfigurationList::GetHostEntryByName ( HostName )

  // ask the archival services to retrieve the specified log file
  ArchivalServices::ArchiveItem ( LogFile, HostName, StartDate, StopDate )
}
```

GetLogFileData - This operation retrieves and returns specified log file data based on a given filter.
Arguments: HostName,StartDate, StopDate

```
PDL:MsMdManager::GetLogFileData ( HostName, StartDate, EndDate )
{
  // first make sure that dates are legal
  if ( StartDate > EndDate ) OR ( StartDate > CurrentDate ) OR ( EndDate > CurrentData )
    return ERROR_ILLEGAL_DATE_SPECIFIED

  // check to see if request is on local site or remote
  if ( HostName is local ) {

    // find the log file
    CurrentLogFile = MsMdAggregateLogFileList ( NULL )
    while ( CurrentLogFile::HostName != HostName )
      CurrentLogFile = MsMdAggregateLogFileList::GetNext ( CurrentLogFile )

    // check to see if data must be collected from the host
    if ( ( CurrentLogFile::StopTime < StartDate ) OR
         ( CurrentLogFile::StopTime < EndDate ) ) {

      // bring any data in the log file of the specified host to the local host
      MsMdManager::TransferLogFile ( HostName )
    }

    // check to see if some of the data requested has already been archived
    if ( ( CurrentLogFile::StartTime > StartDate ) OR
         ( CurrentLogFile::StartTime > EndDate ) ) {

      // retrieve the data from the archive, and store it temporarily
      ArchivedData = ArchiveServies::GetArchivedData ( HostName, StartDate, EndDate )
    }

    // now that the data has been consolidate, return the data to caller
    if ( data was retrieved from archival services ) {
      return  ArchivedData UNION CurrentLogFile::Read ( StartDate, EndDate )
    }
    else {
      return CurrentLogFile::Read ( StartDate, EndData )
    }
  }
  else {
    bind to remote host HostName
```

```
    // execute remote host procedure
    FileSize = MsMdManager::GetLogFileData ( HostName, StartDate, StopDate )
    if ( FileSize >= ACCEPTABLE_REMOTE_FILE_SIZE ) {

      ::AskUserToContinue ( "File is large, Continue )
      if ( continue cancelled )
        interrupt file call to MsMdManager::GetLogFilData ()
    }
  }
}
```

GetLogFileMaxSize - This operation examines the current configuration for a particular host an returns the file size threshold.
Arguments: HostName

```
PDL:int MsMdManager::GetTimeInterval ( HostName )
{
  // get the MsMdConfigurationEntry for the specified host, and return it's current size limit
  return MsMdConfiguration::GetHostByMame(HostName)::SizeLimit
}
```

ProcessLogFile - This operation processes an individual MsMdAggregateLogFile and loads accumulated summary data to the Management RDBMS.
Arguments: HostName

```
PDL:MsMdManager::ProcessLogFile ( HostName )
{
  // get the log file to process
  LogFile = MsMdAggregateLogFileList::GetNextLogFile ( NULL )
  while ( LogFile is legal entry ) {
    if ( LogFile::HostName == HostName )
 break;
  }

  // instruct the Management RDBMS to process the LogFile
  ManagementRDBMS::Ingest ( LogFile::Read ( ) )

  // delete the log file
  MMsMdAggregateLogFileList::DeleteLogFile ( HostName )
}
```

RetrieveArchivedLogFile - This operation utilizes the ECS Archival Services to retrieve an individual archived MsMdAggregateLogFile.
Arguments: HostName, StartDate, StopDate

```
PDL:MsMdManager::RetrieveArchivedLogFile ( HostName, StartDate, StopDate )
{
  // ask the archival services for the specified log
  LogFile = ArchivalServices::RetrieveItem ( HostName, StartDate, StopDate )

  return LogFile
}
```

TransferLogFile - This operation transfers a log file from a managed host to the MSS server, appending it to the correct MsMdAggregateLogFile.
Arguments: HostName

```
PDL:MsMdManager::TransferLogFile ( HostName )
{
  // if log file already exists for HostName, just append
  LogFile = MsMdAggregateLogFileList::GetNextLogFile ( NULL )
  while ( LogFile is a legal value ) {

    // break loop when hostname found
    if ( LogFile::HostName == HostName )
      break
  }

  // copy the remote log file
  if ( hostname found )
    LogFile::Write ( CsFtFTPRelA::TransferFile ( HostName + ":/MSS/MSSLOGFILE" ) )
  else {
    LogFile = new MsMdAggregateLogFile ( CsFtFTPRelA::TransferFile ( HostName + ":/
MSS/MSSLOGFILE" ) )
    MsMdAggregateLogFileList::AddLogFile ( LogFile )
  }
}
```

~MsMdManager
Arguments:

Associations:

The MsMdManager class has associations with the following classes:

Class: MsMdUserInterface accesses
Class: EcAgEvent communicateswith
Class: ManagementRDBMS isloadedby
Class: MsMdAggregateLogFileList manages

Class: MsMdSchedule manages
Class: CsFtFTPRelA uses
Class: ECSDataArchive uses
Class: MsMdConfigurationList uses

6.7.3.10 MsMdSchedule Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the current schedule for MSS logfile transfers.

Attributes:

None

Operations:

AddEntry - This operation adds a scheduled entry to the schedule.
Arguments: ScheduleEntry

```
PDL:MsMdSchedule::AddEntry ( NewScheduleEntry )
{
  // add schedule entry to list
  attach NewScheduleEntry to end of list
}
```

GetNextEntry - This operation returns the next scheduled entry in the schedule.
Arguments: CurrentEntry

```
PDL:MsMdSchedule::GetNextEntry ( CurrentEntry )
{
  // if CurrentEntry is equal to NULL, then return the first log file in the list
  if ( CurrentEntry == NULL )
    return first ScheduleEntry

  if ( CurrentEntry has a next file )
    return CurrentEntries next file
  else
    return NO_MORE_SCHEDULE_ENTRIES
}
```

Load
Arguments: configurationList

~MsMdSchedule
Arguments:

Associations:

The MsMdSchedule class has associations with the following classes:
Class: MsMdManager manages

6.7.3.11 MsMdScheduleEntry Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:
This class represents the schedule entry for a single managed MSS host logfile.

Attributes:

hostName - the host name to which this schedule entry applies
Data Type: String
Privilege: Private
Default Value:

nextTime - the next scheduled time for this log file to be transferred
Data Type: time
Privilege: Private
Default Value:

Operations:

~MsMdScheduleEntry
Arguments:
Return Type: Void
Privilege: Public

Associations:

The MsMdScheduleEntry class has associations with the following classes:
MsMdSchedule (Aggregation)

6.7.3.12 MsMdUserInterface Class

Parent Class: Not Applicable
Public: No Distributed Object: No
Purpose and Description:

This class represents the user interface to the Management Data Access Services. From this interface, MSS logfile data can be browsed, sorted, and filtered. Additionally this interface provides the functionality to update the MDA configuration parameters.

Attributes:

None

Operations:

BrowseLogFileData - This allows a user to retrieve and browse data based on the current filter.

Arguments:

```
PDL:MsMdUserInferface::BrowseLogFileData ( )
{
  // obtain values from user interface
  Host = ::GetHostName
  StartDate = ::GetStartDate
  StopDate = ::GetStopDate

  // display data to user screen
  ::Display ( MsMdManager::GetLogFileData ( HostName, StartDate, StopData )
}
```

DisplayConfiguration - This method provides the ability to edit the current configuration parameters for MDA.

Arguments:

```
PDL:MsMdUserInterface::DisplayConfiguration ( )
{
  // get each Configuration item
  ConfigItem = MsMdConfigurationList::GetNextEntry ( NULL )
  while ( ConfigItem is a legal item ) {

    // display the configuration item to the screen
    ::Display ( ConfigItem )

    ConfigItem = MsMdConfigurationList::GetNextEntry ( ConfigItem )
  }

  // if the user wants to modify a configuration item, then call
  on modify: call MsMdUserInterface::EditConfiguration ( SelectedConfiguration )
}
```

DisplaySchedule - This method displays the current schedule for log file transfers.

Arguments:

```
PDL:MsMdUSerInferface::DisplaySchedule ( )
{
  // retrieve all the schedule entries
  ScheduleEntry = MsMdSchedule::GetNextEntry ( NULL )
  while ( ScheduleEntry is legal entry ) {

    // display the schedule entry
    ::DisplayScheduleEntry ( ScheduleEntry )

