

6. MCI - Management Software CSCI

The Management Software CI consists of the Mode Management Service, Fault Management Service, the Performance Management Service, the Security Management Service, the Accountability Management Service, the Physical Configuration Management Service, the Report Service, the Billing and Accounting Service, the Management DBMS, the Trouble Ticketing Service, the Management Data Access Service, the Management Database, the User Comment Survey Tool, and Backup and Restore Management. The Management Software CI context is provided in Figure 6-1.

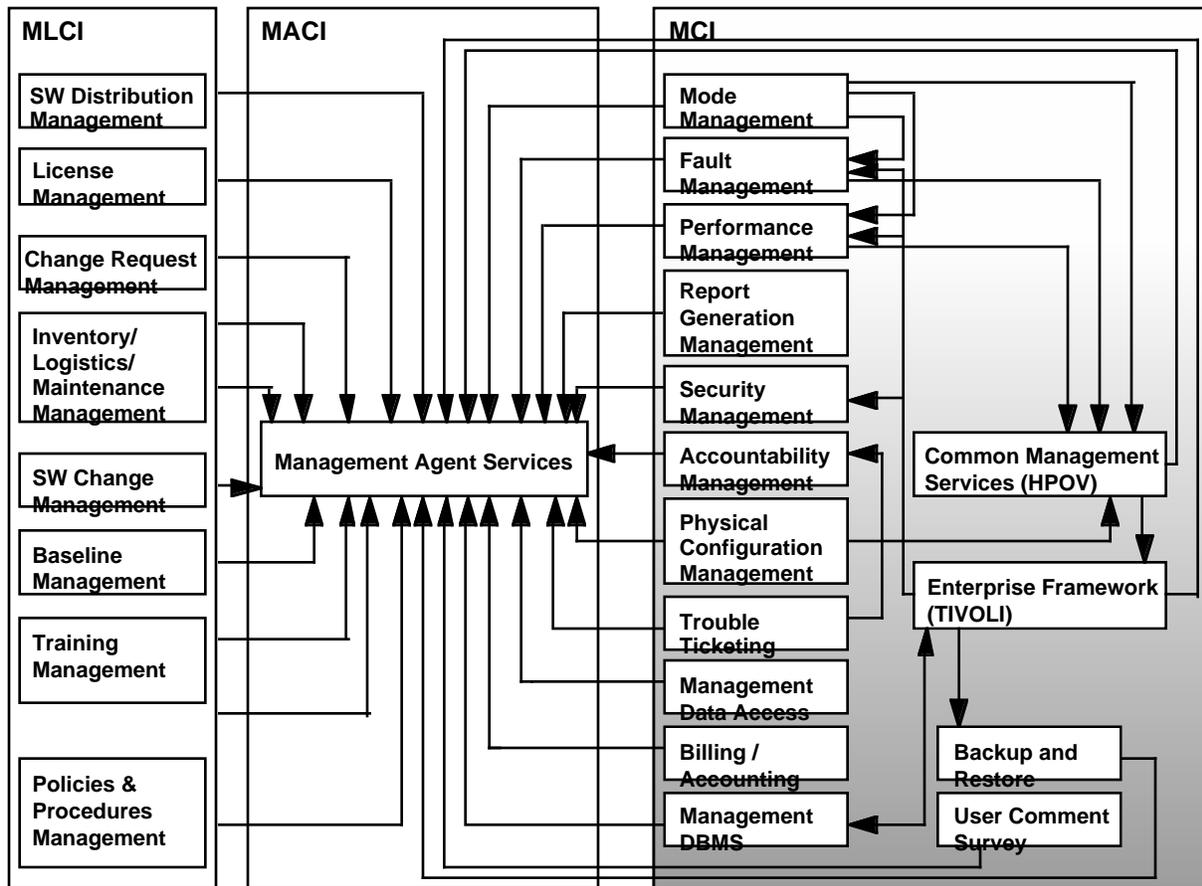


Figure 6-1. Management Software CI Context

6.1 Mode Management

6.1.1 Mode Management Overview

Mode Management addresses the planning, initiation, execution, monitoring, and control of various system activities. These activities include operations, testing, and training. Each unique activity is classified as a mode. Mode Management enables the execution of multiple modes such that each mode functions without interfering with the other and each mode maintains data integrity throughout its execution. For example, testing a data server application within the same system that is supporting operational activities. The test version of the data server must not interfere/interact with the operational version of the data server. In addition, it will only see and have access to interface components that have been specifically set up and initiated under the same test mode.

The mode management design does not limit the number of concurrently executing modes, however, performance considerations need to be addressed prior to the initiation of an additional mode. It will support multiple test and training mode instances, but due to data persistence issues there can only be one operational mode of execution at any given time. Once an application has been initiated within a given mode, it will remain in that mode for the life of the process.

The site Resource Manager will have a view of all the components supporting each mode. This view is provided through HP OpenView and can be configured to display all of the components from every mode on one window or to display the components associated with each mode in separate windows. Software components will be duplicated, and hardware resources will be isolated whenever possible to support an additional mode. However, there will be shared resources, both hardware and software, that require special consideration to enable mode management support.

Section 6.1 focuses on the detailed design for the Mode Management Service (MMS) within MSS. Mode Management compliance within each subsystem is addressed in the subsystem's associated detailed design specification. However, a brief synopsis of the mode management design from a system perspective is provided in the paragraph 6.1.1.1 to enhance the reader's overall understanding of the MMS design. The Mode Management Service design overview is provided in section 6.1.1.2, followed by the Object and Dynamic models.

6.1.1.1 Mode Management within ECS Context

Mode Management, from a system perspective, consists of procedural activities and infrastructure control. The procedural aspects of mode management address mode planning, resource allocation, and system configuration activities. Infrastructure control ensures the software subsystems will recognize the different modes of execution and that data integrity and process distinction will be maintained within each mode.

6.1.1.1.1 Procedural Activities

The procedural activities that are required to support a mode of execution are as follows:

1. Obtain and read the test/training/simulation/etc. plan from the plan originator.
2. Obtain a unique mode identifier from the Resource Manager.
3. Determine the scope of the new activity. Will the test be inter-DAAC or intra-DAAC? What services/components are involved? etc..

4. Determine whether the addition of the new mode will impact the performance of any other simultaneously executing modes. If so, how can this impact be minimized.
5. Coordinate M&O personnel required for the support of the additional mode.
6. Use the Resource planning tool to identify and allocate the necessary hardware and software resources required to support the additional mode.
7. Notify the SMC of the intended plan. If the new mode involves multiple DAACs, the SMC may become involved in the planning process.
8. Configure the directory namespace (i.e. Cell Directory Structure (CDS)) for the new mode based on the mode identifier. This activity may not be required if the CDS had been previously configured using the same mode identifier.
9. Create an HPOV map in support of the new mode. (This activity may not be required if an HPOV map had been previously configured for the same mode).
10. Establish a new HPOV session and load mode specific map into HP OpenView.
11. Identify support data sets, test software, control files, and test procedures necessary for the execution of the new mode.
12. Establish directory partitions within the file system and databases based on the mode identifier.
13. Load support data sets, configuration files, control files (drivers), and test software into mode established partitions.
14. Initiate Mode Management Service from within the HP OpenView management environment.
15. Activate new Mode within system using Mode Management Service.
16. After mode completion backup mode associated output data sets
17. Deactivate mode using the MMS and return system to a pre configured state.

6.1.1.1.2 Infrastructure Control

The System Infrastructure will ensure data integrity between modes and provide process distinction and separation where feasible. In the case of COTS, where running multiple instances of the executable may not be possible, a single application will be required to handle requests from multiple modes. These shared resources require special consideration to ensure integrity between modes. These capabilities will be provided as part of the system infrastructure which have been designed to accommodate mode management.

Infrastructure control is based on the mode identifier. The DCE CDS and all data partitioning will be based on this identifier. These entities will be pre configured, as part of the procedural activities required to support a mode, to accept mode specific requests. Applications are hard coded with a mode attribute variable where mode specific requests are required. The application obtains the mode identifier at startup which it will use for all subsequent mode specific interprocess communications and data I/O requests.

The mode identifier specifications are as follows:

- Maximum of six (6) characters

- Alpha-numeric including the underscore character
- Must not start with an underscore character
- Characters are case sensitive

Examples are "ops", "ts1", "ts2", "tr1", "shared"

Mode Identifier Guidelines:

- The "ops" mode identifier is mandatory for all operational (production) mode activities.
- The "shared" mode identifier is only used internally to designate a common CDS namespace registration point for mode independent applications.

6.1.1.1.2.1 Data Integrity

Data integrity must be maintained between each software mode. For example, operational processes can never read from test/training data sets and test/training data can never be written to operational data sets. All data required to support an additional mode will be duplicated. It will be partitioned by using separate volumes or by a hierarchical directory structure within the same volume such that all reading/writing of data will be segregated between software modes. All data required to support a given mode must be clearly defined, segregated, and duplicated prior to the initiation of the new activity. The segregation of the data will be based on the mode identifier.

For data storage in the UNIX environment, the data will be segregated on the same (or different) disk volume(s) in a hierarchical directory structure based on the mode identifier. The applications will access the data using the mode identifier as part of the directory path.

For data storage within a DBMS, the data will be segregated using a separate database or tape group. Applications accessing the DBMS will pass the mode identifier to the DBMS interface class which will access the corresponding mode specific database.

6.1.1.1.2.2 Process Distinction and Separation

Process distinction and separation for custom developed applications is accomplished via DCE. The Process Framework (PF) will ensure mode specific process communication by registering each server application into the DCE CDS namespace within the appropriate mode hierarchy.

When a server starts up, it registers itself in the CDS directory structure via the PF. Each entry in the CDS is uniquely identified by the server name and it's UUID. If CDS is given the UUID, it will return the rest of the name that is actually registered. Part of the administration for DCE is to setup the CDS directory structure. This is where the group, location, and mode combinations will be initially set up. This way, when registration in CDS occurs, it is known where to place the name. One function of setting up a new mode will be to manually add a new path to the CDS directory structure for the given activity. All applications will then automatically register to this new path based on the mode identifier obtained at startup. The mode identifier will be passed from the Management Framework (HP OpenView) to the remote executable startup scripts as a command line argument. The startup script will set the mode as an environment variable, which is necessary for assigning a mode specific UUID required for ACL management, and then also pass it in as a command line argument to the application's main. Once an application has been initiated within a given mode, it remains in that mode for the life of the process. When clients do server lookup calls, they will only see and find the servers running within the mode they are executing.

Some applications like the Management Data Access (MDA), Subagent, and virtually all COTS products will be mode independent, i.e. a single instantiation of the application will support multiple modes. When mode specific interfacing or data I/O is required the mode identifier will be passed into the application. For example the MDA, which is mode independent, processes events and routes them to the management database. For mode management support it will extract the mode attribute from the event class and then use this to route the event to the mode specific management database via the DBMS interface class. Mode independent applications will register under the "shared" hierarchical directory structure within the CDS namespace. If an event is generated by a mode independent application, the MDA will copy to all active management database(s) independent of mode. This will ensure the autonomy of each mode specific management database.

6.1.1.2 MSS Mode Management Service

The Mode Management Service (MMS) within MSS provides mode initiation, monitoring, and controlling capabilities. These capabilities are provided by HP OpenView with custom code extensions. The MMS is a custom developed application which will interface with HP OpenView via HP's *ovw* APIs and with the agents via HP's *ovsnmp* APIs. A high level overview of this interface is presented in Figure 6.1.1. Communication to and from HP OpenView (and therefore the MMS) is via SNMP protocol. SNMP Gets are sent directly to the remote agent while SNMP Traps and Sets are routed through the local Deputy Agent. The Deputy Agent is used to encapsulate the SNMP call into a more reliable RPC call for transport to/from the remote host.

Through the use of Agents, each process can be controlled and monitored from within HP OpenView. The MMS will incorporate the mode management user interface directly into the HP OpenView GUI, providing methods to activate and deactivate a mode. In addition it provides a mode specific user interface for accessing CSS life-cycle control (suspend, resume, and shutdown). Monitoring capabilities are provided as standard functionality within HP OpenView and will be enhanced to reflect mode specific status propagation of software system, subsystem, application, program, and process level entities. Hardware is mode independent so it's status will be reflected within every mode in which it is configured.

HP OpenView will support multiple modes through the use of separate HPOV sessions. A new session can be brought up on the same host or on separate hosts. Figure 6.1.2 shows a multi-session view of how HP OpenView can be configured on separate hosts to support multiple modes. Every session can load one and only one map. The map can have any number of submaps defined that will decompose the basic high level map representation. Each mode will have it's mode specific map (and associated submaps) predefined to recognize and support the hardware and software components that are supporting the given mode. Submap context will be determined based on the mode identifier.

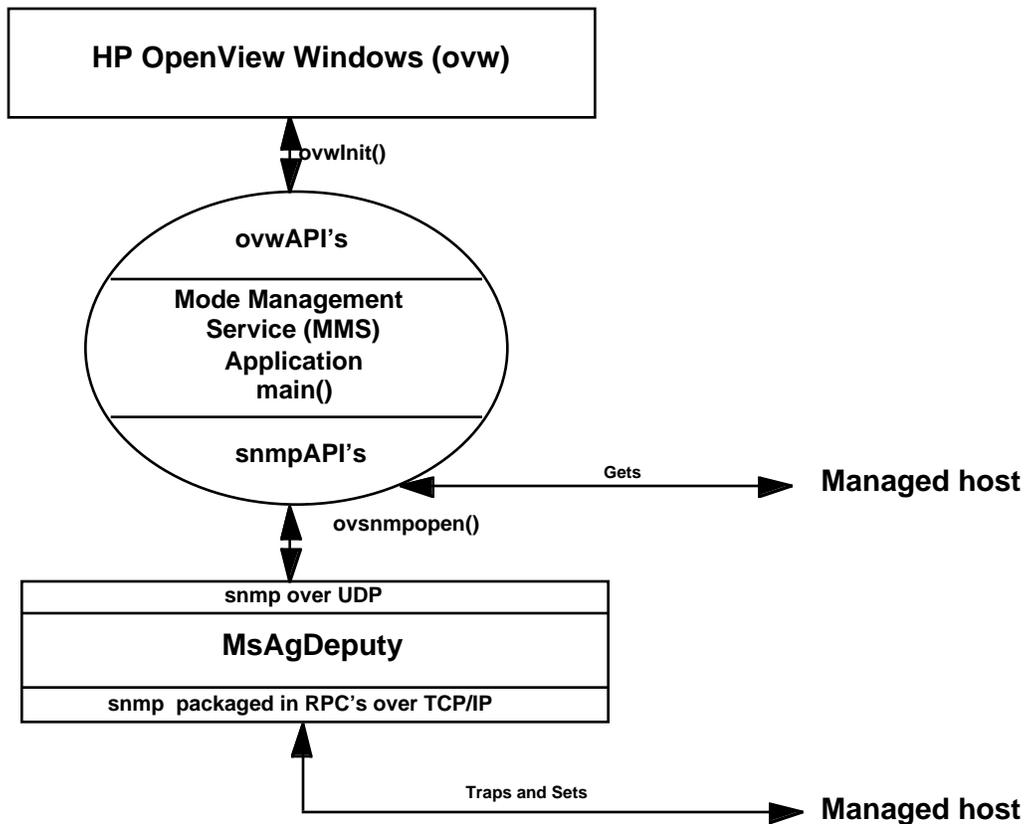


Figure 6.1-1. Mode Management Service Interface Overview Diagram

The MMS, initiated from within HP OpenView, is an event driven application. It will initialize the *ovw* and *ovsnmp* API calls, register the MMS callbacks, and then enter a main event loop. This functionality is similar to X-windows processing. Then when an action occurs, such as an operator selecting "activate mode" from the HP OpenView GUI, an event is triggered and the applicable callback is executed. The objects detailed in the MMS Object diagram are instantiated from within the callback operations. The MMS incorporates the following callback operations (more detail is provided in paragraph 6.1.1.4):

- *ovwActivateMode()*
- prompt operator for new mode identifier
- prompt operator for simulation time if non-ops mode is entered
- add new mode identifier to active mode list
- issue *ovsnmp* API call to Deputy Agent to activate the new mode. (Detailed information is provided on this in the dynamic model).
- *ovwDeactivateMode()*
- Ensure all executables within mode are inactive