    ScheduleEntry = MsMdSchedule::GetNextEntry ( ScheduleEntry )
  }

}
```

EditConfiguration
Arguments: MsMdConfigurationEntry

SetFilter
Arguments: filter

SortLogFile
Arguments: sortMethod

~MsMdUserInterface
Arguments:

Associations:

The MsMdUserInterface class has associations with the following classes:
Class: MsMdConfigurationList accesses
Class: MsMdManager accesses

6.7.4 Management Data Access Dynamic Model

6.7.4.1 User Browses Log File Data

6.7.4.1.1 Beginning Assumptions

None.

6.7.4.1.2 Interfaces with Other Subsystems and Segments

CSS (via CsFtFTPRelA)

Archival Services

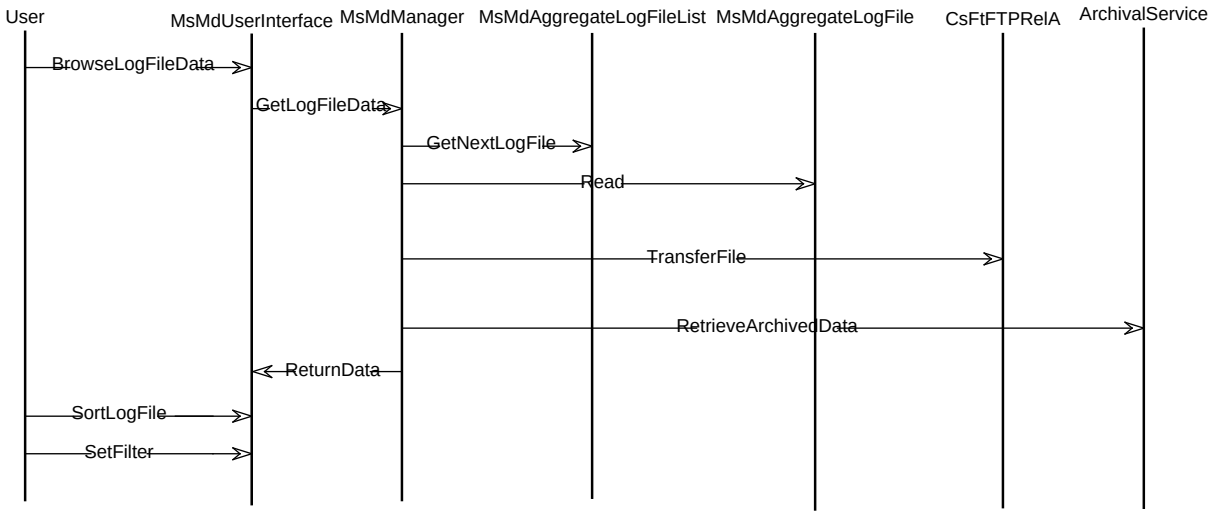


Figure 6.7-3. User Browses Log File Data

6.7.4.1.3 Stimulus

The user requests to browse log file data from the MDA user interface.

6.7.4.1.4 Participating Classes From the Object Model

MsMdUserInterface

MsMdManager

MsMdAggregateLogFileList

MsMdAggregateLogFile

CsFtFTPRelA

ECS Data Archives

6.7.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.7.4.1.6 Ending State

The user is provided the log file browser and the requested data.

6.7.4.1.7 Scenario Description

From the MDA user interface, a user requests to browse log file data. This request specifies a time period, host name and filter. The MsMdManager examines the request criteria to calculate the location of the data. In this scenario, the time period specified is assumed to be large enough to require that data be extracted from: the ECS Data Archives, MSS server, and a managed host. It should be noted that this situation is unlikely, and is demonstrated in this scenario purely to illustrate the method for retrieving data from all sources.

To retrieve the data from archive, a request is made to the ECS Data Archives. When the data has been returned, the required filter is applied to the data. Next the data from the managed host is transferred to the MSS server. This data is appended to the MsMdAggregateLogFile for the particular host. Finally, the same MsMdAggregateLogFile is read and the proper data filtered out. The union of this data and that retrieved from the archives is returned to the MDA user interface.

At this point the interface allows the user to request additional sorting and filtering on the retrieved data.

6.7.5 Management Data Access Structure

Table 6.7-1. Management Data Access Components

Component Name	COTS/Custom
Management Data Access Services	Custom
Management Data Access User Interface	Custom

6.7.5.1 Management Data Access Services CSC

Purpose and Description

The Management Data Access Services CSC provides the ability to centralize and retrieve management log file data. Additionally it is responsible for accumulating metrics and loading them to the Management RDBMS.

Mapping to objects implemented by this component

MsMdManager (C++ code)

MsMdConfiguration (C++ code)

MsMdConfigurationEntry (C++ code)

MsMdSchedule (C++ code)

MsMdAggregateLogFileList (C++ code)

MsMdAggregateLogFile (C++ code)

6.7.5.2 Management Data Access User Interface CSC

Purpose and Description

The Management Data Access User Interface CSC provides the user interface functionality which allow the users to configure MDA and to browse, sort, and filter log file data.

Mapping to objects implemented by this component

MsMdUserInterface (C++ code / X-Windows design)

6.7.6 Management Data Access Management and Operation

6.7.6.1 System Management Strategy

The Management Data Access Service will utilize the MSS Management framework for reporting fault and performance data, as well as lifecycle services. The EcAgManager class will be imported in order to provide this functionality. Example event data this application will log includes:

- start/end of an MDA user interface
- start/end of a browse request
- start/end of a scheduled log file transfer
- MDA Service faults (transfer errors, DCE errors, etc.)

6.7.6.2 Operator Interfaces

The Management Data Access Service will provide the MsMdUserInterface class as its graphical user interface. This interface will allow log file data to be browsed, filtered, sorted, and saved. Additionally, this interface will provide the ability to view and change the configuration of MDA parameters including scheduling, processing and database load times.

6.7.6.3 Reports

The Management Data Access Service does not provide reports.

6.8 Ground Events Planning Service

6.8.1 Ground Events Planning Service Overview

The Ground Events Planning Service is a complete reuse of the Production Planning Workbench that is part of the Production Planning CSCI provided by the Planning and Data Processing Subsystem (PDPS). The Production Planning Workbench is used to prepare a schedule for the production at a site, and forecast the start and completion times of the activities within the schedule. The functions provided by the workbench include creating a candidate plan from production requests, activating a candidate plan, updating the active plan, and cancelling or modifying the active plan.

The Production Planning Workbench provides an interface to submit operations ground events such as production, maintenance, testing, etc. It develops optimum resource utilization plans and schedules based upon approved system configurations and priorities. The “process” for using the Production Planning Workbench is as follows:

- A ground event activity is planned and submitted to the resource manager for resources to be scheduled.
- The resource planner enters the ground event data into the Production Planning Workbench via the system display. This data includes ground event type, description of resources required, start time of ground event implementation, and duration. The operator has the capability to create a new daily, weekly, or monthly site resource schedule or to modify an existing schedule.

- After the required data is entered, the operator submits the data to the Production Planning Workbench.
- The Production Planning Workbench analyzes the requests, schedules the appropriate resources, and presents the candidate integrated schedule to the resource manager.
- Adjudication of schedules is handled locally for sites. The SMC adjudicates for any inter-DAAC problems.
- After adjudication, the final site resource schedule is distributed to the site operators, external users, and the SMC. The SMC enters site schedules into the system resource scheduler in a similar fashion.

The Production Planning Workbench is used by each site to plan ground events for all site resources, including FOS resources at the EOC. Planning ground events at the EOC will be very similar to planning DAAC ground events with the exception that there will not be an electronic interface such as that provided between MSS and PDPS for planning at the DAAC. FOS ground resources are typically dedicated to specific functions, with redundant resources for critical elements, and does not entail the type of real time allocations as is required for PDPS. The FOS planning and scheduling function supports flight operations rather than ground resource scheduling. The flight operations schedules will be used by the Ground Event Planner as input to the planning of resources to support ground events. In addition the ground event schedule and allocated resources will be available to EOC operations staff and is expected to be closely coordinated during the planning process. The design of the Planning Workbench will support multiple users and therefore can be used by the EOC operations staff to allocate resources to specific missions.

6.8.2 Production Planning Workbench Context

6.8.3 Production Planning Workbench Object Model

The Production Planning Workbench Object Model, Figure 6.8-1, indicates the classes in the Production Planning Workbench, the associations among them, their attributes and their operations. The central classes are the P1PlanningWorkbenchUI, P1Plan, P1ResourceManager, P1ResorceRequirement, P1Activity, P1PGEActivity, P1GroundActivity, P1GroundEvent, and P1PublishedPlan.

A full description of the object model and all its classes is provided in the PDPS CDR document.

6.8.3.1 P1PlanningWorkbenchUI Class

This class is an abstraction for the user interface to the production planning workbench application.

6.8.3.2 P1Plan

This class represents an abstraction for a production plan.

6.8.3.3 P1ResourceManager

This class represents an abstraction of the resource management capabilities used when generating a plan, describing the operations required to match resource requirements of an activity to the available resources, and allocating the resource for the activity.

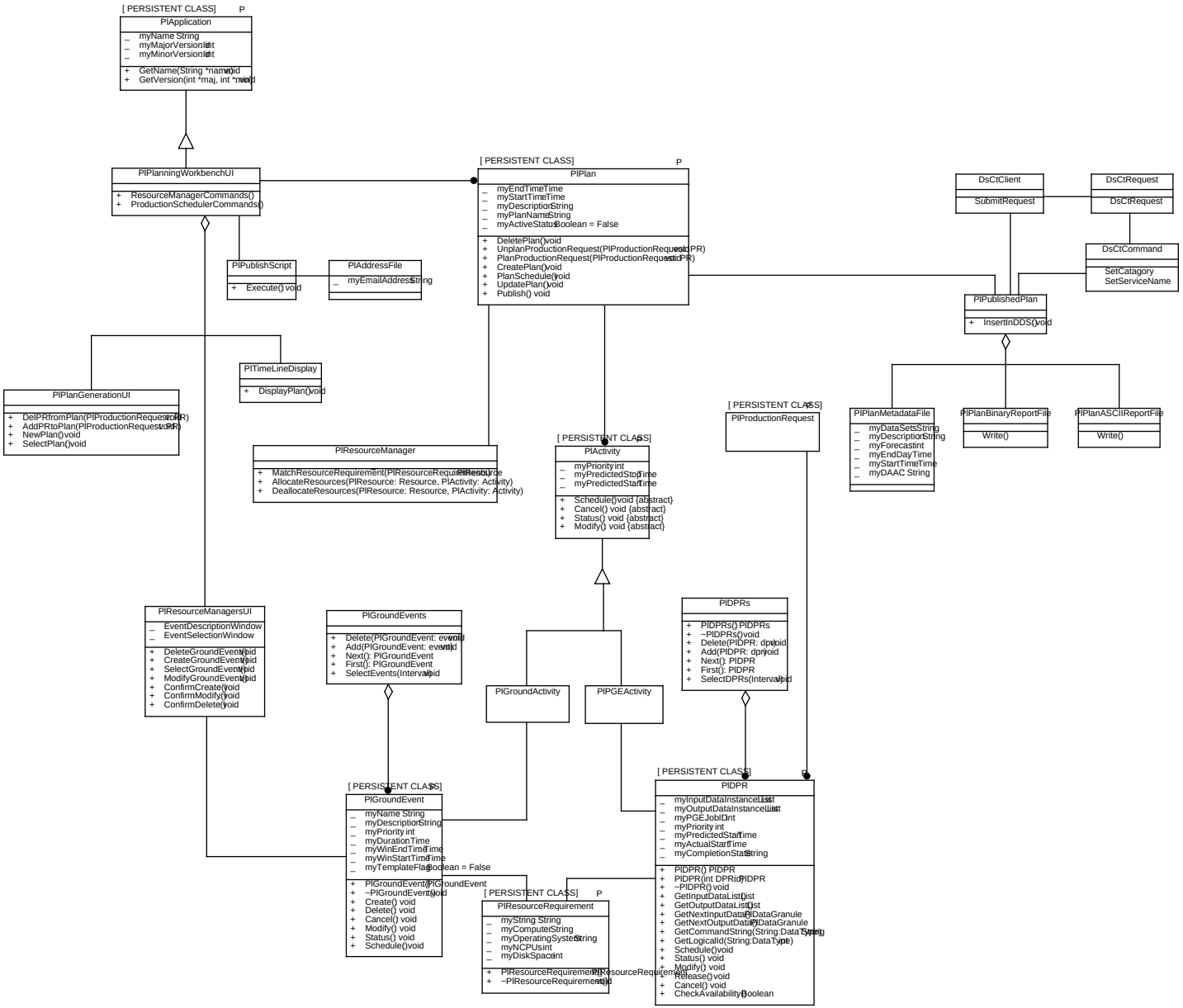


Figure 6.8-1. Production Planning Workbench Object Model

6.8.3.4 P1ResourceRequirement

This class describes the resource requirements for a task.

6.8.3.5 P1Activity

This class describes an item within a plan.

6.8.3.6 P1PGEActivity

This class is a specialization of the P1Activity class. The class describes a data processing request within the plan.

6.8.3.7 P1GroundActivity

This class is a generalization of the P1Activity class. The class describes a ground event within the plan.

6.8.3.8 P1GroundEvent

This class describes a ground event.

6.8.3.9 P1PublishedPlan

This class encapsulates the methods required to insert externalized formats of the plan into the document data server.

6.8.4 Production Planning Workbench Dynamic Model

The following scenarios demonstrate typical Production Planning Workbench functions relating to resource planning.

6.8.4.1 Ground Event Scenario

This scenario describes the system response to a resource manager adding/modifying/deleting a ground event. This scenario is depicted in Figure 6.8-2.

6.8.4.1.1 Beginning Assumptions

None.

6.8.4.1.2 Interfaces with Other Subsystems and Segments

None.

6.8.4.1.3 Stimulus

The resource manager initiates the Production Planning Workbench in order to enter a ground event.

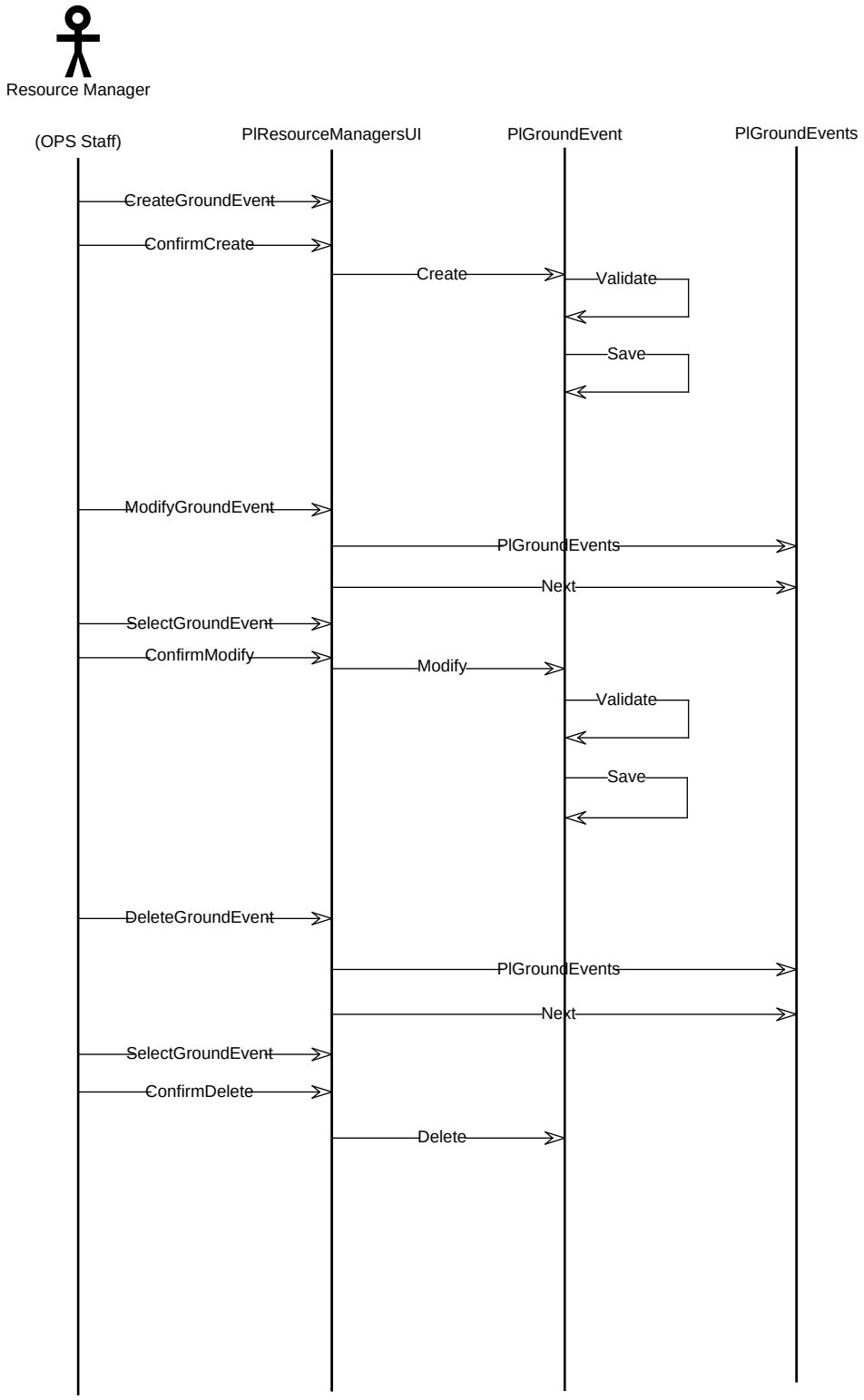


Figure 6.8-2. Ground Event Dynamic Model

6.8.4.1.4 Participating Classes From the Object Model

The following are participating classes from the object model:

- P1PlanningWorkbenchUI
- P1ResourceManagersUI
- P1GroundEvents
- P1GroundEvent

6.8.4.1.5 Beginning System, Segment and Subsystem State(s)

All systems are operating normally.

6.8.4.1.6 Ending State

All systems are operating normally.

6.8.4.1.7 Scenario Description

The user starts the Production Planning Workbench utility.

Addition of a ground event:

- The user is presented with a function to add a new event. This initiates a form window in which the ground event can be described. The user populates the fields within the form with information such as event description, window of opportunity, duration, resource requirements, etc.
- The user confirms the description is complete and the ground event is created and recorded.
- The timeline display of resources will be updated to show the allocation.

Modification of a ground event:

- The user is presented with a function to modify an event. This initiates a window that provides a list of the current defined events.
- The user selects a ground event from the list.
- The user modifies the description and confirms the modification of the ground event.
- The timeline display of resources will be updated to show the allocation.

Deletion of a ground event:

- The user is presented with a function to delete an event. This initiates a window that provides a list of the current define events.
- The user selects a ground event from the list.
- The user confirms the deletion of the ground event
- The timeline display of resources will be updated to show the allocation

6.8.4.2 Plan Creation Scenario

This scenario describes the creation of a plan within the Production Planning Workbench. This scenario is depicted in Figure 6.8-3.

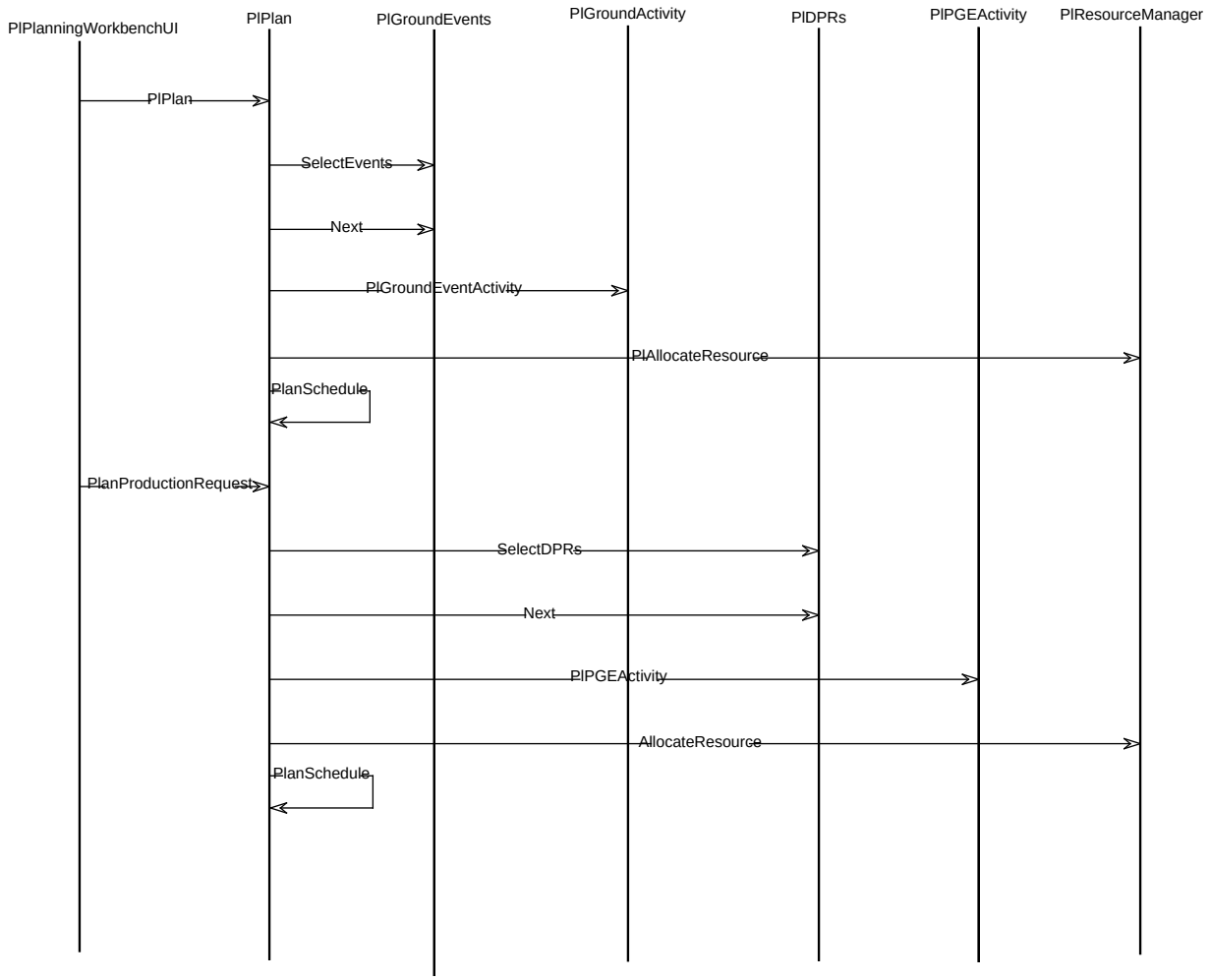


Figure 6.8-3. Plan Creation Dynamic Model

6.8.4.2.1 Beginning Assumptions

None.

6.8.4.2.2 Interfaces with Other Subsystems and Segments

None.

6.8.4.2.3 Stimulus

The resource manager initiates the Production Planning Workbench in order to generate a new plan.

6.8.4.2.4 Participating Classes From the Object Model

The following are participating classes from the object model:

- P1PlanningWorkbenchUI

- P1Plan
- P1GroundEvents
- P1GroundEvent
- P1GroundActivity
- P1DPRs
- P1DPR
- P1PGEActivity
- P1ResourceManager

6.8.4.2.5 Beginning System, Segment and Subsystem State(s)

All systems are operating normally.

6.8.4.2.6 Ending State

All systems are operating normally.

6.8.4.2.7 Scenario Description

- The user starts the Production Planning Workbench utility. The user specifies the time period for which to generate a plan and initiates the creation of a plan object.
- The plan object determines which ground events are defined within the duration of the planning time period.
- The plan object iterates through the ordered list of ground events, creating an activity to fulfill each event.
- The activity is allocated to the appropriate resources by the resource manager to set out a timeline of when the resources are unavailable for production.
- The user then specifies the production requests to be included in the plan
- The plan object iterates through the ordered list of data processing requests, creating an activity.
- The activity is allocated to the appropriate resources by the resource manager.
- The schedule is set up with predicted start and stop times of all the activities.

6.8.5 Production Planning Workbench Structure

The Ground Events Planning Service consists of the Production Planning Workbench, which is COTS.

6.8.5.1 Production Planning Workbench CSC

Purpose and Description

The application is used to prepare a schedule for the production at a site, and forecast the start and completion times of the activities within the schedule. These functions provided by the workbench include the following:

- Candidate Plan Creation

- Plan Activation
- Updating the Active Plan
- Canceling or Modifying the Active Plan

A full description of the Production Planning Workbench CSC is provided in the PDPS CDR document.

6.8.6 Production Planning Management and Operation

A key consideration is that operations personnel will not be required to routinely insert individual data processing requests. That is, once the instructions for processing a type of data have been entered, the error-prone, repetitive entry of data processing instructions for each data product is avoided.

Data processing will be data driven. Plans are constructed based upon the data that is expected to arrive from external sources or based upon the intent to reprocess existing data with revised algorithms. As data arrives, the system will be made aware of the existence of the data and processing will begin when all necessary data files have arrived and are staged.

6.8.6.1 System Management Strategy

A primary consideration is that operator interactions are simplified and automated as much as possible. In addition, it provides operations with the needed flexibility to respond to unexpected tasks as they arise.

A full description of the management and operation and system management strategy is provided in the PDPS CDR document.

6.8.6.2 Operator Interfaces

The operator interfaces will be developed from a combination of COTS graphical user interface (GUI) libraries and custom code.

A full description of the operator interfaces is provided in the PDPS CDR document.

6.8.6.3 Reports

A variety of ad-hoc and canned reports are available to the DAAC operations staff to assist in the monitoring of the activities associated with the Production Planning CSCI. The canned reports include the following:

- Candidate Plan Characteristics Report - This report will be associated with each generated candidate plan and will contain summary information to be used to establish the quality of the produced candidate plan.
- Production and Data Processing Request Status Reports - These reports will contain request status information.
- Planning Workload and Processing Turn-Around Reports - These reports will provide tracking information on planned versus actual processing results.
- Planning Management Reports - These reports will provide the operations staff information on application software events that have occurred.

A full description of all the available reports is provided in the PDPS CDR document.

6.9 Management DBMS

6.9.1 Overview

A COTS Database Management System (DBMS), for which Sybase has been selected, provides a central repository for all ECS management data. This repository is used by management applications to share management data. Additionally, from this central repository the M&O Staff will perform ad hoc statistical analysis and generate ad hoc reports to satisfy the ECS report generation requirements. A COTS Report Generation product is integrated with the DBMS.

6.9.2 Implementation

The Management DBMS architecture is a client/server paradigm and is a COTS Relational DBMS with an integrated report writer. The programming interface to the Management DBMS is provided by the vendor to create, modify, and update the management data tables and fields. This programming interface is Simple Query Language (SQL)-2 compliant. The human/programming interface to the Report Generator is provided by the vendor to generate standard and/or ad hoc reports. The human interface is Motif compliant.

7. MSS HWCI

The MSS Hardware CI (MSS-HCI) is the hardware to host all MSS software described in Sections 3 through 6. The MSS-DCHCI logically includes an enterprise monitoring server, a local management server, and a management workstation. The selection of these configurations are the culmination of requirements analysis, trades analysis, and analysis with specific focus on performance, RMA, security and evolvability considerations. *Note that the rationale for the sizing of the MSS HWCI, which is specific to each DAAC, is in the DAAC-specific portions of this document.*

7.1 MSS-HCI Design Drivers

Major design drivers for management subsystem hardware include the number of managed objects, size and frequency of data to be collected about managed objects, data distribution (between the DAACs and the SMC), data retention policy and archiving design, and COTS choices. In addition, Table 7.1-1 lists the Level 3 requirements that specifically drive the MSS hardware design:

Table 7.1-1. MSS Hardware Requirements Drivers

Requirement Number	Summary	MSS H/W Impact
EOSD 0030	Archive of EOS and related non-EOS data and products	Sizing assumptions of management data holdings
EOSD 3200	Minimum one backup in separate physical location for ECS software and key data items	MSS database partitioning/replication schemes
EOSD 3700	ECS functions Ao of 0.96 (.998 design goal) and MDT of four (4) hours (1.5 hour design goal)	Overall RMA design of hardware strings
EOSD 4030	SMC function Ao of .998 (.999998 design goal) and MDT of 20 minutes (5 minutes design goal	Overall MSS design of EMC and LSM
SMC-0300	100 percent growth in SMC processing speed without modifications or upgrades to software	Sizing and scaleability
SMC-0310	100 percent growth in SMC storage capacity without modifications or upgrades to software	Sizing and scaleability

7.1.1 Key Trade-off studies and Prototypes

Table 7.1-2 lists trades and prototypes that have provided input to the MSS hardware design:

Table 7.1-2. Key Trades and Prototypes

Trade	Conclusion / Result	Impact
Management Data Archive Trade (540-TP-001-001)	Use data server archive for long-term storage of MSS data	Eliminates need for separate large MSS storage.
Common DBMS Server (540-TP-001-001)	Offloading DBMS server functionality to the MSS workstations may be desirable, especially at Release B.	Size the CPU of the MSS workstations to provide DBMS server functionality (assume data stores remain on MSS server).
Management Agents / Protocol Selection (540-TP-001-001)	SNMP protocol will be used, with MIB-II and extensible agents	Performance and sizing model inputs
Selection of Management Framework Product (540-TP-001-001)	HP Openview will provide the management framework	Performance and sizing model inputs

7.1.2 Sizing and performance analysis

This section discusses the method of sizing the MSS hardware at the DAACs and the SMC. Specific sizing calculations, on a per-site basis, are included in the GSFC, MSFC, LaRC, and EDC design documents. The MSS HWCI configuration includes both the MSS and CSS server software (a redundant configuration to enable warm backup); however, this section only describes analysis of the MSS requirements. The complete configuration of the MSS and CSS HWCIs, based on the combined requirements of the subsystems, are presented in the site-specific documents. Additional detail on the analysis of the CSS HWCI sizing and performance is contained in the CSS document.

7.1.2.1 Processing Analysis

The processing requirements for the MSS HWCI are driven by the following types of transactions:

- HP Openview data collection, displays, and ad hoc queries
- Conversion / import of HP Openview and log file data to MSS Sybase DBMS
- DBMS usage for report generation / ad hoc queries
- Usage for configuration management, baseline management, trouble-ticketing, and associated report generation

HP Openview data collection transactions were calculated, based on the number of managed objects, the number of MIB attributes to be monitored, and the frequency of data collection. Based on an estimate provided by HP of approximately 1K instructions per monitoring transaction, the number of MIPs required for an HP Openview data collection was estimated. Estimates of HP Openview requirements for display, query, and reporting functions were based on experience in the EDF (EP3 and EP4), and HP estimates of host requirements. The total estimate of processing capacity and RAM required does not vary significantly by DAAC, since data collection represents only a part of the processing requirement.

Conversion and import of HP Openview and log file data to the MSS DBMS was based on the amount of data to be imported and the amount of intermediate processing required for each data item. The amount of processing of each log file and HP Openview record was based on the number of lines of code in the MSS Management Data Access component responsible for data conversion, and an estimate of Sybase update complexity. The number of log files to be processed was based on the number of system transactions throughout the DAAC. Pull transactions were based on user model estimates of the number of directory, inventory, browse, and other service requests per DAAC. An average of twenty 128 to 256 byte log entries is assumed per pull transaction, including log entries generated by CIDM and data server. Log entries will also be generated by push transactions, including PGEs (estimated 2 entries per PGE), archive requests (estimated 4 entries per request), and external data transfers (estimated 4 entries per transfer). A worst-cast assumption was made that all data collected on a regular basis will be imported to the DBMS, and that 10% of the data requires additional summarization.

Use of Sybase for ad hoc queries, analysis and report generation will exert a moderate (but sporadic) load on the MSS server. The estimate of Sybase client and server requirements was based on vendor recommendations and analysis of DBMS benchmarks in the EDF.

M&O usage of CM tools, the trouble-ticketing system, and utilities (word processor, spreadsheet, graphics) are expected to exert a minor load on the MSS hardware. Estimates were based on vendor-provided information and experience in the use of these products (e.g., Clearcase) in the EDF.

MSS HWCI requirements were then combined with those of the CSS DCHCI to develop an aggregate profile for SMC and LSM servers. Based on the combined requirements of CSS and MSS, medium to high end servers will be used at both the DAACs and the SMC. Processing requirements were analyzed for both Release A and Release B estimates of the number of managed objects. The choice of processors was based on the need to handle Release B loads.

7.1.2.2 Storage Analysis

Major CSMS datastores were analyzed for all sites in the Release A timeframe. MSS data stores analyzed included

- HP Openview internal data storage
- Log files (collected from monitored processes)
- Sybase database (derived from data in HP Openview and log files)
- Clearcase data storage, for ECS software, configuration files, and test data
- Internal data storage for other MSS COTS, including the change request manager, the baseline manager, and the trouble ticketing system.

HP Openview data storage was based on the number of managed objects, the number and size of MIB attributes to be monitored, HP Openview storage overhead, the frequency of data collection for each attribute, and an assumption of 14 days of rolling storage at a time.

Required storage for log files was based on the number of distributed logs to be collected (based on the number of active processes at each DAAC) and the amount of data in each (based on the number of system operations that cause logs to be created). Data maintained in log files addresses needs for performance, fault, accountability and security. Transient processes (e.g., in the

production subsystem) are assumed to provide a minimum of two log entries of 128 bytes each, to indicate the initiation and completion of the process. Non-transient processes provide a 128 byte log entry each time they process a new request (e.g., a file archive request). In addition, major system processes are monitored (polled) at 15 minute intervals, with each polling request generating approximately 512 bytes of data.

A gross estimate of the Sybase database was developed using a worst-cast assumption that all data collected on a regular (i.e., with determined frequency, vs. ad hoc) basis is kept in the DBMS, with an additional 10% for summarized fields and data to be maintained in more than one table (e.g, data relevant to multiple MSS services). The estimate assumed that the DBMS would maintain one month of data at one time, in order to be able to examine medium term trends affecting the DAAC.

Storage required for Clearcase managed data at each DAAC was based on estimates of ECS source custom lines of code, algorithm source lines of code, executables, configuration files and scripts, and test data.

Other COTS products are expected to require a minor amount of storage at each DAAC and the SMC. An estimate of the storage required each product was based on vendor specifications for record sizing and estimates of the number of records required for each (e.g., for the change request manager, an estimate of the number of CCRs per month, DRs per month, and NCRs per month).

In addition, CSS DCE datastores (DCE directory and security information bases) will be replicated on the MSS server to ensure high availability of CSS DCE services. Estimates for these stores were added to the MSS requirements to determine the overall disk sizing.

7.1.3 Scalability and Evolvability to Release B

At Release B, several sizing parameters increase. Increased number of user requests (PGEs, and file archive requests all affect the number of log entries that are collected by the MSS server. An increase in the overall number of managed objects (a worst case assumption of four times the Release A number of managed objects) will increase the amount of HP Openview data collection. Increases in the number of log entries and the amount of HP Openview data collection drive the amount of processing needed to import data to the MSS DBMS, and also increase the volume of network traffic over the DAAC LANs.

CPU and RAM for the CSS and MSS servers were sized to accommodate Release B processing requirements at all DAACs and the SMC. RAID storage was sized based on Release A worst case estimates.

Sizing of network loads through Release B indicates that FDDI will be adequate at the SMC and the DAACs.

7.2 HWCI Structure

The MSS-MHCI contains the following components:

- Enterprise Monitoring Server (SMC)
- Local System Management Servers (MSFC, LaRC, GSFC, EDC DAACs)
- Management Workstations (all locations)
- Printers (all locations)

7.2.1 Connectivity

Each DAAC's LSM will reside on a separate FDDI ring, with connectivity to the rest of the DAAC provided by a redundant FDDI switch/router. The MSS Local Management Server will be equipped with a DAS (dual-attached station) card that is connected to two FDDI concentrators, providing redundancy in the event of a concentrator failure. The MSS monitoring workstations will be equipped with an SAS (single-attached station) interface card, connected to a single FDDI concentrator.

The SMC will reside on a separate FDDI ring at GSFC. Both the Enterprise Communications Server and Enterprise Management Server will be connected to dual FDDI concentrators via DAS interfaces; management workstations and printers will be connected to a single FDDI concentrator via an SAS interface.

Overall network connectivity at each site is discussed in the Release A Overview Design Specification (305-CD-004-001); specific configurations for each site are presented in the site-specific documents.

7.2.2 HWCI Components

The MSS-MHCI monitoring / management server, provided at each DAAC and the SMC, is the primary server for MSS applications and data. It is cross-strapped to the CSS-DCCI communications server (see Figure 7.2-1) to provide for failover (warm standby) capability, and is populated with the MSS-MCI monitoring configuration, the MSS-MLCI logistics monitoring configuration, the MSS-MACI system management agent configuration, and the CSS-DCCI client and server configurations. The MSS-MCI monitoring configuration includes management applications (e.g., HP Openview, Clearcase), common management services (e.g., Sybase, wordprocessing, and spreadsheet packages), and managed object template services software.

The MSS-MHCI management workstation configurations are networked workstations at every ECS site that support all aspects of enterprise management between the M&O staff and the SMC/ LSMs. The MSS-MHCI management workstations are populated with the CSS-DCCI client configuration, the MSS-MACI systems management agent configuration, and user-selected subsets of the MSS-MCI enterprise monitoring configuration software and data.

The MSS-MHCI printers are printers at each site for general administrative purposes, and a specialized printer (e.g., impact) at the SMC to print passwords through sealed envelopes for password administration purposes.

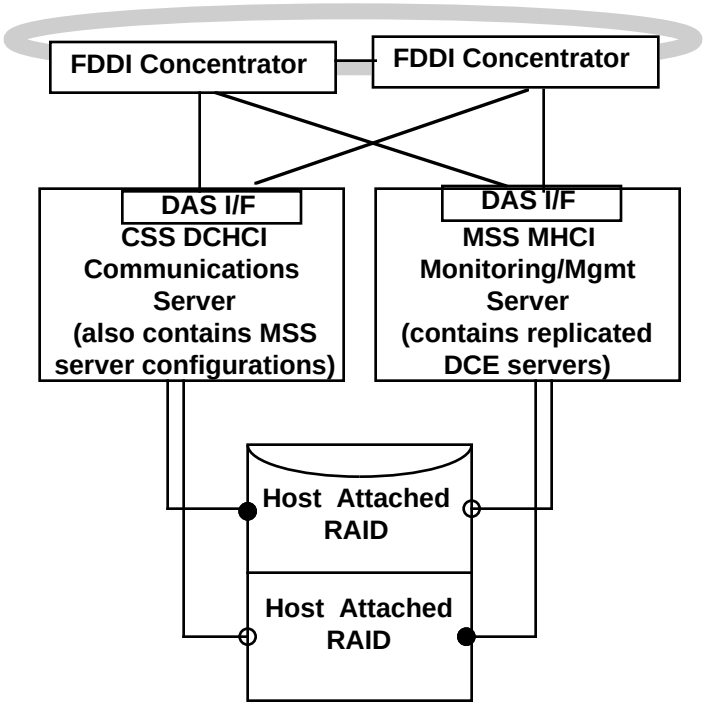


Figure 7.2-1. MSS / CSS Redundant Server Configuration

7.2.3 Failover and Recovery strategy

Analysis of failover strategies supports the integration of the CSS and MSS servers to serve as warm standby to each other, cross-strapped to RAID devices for critical data access by either server. The DCE logical server functions will be replicated and active on the MSS server. MSS logical server functions will be configured but inactive on the CSS server. In the event of a failure of either server, the second RAID can be mounted for use by the backup server. All data is replicated, and is also routinely safestored in the ECS data server archive.

The LSM is designed to continue to function in the event of an EMC failure, and agents at hosts will continue to monitor managed objects in the event of an LSM failure. Dual attached FDDI within the local DAAC LAN designs for critical RMA links.

Specific calculations of reliability and availability of MSS components are provided in 516-CD-001-003, Reliability Predictions for the ECS Project (August 1995), and 518-CD-001-003, Maintainability Predictions for the ECS Project (August 1995).

Appendix A. Requirements Trace

The Interim Release 1 (Ir1) and TRMM Development (Release A) Level 4 requirements listed in the following table reflect the state of the RTM database on July 15, 1995.

Table A-1. Requirements Trace (Page 1 of 45)

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI	Comments
C-HRD-11000	The Enterprise Monitoring Server shall be physically and functionally identical to the Enterprise Communications Server in supporting the CSMS requirements.	MHCI	Section 7
C-HRD-11005	The Enterprise Monitoring Server shall share data with the Local System Management Server in supporting the CSMS requirements.	MHCI	7
C-HRD-11010	The Enterprise Monitoring Server shall preserve DAAC autonomy of operations.	MHCI	7
C-HRD-11015	The Enterprise Monitoring Server shall host the MSS software configuration items to create, with the Enterprise Communications Server and Management Workstations, an enterprise monitoring and coordination center for the ECS.	MHCI	7
C-HRD-11100	The Enterprise Monitoring Server processor shall include a dedicated terminal to be used as a local systems operations console.	MHCI	7
C-HRD-11105	The Enterprise Monitoring Server processor shall be capable of expansion with additional quantities and types of peripherals.	MHCI	7
C-HRD-11110	The Enterprise Monitoring Server processor shall be upgradeable/replaceable within the same product family without major software modification or replacement of any peripheral or attached component.	MHCI	7
C-HRD-11115	The Enterprise Monitoring Server processor shall have the capability to support a POSIX compliant IEEE 1003.1 operating system (UNIX).	MHCI	7
C-HRD-11120	The Enterprise Monitoring Server processor terminal shall be compatible with the Management Workstation display device.	MHCI	7
C-HRD-11300	The Enterprise Monitoring Server data storage shall be compatible with POSIX compliant operating systems from several vendors.	MHCI	7
C-HRD-11310	The Enterprise Monitoring Server data storage shall be compatible with the Local System Management Server short-term data storage.	MHCI	7

Table A-1. Requirements Trace (Page 2 of 45)

C-HRD-11315	The Enterprise Monitoring Server data storage shall support RAID level-5: striping with interleaved parity.	MHCI	7
C-HRD-11320	The Enterprise Monitoring Server data storage shall have the following hot swappable components: a. Disks b. Power Supplies c. Fans d. Disk-array controllers	MHCI	7
C-HRD-11325	The Enterprise Monitoring Server data storage shall be cross-strapped with the Enterprise Communications Server data storage in supporting the CSMS requirements.	MHCI	7
C-HRD-11335	The Enterprise Monitoring Server data storage shall be capable of archiving data to the ECS data server archive for data archive.	MHCI	7
C-HRD-11345	The Enterprise Monitoring Server data archive shall adhere to ECS data server archival requirements for data storage and retrieval.	MHCI	7
C-HRD-11505	The Enterprise Monitoring Server peripheral disk drives shall be capable of retrieving data stored from both the enterprise monitoring server data storage and data archive.	MHCI	7
C-HRD-11530	The Enterprise Monitoring Server peripherals shall support at least one tape drive.	MHCI	7
C-HRD-11535	The Enterprise Monitoring Server peripheral tape drive shall have the following characteristics: a. 4mm Digital Audio Tape format b. Accept industry standard magnetic 4mm DAT (i.e. DDS-90) c. Data transfer rate of 200KB/sec	MHCI	7
C-HRD-11540	The Enterprise Monitoring Server tape drives shall be upgradeable/replaceable within the same product family.	MHCI	7
C-HRD-11565	The Enterprise Monitoring Server peripherals shall support at least one CD-ROM drive.	MHCI	7
C-HRD-11570	The Enterprise Monitoring Server peripheral CD-ROM drive shall have the following characteristic: a. Accept 600MB Compact Disk	MHCI	7
C-HRD-11575	The Enterprise Monitoring Server peripheral CD-ROM drives shall be upgradeable/ replaceable within the same product family.	MHCI	7
C-HRD-12000	The Local Management Server shall be physically and functionally identical to the Local Communications Server in supporting the CSMS requirements.	MHCI	7

Table A-1. Requirements Trace (Page 3 of 45)

C-HRD-12005	The Local Management Server shall share data with the Enterprise Monitoring Server in supporting the CSMS requirements.	MHCI	7
C-HRD-12010	The Local Management Server shall manage only the local DAAC and preserve other DAAC autonomy of operations.	MHCI	7
C-HRD-12015	The Local Management Server shall host the MSS software configuration items to create, with the Local Communications Server and Management Workstations, a local system management center for each ECS DAAC.	MHCI	7
C-HRD-12100	The Local Management Server processor shall include a dedicated terminal to be used as a local systems operations console.	MHCI	7
C-HRD-12105	The Local Management Server processor shall be capable of expansion with additional quantities and types of peripherals.	MHCI	7
C-HRD-12110	The Local Management Server processor shall be upgradeable/replaceable within the same product family without major software modification or replacement of any peripheral or attached component.	MHCI	7
C-HRD-12115	The Local Management Server processor shall have the capability to support a POSIX compliant IEEE 1003.1 operating system (UNIX).	MHCI	7
C-HRD-12120	The Local Management Server processor terminal shall be compatible with the Management Workstation display device.	MHCI	7
C-HRD-12300	The Local Management Server data storage shall be compatible with POSIX compliant operating systems from several vendors.	MHCI	7
C-HRD-12310	The Local Management Server data storage shall be compatible with the Enterprise Monitoring Server intermediate-term data storage.	MHCI	7
C-HRD-12315	The Local Management Server data storage shall support RAID level-5: striping with interleaved parity.	MHCI	7
C-HRD-12320	The Local Management Server data storage shall have the following hot swappable components: a. Disks b. Power Supplies c. Fans d. Disk-array controllers	MHCI	7
C-HRD-12325	The Local Management Server data storage shall be cross-strapped with the Local Communications Server short-term data storage in supporting the CSMS requirements.	MHCI	7

Table A-1. Requirements Trace (Page 4 of 45)

C-HRD-12335	The Local Management Server data storage shall be capable of archiving data to the ECS Data Server archive for data archive.	MHCI	7
C-HRD-12345	The Local Management Server data archive shall adhere to ECS Data Server archival requirements for data storage and retrieval.	MHCI	7
C-HRD-12505	The Local Management Server peripheral disk drives shall be capable of retrieving data stored from both the Local Management server data storage data archive.	MHCI	7
C-HRD-12530	The Local Management Server peripherals shall support at least one tape drive.	MHCI	7
C-HRD-12535	The Local Management Server peripheral tape drive shall have the following characteristics: a. 4mm Digital Audio Tape format b. Accept industry standard magnetic 4mm DAT (i.e. DDS-90) c. Data transfer rate of 200KB/sec	MHCI	7
C-HRD-12540	The Local Management Server tape drives shall be upgradeable/replaceable within the same product family.	MHCI	7
C-HRD-12565	The Local Management Server peripherals shall support at least one CD-ROM drive.	MHCI	7
C-HRD-12570	The Local Management Server peripheral CD-ROM drive shall have the following characteristic: a. Accept 600MB Compact Disk	MHCI	7
C-HRD-12575	The Local Management Server peripheral CD-ROM drives shall be upgradeable/ replaceable within the same product family.	MHCI	7
C-HRD-13000	All Management Workstations and processors shall be capable of operating simultaneously and independently of other workstations and management/ communications servers.	MHCI	7
C-HRD-13100	At a minimum, each processor shall have the capability to support a POSIX compliant IEEE 1003.1 operating system (UNIX).	MHCI	7
C-HRD-13105	Each Management Workstation shall provide one QWERTY keyboard which shall: a. Be detachable and cabled for movement on a desk-top style workstation area b. Provide a minimum of 12 programmable function keys	MHCI	7

Table A-1. Requirements Trace (Page 5 of 45)