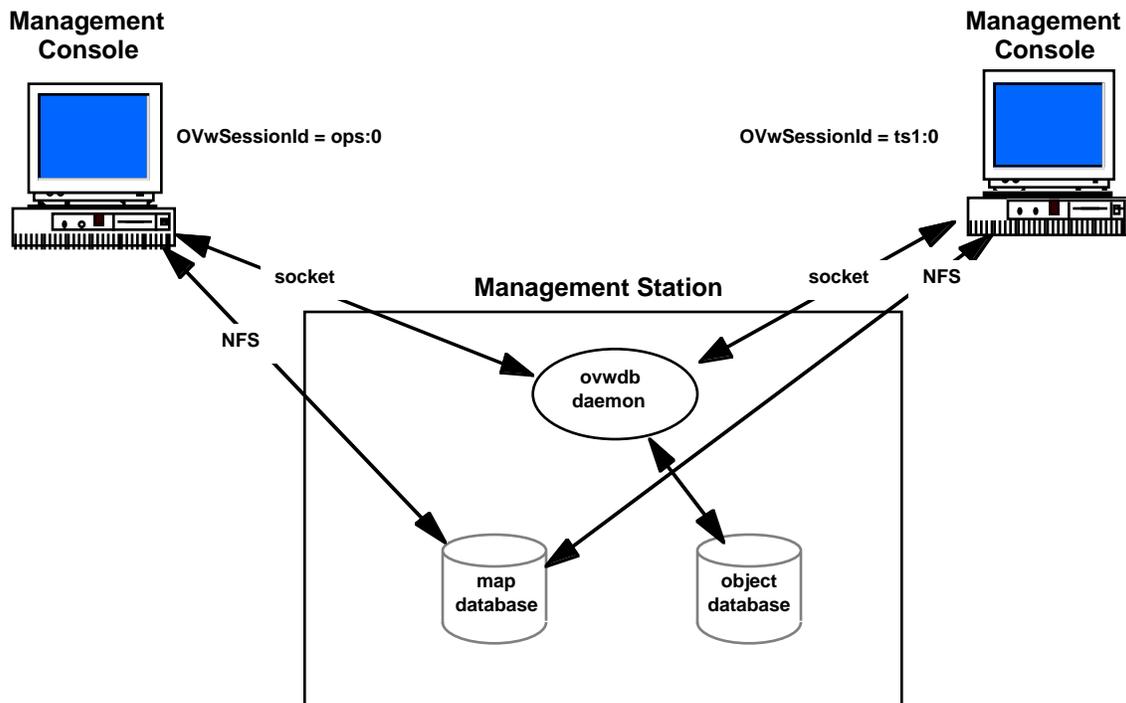


Figure 6.1-2. HP OpenView Multi-Session View Diagram

- issue ovsnmp API call to Deputy Agent to deactivate the selected mode.
- remove deactivated mode from active mode list
- ovwShutdownExec()
- decompose system/subsystem level ovw objects into application/program/process level objects.
- issue ovsnmp API call to Deputy Agent to shutdown executable.
- ovwSuspendExec()
- decompose system/subsystem level ovw objects into application/program/process level objects.
- issue ovsnmp API call to Deputy Agent to suspend executable.
- ovwResumeExec()
- decompose system/subsystem level ovw objects into application/program/process level objects.
- issue ovsnmp API call to Deputy Agent to resume executable.

When a new mode is activated, the subagent's MsAgDiscoverer class will add the new mode identifier to its valid modes list. Since all applications and executables require an associated configuration file in the subagent's configuration directory, the MsAgDiscoverer class uses this list to determine the configuration directory trees to span when it searches for installed applications.

When the new mode is added, it issues a DiscoverNow() call and searches each mode's configuration tree. When new applications are discovered (and they will be since the executables that support the new mode are now recognized by the subagent), the subagent will issue an SNMP trap to HP OpenView to add the new object to the ovw database. Every application and executable is represented internally within HP Openview as an object. This object contains all of the executable's associated attributes based on the information contained in it's corresponding configuration file. The object's mode (i.e. executable's mode) that was passed back from the subagent that discovered it, is also stored in the object database. HP OpenView objects are represented as symbols on the HP OpenView maps/submaps. When it receives the snmp trap to add the new object it will add it's corresponding symbol to the appropriate submap within the applicable mode's session. The symbol will be in an inactive state since the actual executable has not been started yet. It has merely registered within the management framework. This action occurs for all the applications/executables that are in support of the new mode. From this state the operator can start the applications/executables as part of the normal HP OpenView startup process.

In summary, the Mode Management Service will:

- Incorporate Mode Management Service functionality into the HP OpenView GUI.
- Support independent displays for each different mode of execution.
- Provide methods for activating and deactivating the system to a given mode.
- Enable startup/shutdown/suspend/resume activities for each process by utilizing CSS provided life cycle services.
- Provide the capability to enter a simulated time value for any non-ops mode if required.
- Enable application/program/process level monitoring within each mode.

6.1.1.3 Mode Management COTS

HP OpenView Network Node Manager (NNM) has been selected as the ECS Management Framework. This COTS product inherently provides the capabilities for fault and performance management of TCP/IP networks (SNMP devices). Mode Capabilities will be added via a custom Mode Management Service application. In the Object Model, Section 6.1.6, HP OpenView Network Node Manager is represented by the object labeled ManagementFramework. This product provides capabilities and features to allow customization for mode management of the ECS network. This customization, represented by MsMmMode, MsMmModeInit, MsMmModeTerm, MsMmCtrl, MsMmSuspend, MsMmResume, and MsMmShutdown in the Object Model, includes the following tasks:

- creation of maps and submaps to include the separation into different maps of the processes associated with a particular mode of execution
- add discovered managed objects to the appropriate submaps to graphically represent the topology of the ECS network
- change and propagate status of managed object based on faults/events
- control the life-cycle services (startup/shutdown/suspend/resume)
- seamlessly incorporate Mode Management Service functionality into the HP OpenView GUI.

- support independent displays for each different mode of execution.
- methods for activating and deactivating the system to a given mode.
- capability to enter a simulated time value for any none "ops" mode if required.
- application/program/process level monitoring within each mode.
- definition of monitoring criteria
- definition of thresholds on attribute values
- definition of notification mechanisms
- definition of forwarding criteria

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 mode management of non-SNMP entities such as ECS applications. This custom development is illustrated in the appropriate sections of the object model.

6.1.1.4 MMS Main Loop and Callbacks

The PDL for the MMS, event driven, main loop as well as for the MMS callbacks is as follows:

Basic Main Loop Structure:

```
main(argc, argv)
{
    .....
    ovw initializations
    .....
    ovsnmp initializations
    .....
    add callbacks
    .....
    mainEventLoop(...)
}

```

Callback Structure:

OVwActivateMode()

```
{
    // instantiate a mode init object
    MsMmModelInit newMode;

    // call member function to tell agent to activate this new mode
    newMode.ActivateMode();
}

```

OVwDeactivateMode()

```
{
    // instantiate a mode term object with the mode to deactivate
    MsMmModeTerm killMode(RWCString mode);

    // call member function to see if valid and tell agent to deactivate mode
    killMode.DeactivatMode();
}

```

OVwSuspendExec(ovwObject *objectptr)

```
{
    // create an application level object pointer if needed
    ovwObject *appObject;
    EcTint seconds;

    // obtain number of seconds until suspend
    ovwAPIGUI("Enter Number of seconds until suspend: ", &seconds);

    // system and subsystem level ovwObjects need to be broken
    // down into application level objects.
    if (objectptr->objLevel == "system" || objectptr->objLevel == "subsystem")
    {
        traverse down system or subsystem object tree, then
        for each (appObject = objectptr->application level object) do
        {
            // create a suspend object for each application level ovw object
            MsMmSuspend suspendObj(appObject->nTblID, appObject->rowIndex,
                                   appObject->hostID, appObject->seconds);

            // issue suspend to agent
            suspendObj.suspendExec();
        }
    }
    else
    {
        // create a suspend object for the (app, prog, or process level) ovw object
        MsMmSuspend suspendObj(objectptr->nTblID, objectptr->rowIndex,
                               objectptr->hostID, objectptr->seconds);

        // issue suspend to agent
        suspendObj.suspendExec();
    }
}
```

OVwShutdownExec(ovwObject *objectptr)

```
{
    // create an application level object pointer if needed
    ovwObject *appObject;
    EcTint seconds;

    // obtain number of seconds until shutdown
    ovwAPIGUI("Enter Number of seconds until shutdown: ", &seconds);

    // system and subsystem level ovwObjects need to be broken
    // down into application level objects.
    if (objectptr->objLevel == "system" || objectptr->objLevel == "subsystem")
    {
        traverse down system or subsystem object tree, then
        for each (appObject = objectptr->application level object) do
        {
            // create a shutdown object for each application level ovw object
            MsMmShutdown shutdownObj(appObject->nTblID, appObject->rowIndex,
```

```

        appObject->hostID, appObject->seconds);
        // issue shutdown to agent
        shutdownObj.shutdownExec();
    }
}
else
{
    // create a shutdown object for the (app, prog, or process level) ovw object
    MsMmShutdown shutdownObj(objectptr->nTblID, objectptr->RowIndex,
        objectptr->hostID, objectptr->seconds);
    // issue shutdown to agent
    shutdownObj.shutdownExec();
}
}
OVwResumeExec(ovwObject *objectptr)
{
    // create an application level object pointer if needed
    ovwObject *appObject;

    // system and subsystem level ovwObjects need to be broken
    // down into application level objects.
    if (objectptr->objLevel == "system" || objectptr->objLevel == "subsystem")
    {
        traverse down system or subsystem object tree, then
        for each (appObject = objectptr->application level object) do
        {
            // create a resume object for each application level ovw object
            MsMmResume resumeObj(appObject->nTblID, appObject->RowIndex,
                appObject->hostID);
            // issue resume to agent
            resumeObj.resumeExec();
        }
    }
    else
    {
        // create a resume object for the (app, prog, or process level) ovw object
        MsMmResume resumeObj(objectptr->nTblID, objectptr->RowIndex,
            objectptr->hostID);
        // issue resume to agent
        resumeObj.resumeExec();
    }
}
}

```

6.1.2 Mode Management Context

The Mode Management Service provides a means to initiate/terminate modes of execution, and to monitor and control various CSCI components as managed resources. The managed resources include SDPS, FOS, CSMS services.

The Management Agent, Master Agent, and Server Objects co-resident with the managed objects provide/enable the monitoring and control of any process within each mode. The Mode Management Service context diagram is shown in Figure 6.1-3.

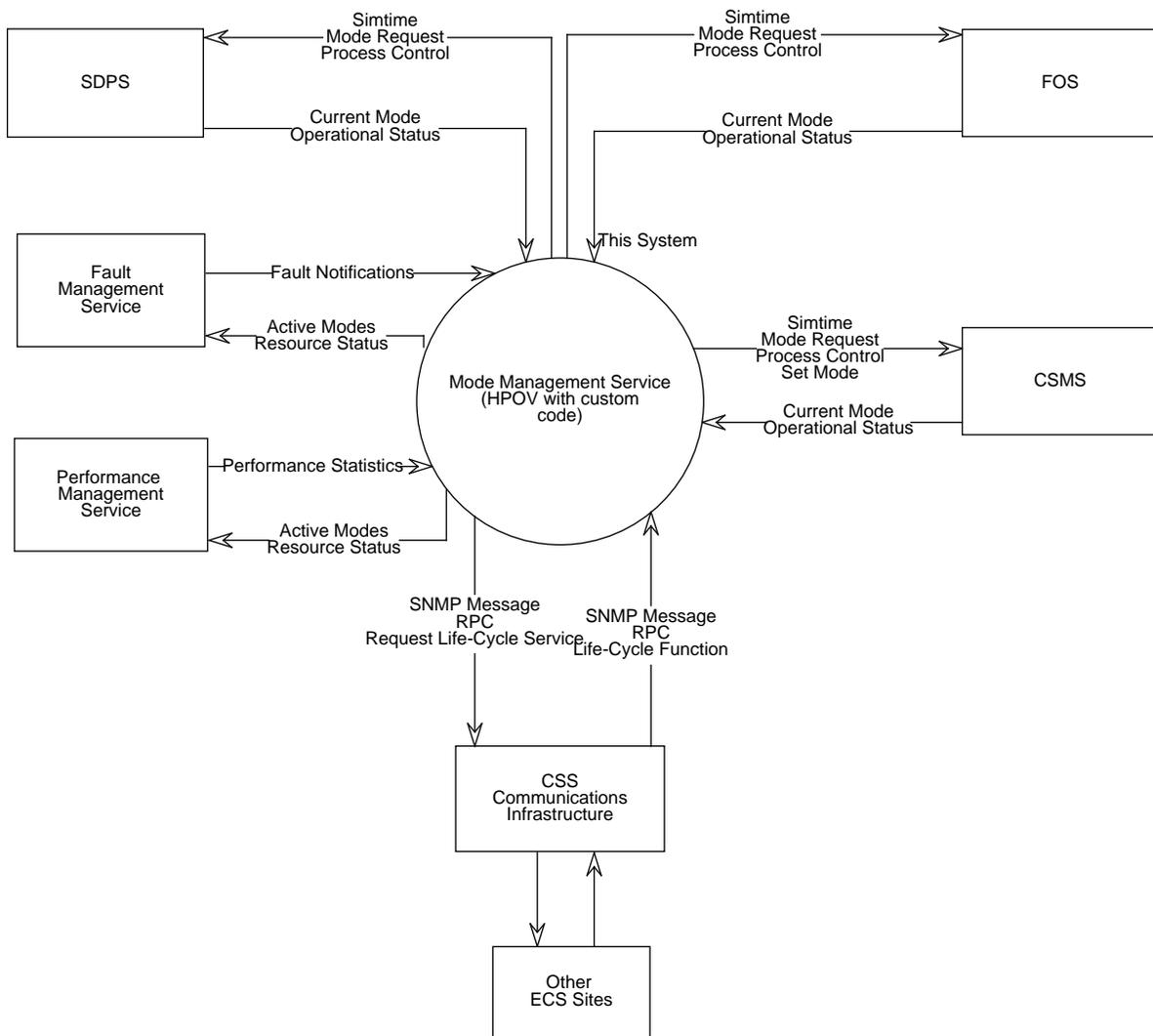


Figure 6.1-3. Mode Management Context Diagram

6.1.3 Mode Management Object Model

The Mode Management object model is shown in Figure 6.1-4.