C-HRD-13110	Each Management Workstation shall provide one color text and graphics display device which shall: a. Display the complete ASCII character set b. Provide a minimum of 1024 pixel x 864 lines resolution display c. Display a minimum of 16 colors d. Display pages 24 lines by 80 characters wide e. Display a minimum of four screen display pages f. Display pages readable from any location along the width of the workstation and up to a distance of 6 feet from the screen g. Provide a minimum of 19 inches diagonal non-glare screen h. Provide RGB video output for hard copy	MHCI	7
C-HRD-13115	The Management Workstation shall provide one cursor pointing device (mouse).	MHCI	7
C-HRD-13120	The Management Workstation shall be upgradeable/replaceable within the same product family.	MHCI	7
C-HRD-13300	The Management Workstation data storage shall be capable of retrieving data from the data storage function of both the Enterprise Monitoring Server and the Local Management Server.	MHCI	7
C-HRD-13505	All Management Workstation disk drives serving a specific function (e.g. local management, enterprise monitoring) shall be identical and will have equal capacity.	MHCI	7
C-HRD-13900	Each Printer shall be physically and functionally identical in supporting the CSMS printing requirements.	MHCI	7
C-HRD-16000	The Enterprise Monitoring Server shall be capable of 100 percent growth in Appendix A processing speed without modifications or upgrades to software.	MHCI	7
C-HRD-16005	The Enterprise Monitoring Server shall be capable of 100 percent growth in Appendix A storage capacity without modifications or upgrades to software.	MHCI	7
C-HRD-16010	The Local Management Server shall be capable of 100 percent growth in Appendix A processing speed without modifications or upgrades to software.	MHCI	7
C-HRD-16015	The Local Management Server shall be capable of 100 percent growth in Appendix A storage capacity without modifications or upgrades to software.	MHCI	7

Table A-1. Requirements Trace (Page 6 of 45)

C-HRD-16020	The Enterprise Monitoring Server shall be capable of meeting the capacity and performance characteristics of Appendix A.	MHCI	7
C-HRD-16025	The Local Management Server shall be capable of meeting the capacity and performance characteristics of Appendix A for all DAAC configurations.	MHCI	7
C-HRD-16030	The Management Workstation shall be capable of meeting the capacity and performance characteristics of Appendix A.	MHCI	7
C-HRD-17000	The MSS-MHCI hardware selection criteria shall meet overall ECS security policies and system requirements.	MHCI	7
C-HRD-18000	The MSS-MHCI Enterprise Monitoring Server shall maintain one backup of all software and key data items in a separate physical location.	MHCI	7
C-HRD-18005	The MSS-MHCI Local Management Server shall maintain one backup of all software and key data items in a separate physical location.	MHCI	7
C-HRD-18010	The MSS-MHCI functional string between the Enterprise Monitoring Server and the Local Management Server shall provide a function Ao (operational availability) of 0.998 and an MDT of 20 minutes.	MHCI	7
C-HRD-18015	The MSS-MHCI functional string between the Local Management Server and ECS managed objects shall provide a function Ao of 0.998 and an MDT of 20 minutes.	MHCI	7
C-HRD-41000	The EDF in the IR-1 timeframe shall provide a Enterprise Monitoring Server configured with: a. Two Fixed Disks b. One Tape Drive c. One CD-ROM Drive d. Storage cross-strapped with Enterprise Communications Server	MHCI	7
C-HRD-41015	The EDF in the IR-1 timeframe shall provide two (2) Data Storage Unit supporting RAID level 5, one for the shared Enterprise Monitoring/Enterprise Communications, and the other for the Bulletin Board Server.	MHCI	7
C-HRD-41020	The EDF in the IR-1 timeframe shall provide four (4) Management Workstations, which can perform any EMC function.	MHCI	7
C-HRD-41025	The EDF in the IR-1 timeframe shall provide 1 system printer.	MHCI	7

Table A-1. Requirements Trace (Page 7 of 45)

C-HRD-42000	The GSFC LSM in the IR-1 timeframe shall provide a Local Management Server configured with: a. Two Fixed Disks b. One Tape Drive c. One CD-ROM Drive	MHCI	7
C-HRD-42010	The GSFC LSM in the R-A timeframe shall provide one Data Storage Unit supporting RAID level 5 cross strapped between the local management and local communications servers.	MHCI	7
C-HRD-42015	The GSFC LSM in the R-A timeframe shall provide two (2) Management Workstations, which can perform any EOC LSM function.	MHCI	7
C-HRD-42020	The GSFC LSM in the R-A timeframe shall provide 1 system printer.	MHCI	7
C-HRD-42700	The GSFC EMC in the R-A timeframe shall provide an enterprise monitoring server, enterprise communications server, four (4) Management Workstations, one (1) printer, and bulletin board server transferred from the IR-1 EDF.	MHCI	7
C-HRD-42705	The GSFC EMC in the R-A timeframe shall provide, via the ECS data server, a Enterprise Monitoring Server long-term data storage capability.	MHCI	7
C-HRD-43000	The EOC LSM in the R-A timeframe shall provide a Local Management Server configured with: a. Two Fixed Disks b. One Tape Drive c. One CD-ROM Drive d. Storage cross-strapped with Local Communications Server	MHCI	7
C-HRD-43010	The EOC LSM in the R-A timeframe shall provide one Data Storage Unit supporting RAID level 5 cross strapped between the local management and local communications servers.	MHCI	7
C-HRD-43015	The EOC LSM in the R-A timeframe shall provide two (2) Management Workstations, which can perform any EOC LSM function.	MHCI	7
C-HRD-43020	The EOC LSM in the R-A timeframe shall provide 1 system printer.	MHCI	7
C-HRD-44000	The MSFC LSM in the IR-1 timeframe shall provide a Local Management Server configured with: a. Two Fixed Disks b. One Tape Drive c. One CD-ROM Drive	MHCI	7

Table A-1. Requirements Trace (Page 8 of 45)

C-HRD-44010	The MSFC LSM in the R-A timeframe shall provide one Data Storage Unit supporting RAID level 5 cross strapped between the local management and local communications servers.	MHCI	7
C-HRD-44015	The MSFC LSM in the R-A timeframe shall provide two (2) Management Workstations, which can perform any EOC LSM function.	MHCI	7
C-HRD-44020	The MSFC LSM in the R-A timeframe shall provide 1 system printer.	MHCI	7
C-HRD-45000	The LaRC LSM in the IR-1 timeframe shall provide a Local Management Server configured with: a. Two Fixed Disks b. One Tape Drive c. One CD-ROM Drive	MHCI	7
C-HRD-45010	The LaRC LSM in the R-A timeframe shall provide one Data Storage Unit supporting RAID level 5 cross strapped between the local management and local communications servers.	MHCI	7
C-HRD-45015	The LaRC LSM in the R-A timeframe shall provide two (2) Management Workstations, which can perform any EOC LSM function.	MHCI	7
C-HRD-45020	The LaRC LSM in the R-A timeframe shall provide 1 system printer.	MHCI	7
C-HRD-46000	The EDC LSM in the IR-1 timeframe shall provide a Local Management Server configured with: a. Two Fixed Disks b. One Tape Drive c. One CD-ROM Drive	MHCI	7
C-HRD-46010	The EDC LSM in the R-A timeframe shall provide one Data Storage Unit supporting RAID level 5 cross strapped between the local management and local communications servers.	MHCI	7
C-HRD-46015	The EDC LSM in the R-A timeframe shall provide two (2) Management Workstations, which can perform any EOC LSM function.	MHCI	7
C-HRD-46020	The EDC LSM in the R-A timeframe shall provide 1 system printer.	MHCI	7
C-MSS-00010	The MSS services shall have an operational availability of .998 and an MDT of 20 minutes or less for critical services.	MHCI	7
C-MSS-00020	The MSS services shall have no single point of failure for functions associated with network databases and configuration data.	MHCI	7
C-MSS-00030	The MSS services shall be extensible in its design to provide capability for growth and enhancement.	MHCI	7
C-MSS-00200	The MSS services shall allocate 10% of development resources for IV&V activity.	MHCI	7

Table A-1. Requirements Trace (Page 9 of 45)

C-MSS-10010	The MSS shall interface with the Ecom systems to exchange data identified in Table 5.1-1 as specified in the ECS/Ecom IRD.	MHCI	7
C-MSS-10020	The MSS shall interface with the Version 0 system to exchange data identified in Table 5.1-1 as specified in the ECS/V0 IRD, 194-219-SE1-004.	MHCI	7
C-MSS-10030	The MSS shall interface with the Science Computing Facility (SCF) to exchange data identified in Table 5.1-1 as specified in ECS/SCF IRD, 194-219-SE1-005.	MHCI	7
C-MSS-10040	The MSS shall interface with the NASA Institutional Support System (NISS) to exchange data identified in Table 5.1-1 as specified in ECS/NISS IRD, 194-219-SE1-020.	MHCI	
C-MSS-10050	The MSS shall interface with the Affiliated Data Centers (ADC) to exchange data identified in Table 5.1-1 as specified in ECS/ADC IRD, 219-CD-006.	MHCI	7
C-MSS-10060	The MSS shall interface with the Tropical Rainfall Measuring Mission (TRMM) to exchange data identified in Table 5.1-1 as specified in ECS/TRMM IRD, 194-219-SE1-018.	MHCI	7
C-MSS-10060	The MSS shall interface with the Tropical Rainfall Measuring Mission (TRMM) to exchange data identified in Table 5.1-1 as specified in ECS/TRMM IRD, 194-219-SE1-018.	MHCI	7
C-MSS-10070	The MSS shall interface with the Landsat 7 System to exchange data identified in Table 5.1-1 as specified in ECS/Landsat 7 IRD, 219-CD-003.	MHCI	7
C-MSS-10080	The MSS shall interface with the NASA Science Internet (NSI) to exchange data identified in Table 5.1-1 as specified in ECS/NSI IRD, 194-219-SE1-001.	MHCI	7
C-MSS-10090	The MSS shall interface with the Program Support Communications Network (PSCN) to exchange data identified in Table 5.1-1 as specified in ECS/PSCN IRD, 193-219-SE1-008.	MHCI	7
C-MSS-10100	The MSS shall interface with the EDOS to exchange data identified in Table 5.1-1 as specified in EDOS/EGS IRD, 560-EDOS-0211.	MsFlExtSys MsScExtSys MsPmExtSys	6.1 6.2 6.3
C-MSS-10110	The MSS shall interface with the International Partners (IP) for Data Interoperability to exchange data identified in Table 5.1-1 as specified in ECS/IP IRD, 194-219-SE1-015.	MsFlExtSys MsScExtSys MsPmExtSys	6.1 6.2 6.3

Table A-1. Requirements Trace (Page 10 of 45)

C-MSS-10200	The MSS shall interface with the SDPS subsystems to exchange the data items in Table 5.1-2 as specified in the ECS internal ICDs, 313-DV3-003.	EcAgEvent MsUsProfile	4.1.3.2 4.3.1.8
C-MSS-10300	The MSS shall interface with the FOS subsystems to exchange the data items in Table 5.1-3 as specified in the ECS internal ICDs, 313-DV3-003.	EcAgEvent	4.1.3.2
C-MSS-10400	The MSS at a site shall interface with the MSS subsystems at the SMC and other sites to exchange management data items in Table 5.1-4.	MsPmExtSys MsFlExtSys MsFlSMC	6.2.3.7 6.1.3.8 6.1.3.10
C-MSS-10410	The MSS shall interface with the CSS subsystems to exchange the data items in Table 5.1-5 as specified in the ECS internal ICDs, 313-DV3-003.	DCERGY_EDIT DCEACL_EDIT EcUtLogger	6.3.3.3 6.3.3.2 4.1.3.6
C-MSS-10420	The MSS shall interface with the ISS subsystems to exchange the data items in Table 5.1-6 as specified in the ECS internal ICDs, 313-DV3-003.	Management Framework Router ACLs	6.2.3.2 6.3.3.17
C-MSS-12005	The MSS Management User Interface (MUI) Service shall be compatible with the ECS management framework.	MUI	3.3.3
C-MSS-12010	The MSS Management User Interface (MUI) Service shall provide a graphical user interface that is OSF/MOTIF compliant	MUI	3.3.3
C-MSS-12020	The MSS MUI Service shall have the capability to respond to keyboard and mouse input devices	MUI	3.3.3
C-MSS-12030	The MSS MUI Service shall provide a capability for the M&O Staff to add/delete a symbol and to modify a symbol's shape, color and position	MUI	3.3.3
C-MSS-12040	The MSS MUI Service shall provide a capability for an application to add/delete a symbol and to modify a symbol's shape, color and position	MUI	3.3.3
C-MSS-12050	The MSS MUI Service shall provide a capability for the M&O Staff to add, delete, and modify text strings	MUI	3.3.3
C-MSS-12060	The MSS MUI Service shall provide a capability for an application to add, delete, and modify text strings	MUI	3.3.3
C-MSS-12070	The MSS MUI Service shall have the capability to provide options and methods to the M&O Staff for screen configuration changes (color, symbol placement, etc.) and for retaining the changes from session to session	MUI	3.3.3
C-MSS-12080	The MSS MUI Service shall provide a capability for an applications to alert the M&O Staff	MUI	3.3.3

Table A-1. Requirements Trace (Page 11 of 45)

C-MSS-12090	The MSS MUI Service shall provide a capability for an applications to establish a dialog session with the M&O Staff	MUI	3.3.3
C-MSS-12100	The MSS MUI Service shall provide a capability for the M&O Staff to load and unload vendor or ECS defined MIB.	MUI	3.3.3
C-MSS-12110	The MSS MUI Service shall provide a capability for an applications to load and unload vendor or ECS defined MIB.	MUI	3.3.3
C-MSS-12120	The MSS MUI Service shall provide a capability for the operator to browse MIB values.	MUI	3.3.3
C-MSS-12130	The MSS MUI Service shall provide the capability for the M&O Staff to register and unregister managed objects.	MUI	3.3.3
C-MSS-12140	The MSS MUI Service shall provide the capability for an application to register and unregister managed objects.	MUI	3.3.3
C-MSS-12170	The MSS MUI Service shall provide the capability to register and unregister management applications.	MUI	3.3.3
C-MSS-12180	The MSS MUI Service shall provide the capability for an application to display on-line help windows	MUI	3.3.3
C-MSS-14010	The MSS Maps/Collection Service shall retain the status of managed objects and their relationship to symbols that comprise a graphical representation of the physical network topology.	Maps/Collections	3.3.4
C-MSS-14020	The MSS Map/Collection Service shall provide a capability to define maps and objects.	Maps/Collections	3.3.4
C-MSS-14030	The MSS Map/Collection Service shall provide a capability to define a hierarchical relationship between maps and sub-maps (i.e., a graphical hierarchical tree)	Maps/Collections	3.3.4
C-MSS-14040	The MSS Map/Collection Service shall propagate events associated with objects up the hierarchical tree	Maps/Collections	3.3.4
C-MSS-16005	The ECS management protocol shall be the SNMP standard as specified in RFC 1157.	Monitor/Control Management Agent Services	3.3.5 4.1
C-MSS-16010	MSS Monitor/Control Service shall communicate via ECS management protocol with the Management Agent Service in test or operational mode.	Monitor/Control	3.3.5
C-MSS-16020	The MSS Monitor/Control Service shall communicate via ECS management protocol with the MSS Management Agent Service to request management data on a managed object.	Monitor/Control	3.3.5

Table A-1. Requirements Trace (Page 12 of 45)

C-MSS-16030	The MSS Monitor/Control Service shall be able to communicate via ECS management protocol with the MSS Management Agent Service to send ECS management set messages to configure and control the processing performed by the ECS management agent.	Monitor/Control	3.3.5
C-MSS-16040	The MSS Monitor/Control Service shall communicate via ECS management protocol with the MSS Management Agent Service to receive ECS management traps/events.	Monitor/Control	3.3.5
C-MSS-16050	The MSS Monitor/Control Service shall allow customized M&O staff-event notifications and automatic actions.	Monitor/Control	3.3.5
C-MSS-16060	The MSS Monitor/Control Service shall allow the capability to set thresholds on managed resources that are monitored	Monitor/Control	3.3.5
C-MSS-16070	The MSS Monitor/Control Service shall automatically report when a threshold has been exceeded by generating a ECS management event	Monitor/Control	3.3.5
C-MSS-16100	The MSS Monitor/Control Service shall perform the following protocol test on managed network nodes: a. IP test b. TCP test c. SNMP test d. UDP test e. ICMP test	Monitor/Control	3.3.5
C-MSS-18040	The MSS Management Data Access Service shall maintain the integrity of the management database.	MsMdManager	6.7.3.9
C-MSS-18050	The MSS Management Data Access Service's shall utilize CSS Services to access/transfer management data.	MsMdManager	6.7.3.9 (via 6.7.3.1 CsFtPRelA)
C-MSS-18060	The Management Data Access Service shall provide the capability for an application to access management data.	ManagementRDBMS	6.7.3.4- Design change MDA provides the ability for users to access management data; the Management RDBMS can provide access for applications
C-MSS-18070	The MSS Management Data Access Service shall provide the capability to selectively access management data.	MsMdUserInterface	6.7.3.11
C-MSS-18200	The MSS Management Data Access Service shall provide the capability for an application via APIs to update fields in the management database.	MsMdManager	6.7.3.9 MDA (through Agent) provides the ability for applications to log events which are summarized into the management database

Table A-1. Requirements Trace (Page 13 of 45)

C-MSS-18220	The MSS Management Data Access Service shall provide the capability for an application via APIs to alter tables and fields in the management database.	ManagementRDBMS	6.7.3.4 The ability to change the management database is provided in the COTS database itself
C-MSS-18260	The MSS Management Data Access Service shall have the capability to schedule the transfer and loading log files into the management database at the site.	MsMdManager	6.7.3.9
C-MSS-18270	The MSS Management Data Access Service shall have the capability to schedule the archiving of log files at the site.	MsMdManager	6.7.3.9
C-MSS-18280	The MSS Management Data Access Service shall have the capability to schedule the transfer of management data at the sites to the SMC.	MsMdManager	6.7.3.9
C-MSS-18330	The MSS Management Data Access Service shall provide the capability for an applications to append records to a log file.	EcAgEvent	4.1.3.2 The ability to log events to the management log file is provided through the Agent design (utilizing CSS logging facility)
C-MSS-18340	The MSS Management Data Access Service shall provide the capability for an application to selectively read a record from a log file	MsMdUserInterface	6.7.3.11- MDA provides the user the ability to read records from the management logfiles through its browser.
C-MSS-18350	The MSS Management Data Access Service shall provide the capability for an application to load log files into the management database at the site	MsMdManager	6.7.3.9
C-MSS-20010	The MSS Discovery Service shall discover (via network protocol) new instances of managed objects.	Discovery	3.3.6
C-MSS-20020	The MSS Discovery Service shall detect missing occurrences of managed objects.	Discovery	3.3.6
C-MSS-20030	The MSS Discovery Service shall report missing occurrences of managed objects.	Discovery	3.3.6
C-MSS-20040	The MSS Discovery Service shall update the object database after the Discovery Service receives a request to register/unregister a managed object.	Discovery	3.3.6
C-MSS-36010	The MSS Management Agent Service shall retrieve data from ECS managed objects in test or operational mode.	MsAgSubAgent EcAgManager	4.1.5.2MsAgSubagent CSC 4.1.5.6EcAgInstrm CSC
C-MSS-36020	The MSS Management Agent Service shall communicate via ECS management protocol with the MSS Monitor/Control Service to respond to requests for managed object MIB attributes	MsAgAgent	4.1.5.1 MsAgAgent CSC
C-MSS-36040	The MSS Management Agent Service shall communicate via ECS management protocol with the MSS Monitor/Control Service to send ECS management traps/events to the Monitor/Control Service.	EcAgEvent	4.1.5.6 EcAgInstrm CSC

Table A-1. Requirements Trace (Page 14 of 45)

C-MSS-36050	The MSS Management Agent Service shall communicate via ECS management protocol with the MSS Monitor/Control Service to receive ECS management set message from the Monitor/Control Service.	MsAgDeputy	4.1.5.5 MsAgDpty CSC
C-MSS-36060	The MSS Management Agent Service shall provide an ECS management agent that is configurable to include: a. Community to respond to and set attributes b. Agent location & contact person c. Traps to send d. Events to log & log file name	MsAgAgent	4.1.5.1 MsAgAgent CSC
C-MSS-36070	The MSS Management Agent Service shall provide an ECS management agent for network devices	MsAgEncps	4.1.5.4 MsAgEncps CSC
C-MSS-36080	The MSS Management Agent Service shall provide an extensible ECS management agent for ECS Host systems	MsAgAgent	4.1.5.1 MsAgAgent CSC
C-MSS-36090	The MSS Management Agent Service shall provide an extensible ECS management agent for ECS applications	MsAgSubAgent	4.1.5.2 MsAgSubAgent CSC
C-MSS-36100	The MSS Management Agent Service shall provide proxy agents for ECS network devices and applications that cannot be managed via SNMP.	EcAgProxy	4.1.5.3 EcAgPrxy CSC
C-MSS-36110	The MSS Management Agent Service shall provide an ECS domain manager agent to coordinate and communicate with multiple ECS management agents.	MsAgAgent	4.1.5.1 MsAgAgent CSC
C-MSS-40000	The MSS configuration management application service at each site shall track the following items at the site by name and identifier: a. ECS subsystems, networks, and configured system and network devices such as workstations, servers, and routers b. ECS releases and site baselines c. ECS hardware and software resources designated as configuration items d. specifications associated with configuration items e. technical documentation and test materials f. scientific algorithms, including software, data and test materials (DAACs only)	BaselineManager Subsystem ConfiguredDevice HardwareControlItem SoftwareControlItem CI Algorithm Toolkit DocumentProfile	5.3.1.4 5.3.1.16 5.3.1.9 5.3.1.11 5.3.1.15 5.3.1.6 5.3.1.1 5.3.1.17 5.3.1.10

Table A-1. Requirements Trace (Page 15 of 45)

C-MSS-40010	The MSS configuration management application service at each site shall identify versions and variants of configuration controlled resources that comprise the site's operational baseline.	BaselineManager Subsystem ConfiguredDevice HardwareControlItem SoftwareControlItem CI Algorithm Toolkit DocumentProfile	5.3.1.4 5.3.1.16 5.3.1.9 5.3.1.11 5.3.1.15 5.3.1.6 5.3.1.1 5.3.1.17 5.3.1.10
C-MSS-40030	The MSS configuration management application service at each site shall make available to the SMC records that identify the site's operational baseline and the versions and implementation status of configuration controlled resources that comprise it.	BaselineManager BaselineManagement Report	5.3.1.4 5.3.1.3
C-MSS-40040	The MSS configuration management application service at each site shall make available to the SMC, “level of assembly” records that describe the composition of configuration items at the site.	BaselineManager BaselineManagement Report	5.3.1.4 5.3.1.3
C-MSS-40060	The MSS configuration management application service at each site shall maintain historical status records about ECS configuration items at the site, identifying each item's: a. current version; b. current version's specifications and technical, operations, and maintenance documentation; c. specification and technical documentation history; d. “level of assembly” representation of the components comprising the items current and release configurations e. version history	BaselineManager DocumentProfile CI	5.3.1.4 5.3.1.10 5.3.1.6
C-MSS-40070	The MSS configuration management application service at the SMC and the sites shall maintain records that establish traceability among operational baselines and releases.	BaselineManager BaselineProfile	5.3.1.4 5.3.1.5
C-MSS-40080	The MSS configuration management application service at the SMC and the sites shall maintain records describing dependencies among baseline objects.	BaselineManager ResourceProfile	5.3.1.4 5.3.1.13

C-MSS-40100	The MSS configuration management application service at the SMC and the DAACs shall maintain SCF-provided configuration data for individual algorithms, including: a. algorithm development version numbers, identification codes, and reference numbers; b. SCF point of contact's name and organization; c. associated files' names, formats, sizes, and descriptions; d. number of files by category and type.	BaselineManager Algorithm	5.3.1.4 5.3.1.1
C-MSS-40110	The MSS configuration management application service shall display and report indentured, “level of assembly” lists that describe the component structure of configuration items.	BaselineManager BaselineManagement Report ResourceProfile	5.3.1.4 5.3.1.3 5.3.1.13
C-MSS-40120	The MSS configuration management application service at the SMC shall track the names and identifiers of the following items deployed at the sites: a. ECS subsystems, networks, and configured system and network devices such as workstations, servers, and routers b. ECS releases and baselines c. ECS hardware and software resources designated as configuration items d. technical documentation and test materials; e. scientific algorithms, including software, data and test materials (DAAC's only) f. algorithm processing logic control and calibration coefficients data; g. algorithm test documentation, including specifications, data files, and	BaselineManager Subsystem ConfiguredDevice HardwareControlItem SoftwareControlItem CI Algorithm Toolkit DocumentProfile	5.3.1.4 5.3.1.16 5.3.1.9 5.3.1.11 5.3.1.15 5.3.1.6 5.3.1.1 5.3.1.17 5.3.1.10
C-MSS-40140	The MSS configuration management application service at the SMC shall maintain, and make available system-wide, information identifying the sites where individual versions of configuration items are located and the operational status of that version at the site.	BaselineManager ConfiguredDevice HardwareControlItem CI SoftwareControlItem	5.3.1.4 5.3.1.9 5.3.1.11 5.3.1.6 5.3.1.15
C-MSS-40150	The MSS configuration management application service at the SMC shall maintain, and make available system-wide, records that identify the current and previous versions of ECS hardware and software resources deployed to the sites.	BaselineManager BaselineManagement Report BaselineProfile BaselineChange HardwareControlItem CI SoftwareControlItem	5.3.1.4 5.3.1.3 5.3.1.5 5.3.1.2 5.3.1.11 5.3.1.6 5.3.1.15

Table A-1. Requirements Trace (Page 17 of 45)

C-MSS-40160	The MSS configuration management application service at the SMC shall maintain records that identify the current and previous versions of ECS documents associated with deployed ECS resources.	BaselineManager BaselineProfile DocumentProfile BaselineManagement Report BaselineChange	5.3.1.4 5.3.1.5 5.3.1.10 5.3.1.3 5.3.1.2
C-MSS-40170	The MSS configuration management application service at the SMC shall maintain, and distribute to each site, records that identify the baseline changes included in each release of ECS hardware and software deployed to the site.	BaselineManager BaselineManagement Report BaselineProfile BaselineChange	5.3.1.4 5.3.1.3 5.3.1.5 5.3.1.2
C-MSS-40180	The MSS configuration management application service at the SMC shall maintain, and distribute to each site, records that identify the specifications and technical, operations, and maintenance documents associated with versions of ECS hardware and software configuration items deployed to the site.	BaselineManager DocumentProfile BaselineProfile CI	5.3.1.4 5.3.1.10 5.3.1.5 5.3.1.6
C-MSS-40190	MSS configuration management application service at the SMC shall maintain, and distribute to each site, records that describe the change requests (enhancements and corrections) satisfied by new versions of ECS hardware, software, and documentation deployed to the sites.	BaselineManager BaselineManagement Report ChangeRequestManager SoftwareControlItem CI HardwareControlItem DocumentProfile	5.3.1.4 5.3.1.3 5.3.1.7 5.3.1.15 5.3.1.6 5.3.1.11 5.3.1.10
C-MSS-40200	The MSS configuration management application service at the SMC shall maintain historical status records about ECS configuration items system-wide, to include each item's: a. current version; b. current version's specifications and technical, operations, and maintenance documentation; c. specifications and technical documentation history d. “level of assembly” representation of components comprising the item's current and release configurations: e. version history	BaselineManager DocumentProfile CI	5.3.1.4 5.3.1.10 5.3.1.6

Table A-1. Requirements Trace (Page 18 of 45)

C-MSS-40210	The MSS configuration management application service at the SMC shall maintain historical status records about ECS system releases, to include each release's: a. latest baseline plus approved changes. b. baseline history. c. latest release documentation. d. "level of assembly" representation of the subsystem and configuration item versions that comprise the release configuration e. history of changes, including changes to subordinate units/components. f. effectivity and installation status at operational sites. g. release configuration	BaselineManager BaselineProfile DocumentProfile Subsystem CI ConfiguredDevice	5.3.1.4 5.3.1.5 5.3.1.10 5.3.1.16 5.3.1.6 5.3.1.9
C-MSS-40220	The MSS configuration management application service at the SMC shall maintain historical status records about ECS baseline changes to include: a. sites affected; b. installation dates c. installation status.	BaselineManager BaselineProfile	5.3.1.4 5.3.1.5
C-MSS-40240	The MSS configuration management application service at the SMC shall maintain software-critical and security-sensitive items lists.	BaselineManager DocumentProfile SoftwareControlItem CI HardwareControlItem	5.3.1.4 5.3.1.10 5.3.1.15 5.3.1.6 5.3.1.11
C-MSS-40250	The MSS configuration management application service at the SMC shall produce, and make available system-wide, reports containing the identity and change status of documents associated with deployed ECS resources.	BaselineManager BaselineManagement Report DocumentProfile	5.3.1.4 5.3.1.3 5.3.1.10
C-MSS-40260	The MSS configuration management application service at the SMC shall produce, and make available system-wide, reports, containing the identity and change status of individual ECS resources deployed to the sites.	BaselineManager BaselineManagement Report ConfiguredDevice HardwareControlItem CI SoftwareControlItem Toolkit	5.3.1.4 5.3.1.3 5.3.1.9 5.3.1.11 5.3.1.6 5.3.1.15 5.3.1.17
C-MSS-40270	The MSS configuration management application service at the SMC shall produce, and make available system-wide, reports containing the identity of resources comprising ECS baselines and releases.	BaselineManager BaselineManagement Report BaselineProfile ConfiguredDevice HardwareControlItem CI SoftwareControlItem	5.3.1.4 5.3.1.3 5.3.1.5 5.3.1.9 5.3.1.11 5.3.1.6 5.3.1.15

Table A-1. Requirements Trace (Page 19 of 45)

C-MSS-40280	The MSS configuration management application service shall characterize ECS-controlled resources as system-wide or site-specific.	BaselineManager ResourceProfile	5.3.1.4 5.3.1.13
C-MSS-40290	The MSS configuration management application service shall accept and store baseline management data records provided via interactive user interface and formatted data files.	BaselineManager SoftwareControlItem SoftwareChangeManager ResourceProfile BaselineProfile DocumentProfile ChangeRequestManager	5.3.1.4 5.3.1.15 5.3.1.14 5.3.1.13 5.3.1.5 5.3.1.10 5.3.1.7
C-MSS-40300	The MSS configuration management application service shall produce formatted data files containing baseline management data records.	BaselineManager BaselineProfile ResourceProfile DocumentProfile	5.3.1.4 5.3.1.5 5.3.1.13 5.3.1.10
C-MSS-40400	The MSS configuration management application service at the sites and the SMC shall maintain software libraries to store files containing versions and platform variants of: a. source code; b. binaries and executables; c. patches; d. calibration coefficients and control data; e. scripts; f. designs and design specifications; g. databases; h. technical documentation (both text and graphics); i. test data; j. test reports; k. interface specifications; l. configuration data. (IR-1)	SoftwareChangeManager SoftwareLibrary LibraryFile	5.3.2.9 5.3.2.11 5.3.2.7
C-MSS-40410	The MSS configuration management application service at each DAAC shall maintain user-definable software configuration status information for each algorithm. (IR-1)	SoftwareChangeManager SoftwareLibrary LibraryFile	5.3.2.9 5.3.2.11 5.3.2.7
C-MSS-40420	The MSS configuration management application service at each site shall maintain M&O staff-definable software configuration status information for each version of every software library file.	SoftwareChangeManager SoftwareLibrary LibraryFile	5.3.2.9 5.3.2.11 5.3.2.7

C-MSS-40460	The MSS configuration management application service at the SMC shall assemble unlicensed toolkit software files for posting to the ECS bulletin board. Files consist of: a. source code; b. linkable object code for selected workstation configurations; c. makefiles that automate installation; d. installation instructions.	SoftwareChangeManager SoftwareLibrary LibraryFile View Association SoftwareChangeReport	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.12 5.3.2.2 5.3.2.10
C-MSS-40470	The MSS configuration management application service shall regulate operations on software library files through use of individual and group permissions.	SoftwareChangeManager SoftwareLibrary LibraryFile AccessProfile	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.1
C-MSS-40480	The MSS configuration management application service shall use a checkout/edit/checkin paradigm to govern changing of software library files.	SoftwareChangeManager SoftwareLibrary LibraryFile	5.3.2.9 5.3.2.11 5.3.2.7
C-MSS-40490	The MSS configuration management application service shall track each software library file that has been changed as a new version of the original file.	SoftwareChangeManager SoftwareLibrary LibraryFile	5.3.2.9 5.3.2.11 5.3.2.7
C-MSS-40500	The MSS configuration management application service shall merge versions of software library files and identify version conflicts, if any.	SoftwareChangeManager SoftwareLibrary LibraryFile	5.3.2.9 5.3.2.11 5.3.2.7
C-MSS-40510	The MSS configuration management application service shall maintain records of actual changes made to ECS software library files in implementing system enhancement requests.	SoftwareChangeManager SoftwareLibrary LibraryFile CmScmCotsLog	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.6
C-MSS-40520	The MSS configuration management application service shall verify that changes to software library files are supported by approved change requests.	SoftwareChangeManager SoftwareLibrary LibraryFile ChangeRequestManager	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.5
C-MSS-40530	The MSS configuration management application service shall identify implementation status for each version of every software library file, reflecting the lifecycle stage to which it has been promoted.	SoftwareChangeManager SoftwareLibrary LibraryFile BaselineManager	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.3
C-MSS-40540	The MSS configuration management application service shall perform builds of baseline systems for ECS platforms and audit the builds such that they can be repeated.	SoftwareChangeManager SoftwareLibrary LibraryFile BuildRecord	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.4
C-MSS-40550	The MSS configuration management application service shall reconstruct previous versions of software library files.	SoftwareChangeManager SoftwareLibrary-LibraryFile BuildRecord	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.4

Table A-1. Requirements Trace (Page 21 of 45)

C-MSS-40560	The MSS configuration management application service shall allow concurrent user access to software library files.	SoftwareChangeManager SoftwareLibrary LibraryFile AccessProfile	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.1
C-MSS-40570	The MSS configuration management application service shall maintain an audit trail of all changes made to software library files.	SoftwareChangeManager SoftwareLibrary LibraryFile CmScmCotsLog	5.3.2.9 5.3.2.11 5.3.2.7 5.3.2.6
C-MSS-40600	The MSS configuration management application service shall provide a capability with which to specify a need for ECS system changes, both for enhancing system capabilities and for correcting non-conformance with system requirements.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40610	The MSS configuration management application service shall store copies of non-conformance reports and requests to modify ECS components and configurations.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40620	The MSS configuration management application service at the sites shall provide a capability with which to forward non-conformance reports and requests for ECS configuration changes to the SMC.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40650	The MSS configuration management application service at the SMC shall receive configuration change requests and non-conformance reports in electronic form from the sites.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40660	The MSS configuration management application service at the SMC shall distribute change evaluation requests to designated organizations system-wide and record evaluation assignments and distribution status.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40670	The MSS configuration management application service at the SMC shall receive and store impact assessments in response to change evaluation requests.	ChangeRequestManager ResourceChangeRequest Attachment	5.3.3.3 5.3.3.6 5.3.3.1
C-MSS-40680	The MSS configuration management service at the SMC shall electronically link impact assessments to their associated change requests.	ChangeRequestManager ResourceChangeRequest Attachment	5.3.3.3 5.3.3.6 5.3.3.1
C-MSS-40690	The MSS configuration management application service at the SMC shall maintain the status of responses to change evaluation requests.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6

Table A-1. Requirements Trace (Page 22 of 45)

C-MSS-40700	The MSS configuration management application service at the SMC shall record summaries of impact assessments received.	ChangeRequestManager ResourceChangeRequest Attachment	5.3.3.3 5.3.3.6 5.3.3.1
C-MSS-40720	The MSS configuration management application service at the SMC shall make non-conformance reports, configuration change requests, assessments, and status available for system-wide viewing.	ChangeRequestManager ResourceChangeRequest ResourceChangeRequestReport	5.3.3.3 5.3.3.6 5.3.3.7
C-MSS-40730	The MSS configuration management application service at the SMC shall maintain historical records of ECS configuration change requests, non-conformance reports, and system impact assessments.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40750	The MSS configuration management application service at the SMC shall track approval and closure status of configuration change requests and non-conformance reports.	ChangeRequestManager ResourceChangeRequest	5.3.3.3 5.3.3.6
C-MSS-40760	The MSS configuration management application service at the SMC shall report, and make available system-wide lists of the identity and disposition of configuration change requests and non-conformance reports against ECS baselines.	ChangeRequestManagerResourceChangeRequestBaselineManager SoftwareChangeManager ResourceChangeRequestReport	5.3.3.3 5.3.3.6 5.3.3.2 5.3.3.8 5.3.3.7
C-MSS-40770	The MSS configuration management application service at the SMC shall collect, and make available system-wide, the allocations, schedules and status of tasks for implementing CCB-approved changes to ECS hardware and software and for correcting non-conformance with system requirements.	ChangeRequestManagerResourceChangeRequest ResourceChangeRequestReport	5.3.3.3 5.3.3.6 5.3.3.7
C-MSS-40990	The MSS configuration management application service shall log the following information for configuration management events: a. operation type; b. userid of initiator; c. date-time stamp; d. host name. (IR-1, at the sites only)	BaselineManager SoftwareChangeManager ChangeRequestManager CmCrmCotsLog CmBmCotsLog CmScmCotsLog	5.3.1.4 5.3.2.9 5.3.3.3 5.3.3.4 5.3.1.8 5.3.2.6