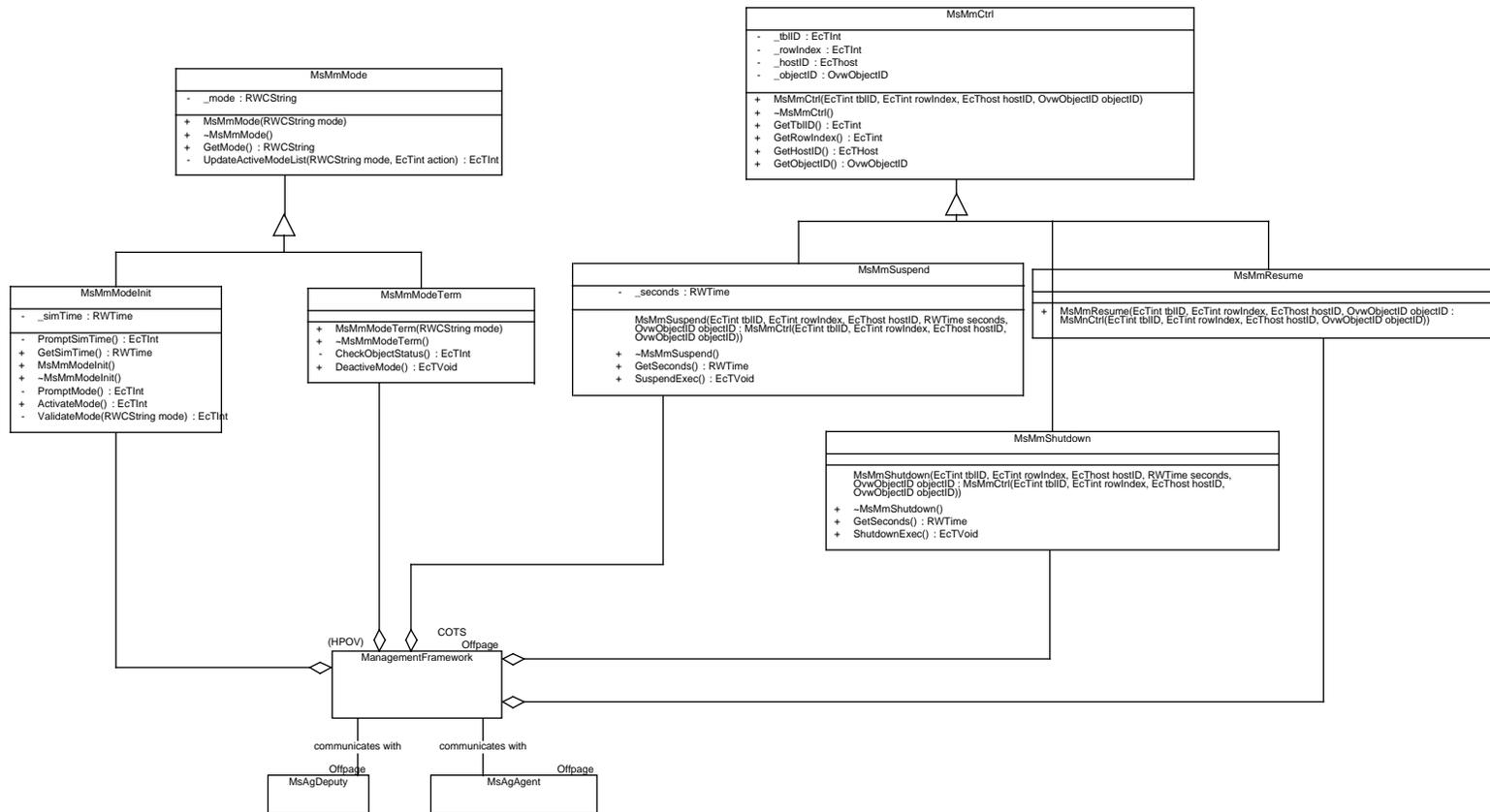


Figure 6.1-4. Mode Management Object Model Diagram

6.1.3.1 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: MsAgAgent communicateswith

Class: MsAgDeputy communicateswith

6.1.3.2 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. 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: ManagementFramework communicateswith

6.1.3.3 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:

None

Associations:

The MsAgDeputy class has associations with the following classes:

Class: ManagementFramework communicateswith

6.1.3.4 MsMmCtrl Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

__tblID

Attributes:

_hostID - The ID of the host where the process is running.

Data Type:EcThost

Privilege:Private

Default Value:

_objectID - The object ID in the HP Open View database.

Data Type:OvwObjectID

Privilege:Private

Default Value:

_rowIndex - The row index of the process table. Used by the MsAgTblMgr to access the table.

Data Type:EcTInt

Privilege:Private

Default Value:

_tblID - The process/application/program table ID.

Data Type:EcTInt

Privilege:Private

Default Value:

Operations:

GetHostID - Accessor function to get the _hostID attribute.

Arguments:

Return Type:EcTHost

Privilege:Public

PDL://{

//return (_hostID);

//}

GetObjectID - Accessor function to get the _objectID attribute.

Arguments:

Return Type:OvwObjectID

Privilege:Public

PDL://{

//return (_objectID);

//}

GetRowIndex - Accessor function to get the _rowIndex attribute.

Arguments:

Return Type:EcTint

```
Privilege:Public
PDL://{
//return ( _rowIndex );
//}
```

GetTblID - Accessor function to get the _tblID attribute.

```
Arguments:
Return Type:EcTint
Privilege:Public
PDL://{
//return ( _tblID );
//}
```

MsMmCtrl - No description

```
Arguments:EcTint tblID, EcTint rowIndex, EcThost hostID, OvwObjectID objectID
Return Type:Void
Privilege:Public
PDL: // {
// }
```

~MsMmCtrl - The destructor for the class.

```
Arguments:
Return Type:Void
Privilege:Public
PDL: // {
// }
```

Associations:

The MsMmCtrl class has associations with the following classes:

None

6.1.3.5 MsMmMode Class

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

Attributes:

_mode - The mode of a process such as ops, ts1, ts2, ...
Data Type:RWCString
Privilege:Private
Default Value:

Operations:

GetMode - Accessor function to get the mode attribute.

Arguments:
Return Type:RWCString
Privilege:Public
PDL: // {
// return (_mode);
// }

MsMmMode - The default constructor for the class. Accepts the mode attribute.

Arguments:RWCString mode
Return Type:Void
Privilege:Public
PDL: // {
// _mode = mode;
// }

UpdateActiveModeList - Write a new mode to (or delete a mode from) Active mode file.

Arguments:RWCString mode, EcTint action
Return Type:EcTint
Privilege:Private
PDL: // {
// lock_file();
// if (action == ADD)
// write new mode to the file
// else
// remove mode from file
// unlock_file();
// }

~MsMmMode - The destructor for the class.

Arguments:
Return Type:Void
Privilege:Public
PDL:// {
// }

Associations:

The MsMmMode class has associations with the following classes:

None

6.1.3.6 MsMmModelnit Class

Parent Class:MsMmMode

Public:No

Distributed Object:No

Purpose and Description:

Attributes:

_simTime - The simulation time for test mode.

Data Type:RWTime

Privilege:Private

Default Value:

Operations:

ActivateMode - Add the new mode to active mode file.

Arguments:

Return Type:EcTInt

Privilege:Public

PDL: // {

// RWCString mode;

// RWCString sim_time;

//

// mode = MsMmMode::GetMode();

//

// sim_time = GetSimTime();

//

// MsMmMode::UpdateActiveModeList (mode , ADD);

//

// SnmpAPIActivateMode (mode, sim_time);

// }

GetSimTime - Accessor function to get the simulation time for non-ops modes.

Arguments:

Return Type:RWTime

Privilege:Public

PDL: // {

//return (_simTime);

```
// }
```

MsMmModeInit - The default constructor.

Arguments:

Return Type:Void

Privilege:Public

PDL:

PromptMode - Get the mode for the process and validate it. Update the ::mode attribute.

Arguments:

Return Type:EcTInt

Privilege:Private

PDL://{

```
//RWCString mode;
```

```
//EcTInt mode_status = -1;
```

```
//
```

```
//OVWAPIGUI ("Enter desired mode");
```

```
//
```

```
//if ( ValidateMode ( mode ) )
```

```
//{
```

```
//MsMmMode::_mode = mode;
```

```
//mode_status = 0;
```

```
//}
```

```
//return (mode_status);
```

```
// }
```

PromptSimTime - Get the simulation time for the test mode, validate it, and update the _simTime attribute.

Arguments:

Return Type:EcTInt

Privilege:Private

PDL://{

```
//RWCString sim_time;
```

```
//sim_time = OVWAPIGUI ("Enter desired simulation time of mode")
```

```
//if ( sim_time.isValid() )
```

```
//_simTime = sim_time;
```

```
// }
```

ValidateMode - Validate the requested mode by consulting the MMS Current modes file.

Arguments:RWCString mode

Return Type:EcTInt

Privilege:Private

PDL: // {

```
// lock file
```

```
//
```

```
// traverse the MMS current mode file to make sure that
// name entered is not in use.
//
// unlock file
// return (-1 or 0 )
// }
```

~MsMmModeInit - The destructor for the class.

Arguments:

Return Type:Void

Privilege:Public

PDL: // {

// }

Associations:

The MsMmModeInit class has associations with the following classes:
ManagementFramework (Aggregation)

6.1.3.7 MsMmModeTerm Class

Parent Class:MsMmMode

Public:No

Distributed Object:No

Purpose and Description:

**

Attributes:

All Attributes inherited from parent class

Operations:

CheckObjectStatus - Checks the object status in the HP Open View database.

Arguments:

Return Type:EcTInt

Privilege:Private

PDL://{

//RWCString mode;

//mode = GetMode (); /* current mode */

//for every object in OVW object DB (ovwdbAPIs)

//{

//if (object.mode == mode) then

//if (object.mode.status != INACTIVE) then

```

//add object to inactive_list
//}
//
//if (inactive_list != NULL )
//{
//ovwAPIGUI ("unable to deactivate mode, mode contains active
// processes. The following processes must be
// first shutdown")
//return (-1);/* NOTE: this should be EcUtStatus */
//
//return (0);
// }

```

DeactiveMode - Deactivate the mode and update the active mode list.

Arguments:

Return Type:EcTVoid

Privilege:Public

PDL:// {

//RWCString mode;

//if (CheckObjectStatus() == 0) /* ensure all executables within
// the mode are terminated */

//{

//mode = GetMode ();

//following function calls MsAgDiscoverer::DeactivateMode() which
//removes the mode from agent's internal table and issues a
//discoverNow operation to update the status

//

//snmpAPIDeactivateMode (mode);

//

//MsMmMode::UpdateActiveModeList (mode, REMOVE)

//

//}

// }

MsMmModeTerm - The default constructor which accepts the mode that needs to be terminated.

Arguments:RWCString mode

Return Type:Void

Privilege:Public

PDL: // {

//MsMmMode::_mode = mode;

// }

~MsMmModeTerm - The destructor for the class.

Arguments:
Return Type:Void
Privilege:Public
PDL: // {
// }

Associations:

The MsMmModeTerm class has associations with the following classes:
ManagementFramework (Aggregation)

6.1.3.8 MsMmResume Class

Parent Class:MsMmCtrl
Public:No
Distributed Object:No
Purpose and Description:
**

Attributes:

All Attributes inherited from parent class

Operations:

MsMmResume -
Arguments:EcTint tblID, EcTint rowIndex, EcThost hostID, OvwObjectID objectID :
MsMnCtrl(EcTint tblID, EcTint rowIndex, EcThost hostID, OvwObjectID objectID)
Return Type:Void
Privilege:Public
PDL://{
//}

Associations:

The MsMmResume class has associations with the following classes:
ManagementFramework (Aggregation)

6.1.3.9 MsMmShutdown Class

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

_seconds

Attributes:

All Attributes inherited from parent class

Operations:

GetSeconds - Accessor function to get the number of seconds.

Arguments:

Return Type:RWTime

Privilege:Public

PDL://{

//return (_seconds);

// }

MsMmShutdown

Arguments:EcTint tblID, EcTint rowIndex, EcThost hostID, RWTime seconds,

OvwObjectID objectID : MsMmCtrl(EcTint tblID, EcTint rowIndex, EcThost hostID,

OvwObjectID objectID)

ShutdownExec - This function initiates the mode based shutdown procedure.

Arguments:

Return Type:EcTvoid

Privilege:Public

PDL: // {

// The following function will call the DeputyGate::SuspendExec(...) operation

//SnpAPIsuspend (GetTblID (), GetRowIndex (), GetHost (), GetSeconds());

// }

~MsMmShutdown - The destructor for the class.

Arguments:

Return Type:Void

Privilege:Public

PDL: // {

// }

Associations:

The MsMmShutdown class has associations with the following classes:

ManagementFramework (Aggregation)

6.1.3.10 MsMmSuspend Class

Parent Class:MsMmCtrl
Public:No
Distributed Object:No
Purpose and Description:
_seconds

Attributes:

_seconds - This attribute represents the number of seconds required to suspend the process.
Data Type:RWTime
Privilege:Private
Default Value:

Operations:

GetSeconds - Accessor function to obtain the number of seconds.

Arguments:
Return Type:RWTime
Privilege:Public
PDL://{
//return (_seconds);
// }

MsMmSuspend

Arguments:EcTint tblID, EcTint rowIndex, EcThost hostID, RWTime seconds,
OvwObjectID objectID : MsMmCtrl(EcTint tblID, EcTint rowIndex, EcThost hostID,
OvwObjectID objectID)

SuspendExec - Starts the suspend procedure by calling the SnmpAPISuspend and the DeputyGate suspend operations.

Arguments:
Return Type:EcTvoid
Privilege:Public
PDL: // {
//The following function will call the DeputyGate::SuspendExec(...) operation
SnmpAPISuspend (GetTblID (), GetRowIndex (), GetHost (), GetSeconds());
// }

~MsMmSuspend - The destructor for the class.

Arguments:
Return Type:Void
Privilege:Public

```
PDL: // {  
// }
```

Associations:

The MsMmSuspend class has associations with the following classes:
ManagementFramework (Aggregation)

6.1.4 Mode Management Dynamic Model

6.1.4.1 Mode Management by an ECS Application

This scenario traces the events associated activating a new mode within the system. This action causes the subagent to discover the new applications that have been designated to support the mode. It notifies HP OpenView of these new applications and HP OpenView will register their associated symbols (icons) on their mode specific submaps. The scenario is depicted in Figure 6.1-5.

6.1.4.1.1 Beginning Assumptions

The system has been configured for the new mode (refer to paragraph 6.1.1.1.1 for the procedural activities required). The MMS has been initiated and is waiting in the main event loop for an operator action.

6.1.4.1.2 Interfaces with Other Subsystems and Segments

CSS - at the application level the CSS services ensure proper mode specific CDS Directory namespace registration.

6.1.4.1.3 Stimulus

An operator selects Activate Mode from the HP OpenView GUI.

6.1.4.1.4 Participating Classes From the Object Model

HPOV (ManagementFramework)

MsMmModeInit

MsMmMode

MsAgDeputy

6.1.4.1.5 Beginning System, Segment and Subsystem State(s)

The MMS has been initiated and is waiting in the main event loop for an operator action. The mode entered is a non-ops mode. This is reflected in the fact that a simulation time is prompted for. All inputs are valid and the mode identifier entered is unique.

Mode Initiation

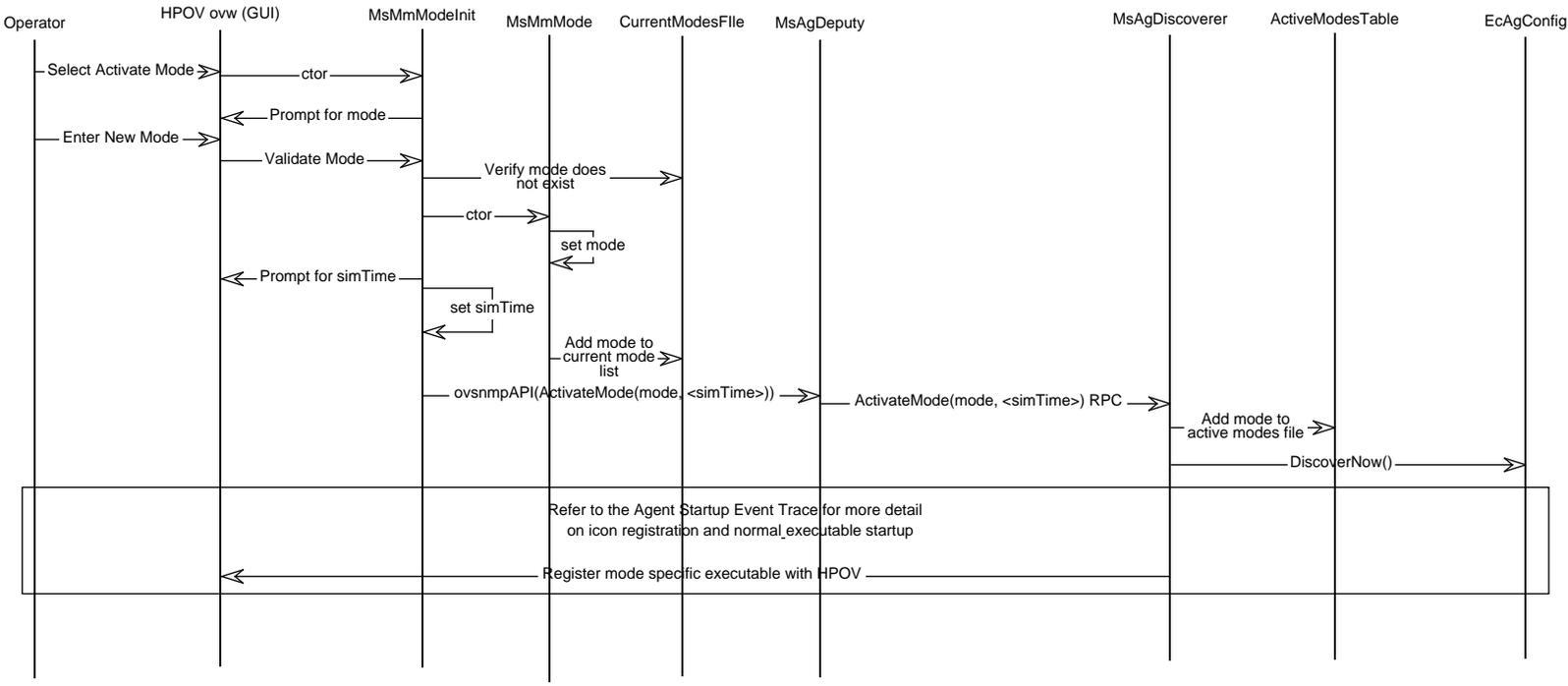


Figure 6.1-5. MMS Mode Activation Event Trace

6.1.4.1.6 Ending State

The system has been initialized for the given mode and all of the mode specific applications are registered within the applicable HP OpenView Session. The symbols that represent the mode's executables are present on the submap in an inactive state.