C-MSS-40995	The MSS configuration management application service shall generate chronological reports of logged CM events associated with M&O staff-selectable: a. time frames; b. operation types; c. userids; d. hosts.	BaselineManager SoftwareChangeManager ChangeRequestManager CmCrmCotsLog CmBmCotsLog CmScmCotsLog	5.3.1.4 5.3.2.9 5.3.3.3 5.3.3.4 5.3.1.8 5.3.2.6

Table A-1. Requirements Trace (Page 23 of 45)

C-MSS-60010	The MSS Fault Management Application Service shall provide the capability to create and display graphical representations of a given network topology consisting of the following: a. routers b. communication lines c. hosts d. peripherals e. applications	ManagementFramework	6.1.3.4
C-MSS-60020	The MSS Fault Management Application Service shall provide the capability to define categories of faults.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60030	The MSS Fault Management Application Service shall provide the capability to assign faults to categories.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60040	The MSS Fault Management Application Service shall provide the capability to assign severity levels to faults.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60050	The MSS Fault Management Application Service shall be capable of providing the Management Data Access Service with a configurable list of fault categories that specify whether to enable or disable the logging of fault notifications for that fault category.	MsAgSubAgent	4.1.5.2
C-MSS-60060	The MSS Fault Management Application Service shall provide the capability to enable or disable the display of fault notifications received from a specific managed object based on fault category assigned to that fault.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60070	The MSS Fault Management Application Service shall provide the capability to specify additional information to be added to a disk log file, based on the fault category, when the notification of a fault is received.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60080	The MSS Fault Management Application Service shall have the capability to establish, view, modify and delete thresholds on performance metrics it measures.	ManagementFramework	6.1.3.4
C-MSS-60100	The MSS Fault Management Application Service shall have the capability to poll for the detection of fault/performance information.	ManagementFramework	6.1.3.4
C-MSS-60110	The MSS Fault Management Application Service shall be capable of receiving fault notifications.	ManagementFramework	6.1.3.4
C-MSS-60120	The MSS Fault Management Application Service shall have the capability to define the frequency with which polling is done for the detection of fault/performance information.	ManagementFramework	6.1.3.4

C-MSS-60130	The MSS Fault Management Application Service shall provide the capability to detect the following types of faults, errors and events: a. communications software version mismatch errors b. communication software configuration errors c. the following errors in communication hardware: 1. host not reachable 2. router not reachable 3. errors and failures of communication links d. Errors in the communications protocols supported e. degradation of performance due to established thresholds being exceeded	ManagementFramework	6.1.3.4
C-MSS-60140	The MSS Site Fault Management Application Service shall have the capability to generate a fault notification when a predefined threshold on a performance metric is exceeded.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60150	The MSS Fault Management Application Service shall have the capability to receive fault notifications from the Management Agent Service.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60160	The MSS EMC Fault Management Application Service shall have the capability to receive notifications of detected faults and degradation of performance from: a. Site fault management applications b. Other external systems as defined in Section 5.1.	ManagementFramework MsFlConfig MsFlSMC MsFlExtSys	6.1.3.4 6.1.3.7 6.1.3.10 6.1.3.8
C-MSS-60170	The MSS EMC Fault Management Application Service shall be capable of requesting fault notification and performance degradation data from : a. Site Fault Management Applications b. Other external systems as defined in Section 5.1.	ManagementFramework MsFlConfig MsFlSMC MsFlExtSys	6.1.3.4 6.1.3.7 6.1.3.10 6.1.3.8
C-MSS-60180	The MSS EMC Fault Management Application Service shall be capable of receiving summarized fault notification and performance degradation data from: a. Site fault management applications b. Other external systems as defined in Section 5.1.	MsFlSMC	6.1.3.10
C-MSS-60190	The MSS Fault Management Application Service shall use the Logging Services to record each detected fault.	EcAgEvent	4.1

C-MSS-60200	The MSS Fault Management Application Service shall have the capability to generate the following types of notifications for detected faults: a. a change in the color of an icon on a display b. a message in a pop-up notification window c. logging the following fault information to a disk log file: 1. fault type 2. date and time of occurrence of the fault 3. identification of the source of the notification (e.g. IP address, process name, etc.) 4. fault data received with the notification 5. operator-defined	ManagementFramework	6.1.3.4
C-MSS-60210	The MSS Fault Management Application Service shall maintain a list of external service providers, M&O operators, and applications to be notified in the event that a specified fault is detected.	MsFlConfig	6.1.3.7
C-MSS-60220	The MSS Fault Management Application Service shall have the capability to send the notification of a fault to registered recipients.	MsFlConfig	6.1.3.7
C-MSS-60230	The MSS Fault Management Application Service shall have the capability of generating a notification within a maximum of five minutes of fault detection.	MsFlConfig ManagementFramework	6.1.3.7 6.1.3.4
C-MSS-60300	The MSS Fault Management Application Service shall provide the capability to identify routes between selected pairs of hosts on the ESN.	MsFlTest	6.1.3.11
C-MSS-60310	The MSS Fault Management Application Service shall provide utilities to perform diagnostics and testing of the following for the purpose of fault isolation: a. connectivity between pairs of ECS hosts and ECS routers b. ability to reach hosts and routers c. availability of network services at hosts	MsFlTest	6.1.3.11
C-MSS-60320	The MSS Fault Management Application Service shall provide, for selective use as a debugging aid, the capability to perform packet tracing of protocols used in ECS.	MsFlTest	6.1.3.11

C-MSS-60330	The MSS Fault Management Application Service at each site shall have the capability to perform periodic testing of all ECS communication links at that site to verify that they are operational.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60340	The MSS Fault Management Application Service shall be capable of verifying the operational status of a host.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60350	The MSS Fault Management Application Service shall have the capability to periodically execute diagnostic tests in order to isolate, characterize and identify a fault.	ManagementFramework MsFlConfig MsFlTest	6.1.3.4 6.1.3.7 6.1.3.11
C-MSS-60360	The MSS Fault Management Application Service shall provide the capability to execute vendor diagnostics in order to diagnose faults traced to hardware equipment.	MsFlTest	6.1.3.11
C-MSS-60370	The MSS Fault Management Application Service at the SMC shall be capable of sending gathered isolation, location, identification and characterization of reported faults data to the level of subsystem and equipment to the following: a. the site Fault Management Applications b. other external systems as defined in Section 5.1.	MsFlExtSys MsFlSMC	6.1.3.8 6.1.3.10
C-MSS-60380	The MSS Fault Management Application Service at the sites shall isolate, locate, and identify faults, identify subsystem, equipment and software faults, and identify the nature of the faults detected within its site.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60390	The MSS Fault Management Application Service at the sites shall, for faults detected within its site, isolate, locate, and identify faults to the level of: a. subsystem b. equipment c. software	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60395	The MSS Fault Management Application Service shall be capable of retrieving records of detected fault.	ManagementFramework	6.1.3.4
C-MSS-60400	The MSS EMC Fault Management Application Service shall support, maintain, and update system fault management policies and procedures, to include: a. Fault Identification b. Fault priorities c. Recovery or corrective actions	MsFlSMC	6.1.3.10
C-MSS-60410	The MSS Site Fault Management Application Service shall have the capability to receive Fault Management Policies and Procedures from the EMC.	MsFlSMC	6.1.3.10

Table A-1. Requirements Trace (Page 27 of 45)

C-MSS-60420	The MSS Fault Management Application Service shall interface with the MSS Configuration Management Application Service and schedule a change in the configuration of the site when such a change in the configuration of the site is deemed necessary to recover from a fault.	MsFlAction	6.1.3.6
C-MSS-60500	The MSS EMC Fault Management Application Service shall coordinate the recovery from conditions of performance degradation and faults with the sites and external network service providers.	MsFlExtSys	6.1.3.8
C-MSS-60510	The MSS EMC Fault Management Application Service at the SMC shall coordinate, as necessary via directives and instructions, the recovery from faults reported from a site.	MsFlSMC	6.1.3.10
C-MSS-60520	The MSS Fault Management Application Service shall provide the capability to allow the specification and execution of action routines in response to the notification of a fault.	ManagementFramework MsFlConfig	6.1.3.4 6.1.3.7
C-MSS-60530	The MSS Fault Management Application Service shall provide the capability to pass parameters to action routines.	ManagementFramework	6.1.3.4
C-MSS-60540	The MSS Fault Management Application Service shall utilize office automation support tools for the generation of directives and instructions for recovery from faults within its site.		will be satisfied by OA tools in Release A
C-MSS-60600	The MSS Fault Management Application Service shall have the capability to generate, on an interactive and on a scheduled basis, reports on performance/error data that it has been configured to collect.	ManagementFramework	6.1.3.4
C-MSS-60600	The MSS Fault Management Application Service shall have the capability to generate, on an interactive and on a scheduled basis, reports on performance/error data that it has been configured to collect.	ManagementFramework	6.1.3.4
C-MSS-60610	The MSS Fault Management Application Service shall have the capability to build histories for different types of errors and events detected, for the purpose of analysis.	Mgt Database	6.9
C-MSS-60620	The MSS Fault Management Application Service shall have the capability to redirect reports to: a. console b. disk file c. printer	ManagementFramework	6.1.3.4

Table A-1. Requirements Trace (Page 28 of 45)

C-MSS-66000	The MSS performance management application service shall be capable of monitoring the performance of the following ECS components a. network components 1. routers 2. links 3. bridges 4. gateways b. hosts c. operating systems d. peripherals e. databases f. applications	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5
C-MSS-66010	The MSS performance management application service shall be capable of monitoring ECS component protocol stack performance parameters defined in IETF RFC 1213.	ManagementFramework	6.2.3.3
C-MSS-66020	The MSS Performance Management Application Service shall be capable of monitoring ethernet-like device performance parameters as specified in IETF RFC 1623.	ManagementFramework	6.2.3.3
C-MSS-66020	The MSS Performance Management Application Service shall be capable of monitoring ethernet-like device performance parameters as specified in IETF RFC 1623.	ManagementFramework	6.2.3.3
C-MSS-66030	The MSS performance management application service shall be capable of receiving managed object definitions for each managed object.	ManagementFramework	6.2.3.3
C-MSS-66040	The MSS performance management application service shall be capable of specifying which available performance metrics are to be gathered from each individual managed object.	ManagementFramework MsPmConfig	6.2.3.3 6.2.3.7
C-MSS-66050	The MSS performance management application service shall be capable of requesting performance data from each individual managed object: a. at configurable intervals b. on demand.	ManagementFramework MsPmConfig	6.2.3.3 6.2.3.7
C-MSS-66060	The MSS performance management application service shall be capable of receiving requested performance data from ECS components.	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5
C-MSS-66070	The MSS Performance Management Application Service shall be capable of receiving unrequested performance data from ECS managed objects.	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5

C-MSS-66080	The MSS performance management application service shall be capable of retrieving the following data for all network component interfaces: a. operational status b. type c. speed d. octets in/out e. packets in/out f. discards in/out g. errors in/out	ManagementFramework	6.2.3.3
C-MSS-66090	The MSS Performance Management Application Service shall have the capability to collect the following performance information about communication protocol stacks on managed devices: a. number of transport layer messages received with errors b. number of transport layer messages requiring retransmission c. number of transport layer messages received that could not be delivered d. number of network layer messages received with errors e. number of network layer messages received that could not be delivered f. number of network layer messages that were discarded	ManagementFramework	6.2.3.3
C-MSS-66100	The MSS performance management application service shall be capable of retrieving the following data for all hosts: a. total CPU utilization b. memory utilization c. physical disk i/o's d. disk storage size e. disk storage used f. number of active processes g. length of run queue h. network i/o's (packets) i. network errors	MsPmApplManager	6.2.3.5
C-MSS-66120	The MSS performance management application service shall be capable of determining the operational state of all network components, hosts, and peripherals to be: a. on-line b. off-line c. in test mode	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5

C-MSS-66130	The MSS performance management application service shall be capable of receiving operational state change notifications from network components, hosts, applications, and peripherals.	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5
C-MSS-66135	The MSS Performance Management Application Service shall have the capability to calculate the following statistics for the purpose of supporting RMA analysis for managed objects: a. Mean Down time (MDT) b. Mean Time Between Maintenance (MTBM) 1. Mean Time Between Preventive Maintenance (MTBPM) 2. Mean Time Between Corrective Maintenance (MTBCM) c. Mean Time to Repair (MTTR)	MsPmManager	6.2.3.11
C-MSS-66137	The MSS Performance Management Application Service shall retain the calculated RMA statistics in a repository accessible for further analysis by the M&O Staff.	MsPmManager	6.2.3.11
C-MSS-66140	The MSS EMC Performance Management Application Service shall have the capability to request performance data from: a. Site performance management applications b. Other external systems as defined in Section 5.1.	MsPmSMC MsPmExtSys	6.2.3.13 6.2.3.9
C-MSS-66150	The MSS EMC Performance Management Application Service shall be capable of receiving performance data from: a. Site performance management applications b. Other external systems as defined in Section 5.1.	MsPmSMC MsPmExtSys	6.2.3.13 6.2.3.9
C-MSS-66160	The MSS EMC Performance Management Application Service shall be capable of receiving summarized performance data from: a. Site performance management applications b. Other external systems as defined in Section 5.1.	MsPmSMC MsPmExtSys	6.2.3.13 6.2.3.9
C-MSS-66170	The MSS performance management application service shall log ECS performance data pertaining to ECS network components and operating system resources.	MsAgSubagent_C	6.2.3.4

Table A-1. Requirements Trace (Page 31 of 45)

C-MSS-66180	The MSS performance management application service shall have the capability to generate the following types of statistics for a configurable period of time for performance data stored in the Management Database: a. average b. median c. maximum d. minimum e. ratios f. rates g. standard deviations.	MsPmApplManager ManagementFramework	6.2.3.5 6.2.3.3
C-MSS-66190	The MSS performance management application service shall provide a configurable number of thresholds for each performance metric.	MsPmConfig	6.2.3.7
C-MSS-66200	The MSS EMC performance management application service shall be capable of creating a list of suggested initial threshold values for each performance metric.	MsPmList	6.2.3.10
C-MSS-66210	The MSS EMC performance management application service shall be capable of sending a list of suggested initial thresholds for each performance metric to the MSS site performance management application service.	MsPmExtSys	6.2.3.9
C-MSS-66220	The MSS site performance management application service shall be capable of receiving a list of suggested initial thresholds for each performance metric from the MSS EMC performance management application service.	MsPmList	6.2.3.10
C-MSS-66230	The MSS performance management application service shall allow each performance metric threshold to be configurable.	MsPmConfig MsAgSubagent_C	6.2.3.7 6.2.3.4
C-MSS-66240	The MSS performance management application service shall be capable of evaluating each performance metric against defined thresholds.	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5
C-MSS-66250	The MSS performance management application service shall record an event in the local History Log whenever a threshold is crossed.	MsPmEvent	6.2.3.8
C-MSS-66260	The MSS performance management application service shall provide queries that generate performance statistics from performance data stored in the Management Database.	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5
C-MSS-66270	The MSS performance management application service shall store generated performance statistics.	ManagementFramework MsPmApplManager	6.2.3.3 6.2.3.5

C-MSS-66280	The MSS site performance management application service shall be capable of extracting summarized site status information from logged performance data.	MsPmManager	6.2.3.11
C-MSS-66290	The MSS site performance management application service shall be capable of sending summarized status information for that site to the MSS EMC performance management application service.	MsPmSMC	6.2.3.13
C-MSS-66300	The MSS EMC performance management application service shall log received summarized site status.	MsPmSMC	6.2.3.13
C-MSS-66310	The MSS performance management application service shall be capable of retrieving the following science algorithm performance data via the Management Data Access Service: a. algorithm name b. algorithm version c. start time d. stop time e. CPU utilization f. memory utilization g. disk reads h. disk writes	MsPmManager	6.2.3.11
C-MSS-67000	The MSS performance management application service shall be capable of extracting values of performance metrics gathered for a specified managed objects over a configurable period of time from the Management Database.	MsPmManager	6.2.3.11
C-MSS-67010	The MSS performance management application service shall be capable of generating a graph of the extracted performance metric values.	MsPmManager	6.2.3.11
C-MSS-68000	The MSS performance management application service shall be capable of graphically displaying the operational state of managed objects through the MUI service.	ManagementFramework	6.2.3.3
C-MSS-68010	The MSS performance management application service shall be capable of displaying M&O staff-selected performance statistics through the MUI in tabular and graphical formats.	MsPmManager ManagementFramework	6.2.3.11 6.2.3.3
C-MSS-68020	The MSS performance management application service shall be capable of printing M&O staff-selected performance statistics.	ManagementFramework	6.2.3.3
C-MSS-68030	The MSS performance management application service shall be capable of receiving system resource utilization information requests from the SDPS Data Processing subsystem via the Management Agent Service.	MsPmApplManager	6.2.3.5

C-MSS-68040	The MSS performance management application service shall be capable of providing the following current system resource utilization information to the SDPS Data Processing subsystem via Management Agent Service: a. CPU utilization b. memory utilization c. disk i/o's (per second)	MsPmApplManager	6.2.3.5
C-MSS-68050	The MSS performance management application service shall be capable of receiving resource utilization information requests from the SDPS Data Server subsystems via Management Agent Service.	MsPmApplManager	6.2.3.5
C-MSS-68060	The MSS performance management application service shall be capable of providing the following current resource utilization information to the SDPS Data Server subsystem via the Management Agent Service: a. CPU utilization b. memory utilization c. disk I/O's (per second)	MsPmApplManager	6.2.3.5
C-MSS-68070	The MSS performance management application service shall be capable of receiving resource utilization information requests from the SDPS Client subsystem via the Management Agent Service.	MsPmApplManager	6.2.3.5
C-MSS-68080	The MSS performance management application service shall be capable of providing the following current resource utilization information to the SDPS Client subsystem via the Management Agent Service. a. CPU utilization b. memory utilization c. disk I/O's (per second)	MsPmApplManager	6.2.3.5
C-MSS-68090	The MSS Performance Management Application Service shall have the capability to generate reports from collected management data.	MsPmManagerManagement FrameworkMs PmApplManager	6.2.3.11 6.2.3.3 6.2.3.5
C-MSS-68100	The MSS Performance Management Application Service shall have the capability to redirect reports to: a. console b. disk file c. printer	MsPmManager ManagementFrameworkMsPmApplManager	6.2.3.11 6.2.3.3 6.2.3.5
C-MSS-69000	The MSS performance management application service shall maintain operational benchmark test procedures.	MsPmTest	6.2.3.14
C-MSS-69010	The MSS performance management application service shall receive and maintain operational benchmark test results.	MsPmTest	6.2.3.14

Table A-1. Requirements Trace (Page 34 of 45)

C-MSS-69020	The MSS performance management application service shall be capable of performing operational benchmark tests.	MsPmTest	6.2.3.14
C-MSS-69030	The MSS performance management application service shall be capable of providing results of benchmark tests and results of predefined tests to the M&O staff for validation.	MsPmTest ManagementFramework	6.2.3.144.2.3.3
C-MSS-70010	The MSS Security Management Application Service shall provide the capability to create, modify and delete user accounts with the following attributes: a. username b. password c. group identification code d. user identification code e. login directory f. command line interpreter	MsAuthenticationDB OS	6.3.3.7 6.3.3.15
C-MSS-70020	The MSS Security Management Application Service shall enable the assignment of user accounts to groups based on the group identification code.	MsAuthenticationDB OS DCE RGY_EDIT HalDcm	6.3.3.7 6.3.3.15 6.3.3.3 6.3.3.5
C-MSS-70100	The MSS site Security Management Application Service shall provide the capability to set, maintain, and update access control information for ECS resources.	MsScAuthorizationDB Router ACLs OS ACL TCP WrapperConfig DCE ACL_EDIT HalDcm	6.3.3.8 6.3.3.17 6.3.3.15 6.3.3.18 6.3.3.2 6.3.3.5
C-MSS-70110	The MSS site Security Management Application Service shall provide the capability to specify privileges for authorized users and user groups for access to ECS resources.	MsScAuthorizationDB OS ACL DCE ACL_EDIT HalDcm	6.3.3.8 6.3.3.15 6.3.3.2 6.3.3.5
C-MSS-70120	The MSS site Security Management Application service shall provide the mechanism, for each ECS host, to allow or deny incoming requests from specific hosts to services.	TCP WrapperConfig	6.3.3.18
C-MSS-70130	The MSS site Security Management Application Service shall provide a command line interface and a GUI for the management of the following security databases: a. Authentication Database b. Authorization Database c. Network Database	OS DCE RGY_EDIT HalDcm OS ACL DCEACL_EDIT Router ACLs	6.3.3.15 6.3.3.3 6.3.3.5 6.3.3.15 6.3.3.2 6.3.3.17

C-MSS-70300	The MSS site Security Management Application Service shall have the capability to perform the following types of security tests: a. password auditing b. file system integrity checking c. auditing of user privileges d. auditing of resource access control information	ComplianceTest	6.3.3.1
C-MSS-70310	The MSS site Security Management Application Service shall have the capability to perform security testing on a periodic and on an interactive basis.	MsScManager	6.3.3.10
C-MSS-70320	The MSS site Security Management Application Service shall have the capability to send the results of the tests to the EMC Security Management Application Service.	MsScSMC	6.3.3.13
C-MSS-70330	The MSS EMC Security Management Application Service shall have the capability to request, support, coordinate and maintain security testing for sites.	MsScSMC	6.3.3.13
C-MSS-70340	The MSS EMC Security Management Application Service shall have the capability to request security testing of the sites on a scheduled and an interactive basis	MsScSMC	6.3.3.13
C-MSS-70350	The MSS EMC Security Management Application Service shall have the capability to receive the results of security tests performed at the sites.	MsScSMC	6.3.3.13
C-MSS-70400	The MSS EMC Security Management Application Service shall have the capability to receive notifications of security events from the site Security Management Application Services.	MsScSMC	6.3.3.13
C-MSS-70410	The MSS EMC Security Management Application Service shall have the capability to receive security audit trails from the site Security Management Application Services.	MsScSMC	6.3.3.13
C-MSS-70420	The MSS EMC Security Management Application Service shall have the capability to analyze security audit trails for the purpose of detecting intrusions.	TCP WrapperConfig	6.3.3.18
C-MSS-70430	The MSS site Security Management Application Service shall provide the capability to designate a user or a group of users to receive a notification upon the detection of an intrusion.	MsScExtSys MsScSMC	6.3.3.9 6.3.3.13
C-MSS-70440	The MSS site Security Management Application Service shall provide the capability to notify designated M&O staff(s) upon the detection of an intrusion.	MsScExtSys MsScSMC	6.3.3.9 6.3.3.13

C-MSS-70450	The MSS site Security Management Application Service shall have the capability to detect the following types of intrusions: a. Login failures b. Unauthorized access to ECS resources c. Break-ins d. Viruses and worms	a) TCP WrappersConfig b) Move to Agent c) IntrusionDetectionTest d) Need to delete.	6.3.3.18 6.3.3.6
C-MSS-70460	The MSS site Security Management Application Service shall have the capability of generating a notification within a maximum of five minutes of the detection of an intrusion.	IntrusionDetectionTest	6.3.3.6
C-MSS-70500	The MSS EMC Security Management Application Service shall have the capability to coordinate with the site Security Management Application Services, via directives and instructions, the recovery from security compromises.	MsScSMC	6.3.3.13
C-MSS-70510	The MSS site Security Management Application Service shall, upon the detection of a compromise, isolate the compromised input I/O, and the compromised area's output I/O until the compromise has been eliminated.	Operational requirement.	6.3
C-MSS-70520	The MSS EMC Security Management Application Service shall provide office automation support tools to enable the generation of directives and instructions for recovery from detected security events.	MsScSMC	6.3.3.13
C-MSS-70530	The MSS EMC Security Management Application Service shall coordinate, as necessary via directives and instructions, the recovery from security events reported from a site.	MsScSMC	6.3.3.13
C-MSS-70700	The MSS Security Management Application Service shall have the capability to generate intrusion reports on the following: a. Login failures b. Unauthorized access to ECS resources c. Break-ins d. Viruses and worms	MsScReport	6.3.3.12
C-MSS-70710	The MSS Security Management Application Service shall have the capability to generate reports from collected management data.	MsScReport	6.3.3.12
C-MSS-70720	The MSS Security Management Application Service shall have the capability to redirect reports to: a. console b. disk file c. printer	MsScManager	6.3.3.10

C-MSS-75000	The MSS accountability management service shall provide the capability to maintain a user profile database that stores the following information for each registered user: a. Name b. User ID c. Password information 1. password 2. password expiration date d. Assigned privileges e. Mailing address f. Telephone number g. Product shipping address h. E-mail address i. Organization (optional) j. Project affiliation(s) (optional) 1. project name 2. project principal investigator k. User group	MsUsProfile	6.4.3.8
C-MSS-75010	The MSS accountability management service shall be capable of receiving user profile records entered by M&O personnel.	MsAcGuestUserReq	6.4.3.2
C-MSS-75020	The MSS Accountability Management Service shall create a new user account whenever a new record is added to the user profile database.	MsUsProfile	6.4.3.8
C-MSS-76000	The MSS accountability management service shall be capable of retrieving user activity data (user id, type of user activity, data items used (browsed, searched, or ordered), and date/time of activity) from records generated by the SDPS Data Server, Data Processing, and Client subsystems.	MgDatabase	6.9
C-MSS-76010	The MSS accountability management service shall be capable of querying via the Management Data Access service user activity data stored in the Management Database.	MgDatabase	6.9
C-MSS-76020	The MSS accountability management service shall be capable of retrieving all activities associated with a particular user or data item via the Management Data Access service.	MgDatabase	6.9
C-MSS-76030	The MSS Accountability Management Service shall log, for each ECS host, incoming access attempts via: a. telnet b. FTP c. rlogin d. finger.	TCP WrapperConfig	6.3.3.18
C-MSS-76040	The MSS Accountability Management Service shall be capable of reporting audit information to M&O staff via the MUI service.	MgDatabase	6.9

Table A-1. Requirements Trace (Page 38 of 45)

C-MSS-77000	The MSS accountability management service shall be capable of retrieving data processing information (instrument used and date/time of ingest or algorithm used (name and version) and date/time or processing) from records generated by the SDPS Data Processing subsystem.	MgDatabase	6.9
C-MSS-77010	The MSS accountability management service shall be capable of querying via the Management Data Access service all data processing information stored in the Management database.	MgDatabase	6.9
C-MSS-77030	The MSS accountability management service shall be capable of retrieving all data processing information logged for a specified data item.	MgDatabase	6.9
C-MSS-77040	The MSS accountability management service shall be capable of accepting queries for the status of a particular ordered data item from the SDPS Client subsystem.	MgDatabase	6.9
C-MSS-77050	The MSS accountability management service shall be capable of interfacing with the SDPS subsystems to determine the status of an ordered data item to be: a. Item in queue for processing b. Item currently being processed c. Item successfully processed d. Error in processing e. Error in request	MgDatabase	6.9
C-MSS-77060	The MSS accountability management service shall be capable of reporting the requested status of an ordered data item to the SDPS Client subsystem.	MgDatabase	6.9
C-MSS-77070	The MSS accountability management service shall be capable of searching local history logs to find processing data for an ordered data item.	MgDatabase	6.9
C-MSS-77080	The MSS Accountability Management Service shall have the capability to generate reports from collected management data.	MgDatabase	6.9
C-MSS-77090	The MSS Accountability Management Service shall have the capability to redirect reports to: a. console b. disk file c. printer	MgDatabase	6.9
C-MSS-90020	The DBMS shall support a client-server design paradigm with distributed data allocation.	MgDatabase	6.9

C-MSS-90030	The DBMS shall provide security access control based upon userid, role and privileges for the following: a. database b. database object c. database operations	MgDatabase	6.9
C-MSS-90060	The DBMS shall provide an SQL interface with query, update, and administrative functions capabilities.	MgDatabase	6.9
C-MSS-90070	The DBMS shall be in compliance with the SQL-2 of Federal Information Processing System Publication (FIPS PUB) 127-1.	MgDatabase	6.9
C-MSS-90080	The DBMS shall support mathematical operations to generate statistics from management data to include: a. average b. maximum c. minimum d. standard deviation e. sum f. count g. variance	MgDatabase	6.9
C-MSS-90120	The DBMS shall be compatible with the ECS management framework to support the import of the ECS management framework data.	MgDatabase	6.9
C-MSS-90140	The DBMS shall support, or be accessed via, CSS session-establishment services.	MgDatabase	6.9
C-MSS-90150	The DBMS shall support access structures (i.e., single-level indexes, multilevel indexes) to improve the efficiency of retrieval of management data.	MgDatabase	6.9
C-MSS-90160	The DBMS shall support features in compliance with X/Open environment to include the following: a. Hardware independence b. Operating systems independence c. Network protocols independence	MgDatabase	6.9
C-MSS-90170	The DBMS shall provide the following bulk data load capabilities: a. direct writes from data files to database b. loading of files containing fixed and variable length records c. incremental bulk load d. Maintain indexes during data loads	MgDatabase	6.9
C-MSS-90180	The DBMS shall provide the following database backup capabilities: a. Entire database b. Incremental data c. User specified database items.	MgDatabase	6.9
C-MSS-90190	The DBMS shall provide capabilities for specifying frequency, time, and type of backups.	MgDatabase	6.9

Table A-1. Requirements Trace (Page 40 of 45)

C-MSS-90200	The DBMS shall perform on-line disk management functions to include: a. Relocation of database files to different disks b. Expansion of database size by adding new physical data files to it on-line c. Dynamic pre-allocation of contiguous space for tables d. Database objects and indexes can span physical files e. Database objects and indexes can exist on different disks	MgDatabase	6.9
C-MSS-90210	The DBMS shall support the following features: a. Data compression of nulls and variable length character strings, and indexes b. Space reclaimed from deleted records automatically c. Variable-length column storage	MgDatabase	6.9
C-MSS-90230	The DBMS shall provide a transaction roll backward capability to a specified time or state: a. Restore a database b. Restore all or operator selected database objects of any database	MgDatabase	6.9
C-MSS-90240	The DBMS shall provide for automatic database recovery including a means to: a. automatically restore undamaged portions of a database and recover work in progress after a system or component failure b. achieve dynamic backout of database modifications, performed by a failing transaction, that does not affect separate, concurrent tasks	MgDatabase	6.9
C-MSS-90260	The DBMS shall provide a capability to export, archival, and restore a database.	MgDatabase	6.9
C-MSS-90280	The DBMS shall provide the capability to issue and record a database checkpoint.	MgDatabase	6.9
C-MSS-90290	The DBMS shall provide an audit trail of chronological activities in the database.	MgDatabase	6.9
C-MSS-90500	The Report Generator shall be compatible with the DBMS.	MgDatabase	6.9
C-MSS-90510	The Report Generator shall provide a Motif based Graphical User Interface (GUI) for creating ad hoc reports.	MgDatabase	6.9
C-MSS-90520	The Report Generator shall have the capability to generate ad hoc reports from management data maintained in the DBMS.	MgDatabase	6.9

C-MSS-90530	The Report Generator shall provide the capability to format reports to include the report: a. title b. header c. footer d. page number e. date/time of report	MgDatabase	6.9
C-MSS-90570	The Report Generator shall have the capability to generate charts and graphs (e.g., bar, pie, line, etc.) from management data maintained in the DBMS.	MgDatabase	6.9
C-MSS-90600	The Report Generator shall provide the capability to redirect generated reports to: a. console b. disk file c. printer	MgDatabase	6.9
C-MSS-91010	The MSS Office Automation word processing capability shall facilitate the: a. preparation, revision, and recording of documents, messages, reports, and data b. import, transformation, and editing of documents produced by other word processing packages c. insertion of worksheet and graphic images into documents, messages, and reports d. transfer of document, message, and report information to spreadsheet and graphics applications e. printing of documents, messages, reports, and data	MgDatabase	6.9
C-MSS-91020	The MSS Office Automation shall provide a spreadsheet capability that: a. simulates and displays an accountant's worksheet b. enables revisions and calculations on the displayed worksheet's data c. enables transfer of the worksheet data to database, word processing and graphics applications d. enables printing of worksheet information.	MgDatabase	6.9
C-MSS-91030	The MSS Office Automation shall provide a graphics capability that enables: a. the development, modification, recording, and printing of graphic images b. the transfer of graphics images to word processing documents, messages, and reports.	MgDatabase	6.9

C-MSS-XXXXX	The Trouble Ticketing Service shall have a graphical user interface to support the entry and editing of trouble tickets	MsTtManager MsTtUserInterface	6.3.3.3 - provided by Remedy COTS package (additional entry capabilities provided by custom HTML interface); 6.3.3.6
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to automatically notify the originator of the trouble ticket of changes in status	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide an Application Program Interface which supports integration of entry of trouble tickets by other packages.	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to search historical and current trouble tickets by various criteria including keyword, userid, and trouble ticket id.	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to forward trouble tickets from one organization to another to facilitate the escalation of trouble tickets (e.g. from DAAC to SMC)	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to maintain different trouble ticket statuses including : , Open, Work-In-Progress, Closed, Archived.	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to search for trouble tickets relating to the same resource (equipment).	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to search for trouble tickets relating to the same resource (equipment).	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the ability to store the following minimum set of information : unique trouble ticket ID, status, description, associated resources, problem solution, originator, keywords	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall integrate with the MSS framework to allow management and monitoring of its services.	MsTtProxy	4.6.3.6 -specialization of the standard MSS provided proxy agent