6.1.4.1.7 Scenario Description

This description describes the accompanying event trace.

- An operator selects "Activate Mode" from the HP Openview GUI ManagementFramework which triggers the ActivateMode callback.
- MsMmModeInit is instantiated and prompts the operator (PromptMode operation) to enter a mode identifier via the HP Openview GUI ManagementFramework.
- MsMmModeInit validates the mode identifier by checking to make sure the mode is unique and is not currently active via the ValidateMode operation.
- MsMmModeInit instantiates the MsMmMode object and sets the mode identifier attribute within the MsMmMode constructor.
- MsMmModeInit prompts the operator to enter a simulation time via the PromptSimTime operation.
- MsMmModeInit sets the simTime attribute.
- MsMmMode adds the new mode identifier to the active modes file via the UpdateActiveModeList operation .
- MsMmModeInit issues an ovsnmpAPI call to send a ActiveMode command to the MsAgDeputy.
- MsAgDeputy bundles the snmp call within an RPC and sends it to the MsAgDiscoverer
- MsAgDiscoverer adds the mode identifier to it's active mode list and then issues a DiscoverNow operation to look for newly installed applications. Since it will now search the new configuration directory established in support of the newly activated mode, it will generate an snmp trap event to HPOV ManagementFramework to register the new applications/executables. Details from this point on are contained in the Agent event trace diagrams.

6.1.5 Mode Management Structure

Table 6.1-2 lists the components of Mode Management Service.

Table 6.1-2. Mode Management Components (1 of 2)

Component Name	COTS/Custom
MsMmMode	Custom (C++ code)
HPOV (Management Framework)	COTS + custom (ovwAPIs, ovsnmpAPIs, scripts, callbacks)
MsMmModelnit	Custom (C++ code)
MsMmModeTerm	Custom (C++ code)

Table 6.1-2. Mode Management Components (2 of 2)

Component Name	COTS/Custom
MsAgDeputy	C/C++ code p/o Management Agent Services
MsAgAgent	COTS
MsMmCtrl	Custom (C++ code)
MsMmSuspend	Custom (C++ code)
MsMmResume	Custom (C++ code)
MsMmShutdown	Custom (C++ code)

6.1.6 Mode Management Management and Operation

6.1.6.1 System Management Strategy

The Mode Management Service is based on HP OpenView NNM, which generates notifications when it detects partial failures of its managed components. Components of HP OpenView may be individually restarted. In the case of a total failure, the managed component 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 Mode Management is the graphical user interface provided by HP OpenView Network Node Manager.

6.2 Accountability Management

6.2.1 Accountability Management Overview

The Accountability Management Service provides the capabilities of User Registration, User Account/Profile Maintenance, the generation of reports from audit trails, and Request Tracking (includes near real-time reporting of request states as well as non-near real-time reporting of request resource utilization).

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.

User Account/Profile Maintenance includes providing the operator the means to maintain the created accounts and the user profile information. 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 system provides the capabilities for the modification and maintenance of accounts with user profiles.

User Account/Profile Maintenance also includes making the user profile available to the various subsystems, such as the Data Server subsystem and the Billing and Accounting Application Service, 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.

Request Tracking provides the operator the capability to see the status of any of the trackable types of user requests in near real-time. Trackable types of user requests include: Product Orders, Ingest Requests, User Requests, and Operator Requests. Request tracking information is provided to this service by the ECS applications which perform processing on the above specified request types. In addition to tracking the state of the requests in near real-time, this service also collects resource utilization on each request and stores the information in a database for cost analysis reports. The resource utilization is reported back to the request tracking database after the request has completed processing. The ECS applications report tracking information via the Request Tracking Key Mechanism. In addition, Request Tracking provides a cost account reporting mechanism which will generate various reports on the resource utilization cost of the requests which were tracked.

The Request Tracking Key Mechanism is intended for use by the developers of ECS applications to report request status changes back to a central database to be displayed to an operator in near real-time. The mechanism is also used to report resource cost that was collected during the life of the request. The cost data is sent from the ECS application after the request has completed processing and the utilization data used for cost accounting. The mechanism also accounts for spawning of requests. When an ECS application creates one or more sub-requests for a request, a parent-child type of relationship is established in the mechanism so that the spawned requests can be tracked independently of each other and the operator will be able to get request state information for the entire tree of requests in near real-time.

6.2.2 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 as well as to receive updates to request tracking information. 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. The Accountability Management context diagram is shown in Figure 6.2-1.

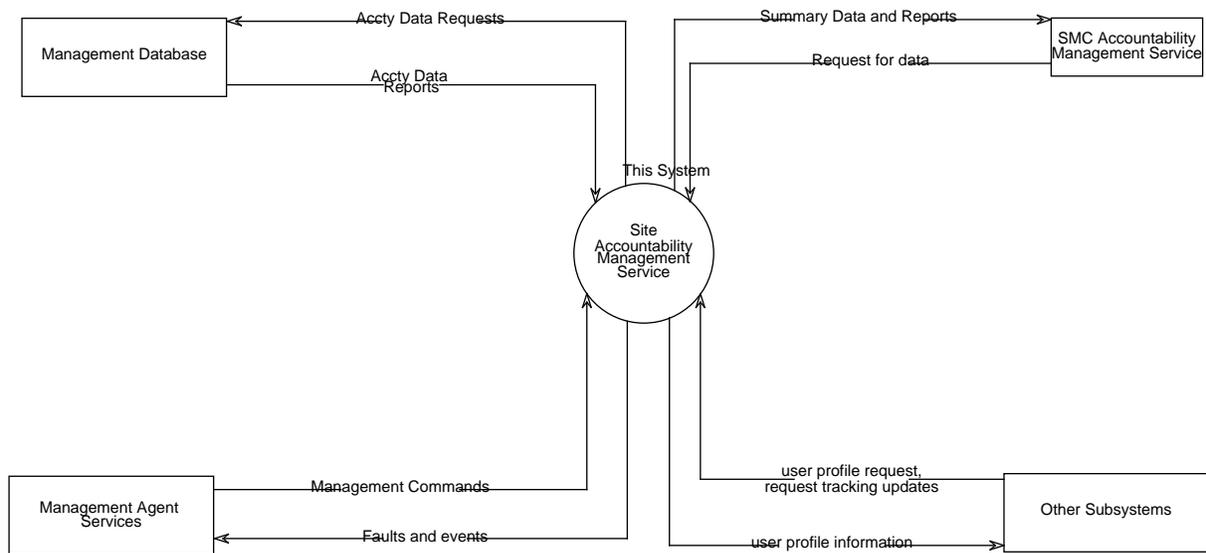


Figure 6.2-1. Accountability Management Context Diagram

6.2.3 Accountability Management Object Model

The overview of the Accountability Management Service's object model is shown in Figure 6.2-2. The details of the classes shown in the overview are found in Figure 6.2-3 through Figure 6.2-7.

6.2.3.1 EcAgEvent Class

Parent Class: Not Applicable

Public: Yes

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.

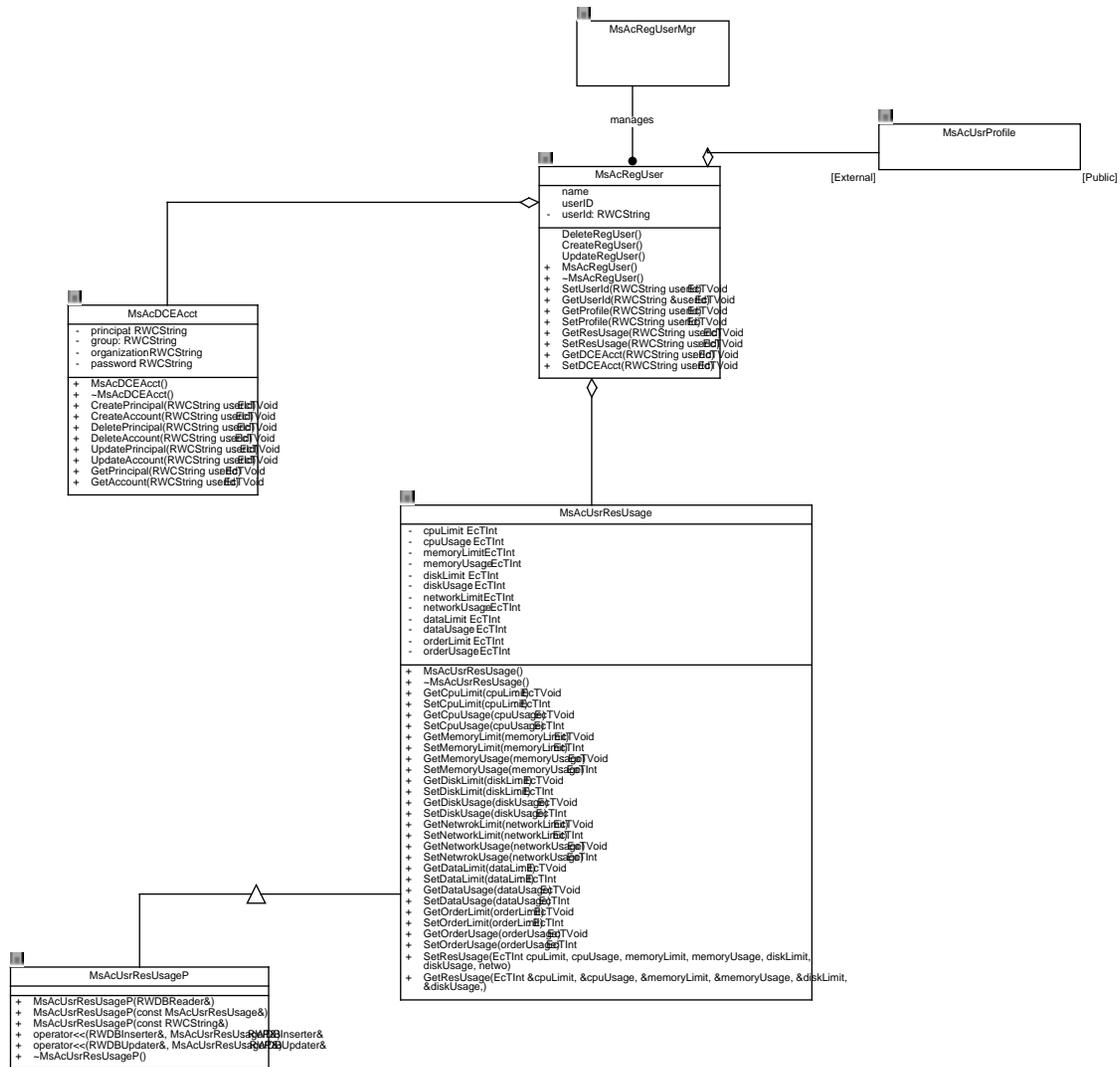


Figure 6.2-5. Accountability Management Object Model Detail 3

Attributes:

None

Operations:

None

Associations:

The EcAgEvent class has associations with the following classes:

None

6.2.3.2 EcOrder Class

Parent Class:EcRequest

Public:Yes

Distributed Object:No

Purpose and Description:

This is a public class which is used by ECS applications to collect resource utilizations associated with an order type of request. The class is also used by the application to report the state of the order when the order state changes. Objects in this class should remain until the application has finished processing the associated order request. An order type of request is the top most root of a hierarchy of sub-classes and services that is associated with a Product Data Order request from an ECS user.

Attributes:

distList - The distribution list for the product.

Data Type:DistListType

Privilege:Private

Default Value:

estimatedPrice - The price which was reported to the ECS user and the price which is to be decremented from the available balance of the user.

Data Type:EcTLong

Privilege:Private

Default Value:

homeDAAC - The site at which the user is registered, who placed the product order.

Data Type:RWCString

Privilege:Private

Default Value:

orderUR - This is the UR for the order which is reported back to the ECS user. This is stored in the tracking database so that the order tracking information can be retrieved by the Order UR.

Data Type:EcTUR

Privilege:Private

Default Value:

shipAddress - Mailing address that the products produced/retrieved for the order are to be shipped.

Data Type:RWCString

Privilege:Private

Default Value:

shipMethod - The method of shipment - how the product(s) are to be sent to the requesting user.

Data Type:RWCString

Privilege:Private

Default Value:

shipToName - The name to which the products are to be addressed.

Data Type:RWCString

Privilege:Private

Default Value:

userId - The unique ECS user identification of the user who placed the order.

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

EcOrder - The constructor for the class. The order related information is initialized and the information, including the starting state, are sent to the request tracking server. In addition, the resource utilization counters are initialized to zero.

Arguments:RWCString description, enum type, enum state, EcTUR orderUR, RWCString userId, RWCString homeDAAC, RWCString shipAddress, RWCString shipToName, RWCString shipMethod, DiskListType distList, EcTLong estimatedPrice

Return Type:Void

Privilege:Public

PDL: No PDL

~EcOrder - Default destructor for the class. The collected resource utilization and the final state of the order are sent to the request tracking server.

Arguments:

Return Type:Void
Privilege:Public
PDL: No PDL

Associations:

The EcOrder class has associations with the following classes:

Class: EcOrderEvent generates

Class: EcPfManagedServer processnon-statechangeorderevents

6.2.3.3 EcOrderEvent Class

Parent Class:EcRequestEvent

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is a public, distributed object whose purpose is to report information collected about an order type of request. An order type of request is the root of a request hierarchy structure that was generated based on a Product Order request from an ECS user. Objects of this class are created with the information to be reported and processed (sent to the request tracking server) and then destroyed. These objects only need to stay around long enough for the event to be processed.

Attributes:

distList - The distribution list for the product.

Data Type:DistListType

Privilege:Private

Default Value:

estimatedPrice - The price which was reported to the ECS user and the price which is to be decremented from the available balance of the user.

Data Type:EcTLong

Privilege:Private

Default Value:

homeDAAC - The site at which the user is registered, who placed the product order.

Data Type:RWCString

Privilege:Private

Default Value:

orderUR - This is the UR for the order which is reported back to the ECS user. This is stored in the tracking database so that the order tracking information can be retrieved by the Order UR.

Data Type:EcTUR
Privilege:Private
Default Value:

shipAddress - Mailing address that the products produced/retrieved for the order are to be shipped.

Data Type:RWCString
Privilege:Private
Default Value:

shipMethod - The method of shipment - how the product(s) are to be sent to the requesting user.

Data Type:RWCString
Privilege:Private
Default Value:

shipToName - The name to which the products are to be addressed.

Data Type:RWCString
Privilege:Private
Default Value:

userId - The unique ECS user identification of the user who placed the order.

Data Type:RWCString
Privilege:Private
Default Value:

Operations:

EcOrderEvent - Constructor for the class. The information to be reported to the request tracking server is initialized.

Arguments:EcTLong itemID, RWString description, requestType type, requestStateType state, EcTTime timeStateUpdated, requestCost rCost, EcTUR orderUR, RWString userId, RWString homeDAAC, distList, EcTLong estimatedPrice
PDL: No PDL

ProcessOrderEvent - This method takes the information which was set in the class and sends the information to the request tracking server.

Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

~EcOrderEvent - Default constructor for the class. This method cleans up the order event object by removing any memory allocated for the attributes.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

Associations:

The EcOrderEvent class has associations with the following classes:
Class: EcOrder generates

6.2.3.4 EcPfManagedServer Class

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

This is the container class that starts up the event Manager, table Manager, monitor, port monitor, discoverer, subagent configuration, static buffer, and the deputy gate. This class also starts a thread that triggers scheduled events (i.e. polling ECS application's performance metrics).

Attributes:

None

Operations:

None

Associations:

The EcPfManagedServer class has associations with the following classes:
Class: EcOrder processnon-statechangeorderevents
Class: EcService processnon-statechangeserviceevents
Class: EcSubOrder processnon-statechangesub-orderevents

6.2.3.5 EcPriceTableB Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:Yes
Purpose and Description:

This class represents a public and distributed class that holds the prices of every billable item in the ECS inventory of products and services. Price of hard media and standard

shipping costs are also maintained in this table.

Attributes:

None

Operations:

None

Associations:

The EcPriceTableB class has associations with the following classes:

Class: MsAcTrackingMgr providepriceforcancelledrequests

6.2.3.6 EcRequest Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

This is an abstract class which represents all types of requests which are tracked in the ECS system. This class contains the attributes and operations which are common to all of the request types. The objects which are created from the sub-classes are used to track resource utilization of the associated system request types as well as to maintain and report the current state of the associated system request to the request tracking server.

Attributes:

activeTime - This is a resource utilization counter which contains the approximate amount of real-time that the request has been actively processed.

Data Type:EcTTime

Privilege:Private

Default Value:

cpuUtilAtMethodStart - This attribute is set to the current value of the cpu counter of the system when resource utilization collection is started. This value is then used to calculate the amount of cpu which was used during the collection period.

Data Type:EcTLong

Privilege:Private

Default Value:

cpuUtilization - The running total amount of cpu processing which has been used while processing this request.

Data Type:EcTLong

Privilege:Private

Default Value:

description - A textual description of the request.

Data Type:RWCString

Privilege:Private

Default Value:

diskUtilization - The running total amount of disk utilization which has been used while processing this request.

Data Type:EcTLong

Privilege:Private

Default Value:

idleTime - This is a resource utilization counter which contains the approximate amount of real-time that the request has been idle.

Data Type:EcTTime

Privilege:Private

Default Value:

ioUtilAtMethodStart - This attribute is set to the current value of the I/O utilization counter of the system when resource utilization collection is started. This value is then used to calculate the amount of I/O utilization which was used during the collection period.

Data Type:EcTLong

Privilege:Private

Default Value:

ioUtilization - The running total amount of I/O utilization which has been used while processing this request.

Data Type:EcTLong

Privilege:Private

Default Value:

lastEventID - This is the event identification of the last event that was reported to the MSS event logging capability. This event ID allows an operator to browse through the event log chain for the request in order to show the history of state changes as well as to see any other significant events associated with this request.

Data Type:EcTLong

Privilege:Private

Default Value:

requestDate - The date/time at which the request started to be processed.

Data Type:EcTTime

Privilege:Private

Default Value:

requestID - A unique identification of the request.

Data Type:EcTLong

Privilege:Private

Default Value:

requestStartTime - The date/time at which the request started to be processed.

Data Type:EcTTime

Privilege:Private

Default Value:

state - This is the current state of the request.

Data Type:enum

Privilege:Private

Default Value:

timeOfLastStateUpdate - This is the time at which the current state was changed.

Data Type:EcTTime

Privilege:Private

Default Value:

type - The type of the request being processed.

Data Type:enum

Privilege:Private

Default Value:

Operations:

EcRequest - This is the constructor for the object. This method sets the attributes of the request object to the passed values and initializes the resource utilization totals to zero.

Arguments:RWCString description, enum type, enum state

Return Type:Void

Privilege:Public

PDL: No PDL

GetDiskUtilization - Returns the current value of the disk utilization attribute.

Arguments:

Return Type:EcTLong

Privilege:Public

PDL: No PDL

GetRequestID - Returns the value of the request ID attribute.

Arguments:

Return Type:EcTLong

Privilege:Public

PDL: No PDL

GetState - Returns the current value of the request state attribute.

Arguments:

Return Type:trackingStateType

Privilege:Public

PDL: No PDL

ProcessEvent - No description

Arguments:RWCString eventDescription

PDL: No PDL

SetDiskUtilization -

Arguments:diskUtilization

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

SetState - This method sets the current value of the request state attribute to the passed value.

Arguments:state

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

SetState - This method sets the current value of the request state attribute to the passed value.

Arguments:trackingStateType newState

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

StartCollecting - This method reads from the system the current values of particular resource counters and stores them in attributes. This method should be called at the beginning of a method which processes the request associated with this object.

Arguments:

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

StopCollecting - This method reads from the system the current values of particular resource counters and subtracts from them the associated values stored in the attributes. The resulting value will be added to the running total utilization attribute. This method should be called at the end of a method which processes the request associated with this object.

Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

~EcRequest - This is the default destructor of the object. This method cleans up any memory which was allocated to attributes within this object.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

Associations:

The EcRequest class has associations with the following classes:
None

6.2.3.7 EcRequestEvent Class

Parent Class:EcAgEvent
Public:Yes
Distributed Object:Yes
Purpose and Description:

This is an abstract class which represents all types of requests event objects which are used to report information collected about a request. This class contains the attributes and operations which are common to all of the request event types. The objects which are created from the sub-classes are used to report resource utilization of the associated system request types as well as to report the current state of the associated system request to the request tracking server.

Attributes:

activeTime - This is a resource utilization counter which contains the approximate amount of real-time that the request has been actively processed.

Data Type:EcTTime
Privilege:Private
Default Value:

cpuUtilization - The running total amount of cpu processing which has been used while processing this request.

Data Type:EcTLong
Privilege:Private
Default Value:

description - A textual description of the request.

Data Type:RWCString

Privilege:Private

Default Value:

diskUtilization - The running total amount of disk utilization which has been used while processing this request.

Data Type:EcTLong

Privilege:Private

Default Value:

ioUtilization - The running total amount of I/O utilization which has been used while processing this request.

Data Type:EcTLong

Privilege:Private

Default Value:

requestID - A unique identification of the request.

Data Type:EcTLong

Privilege:Private

Default Value:

requestStartTime - The date/time at which the request started to be processed.

Data Type:EcTTime

Privilege:Private

Default Value:

sleepTime - This is a resource utilization counter which contains the approximate amount of real-time that the request has not been actively processed.

Data Type:EcTTime

Privilege:Private

Default Value:

state - This is the current state of the request.

Data Type:enum

Privilege:Private

Default Value:

timeOfLastStateUpdate - This is the time at which the current state was changed.

Data Type:EcTTime

Privilege:Private

Default Value:

totalTime - This is the total amount of real-time which was required to process the request.

Data Type:EcTTime

Privilege:Private
Default Value:

type - The type of the request being processed.
Data Type:enum
Privilege:Private
Default Value:

Operations:

EcRequest - No description

Arguments:EcTLong, requestID, RWCString descrip, enum type, EcTTime timeOfLast, EcTLong cpuUtil, EcTLong ioUtil, EcTLong diskUtil, EcTTime rqStart, EcTTime SleepT, EcTTime activeT, EcTTime totalTime
PDL: No PDL

~EcRequest - No description

Arguments:
PDL: No PDL

Associations:

The EcRequestEvent class has associations with the following classes:

None

6.2.3.8 EcService Class

Parent Class:EcRequest

Public:Yes

Distributed Object:No

Purpose and Description:

This is a public class which is used by ECS applications to collect resource utilizations associated with service type of request. The class is also used by the application to report the state of the service when the service state changes. Objects in this class should remain until the application has finished processing the associated service request. A service type of request is a request which is not associated with retrieving a specific product, a service could be spawned from a sub-order type of request if the ECS application spawns a set of processing which does not result in product. a service type of request could be the root of a service request hierarchy for tracking ECS requests which are not Product Orders.

Attributes:

homeDAAC - The site at which the user is registered, who placed the product order.
Data Type:RWCString

Privilege:Private
Default Value:

parentId - This is the request ID of the service or sub-order type of request which spawned this service type of request.

Data Type:EcTLong
Privilege:Private
Default Value:

serviceUR - This is the UR for the service which is reported back to the ECS user. This is stored in the tracking database so that the service tracking information can be retrieved by the Service UR. This attribute is only used if the service is the root of the service hierarchy.

Data Type:EcTUR
Privilege:Private
Default Value:

userId - The unique ECS user identification of the user who placed the service. This attribute is only used if the service is the root of the service hierarchy.

Data Type:RWCString
Privilege:Private
Default Value:

Operations:

EcService - The constructor for the class. The service related information is initialized and the information, including the starting state, are send to the request tracking server. In addition, the resource utilization counters are initialized to zero.

Arguments:RWCString description, enum type, enum state, EcTLong parentId, RWCString userId, EcTUR serviceUR, RWCString homeDAAC
Return Type:Void
Privilege:Public
PDL: No PDL

~EcService - Default destructor for the class. The collected resource utilization and the final state of the service are sent to the request tracking server.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

Associations:

The EcService class has associations with the following classes:

Class: EcServiceEvent generates

Class: EcPfManagedServer processnon-statechangeserviceevents
EcOrder (Aggregation)
EcSubOrder (Aggregation)

6.2.3.9 EcServiceEvent Class

Parent Class:EcRequestEvent

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is a public, distributed object whose purpose is to report inform

Attributes:

homeDAAC - The site at which the user is registered, who placed the product order or service.

Data Type:RWCString

Privilege:Private

Default Value:

parentId - This is the request ID of the service or sub-order type of request which spawned this service type of request.

Data Type:EcTLong

Privilege:Private

Default Value:

serviceUR - This is the UR for the service which is reported back to the ECS user. This is stored in the tracking database so that the service tracking information can be retrieved by the Service UR. This attribute is only used if the service is the root of the service hierarchy.

Data Type:EcTUR

Privilege:Private

Default Value:

Operations:

EcOrderEvent - Constructor for the class. The information to be reported to the request tracking server is initialized.

Arguments:EcTLong itemID, RWString description, requestType type, requestStateType state, EcTTime timeStateUpdated, requestCost rCost, RWString userId, RWString homeDAAC, EcTLong parentId, EcTUR serviceUR

PDL: No PDL

ProcessOrderEvent - This method takes the information which was set in the class and sends the information to the request tracking server.

Arguments:
PDL: No PDL

~EcOrderEvent - Default constructor for the class. This method cleans up the service event object by removing any memory allocated for the attributes.

Arguments:
PDL: No PDL

Associations:

The EcServiceEvent class has associations with the following classes:

Class: EcService generates

6.2.3.10 EcSubOrder Class

Parent Class:EcRequest

Public:Yes

Distributed Object:No

Purpose and Description:

This is a public class which is used by ECS applications to collect resource utilizations associated with sub-order type of request. The class is also used by the application to report the state of the sub-order when the sub-order state changes. Objects in this class should remain until the application has finished processing the associated sub-order request. A sub-order type of request is a child of a hierarchy of sub-orders and services that is associated with a Product Data Order request from an ECS user. At the top of this hierarchy is an order type of request.

Attributes:

archiveUtilization - This attribute contains the total amount of achive utilization which has been collected for this sub-order.

Data Type:EcTLong

Privilege:Private

Default Value:

granualFormatList - This attribute is a list of granual formats. There is one format list entry for each granual which is associated with this sub-order.

Data Type:GranFormatListType

Privilege:Private

Default Value:

granualList - This attribute is a list of granual identifications which are associated with this sub-order.

Data Type:GranListType

Privilege:Private

Default Value:

granualMediaList - This attribute is a list of media types. There is one media type list entry for each granual which is associated with this sub-order.

Data Type:GranMediaListType

Privilege:Private

Default Value:

granualSizeList - This attribute is a list of granual sizes.

Data Type:GranSizeListType

Privilege:Private

Default Value:

lastEventId - This is the event identification of the last event that was reported to the MSS event logging capability for this sub-order. This event ID allows the operator to browse through the event log chain for the sub-order in order to show the history of state changes as well as to see any other significant events associated with this sub-order.

mediaCountList - This attribute contains a list of media counts. A sub-order could have more than one type of media being produced from it. This attribute contains the total number of pieces of media which have been produced of each media type.

Data Type:RWCollectionList(EcTLong)

Privilege:Private

Default Value:

mediaTypeList - This attribute contains a list of media types. A sub-order could have more than one type of media being produced from it.

Data Type:MediaListType

Privilege:Private

Default Value:

numGranuals - This attribute contains the number of data granuals which will be produced/retrieved as part of the processing of this sub-order.

Data Type:EcTLong

Privilege:Private

Default Value:

parentID - This is the unique request ID of the order or sub-order which spawned the sub-order associated with this object.

Data Type:EcTLong

Privilege:Private

Default Value:

shipDateTime - This is the actual date and time when the products associated with this

request were prepared for shipment.

Data Type:EcTTime

Privilege:Private

Default Value:

Operations:

EcRequestTracker - No description

Arguments:

PDL: No PDL

GetGranualInfo - This method returns the detailed information for a requested granual.

Arguments:EcTLong granualId, enum &granMedia, RWCString &granFormat, EcTLong &granSize

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

GetGranualList - This method returns the list of granual IDs associated with this sub-order.

Arguments:GranListType &granualList

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

GetMediaCount - This method returns the number of media that have been used when producing the data associated with this sub-order.

Arguments:

Return Type:EcTLong

Privilege:Public

PDL: No PDL

SetGranualSize - This sets the size of a particular granual associated with this sub-order.

Arguments:EcTLong granualId, EcTLong granualSize

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

SetMediaCount - This method sets the number of media that have been used when producing the data associated with this sub-order.

Arguments:EcTInt mediaCount

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

~**EcRequestTracker** - No description

Arguments:

PDL: No PDL

Associations:

The EcSubOrder class has associations with the following classes:

Class: EcSubOrderEvent generates

Class: EcPfManagedServer processnon-statechangesub-orderevents

EcOrder (Aggregation)

EcSubOrder (Aggregation)

6.2.3.11 EcSubOrderEvent Class

Parent Class:EcRequestEvent

Public:Yes

Distributed Object:Yes

Purpose and Description:

This is a public, distributed object whose purpose is to report information collected about a sub-order type of request. A sub-order type of request is a child of a hierarchy of sub-orders and services that is associated with a Product Data Order request from an ECS user. At the top of this hierarchy is an order type of request. Objects of this class are created with the information to be reported and processed (sent to the request tracking server) and then destroyed. These objects only need to stay around long enough for the event to be processed.

Attributes:

archiveUtilization - This attribute contains the total amount of achive utilization which has been collected for this sub-order.

Data Type:EcTLong

Privilege:Private

Default Value:

granualFormatList - This attribute is a list of granual formats. There is one format list entry for each granual which is associated with this sub-order.

Data Type:GranFormatListType

Privilege:Private

Default Value:

granualList - This attribute is a list of granual identifications which are associated with this sub-order.

Data Type:GranListType

Privilege:Private

Default Value:

granualMediaList - This attribute is a list of media types. There is one media type list entry for each granual which is associated with this sub-order.

Data Type:GranMediaListType

Privilege:Private

Default Value:

granualSizeList - This attribute is a list of granual sizes.

Data Type:GranSizeListType

Privilege:Private

Default Value:

mediaCountList - This attribute contains a list of media counts. A sub-order could have more than one type of media being produced from it. This attribute contains the total number of pieces of media which have been produced of each media type.

Data Type:RWCollectionList(EcTLong)

Privilege:Private

Default Value:

mediaTypeList - This attribute contains a list of media types. A sub-order could have more than one type of media being produced from it.

Data Type:MediaListType

Privilege:Private

Default Value:

numGranuals - This attribute contains the number of data granuals which will be produced/retrieved as part of the processing of this sub-order.

Data Type:EcTLong

Privilege:Private

Default Value:

parentID - This is the unique request ID of the order or sub-order which spawned the sub-order associated with this object.

Data Type:EcTLong

Privilege:Private

Default Value:

shipDateTime - This is the actual date and time when the products associated with this request were prepared for shipment.

Data Type:EcTTime

Privilege:Private

Default Value:

userId - The unique user Identification of the ECS user who submitted the original Product Order request.

Operations:

EcSubOrderEvent - Constructor for the class. The information to be reported to the request tracking server is initialized.

Arguments: EcTLong itemID, RWString description, requestType type, requestStateType state, EcTTime timeStateUpdated, requestCost rCost, long parentID, granListType granualList, granListSizeType granualSizeList, mType mediaType, long mediaCount, date shipDateTime

PDL: No PDL

ProcessSubOrderEvent - This method takes the information which was set in the class and sends the information to the request tracking server.

Arguments:

Return Type: EcTVoid

Privilege: Public

PDL: No PDL

~EcSubOrderEvent - Default constructor for the class. This method cleans up the sub-order event object by removing any memory allocated for the attributes.