Table A-1. Requirements Trace (Page 43 of 45)

C-MSS-XXXXX	The Trouble Ticketing Service shall allow entry of a trouble ticket by any registered user of the system.	MsTtUserInterface	6.6.3.4 - user entry capabilities provided by custom HTML interface or e-mail template
C-MSS-XXXXX	The Trouble Ticketing Service shall provide the capability to generate reports from the its data.	MsTtManager	6.3.3.3 - provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall allow output of reports to either the screen or printer.	MsTtManager	6.3.3.3 -provided by Remedy COTS package
C-MSS-XXXXX	The Trouble Ticketing Service shall provide customization features to allow sites to specify notification and escalation rules.	MsTtManager	6.3.3.3 -provided by Remedy COTS package
C-MSS-xxx10	The Physical Configuration Management Service shall be capable of importing floor plans from existing files	PhysicalConfigurationManager	6.5.1
C-MSS-xxx20	The Physical Configuration Management Service shall provide a graphical interface for adding to and editing the existing floor plan.	PhysicalConfigurationManager	6.5.4.1 and 6.5.4.2
C-MSS-xxx30	The Physical Configuration Management Service shall be capable, through interfacing with the ECS Management framework, of determining and storing information regarding physical components.	PhysicalConfigurationManager	6.5.4.1
C-MSS-xxx35	The Physical Configuration Management Service shall be capable of determining and storing the following information regarding physical components: a. physical device identification b. physical device information c. physical device location d. physical device status	PhysicalConfigurationManager	6.5.3.1
C-MSS-xxx40	The Physical Configuration Management Service shall have the capability to augment the information obtained from ECS Management framework on each component with additional information	PhysicalConfigurationManager	6.5.4.1

C-MSS-xxx50	The Physical Configuration Management Service shall have the capability to allow the entry and storage of information regarding additional physical components that cannot be discerned through the ECS Management framework.	PhysicalConfigurationManager	6.5.4.1
C-MSS-xxx60	The Physical Configuration Management Service shall provide a graphical interface for viewing the physical location of system components on the floor plans.	PhysicalConfigurationManager	6.5.1
C-MSS-xxx70	The Physical Configuration Management Service shall provide a graphical interface for changing the location of the system components.	PhysicalConfigurationManager	6.5.4.2
C-MSS-xxx80	The Physical Configuration Management Service shall be capable of maintaining the following information for all of the physical system components: a. Inventory data (name, purchase date, purchase price, installation date, manufacturer, serial number, physical location) b. Network data (network location, protocols) c. Maintenance data (maintenance date)	PhysicalConfigurationManager	6.5.6.3
C-MSS-xxx90	The Physical Configuration Management Service shall be capable of interfacing with the Management Database in order to store and retrieve data.	PhysicalConfigurationManager	6.5.1
C-MSS-xx100	The Physical Configuration Management Service shall provide a standard set of reports against this data.	PhysicalConfigurationManager	6.5.6.3
C-MSS-xx110	The Physical Configuration Management Service shall provide the ability to produce custom reports against this data.	PhysicalConfigurationManager	6.5.6.3
C-MSS-xx120	The Physical Configuration Management Service shall provide tight integration with the Trouble Ticketing (TT) System including allowing direct access of the TT through the Physical Configuration Management Interface.	PhysicalConfigurationManager	6.5.1

C-MSS-xx140	The Physical Configuration Management Service shall provide the ability to interface with the ECS Management framework to capture status information on each component.	PhysicalConfigurationManager	6.5.1
C-MSS-xx150	The Physical Configuration Management Service shall provide the ability to display the status obtained above within the graphical interface.	PhysicalConfigurationManager	6.5.1

Note: Trouble Ticketing and Physical Configuration Management were not included in the PDR Baseline Level-4 requirements. A CCR has been prepared to add these requirments to the CDR Baseline.

This page intentionally left blank.

Appendix B. Application MIB Definition

B.1 Application MIB

This MIB contains three levels of managed objects: Application Package class, Program class, and Process class. An Application Package may have one or more Programs. A Program may have zero or more Processes. Each object class is an SNMP group in the MIB.

Agent Group

Managed Object Attribute	Description
appAgtUpTime	The value of sysUpTime at the time this agent was last initialized
appAgtLocalPollInterval	The time interval of the polling (e.g. secd) on this host
appAgtLocalPollScope	The scope of the polling
appAgtRetInfoLevel	The level of information which is supposed to return (e.g. fatal-only)

ECS Application Group:

Managed Object Attribute	Description
appTable	A table of the application instances
appIndex	The index which identifies the entry of the application instance
appID	The unique ID which identifies this application package
appName	The name by which the application package chooses to be known
appMajorVersion	The major version of the application package
appMinorVersion	The minor version of the application package
appRevision	The lowest level of version information
appMaintLevel	The level to which this program has been patched
appInstallation	The date and time that this application was installed
appType	The type of application software
appContact	The contact information for this application
appLanguage	The language version of the application software for multilingual software

appUpTime	The value of sysUpTime at the time the application was last initialized
appStatus	The current status of the application. Valid values are...
	(1) Up - indicates the application is currently operational and available.
	(2) Down - indicates the application is unavailable.
	(3) Halted - indicates the application is operational but unavailable.
	(4) Restarting - indicates the process is operation but unavailable but in the process of initializing and will be available soon.
	(5) Not-installed - indicates the application is not installed
appStatusLastChange	The value of sysUpTime the last time the application changed its operational state
appNumOfProgs	The number of program objects which are contained in the application package
appPerfTable	A table which contains the various types and values of the application specific performance metrics.
appIndex	The index which identifies the instance of the application
appPerfIndex	The index which uniquely identifies the performance metric being tracked
appPerfType	The type of performance metric being tracked
appPerfValue	The current value for this performance metric
appPerfThreshold	The threshold value for this performance metric
appFaultTable	A table which contains the various types and values of the application specific faults.
appIndex	The index which identifies the instance of the application
appFaultIndex	The index which uniquely identifies the fault being tracked
appFaultType	The type of fault being tracked
appFaultValue	The current value for this fault
appCfgTable	A table which contains the configuration parameters and their values for this application
appIndex	The index which identifies the instance of the application
appCfgIndex	The index which uniquely identifies the configuration parameter
appCfgParam	The configuration parameter name
appCfgValue	The value of the configuration parameter
appProgTable	A table which track the programs this application contains
appIndex	The index which identifies the instance of the application
appProgID	The program ID

Program Group:

Managed Object Attribute	Description
progTable	A table of the program instances
progIndex	The index which identifies the instance of the program

progID	The unique ID which identifies this program
progName	The name by which the program chooses to be known
progAppID	The ID which refers to the application this program belongs to
progAppName	The name by which the application package chooses to be known
progMajorVersion	The major version of the program
progMinorVersion	The minor version of the program
progRevision	The lowest level of version information
progMaintLevel	The level to which this program has been patched
progExecPath	The execution path of the program
progExecFile	The filename of the execution image of the program
progType	The type of program this is (e.g. CDSD)
progContact	The contact information for this program
progUpTime	The value of sysUpTime at the time the program was last initialized
progStatus	The status of the program currently. Valid values are...
	(1) Up - indicates the program is currently operational and available.
	(2) Down - indicates the program is unavailable.
	(3) Halted - indicates the program is operational but unavailable.
	(4) Restarting - indicates the process is operation but unavailable but in the process of initializing and will be available soon.
	(5) Not-installed - indicates the program is not installed
progTimeOut	The timeout value
progLogPath	The path to the logfile for this program
progNumofProcs	The number of processes currently executing within this program
progFaultTable	A table which contains the various types and values of the program specific faults.
progIndex	The index which identifies the instance of the program
progFaultIndex	The index which uniquely identifies the fault being tracked
progFaultType	The type of fault being tracked
progFaultValue	The current value for this fault
progCfgTable	A table which contains the configuration parameters and their values for this program
progIndex	The index which identifies the instance of the program
progCfgIndex	The index which uniquely identifies the configuration parameter
progCfgParam	The configuration parameter name
progCfgValue	The value of the configuration parameter
progProcTable	A table which contains entries for each of the processes this program contains
progIndex	The index which identifies the instance of the program
progPID	The PID

progAssocTable	A table which contains entries for the applications associated with this program
progIndex	The index which identifies the instance of the program
progAssocIndex	The index which uniquely identifies the associated application
progAssocRemoteApp	The associated application
progAssocAppProtocol	The protocol the program uses to communicate with the application
progAssocAppType	The type of associated application
progAssocDuration	The duration of the association

Process Group:

Managed Object Attribute	Description
procTable	A table of the process instances
procIndex	The index which identifies the instance of the process
procProgID	The ID which refers to the program this process belongs to.
procUpTime	The value of sysUpTime at the time the application was last initialized
procCPUUtil	The current CPU utilization for this process
procMemory	The current memory utilization for this process
procTTY	The TTY of the process
procGID	The GID of the process
procUID	The UID of the process
procPID	The PID of the process
procPPID	The PID of the parent process
procPriority	The priority at which this process is executing
procStatus	The current status of this process
procInRPCCalls	The number of RPC calls received
procOutRPCCalls	The number of RPC calls sent
procInRPCPkts	The number of RPC packets received
procOutRPCPkts	The number of RPC packets sent
procThreads	The number of threads currently executing within this process
procPerfTable	A table which contains the various types and values of the process specific performance metrics.
procIndex	The index which identifies the instance of the process
procPerfIndex	The index which uniquely identifies the performance metric being tracked pertaining to this process
procPerfType	The type of performance metric being tracked
procPerfValue	The current value for this performance metric
procPerfThreshold	The threshold value for this performance metric
procFaultTable	A table which contains the various types and values of the process specific faults.

procIndex	The index which identifies the instance of the program
procFaultIndex	The index which uniquely identifies the fault being tracked
procFaultType	The type of fault being tracked
procFaultValue	The current value for this fault

Trap Group:

Managed Object Attribute	Description
appDiscoveryTrap	A trap for the discovery of an application
appID	The application ID
appIndex	The application instance
appStartupTrap	A type for holding application start traps
appID	The application ID
appIndex	The application instance
appShutdownTrap	A type for holding application stop traps
appID	The application ID
appIndex	The application instance
appProcessFailedTrap	A trap for a fault in a process
appID	The application ID
appIndex	The application instance
progID	The program ID
progIndex	The program instance
procID	The process ID
procIndex	The process instance
appUID	The ID of the user running the process
appRc	The return code from the application
appProcessMissingTrap	A trap for the missing of a process
appThresholdExcd	A type for holding application threshold exceeded traps
appId	The application ID
appIndex	The application instance
progID	The program ID
progIndex	The program instance
procID	The process ID
procIndex	The process instance
appUID	The ID of the user running the process
appPerfIndex	The application performance index
appPerfType	The application performance type
appPerfValue	The application performance value
appPerfThreshold	The application performance threshold

appSecAlert	There is a security incident notified.
appId	The application ID
appIndex	The application instance
progID	The program ID
progIndex	The program instance
procID	The process ID
procIndex	The process instance
appUID	The ID of the user running the process
appSecType	The type of security violation

Abbreviations and Acronyms

ACL	Access Control List
AFS	Andrews File System
AI&T	Algorithm Integration and Test
AIT	Algorithm Integration Team
ANSI	American National Standards Institute
API	Application program (or programming) interface
ASCII	American Standard Code for Information Exchange
ATM	Asynchronous Transfer Mode
ARP	Address Resolution Protocol
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
BB	Bulletin Board
BBS	Bulletin Board Service
BIND	Berkeley Internet Name Domain
BGP	Border Gateway Protocol
BOA	Basic Object Adapter
CAC	Command and Activity Controller
CCB	Change Control Board (Hughes Convention)
CCB	Configuration Control Board (NASA Convention)
CCR	Configuration Change Request
CDS	Cell Directory Service
CDR	Critical Design Review
CDRL	Contract data requirements list
CERES	Clouds and Earth's Radiant Energy System
CIDR	Classless Interdomain Routing
CM	Configuration management
CMAS	Configuration Management Application Service
CMIP	Common Management Information Protocol
CNE	Campus Network Environment
CORBA	Common object request broker architecture
COTS	Commercial off-the-shelf (hardware or software)
CPU	Central processing unit
CSMS	Communications and System Management Subsystem

CSS	Communication Subsystem
DAAC	Distributed Active Archive Center
DADS	Data Archive and Distribution System
DB	Database
DBMS	Database management system
DCE	Distributed computing environment (OSF)
DEC	Digital Equipment Corporation
DECOM	FOS Decommutation Process
DFS	Distributed File System
DID	Data item description
DME	Distributed Management Environment
DNS	Directory Name Service
DOF	Distributed Object Framework
DPR	December Progress Review
DS	Data Server (FOS)
DTS	Distributed Time Server (part of DCE)
ECS	EOSDIS Core System
EDOS	EOS Data and Operations Center
EDF	ECS Development Facility
E-Mail	Electronic Mail
EMC	Enterprise Monitoring and Coordination
EOC	EOS Operations Center (ECS)
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
EP	Evaluation Prototype
ESN	EOSDIS Science Network
EPV	Endpoint Vector
FDDI	Fiber distributed data interface
FDF	Flight Dynamics Facility
FOS	Flight Operations Segment
ftp	File Transfer Protocol
GB	Gigabyte (10^9)
GCDIS	GDS Global Directory Service
GDS	Global Directory Service
GSFC	Goddard Space Flight Center
GUI	Graphic user interface

HAIS	Hughes Applied Information Systems (ECS)
HiPPI	High Performance Parallel Interface
HP	Hewlett Packard
Http	Hyper Text Transfer Protocol
I/F	Interface
I&T	Integration & Test
IBM	International Business Machines, Inc.
ICD	Interface control document
ICMP	Internet Control Messaging Protocol
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IR-1	Interim Release 1
ISO	International Standards Organization
ISO+	IsoCELL (Isolation Cell)
ISS	Internetworking Subsystem of CSMS
IST	Instrument Support Toolkit
IST	Instrument Support Terminal
Kerberos	Security protocol developed by MIT; base for DCE security
Kftp	Kerberized file transfer protocol
KLOC	Kilolines (10^3) of code
Ktelnet	Kerberized telnet
LAN	Local area network
LaRC	Langley Research Center
LLC	Logical Link Control
LOC	Lines of code
LSM	Local System Management
M&O	Maintenance and operations
MBONE	Multicast Backbone
MIB	Management Information Base
MIME	Multimedia Internet Mail
MLM	Mid-Level Manager
MOPITT	Measurement of pollution in the troposphere
MOSPF	Multicast Open Shortest Path First
MR-AFS	Multi-Resident Andrew File System

MSFC	Marshall Space Flight Center
MSS	Systems Management Subsystem
MUI	Management User Interface
NCR	Non-conformance Report
NFS	Network file system
NIC	Network Interface Card
NNTP	Network New Transfer Protocol
NOAA	National Oceanic and Atmospheric Administration
NOLAN	Nascom Operational Local Area Network
NSI	NASA Science Internet
NTP	Network Time Protocol
OA	Off-Line Analysis Process
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
OMG	Object Management Group
OMT	Object Modeling Technique
OO	Object-oriented
OODCE	Object-oriented DCE
OODBMS	Object-oriented database management system
ORB	Object Request Broker
OS	Object Services (CSS building blocks)
OSF	Open Software Foundation
OSI	Open System Interconnect
OSI-RM	OSI Reference Model
OSPF	Open Shortest Path First
PAC	Privilege Attribute Certificate
PDR	Preliminary Design Review PDR-A
PDU	Protocol Data Unit
PPP	Point-to-Point Protocol
POSIX	Portable Operating System Interface for Computer Environments
PSC	Pittsburgh Supercomputing Center
PTGT	Privilege Ticket Granting Ticket
RDBMS	Relational database management system
RFA	Remote File Access
RFC	Request for comments
RIP	Routing Information Protocol

RMA	Reliability, Maintainability, Availability
RMON	Remote Monitoring
RMP	Reliable Multicast Protocol
RPC	Remote procedure call
RTS	Real-Time Server (FOS)
SCF	Science Computing Facility
SDPF	Sensor Data Processing Facility
SDR	Software/System Design Review
SDR	Sensor data record
SGI	Silicon graphics
SLOC	Source lines of code
SMC	System Monitoring and Coordination Center
SMDS	Switched Multi-megabit Data Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network
SQL	Simple Query Language
TBD	To be determined
TCP/IP	Transmission Control Protocol/Internet Protocol
TGT	Ticket Granting Ticket
TMN	Telecommunications Management Network
TRMM	Tropical Rainfall Measurement Mission
TSDIS	TRMM Science Data Information System
UDP	User Datagram Protocol
UIOAR	User Interface Off-Line Analysis Request Window
URL	Universal Resource Locator
US	User Station (FOS)
UUID	Universal Unique Identifier
UTC	Universal time code
V0	Version 0
VT	Virtual Terminal
WAN	Wide area network
WWW	World Wide Web
X	X Protocol
X.500	OSI standard for directory services (207)
XDS	X/Open Directory Service
XFN	X/Open Federated Naming
XOM	X/Open OSI-Abstract-Data Manipulation