Arguments:

Return Type: Void

Privilege: Public

PDL: No PDL

Associations:

The EcSubOrderEvent class has associations with the following classes:

Class: EcSubOrder generates

6.2.3.12 MsAcAddress Class

Parent Class: Not Applicable

Public: Yes

Distributed Object: No

Purpose and Description:

Attributes:

city

Data Type: RWCString

Privilege: Private

Default Value:

country

Data Type:RWCString

Privilege:Private

Default Value:

fax

Data Type:RWCString

Privilege:Private

Default Value:

phone

Data Type:RWCString

Privilege:Private

Default Value:

state

Data Type:RWCString

Privilege:Private

Default Value:

street1

Data Type:RWCString

Privilege:Private

Default Value:

street2

Data Type:RWCString

Privilege:Private

Default Value:

zip

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

GetCity

Arguments:

Return Type:const RWCString

Privilege:Public

GetCountry

Arguments:

Return Type:const RWCString

Privilege:Public

GetFax

Arguments:

Return Type:const RWCString

Privilege:Public

GetPhone

Arguments:

Return Type:const RWCString

Privilege:Public

GetStreet1

Arguments:

Return Type:const RWCString

Privilege:Public

GetStreet2

Arguments:

Return Type:const RWCString

Privilege:Public

GetZip

Arguments:

Return Type:const RWCString

Privilege:Public

MsAcAddress

Arguments:

Return Type:Void

Privilege:Public

SetCity

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

SetCountry

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

SetFax

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

SetPhone

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

SetStreet1

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

SetStreet2

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

SetZip

Arguments:const RWCString

Return Type:EcTInt

Privilege:Public

~MsAcAddress

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcAddress class has associations with the following classes:

MsAcUsrProfile (Aggregation)

6.2.3.13 MsAcAddressP Class

Parent Class:MsAcAddress

Attributes:

All Attributes inherited from parent class

Operations:

MsAcAddressP

Arguments:RWDBReader&

Return Type:Void

Privilege:Public

MsAcAddressP

Arguments:const MsAcAddress&

Return Type:Void

Privilege:Public

Update

Arguments:RWDBUpdater, RWCString, ...

Return Type:Void

Privilege:Public

operator<<

Arguments:RWDBInserter& MsAcAddressP&

Return Type:Void

Privilege:Public

~MsAcAddressP

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcAddressP class has associations with the following classes:

None

6.2.3.14 MsAcAuditTrail Class

Parent Class:Not Applicable

Attributes:

None

Operations:

GenerateAuditTrail

Arguments:

Return Type:Void
Privilege:Public

MsAcAuditTrail

Arguments:
Return Type:Void
Privilege:Public

~MsAcAuditTrail

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsAcAuditTrail class has associations with the following classes:

Class: MsAcManager manages

6.2.3.15 MsAcCostAcctReport Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents the reports generated to provide detailed and summary cost account information.

Attributes:

None

Operations:

GenCostAcctReport - This method generates cost accounting reports based on the resource utilization information first collected by the MsAcTrackingMgr class and organized into cost accounts by the MsBaCostAcctB class. The type(s) of resources reported is included in the ResourceId parameter passed to this method which include but are not limited to I/O utilization, CPU utilization, disk utilization, archive costs, media costs and fixed costs.

Arguments:RWCString myDAAC, RWString groupId, RWCString userId, RWCString ResourceId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

MsAcCostAcctReport - This method generates cost accounting reports based on the resource utilization information collected by the MsAcTrackingMgr. The type(s) of resources reported is included in the ResourceId parameter passed to this method which include but are not limited to I/O utilization, CPU utilization, disk utilization, archive costs, media costs and fixed costs.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

Print - This method represents printing of detailed and summary account information in the following formats: hardcopy, in response to on-line queries, to extract data files or to disk.

Arguments:RWCString reportId, RWCString destinationID, RWCString reportFormat

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

~MsAcCostAcctReport - This method represents the constructor for this class.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

Associations:

The MsAcCostAcctReport class has associations with the following classes:

Class: MsAcTrackingDB providecostdata

Class: MsAcTrackingUI requestcostreport

6.2.3.16 MsAcDCEAcct Class

Parent Class:Not Applicable

Attributes:

group

Data Type:RWCString

Privilege:Private

Default Value:

organization

Data Type:RWCString

Privilege:Private
Default Value:

password

Data Type:RWCString
Privilege:Private
Default Value:

principal

Data Type:RWCString
Privilege:Private
Default Value:

Operations:

CreateAccount

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public

CreatePrincipal

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public

DeleteAccount

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public

DeletePrincipal

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public

GetAccount

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public

GetPrincipal

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public

MsAcDCEAcct

Arguments:

Return Type:Void

Privilege:Public

UpdateAccount

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

UpdatePrincipal

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

~MsAcDCEAcct

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcDCEAcct class has associations with the following classes:

MsAcRegUser (Aggregation)

6.2.3.17 MsAcManager Class

Parent Class:EcPfManagedServer

Attributes:

All Attributes inherited from parent class

Operations:**GenerateAduitTrail**

Arguments:

Return Type:EcTVoid

Privilege:Public

GenerateReport

Arguments:

Return Type:EcTVoid

Privilege:Public

MsAcManager

Arguments:

Return Type:Void

Privilege:Public

SendSummaryDataToSMC

Arguments:

Return Type:EcTVoid

Privilege:Public

~MsAcManager

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcManager class has associations with the following classes:

Class: MsAcManagerUI communicatewith

Class: MsAcAuditTrail manages

Class: MsAcRegUserMgr manages

Class: MsAcReport manages

Class: MsAcUsrProfileMgr manages

Class: MsAcUsrRequestMgr manages

6.2.3.18 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 peinding requests for registered accounts, create a registered user account from an entry in the pending requests list.

Attributes:

None

Operations:

ApproveRequest

Arguments:RWCString regUserId

Return Type:EcTVoid

Privilege:Public

DeletePendingRequest

Arguments:EcTInt guestUserId

Return Type:EcTVoid

Privilege:Public

DeleteRegUser

Arguments:RWCString regUserId

Return Type:EcTVoid

Privilege:Public

DisplayPendingRequest

Arguments:EcTInt guestUserId

Return Type:EcTVoid

Privilege:Public

DisplayPendingRequestList

Arguments:

Return Type:EcTVoid

Privilege:Public

DisplayRegUser

Arguments:EcTInt guestUserId

Return Type:EcTVoid

Privilege:Public

DisplayRegUserList

Arguments:RWCString regUserId

Return Type:EcTVoid

Privilege:Public

DsisplayProfile

Arguments:

Return Type:EcTVoid

Privilege:Public

GenerateAuditTrail

Arguments:

Return Type:EcTVoid

Privilege:Public

GenerateReport

Arguments:

Return Type:EcTVoid

Privilege:Public

GenerateReport

Arguments:EcTInT RepordId

Return Type:EcTVoid

Privilege:Public

MsAcManagerUI

Arguments:

Return Type:Void

Privilege:Public

RetrievePendingReqstList

Arguments:

Return Type:EcTVoid

Privilege:Public

RetrieveRegUserList

Arguments:

Return Type:EcTVoid

Privilege:Public

SendSummaryDataToSMC

Arguments:

Return Type:EcTVoid

Privilege:Public

~MsAcManagerUI

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcManagerUI class has associations with the following classes:

Class: MsAcRegUserMgr

Class: MsAcManager communicatewith

Class: MsAcUsrProfileMgr uses

Class: MsAcUsrRequest uses

Class: MsAcUsrRequestMgr uses

6.2.3.19 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

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

GetDCEAcct

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

GetProfile

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

GetResUsage

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

GetUserId

Arguments:RWCString &userId

Return Type:EcTVoid

Privilege:Public

MsAcRegUser

Arguments:

Return Type:Void

Privilege:Public

SetDCEAcct

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

SetProfile

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

SetResUsage

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

SetUserId

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

~MsAcRegUser

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcRegUser class has associations with the following classes:

Class: MsAcRegUserMgr manages

6.2.3.20 MsAcRegUserDB Class

Parent Class:Not Applicable

Attributes:

None

Operations:**AddUserProfile**

Arguments:MsAcUsrProfileP&

Return Type:EcTInt

Privilege:Public

Close

Arguments:

Return Type:EcTVoid

Privilege:Public

CreateRegUser

Arguments:RWCString&

Return Type:EcTInt

Privilege:Public

DeleteRegUser

Arguments:RWCString&

Return Type:EcTInt

Privilege:Public

DeleteUserProfile

Arguments:RWCString&

Return Type:EcTInt

Privilege:Public

GetProfileList

Arguments:RWSListCollectables&

Return Type:EcTVoid

Privilege:Public

GetRegUser

Arguments:RWCString&

Return Type:MsAcRegUserP*

Privilege:Public

GetUserProfile

Arguments:RWCString&

Return Type:MsAcUsrProfile*

Privilege:Public

Initialize

Arguments:RWCString&, RWCString&, RWCString&, RWCString&, RWCString&

Return Type:EcTVoid

Privilege:Public

Instance

Arguments:

Return Type:MsAcRegUserDB&

Privilege:Public

MsAcRegUserDB

Arguments:RWCString&, RWCString&, RWCString&, RWCString&, RWCString&

Return Type:Void

Privilege:Public

ProfileExists

Arguments:RWCString&

Return Type:EcTInt

Privilege:Public

RegUserExists

Arguments:RWCString&

Return Type:EcTInt

Privilege:Public

ResourceDataExists

Arguments:RWCString&

Return Type:EcTInt

Privilege:Public

UpdateRegUser

Arguments:MsAcRegUserP&

Return Type:EcTInt

Privilege:Public

UpdateUserProfile

Arguments:MsAcUsrProfile&

Return Type:EcTInt

Privilege:Public

~MsAcRegUserDB

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcRegUserDB class has associations with the following classes:

MsAcRegUserMgr (Aggregation)

6.2.3.21 MsAcRegUserMgr 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:

None

Operations:

CreateRegUser

Arguments:

DeleteRegUser

Arguments:

MsAcRegUserMgr

Arguments:

Return Type:EcTVoid

Privilege:Public

NotifyUser

Arguments:

Return Type:EcTVoid

Privilege:Public

PrintUserInfor

Arguments:

Return Type:EcTVoid

Privilege:Public

RetrieveRegUser

Arguments:

RetrieveRegUserList

Arguments:

UpdateRegUser

Arguments:

~MsAcRegUserMgr

Arguments:

Return Type:EcTVoid

Privilege:Public

Associations:

The MsAcRegUserMgr class has associations with the following classes:

- Class: MsAcManagerUI
- Class: MsAcManager manages
- Class: MsAcRegUser manages
- Class: MsAcUsrProfileMgr uses

6.2.3.22 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:

reportId

Data Type:

Privilege:Private

Default Value:

Operations:

GenerateReport

Arguments:EcTInt reportId

Return Type:Void

Privilege:Public

MsAcReport

Arguments:

Return Type:Void

Privilege:Public

~MsAcReport

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcReport class has associations with the following classes:

Class: MsAcManager manages

6.2.3.23 MsAcTrackingDB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This is the interface class to the request tracking database. This class provides operations that maintain the data in the database. The methods provide access to retrieve, update, and query the request tracking information.

Attributes:

None

Operations:

RetrieveOrderList - This method will retrieve a list of orders from the database according to the specified query.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

RetrieveOrderRecord - This method will retrieve the specified order record from the request tracking database.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

RetrieveRequestList - This method will retrieve a list of sub-orders from the database according to the specified query.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

RetrieveRequestRecord - This method will retrieve the specified sub-order record from the request tracking database.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

RetrieveServiceList - This method will retrieve a list of services from the database according to the specified query.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

RetrieveServiceRecord - This method will retrieve the specified service record from the request tracking database.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

UpdateOrderRecord - This method will update the specified order record in the database with the specified information. This method will also be used to create a new order record in the database.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

UpdateRequestRecord - This method will update the specified sub-order record in the database with the specified information. This method will also be used to create a new sub-order record in the database.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

UpdateServiceRecord - This method will update the specified service record in the database with the specified information. This method will also be used to create a new service record in the database.

Arguments:
Return Type:Void
Privilege:Public
PDL: No PDL

Associations:

The MsAcTrackingDB class has associations with the following classes:

Class: MsAcTrackingMgr exchanges data with
Class: MsAcCostAcctReport provide cost data

6.2.3.24 MsAcTrackingMgr Class

Parent Class: Not Applicable

Public: No

Distributed Object: No

Purpose and Description:

This class represents the manager class that collects order, request and service resource utilization statistics and status for ECS processes. This object is the interface that the request tracking event reporting objects (EcRequestEvent and its subclasses) as well as other ECS applications have to the request tracking database. The database will have near-real time status information about the requests as well as the final resource utilization of each request.

Attributes:

None

Operations:

CreateRequestItem - This method adds a sub-order type of request to request tracking. This method is called when a new sub-order tracking object has been created. The information received will be used to create a new entry in the request tracking database.

Arguments: requestStruct

Return Type: EcTVoid

Privilege: Public

PDL: No PDL

CreateServiceItem - This method adds a service type of request to request tracking. This method is called when a new service tracking object has been created. The information received will be used to create a new entry in the request tracking database.

Arguments: serviceStruct

Return Type: EcTVoid

Privilege: Public

PDL: No PDL

GetOrderInfo - This returns the detailed information for an order-type of request based on the passed request ID.

Arguments: itemID

Return Type: orderStruct

Privilege: Public

PDL: No PDL

GetOrderInfoByUR - This returns the detailed information for an order-type of request based on the passed order UR.

Arguments:orderUR
Return Type:orderStruct
Privilege:Public
PDL: No PDL

GetOrdersBySite - This method returns a list of order-type request IDs which were run at the specified site.

Arguments:siteId
Return Type:Void
Privilege:Public
PDL: No PDL

GetOrdersByUser - This method returns a list of order-type request IDs which were requested by the specified user.

Arguments:userID
Return Type:itemIDList
Privilege:Public
PDL: No PDL

GetRequestInfo - This method returns the detailed information for a sub-order type of request based on the passed request ID.

Arguments:itemID
Return Type:requestStruct
Privilege:Public
PDL: No PDL

GetRequestInfo - This method returns the detailed information for a sub-order type of request based on the passed request ID.

Arguments:orderStruct
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

GetRequestsBySite - This method returns a list of sub-order type request IDs which were processed by the specified site.

Arguments:siteId
Return Type:Void
Privilege:Public
PDL: No PDL

GetServiceInfo - This method returns the detailed information for a service type of request based on the passed request ID.

Arguments:itemID
Return Type:serviceStruct
Privilege:Public
PDL: No PDL

GetServiceInfoByUR - This method returns the detailed information for a service type of request based on the passed request UR.

Arguments:serviceUR
Return Type:serviceStruct
Privilege:Public
PDL: No PDL

GetServicesBySite - This method returns a list of service type request IDs which were processed by the specified site.

Arguments:siteId
Return Type:Void
Privilege:Public
PDL: No PDL

GetServicesByUser - This method returns a list of service type of request IDs which were requested by the specified user.

Arguments:userID
Return Type:itemIDList
Privilege:Public
PDL: No PDL

GetTrackableItemByParent - This method returns a list of request IDs which are the children requests for the specified request.

Arguments:itemID
Return Type:itemIDList
Privilege:Public
PDL: No PDL

UpdateItemStatus - This method sets the state of the specified request to the passed state.

Arguments:itemID, state
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

UpdateOrderCost - This method sets the resource utilization data of the specified order type of request to the passed values.

Arguments:itemID, orderCost
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

UpdateRequestCost - This method sets the resource utilization data of the specified sub-order type of request to the passed values.

Arguments:itemID, requestCost

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

UpdateServiceCost - This method sets the resource utilization data of the specified service type of request to the passed values.

Arguments:itemID, serviceCost

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

Associations:

The MsAcTrackingMgr class has associations with the following classes:

Class: MsAcTrackingDB exchangesdatawith

Class: EcPriceTableB providepriceforcancelledrequests

Class: MsAcUsrProfile updateaccountbalance

Class: MsAcUsrProfileMgr updatesuserprofile

Class: MsAcTrackingUI uses

6.2.3.25 MsAcTrackingUI Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class is the user interface to the request tracking server. This class receives input from the operator and based on that input, will perform the action (with such actions as sorting a list) or will issue requests to the MsAcTrackingMgr to request data to be displayed.

Attributes:

detailScreenCmd - This attribute contains the command entered by the operator from the screen which displays the detail of a request.

Data Type:enum

Privilege:Private

Default Value:

menuScreenCmd - This attribute contains the command entered by the operator from the screen which displays a list of requests.

Data Type:enum
Privilege:Private
Default Value:

reportID

requestIdQuery - This attribute contains the unique request ID which is being asked for.
Data Type:EcTLong
Privilege:Private
Default Value:

searchString - This is a search string that can be input by the user to perform searches.
Data Type:RWCString
Privilege:Private
Default Value:

shipScreenCmd - This attribute contains the command entered by the operator from the screen which displays the shipping information for an order type of request.
Data Type:enum
Privilege:Private
Default Value:

sortBy - This attribute is used to specify how a list displayed to the operator is to be sorted.
Data Type:enum
Privilege:Private
Default Value:

userIdQuery - This attribute stores the ECS user identification which is used to query for Orders or root-level services which have been initiated by the user.
Data Type:RWCString
Privilege:Private
Default Value:

Operations:

DisplayItemsForSite - This method displays to the user a list of orders, sub-orders, and services which have been or are being processed at this site.
Arguments:RWCString siteName, StateListType stateList
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayOrderByUR - This method displays to the operator the detailed information of the order specified by UR.

Arguments:EcTUR orderUR
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayOrderInfo - This method displays to the operator the detailed information of the order specified by the passed request ID.

Arguments:EcTLong itemId
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayOrdersByUser - This method displays to the operator a list of orders, sub-orders, and services which have been initiated by the specified user.

Arguments:RWCString userId
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayOrdersForSite - This method displays to the user a list of orders which have been or are being processed at this site.

Arguments:RWCString siteName, StateListType stateList
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayRequestInfo - This method displays to the operator the detailed information of the sub-order specified by the passed request ID.

Arguments:EcTLong itemId
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayRequestsForSite - This method displays to the user a list of sub-orders which have been or are being processed at this site.

Arguments:RWCString siteName, StateListType stateList
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

DisplayRqstServByOrder - This method displays to the user a list of sub-orders and services which are subparts of the specified order.

Arguments:EcTLong itemId
Return Type:EcTVoid
Privilege:Public

PDL: No PDL

DisplayRqstServByRqst - This method displays to the user a list of sub-orders and services which are subparts of the specified sub-order.

Arguments:EcTLong itemId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

DisplayRqstServByServ - This method displays to the user a list of services which are subparts of the specified service.

Arguments:EcTLong itemId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

DisplayServiceByUR - This method displays to the operator the detailed information of the service specified by the passed request ID.

Arguments:EcTUR serviceUR

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

DisplayServiceByUser - This method displays to the operator the detailed information of the service specified by the passed service UR.

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

DisplayServiceForSite - This method displays to the user a list of root-level services which have been or are being processed at this site.

Arguments:RWCString siteName, StateListType stateList

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

DisplayServiceInfo - This method displays to the operator the detailed information of the service specified by the passed request ID.

Arguments:EcTLong itemId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

Associations:

The MsAcTrackingUI class has associations with the following classes:

Class: MsAcCostAcctReport requestcostreport

Class: MsAcTrackingMgr uses

6.2.3.26 MsAcUserAuditTrail Class

Parent Class:MsAcAuditTrail

Attributes:

All Attributes inherited from parent class

Operations:

MsAcUserAuditTrail

Arguments:

Return Type:Void

Privilege:Public

~MsAcUserAuditTrail

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcUserAuditTrail class has associations with the following classes:

None

6.2.3.27 MsAcUsrName Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

Attributes:

firstName

Data Type:

Privilege:Private

Default Value:

lastName

Data Type:

Privilege:Private

Default Value:

middleInit

Data Type:

Privilege:Private

Default Value:

title

Data Type:

Privilege:Private

Default Value:

Operations:**GetFirstName**

Arguments:

Return Type:Void

Privilege:Public

GetLastName

Arguments:

Return Type:Void

Privilege:Public

GetMiddleInit

Arguments:

Return Type:Void

Privilege:Public

GetTitle

Arguments:

Return Type:Void

Privilege:Public

MsAcUserName

Arguments:

Return Type:Void

Privilege:Public

SetFirstName

Arguments:RWCString
Return Type:Void
Privilege:Public

SetLastName

Arguments:RWCString
Return Type:Void
Privilege:Public

SetMiddleInit

Arguments:RWCString
Return Type:Void
Privilege:Public

SetTitle

Arguments:RWCString
Return Type:Void
Privilege:Public

~MsAcUserName

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsAcUsrName class has associations with the following classes:
MsAcUsrProfile (Aggregation)

6.2.3.28 MsAcUsrNameP Class

Parent Class:MsAcUsrName

Attributes:

All Attributes inherited from parent class

Operations:

MsAcUsrNameP

Arguments:MsAcUsrName&
Return Type:Void
Privilege:Public

MsAcUserNameP

Arguments:RWDBReader&

Return Type:Void

Privilege:Public

operator<<

Arguments:RWDBInserter&, MsAcUserNameP&

Return Type:RWDBInserter&

Privilege:Public

update

Arguments:RWDBUpdater&, RWCString, RWCString, RWCString, RWCString

Return Type:EcTVoid

Privilege:Public

~MsAcUserNameP

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcUserNameP class has associations with the following classes:

None

6.2.3.29 MsAcUserProfile Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

Attributes:**PI**

Data Type:

Privilege:Private

Default Value:

accountBalance

Data Type:

Privilege:Private

Default Value:

accountNumber

Data Type:

Privilege:Private

Default Value:

affiliation

Data Type:

Privilege:Private

Default Value:

altMailAddr

Data Type:

Privilege:Private

Default Value:

altShipAddr

Data Type:

Privilege:Private

Default Value:

billAddr

Data Type:

Privilege:Private

Default Value:

creationDate

Data Type:

Privilege:Private

Default Value:

emailAddr

Data Type:

Privilege:Private

Default Value:

expirationDate

Data Type:

Privilege:Private

Default Value:

homeDAAC

Data Type:

Privilege:Private

Default Value:

mailAddr

Data Type:
Privilege:Private
Default Value:

mediaPref

Data Type:
Privilege:Private
Default Value:

organization

Data Type:
Privilege:Private
Default Value:

privilegeLevel

Data Type:
Privilege:Private
Default Value:

projectName

Data Type:
Privilege:Private
Default Value:

researchFiled

Data Type:
Privilege:Private
Default Value:

shipAddr

Data Type:
Privilege:Private
Default Value:

sponsor

Data Type:
Privilege:Private
Default Value:

telNum

Data Type:
Privilege:Private
Default Value:

userId

Data Type:

Privilege:Private

Default Value:

userName

Data Type:

Privilege:Private

Default Value:

Operations:

GetAccountBalance

Arguments:

Return Type:EcTLong accountBalance

Privilege:Public

GetAccountNumber

Arguments:

Return Type:Void

Privilege:Public

GetAffiliation

Arguments:

Return Type:Void

Privilege:Public

GetAltMailAddrCity

Arguments:

Return Type:Void

Privilege:Public

GetAltMailAddrFax

Arguments:

Return Type:Void

Privilege:Public

GetAltMailAddrPhone

Arguments:

Return Type:Void

Privilege:Public

GetAltMailAddrState

Arguments:

Return Type:Void
Privilege:Public

GetAltMailAddrStreet1

Arguments:
Return Type:Void
Privilege:Public

GetAltMailAddrStreet2

Arguments:
Return Type:Void
Privilege:Public

GetAltMailAddrZip

Arguments:
Return Type:Void
Privilege:Public

GetAltMailCountry

Arguments:
Return Type:Void
Privilege:Public

GetAltShipAddrCity

Arguments:
Return Type:Void
Privilege:Public

GetAltShipAddrFax

Arguments:
Return Type:Void
Privilege:Public

GetAltShipAddrPhone

Arguments:
Return Type:Void
Privilege:Public

GetAltShipAddrState

Arguments:
Return Type:Void
Privilege:Public

GetAltShipAddrStree2

Arguments:

Return Type:Void
Privilege:Public

GetAltShipAddrStreet1

Arguments:
Return Type:Void
Privilege:Public

GetAltShipAddrZip

Arguments:
Return Type:Void
Privilege:Public

GetAltShipCountry

Arguments:
Return Type:Void
Privilege:Public

GetAltShipState

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrCity

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrCountry

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrFax

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrPhone

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrState

Arguments:

Return Type:Void
Privilege:Public

GetBillAddrStreet1

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrStreet2

Arguments:
Return Type:Void
Privilege:Public

GetBillAddrZip

Arguments:
Return Type:Void
Privilege:Public

GetCreationDate

Arguments:
Return Type:Void
Privilege:Public

GetEmailAddress

Arguments:
Return Type:Void
Privilege:Public

GetExpirationDate

Arguments:
Return Type:Void
Privilege:Public

GetHomeDAAC

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrCity

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrCountry

Arguments:

Return Type:Void
Privilege:Public

GetMailAddrFax

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrPhone

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrState

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrStreet1

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrStreet2

Arguments:
Return Type:Void
Privilege:Public

GetMailAddrZip

Arguments:
Return Type:Void
Privilege:Public

GetMailAddress

Arguments:
Return Type:Void
Privilege:Public

GetMediaPref

Arguments:
Return Type:Void
Privilege:Public

GetOrganization

Arguments:

Return Type:Void
Privilege:Public

GetPIFirstName

Arguments:
Return Type:Void
Privilege:Public

GetPILastName

Arguments:
Return Type:Void
Privilege:Public

GetPIMiddleInit

Arguments:
Return Type:Void
Privilege:Public

GetPITitle

Arguments:
Return Type:Void
Privilege:Public

GetPrivilegeLevel

Arguments:
Return Type:Void
Privilege:Public

GetProjectName

Arguments:
Return Type:Void
Privilege:Public

GetResearchField

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrCity

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrCountry

Arguments:

Return Type:Void
Privilege:Public

GetShipAddrFax

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrPhone

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrState

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrStreet1

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrStreet2

Arguments:
Return Type:Void
Privilege:Public

GetShipAddrZip

Arguments:
Return Type:Void
Privilege:Public

GetSponsor

Arguments:
Return Type:Void
Privilege:Public

GetTelNum

Arguments:
Return Type:Void
Privilege:Public

GetUserFirstName

Arguments:

Return Type:Void
Privilege:Public

GetUserId

Arguments:
Return Type:Void
Privilege:Public

GetUserLastName

Arguments:
Return Type:Void
Privilege:Public

GetUserMiddleInit

Arguments:
Return Type:Void
Privilege:Public

GetUserTitle

Arguments:
Return Type:Void
Privilege:Public

MsUserProfile

Arguments:EcTVoid
Return Type:Void
Privilege:Public

SetAccountBalance

Arguments:EcTLong newBalance
Return Type:EcTVoid
Privilege:Public

SetAccountNumber

Arguments:RWCString
Return Type:Void
Privilege:Public

SetAffiliation

Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailAddrCity

Arguments:RWCString

Return Type:Void
Privilege:Public

SetAltMailAddrFax
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailAddrPhone
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailAddrState
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailAddrStreet1
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailAddrStreet2
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailAddrZip
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltMailCountry
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipAddrCity
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipAddrFax
Arguments:RWCString

Return Type:Void
Privilege:Public

SetAltShipAddrPhone
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipAddrState
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipAddrStree2
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipAddrStreet1
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipAddrZip
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipCountry
Arguments:RWCString
Return Type:Void
Privilege:Public

SetAltShipState
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddrCountry
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddrFax
Arguments:RWCString

Return Type:Void
Privilege:Public

SetBillAddrPhone
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddrState
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddrStreet1
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddrZip
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddtCity
Arguments:RWCString
Return Type:Void
Privilege:Public

SetBillAddtStreet2
Arguments:RWCString
Return Type:Void
Privilege:Public

SetCreationDate
Arguments:const RWDate
Return Type:Void
Privilege:Public

SetEmailaddress
Arguments:RWCString
Return Type:Void
Privilege:Public

SetExpirationDate
Arguments:const RWDate

Return Type:Void
Privilege:Public

SetHomeDAAC
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrCity
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrCountry
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrFax
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrPhone
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrState
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrStreet1
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrStreet2
Arguments:RWCString
Return Type:Void
Privilege:Public

SetMailAddrZip
Arguments:RWCString

Return Type:Void
Privilege:Public

SetMailAddress

Arguments:RWCString
Return Type:Void
Privilege:Public

SetMediaPref

Arguments:RWCString
Return Type:Void
Privilege:Public

SetOrganization

Arguments:RWCString
Return Type:Void
Privilege:Public

SetPIFirstName

Arguments:RWCString
Return Type:Void
Privilege:Public

SetPILastName

Arguments:RWCString
Return Type:Void
Privilege:Public

SetPIMiddleInit

Arguments:RWCString
Return Type:Void
Privilege:Public

SetPITitle

Arguments:RWCString
Return Type:Void
Privilege:Public

SetPrivilegeLevel

Arguments:RWCString
Return Type:Void
Privilege:Public

SetProjectName

Arguments:RWCString

Return Type:Void
Privilege:Public

SetResearchField
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrCity
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrCountry
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrFax
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrPhone
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrState
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrStreet1
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrStreet2
Arguments:RWCString
Return Type:Void
Privilege:Public

SetShipAddrZip
Arguments:RWCString

Return Type:Void
Privilege:Public

SetSponsor

Arguments:RWCString
Return Type:Void
Privilege:Public

SetTelNum

Arguments:RWCString
Return Type:Void
Privilege:Public

SetUserFirstName

Arguments:RWCString
Return Type:Void
Privilege:Public

SetUserId

Arguments:RWCString
Return Type:Void
Privilege:Public

SetUserLastName

Arguments:RWCString
Return Type:Void
Privilege:Public

SetUserMiddleInit

Arguments:RWCString
Return Type:Void
Privilege:Public

SetUserTitle

Arguments:RWCString
Return Type:Void
Privilege:Public

~MsUserProfile

Arguments:EcTVoid
Return Type:Void
Privilege:Public

Associations:

The MsAcUsrProfile class has associations with the following classes:

Class: MsAcUsrProfileMgr manages

Class: MsAcTrackingMgr updateaccountbalance

MsAcRegUser (Aggregation)

6.2.3.30 MsAcUsrProfileMgr Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This class represents the User Profile Manager class that governs the update and maintenance of information in the MsAcUsrProfile class. An ECS science user's available balance will be retrieved using this class and be debited by the amount of each data product request received by MSS.

Attributes:

None

Operations:

DeleteProfile

Arguments:

Return Type:EcTVoid

Privilege:Public

InsertProfile

Arguments:

Return Type:EcTVoid

Privilege:Public

MsAcUserProfileMgr

Arguments:

Return Type:Void

Privilege:Public

ReplicateProfileToSMC

Arguments:

Return Type:EcTVoid

Privilege:Public

RetrieveProfile

Arguments:RWCString userId

Return Type:EcTVoid

Privilege:Public

RetrieveProfile

Arguments:RWCString lastname, firstName, middleInitial

Return Type:EcTVoid

Privilege:Public

RetrieveProfile

Arguments:RWCString accountNumber

Return Type:EcTVoid

Privilege:Public

RetrieveProfileList

Arguments:

Return Type:EcTVoid

Privilege:Public

UpdateProfile

Arguments:

Return Type:EcTVoid

Privilege:Public

~MSAcUserProfileMgr

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcUsrProfileMgr class has associations with the following classes:

Class: MsAcManager manages

Class: MsAcUsrProfile manages

Class: MsAcTrackingMgr updatesuserprofile

Class: MsAcManagerUI uses

Class: MsAcRegUserMgr uses

6.2.3.31 MsAcUsrProfileP Class

Parent Class:MsAcUsrProfile

Attributes:

All Attributes inherited from parent class

Operations:

MsAcUsrProfileP

Arguments:const RWCString&

Return Type:Void

Privilege:Public

MsAcUsrProfileP

Arguments:RWDBReader&

Return Type:Void

Privilege:Public

MsAcUsrProfileP

Arguments:MsAcUsrProfile&

Return Type:Void

Privilege:Public

operator<<

Arguments:RWDBUpdater&, MsAcUsrProfileP&

Return Type:RWDBUpdater&

Privilege:Public

operator<<

Arguments:RWDBInserter&, MsAcUsrProfileP&

Return Type:RWDBInserter&

Privilege:Public

Associations:

The MsAcUsrProfileP class has associations with the following classes:

None

6.2.3.32 MsAcUsrRequest Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

Attributes:

PI

accountNumber

affiliation

billAddr

emailAddr

expirationDate

homeDAAC

mailAddr

mediaPref

operator

organization

processDate

projectName

requestDate

researchFiled

shipAddr

sponsor

status

Data Type:RWCString

Privilege:Private

Default Value:

telNum

userName

usrRequestId

Operations:

GetAccountNumber

Arguments:

GetAffiliation

Arguments:

GetBillAddrCity

Arguments:

GetBillAddrCountry

Arguments:

GetBillAddrFax

Arguments:

GetBillAddrPhone

Arguments:

GetBillAddrState

Arguments:

GetBillAddrStreet1

Arguments:

GetBillAddrStreet2

Arguments:

GetBillAddrZip

Arguments:

GetEmailAddress

Arguments:

GetExpirationDate

Arguments:

GetHomeDAAC

Arguments:

GetMailAddrCity

Arguments:

GetMailAddrCountry

Arguments:

GetMailAddrFax

Arguments:

GetMailAddrPhone

Arguments:

GetMailAddrState

Arguments:

GetMailAddrStreet1

Arguments:

GetMailAddrStreet2

Arguments:

GetMailAddrZip

Arguments:

GetMailAddress

Arguments:

GetMediaPref

Arguments:

GetOperator

Arguments:

GetOrganization

Arguments:

GetPIFirstName

Arguments:

GetPILastName

Arguments:

GetPIMiddleInit

Arguments:

GetPITitle

Arguments:

GetProcessDate

Arguments:

GetProjectName

Arguments:

GetRequestDate

Arguments:

GetResearchField

Arguments:

GetShipAddrCity

Arguments:

GetShipAddrCountry

Arguments:

GetShipAddrFax

Arguments:

GetShipAddrPhone

Arguments:

GetShipAddrState

Arguments:

GetShipAddrStreet1

Arguments:

GetShipAddrStreet2

Arguments:

GetShipAddrZip

Arguments:

GetSponsor

Arguments:

GetStatus

Arguments:

GetTelNum

Arguments:

GetUserFirstName

Arguments:

GetUserLastName

Arguments:

GetUserMiddleInit

Arguments:

GetUserTitle

Arguments:

GetUsrRequestId

Arguments:

MsAcUsrRequest

Arguments:

SetAccountNumber

Arguments:RWCString

SetAffiliation

Arguments:RWCString

SetBillAddrCountry

Arguments:RWCString

SetBillAddrFax

Arguments:RWCString

SetBillAddrPhone

Arguments:RWCString

SetBillAddrState

Arguments:RWCString

SetBillAddrStreet1

Arguments:RWCString

SetBillAddrZip

Arguments:RWCString

SetBillAddtCity

Arguments:RWCString

SetBillAddtStreet2
Arguments:RWCString

SetEmailaddress
Arguments:RWCString

SetExpirationDate
Arguments:const RWDate

SetHomeDAAC
Arguments:RWCString

SetMailAddrCity
Arguments:RWCString

SetMailAddrCountry
Arguments:RWCString

SetMailAddrFax
Arguments:RWCString

SetMailAddrPhone
Arguments:RWCString

SetMailAddrState
Arguments:RWCString

SetMailAddrStreet1
Arguments:RWCString

SetMailAddrStreet2
Arguments:RWCString

SetMailAddrZip
Arguments:RWCString

SetMailAddress
Arguments:RWCString

SetMediaPref
Arguments:RWCString

SetOperator
Arguments:RWCString

SetOrganization
Arguments:RWCString

SetPIFirstName
Arguments:RWCString

SetPILastName
Arguments:RWCString

SetPIMiddleInit
Arguments:RWCString

SetPITitle
Arguments:RWCString

SetProcessDate
Arguments:RWCString

SetProjectName
Arguments:RWCString

SetRequestDate
Arguments:RWCString

SetResearchField
Arguments:RWCString

SetShipAddrCity
Arguments:RWCString

SetShipAddrCountry
Arguments:RWCString

SetShipAddrFax
Arguments:RWCString

SetShipAddrPhone
Arguments:RWCString

SetShipAddrState
Arguments:RWCString

SetShipAddrStreet1
Arguments:RWCString

SetShipAddrStreet2
Arguments:RWCString

SetShipAddrZip
Arguments:RWCString

SetSponsor
Arguments:RWCString

SetStatus
Arguments:RWCString

SetTelNum
Arguments:RWCString

SetUserFirstName
Arguments:RWCString

SetUserLastName
Arguments:RWCString

SetUserMiddleInit
Arguments:RWCString

SetUserTitle
Arguments:RWCString

SetUstRequestId
Arguments:RWCString

~MsAcUsrRequest
Arguments:

Associations:

The MsAcUsrRequest class has associations with the following classes:

Class: MsAcUsrRequestMgr manages

Class: MsAcManagerUI uses

6.2.3.33 MsAcUsrRequestMgr Class

Parent Class:Not Applicable

Public:No

Distributed Object:Yes

Purpose and Description:

Attributes:

None

Operations:

CreateUserRequest

Arguments:RWCString userReqId

Return Type:EcTVoid

Privilege:Public

DeleteUserRequest

Arguments:RWCString userReqId

Return Type:EcTVoid

Privilege:Public

MsAcUserRequestMgr

Arguments:

Return Type:EcTVoid

Privilege:Public

RetrieveUserRequest

Arguments:RWCString userReqId

Return Type:EcTVoid

Privilege:Public

RetrieveUserRequestList

Arguments:

Return Type:EcTVoid

Privilege:Public

UpdateUserRequest

Arguments:RWCString userReqId

Return Type:EcTVoid

Privilege:Public

~MsAcUserRequestMgr

Arguments:

Return Type:EcTVoid

Privilege:Public

Associations:

The MsAcUsrRequestMgr class has associations with the following classes:

Class: MsAcManager manages

Class: MsAcUsrRequest manages

Class: MsAcManagerUI uses

6.2.3.34 MsAcUsrRequestP Class

Parent Class:MsAcUsrRequest

Attributes:

All Attributes inherited from parent class

Operations:

MsAcUsrRequestP

Arguments:

Return Type:Void

Privilege:Public

~MsAcUsrRequestP

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcUsrRequestP class has associations with the following classes:

None

6.2.3.35 MsAcUsrResUsage Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

Attributes:

cpuLimit

Data Type:EcTInt

Privilege:Private

Default Value:

cpuUsage

Data Type:EcTInt

Privilege:Private

Default Value:

dataLimit

Data Type:EcTInt

Privilege:Private

Default Value:

dataUsage

Data Type:EcTInt

Privilege:Private

Default Value:

diskLimit

Data Type:EcTInt

Privilege:Private

Default Value:

diskUsage

Data Type:EcTInt

Privilege:Private

Default Value:

memoryLimit

Data Type:EcTInt

Privilege:Private

Default Value:

memoryUsage

Data Type:EcTInt

Privilege:Private

Default Value:

networkLimit

Data Type:EcTInt

Privilege:Private

Default Value:

networkUsage

Data Type:EcTInt

Privilege:Private

Default Value:

orderLimit

Data Type:EcTInt

Privilege:Private

Default Value:

orderUsage

Data Type:EcTInt

Privilege:Private

Default Value:

Operations:

GetCpuLimit

Arguments:cpuLimit

Return Type:EcTVoid

Privilege:Public

GetCpuUsage

Arguments:cpuUsage

Return Type:EcTVoid

Privilege:Public

GetDataLimit

Arguments:dataLimit

Return Type:EcTVoid

Privilege:Public

GetDataUsage

Arguments:dataUsage

Return Type:EcTVoid

Privilege:Public

GetDiskLimit

Arguments:diskLimit

Return Type:EcTVoid

Privilege:Public

GetDiskUsage

Arguments:diskUsage

Return Type:EcTVoid

Privilege:Public

GetMemoryLimit

Arguments:memoryLimit

Return Type:EcTVoid

Privilege:Public

GetMemoryUsage

Arguments:memoryUsage

Return Type:EcTVoid

Privilege:Public

GetNetworkUsage

Arguments:networkUsage

Return Type:EcTVoid

Privilege:Public

GetNetworkLimit

Arguments:networkLimit

Return Type:EcTVoid

Privilege:Public

GetOrderLimit

Arguments:orderLimit

Return Type:EcTVoid

Privilege:Public

GetOrderUsage

Arguments:orderUsage

Return Type:EcTVoid

Privilege:Public

MsAcUsrResUsage

Arguments:

Return Type:Void

Privilege:Public

SetCpuLimit

Arguments:cpuLimit

Return Type:EcTInt

Privilege:Public

SetCpuUsage

Arguments:cpuUsage

Return Type:EcTInt

Privilege:Public

SetDataLimit

Arguments:dataLimit

Return Type:EcTInt

Privilege:Public

SetDataUsage

Arguments:dataUsage

Return Type:EcTInt

Privilege:Public

SetDiskLimit

Arguments:diskLimit

Return Type:EcTInt

Privilege:Public

SetDiskUsage

Arguments:diskUsage

Return Type:EcTInt

Privilege:Public

SetMemoryLimit

Arguments:memoryLimit

Return Type:EcTInt

Privilege:Public

SetMemoryUsage

Arguments:memoryUsage

Return Type:EcTInt

Privilege:Public

SetNetworkLimit

Arguments:networkLimit

Return Type:EcTInt

Privilege:Public

SetNetworkUsage

Arguments:networkUsage

Return Type:EcTInt

Privilege:Public

SetOrderLimit

Arguments:orderLimit

Return Type:EcTInt

Privilege:Public

SetOrderUsage

Arguments:orderUsage

Return Type:EcTInt

Privilege:Public

~MsAcUsrResUsage

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcUsrResUsage class has associations with the following classes:

MsAcRegUser (Aggregation)

6.2.3.36 MsAcUsrResUsageP Class

Parent Class:MsAcUsrResUsage

Attributes:

All Attributes inherited from parent class

Operations:**MsAcUsrResUsageP**

Arguments:RWDBReader&

Return Type:Void

Privilege:Public

MsAcUsrResUsageP

Arguments:const MsAcUsrResUsage&

Return Type:Void

Privilege:Public

MsAcUsrResUsageP

Arguments:const RWCString&

Return Type:Void

Privilege:Public

operator<<

Arguments:RWDBInserter&, MsAcUsrResUsageP&

Return Type:RWDBInserter&

Privilege:Public

operator<<

Arguments:RWDBUpdater&, MsAcUsrResUsageP&

Return Type:RWDBUpdater&

Privilege:Public

~MsAcUsrResUsageP

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsAcUsrResUsageP class has associations with the following classes:

None

6.2.4 Accountability Management Dynamic Model

6.2.4.1 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 6.2-8.

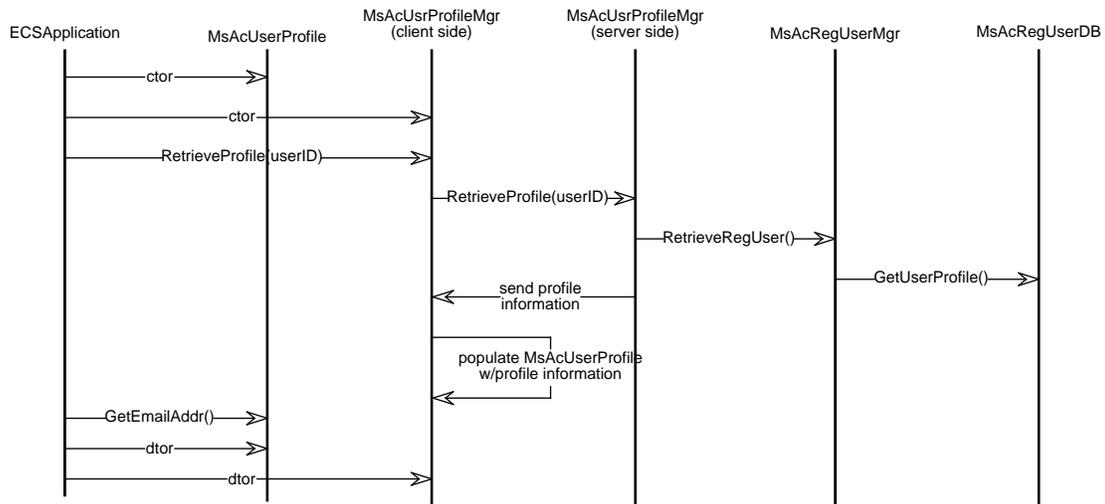


Figure 6.2-8. Retrieving a User's Email Address

6.2.4.1.1 Beginning Assumptions

None.

6.2.4.1.2 Interfaces with Other Subsystems and Segments

An ECS application that needs to retrieve a User Profile for the Email Address

6.2.4.1.3 Stimulus

An ECS application initiates a call to retrieve the User Profile for a registered user.

6.2.4.1.4 Participating Classes From the Object Model

MsAcUserProfile,
MsAcUsrProfileMgr,
MsAcRegUserMgr,
MsAcRegUserDB

6.2.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.2.4.1.6 Ending State

The calling ECS application retrieves the Email address, and the system, segment and subsystem are in a normal steady state.

6.2.4.1.7 Scenario Description

An ECS application creates an instance of MsAcUserProfile (a public class exported by the Accountability Application Service) and creates an instance of MsAcUsrProfileMgr (a public, distributed class). The ECS application then asks MsAcUsrProfileMgr to populate the MsAcUserProfile object with the RetrieveProfile method.

The MsAcUsrProfileMgr in the ECS application requests the profile information from the MsAcUsrProfileMgr which is residing in the Accountability Management server. The profile information is retrieved through the registered user manager object (MsAcRegUserMgr) who gets the information from the registered user database interface object (MsAcRegUserDB).

The information is passed back to the MsAcUsrProfileMgr object who populates the MsAcUserProfile object with the information. The ECS application is then returned control from its method call and the application calls the GetEmailAddr method to get the e-mail address. After the ECS application has gotten all the information it needs from the user profile, the MsAcUserProfile and MsAcUsrProfileMgr objects are deleted.

6.2.4.1 Request Tracking Overview

This scenario shows how the ECS application will use the Request Tracking key mechanism to report request state changes in near real-time and collect and report resource utilization for a Request. This scenario describes how a Product Order type of request would be processed. Ingest Request, User Request, and Operator Request types would be processed in a similar fashion, except the classes EcService and EcServiceEvent would be used. The scenario is depicted in Figure 6.2-9.

The following scenarios show the details of the overview scenario:

1. Creation of an order - Figure 6.2.10.
2. Collection of resource utilization for an order - Figure 6.2.11.

3. Updating the state of an order. This state reporting occurs in near real-time so that the operator can view the current state of the requests in the system. - Figure 6.2.12.
4. Spawning a sub-order from an order - Figure 6.2.13.
5. Collection of resource utilization for a sub-order - Figure 6.2.14.
6. Cancellation of a sub-order - Figure 6.2.15.
7. Completion of processing on the order - Figure 6.2.16.

6.2.4.1.1 Beginning Assumptions

None.

6.2.4.1.2 Interfaces with Other Subsystems and Segments

ECS Application1 and ECS Application2 - these could be any ECS applications which process any part of an ECS Request Type.

Request Tracking Server - this is the part of the Accountability Management CI (in the Management Subsystem) which receives and stores the Request Tracking information in the management database and displays the information to User Services personnel as requested.

6.2.4.1.3 Stimulus

An ECS User submits a request for an ECS product.

6.2.4.1.4 Participating Classes From the Object Model

EcOrder

EcSubOrder

The classes of the Request Tracking Server

6.2.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.2.4.1.6 Ending State

The system, segment and the subsystem are in a normal, steady state.

6.2.4.1.7 Scenario Description

1. ECS Application1 receives a Product Order type of request.
2. ECS Application1 constructs an EcOrder object. This object will stay around as long as this application is processing the associated product order.
3. EcOrder sends the information about the order (received when the object was created) to the Request Tracking Server to be stored in the management database which can then be displayed to the operator, providing a near real-time list of requests in the system.
4. As the order changes state during its processing, that state change is reported to EcOrder. EcOrder immediately sends the state change information to the Request Tracking Server at the MSS server. This allows the operator to see the current state of the request in near real-time.

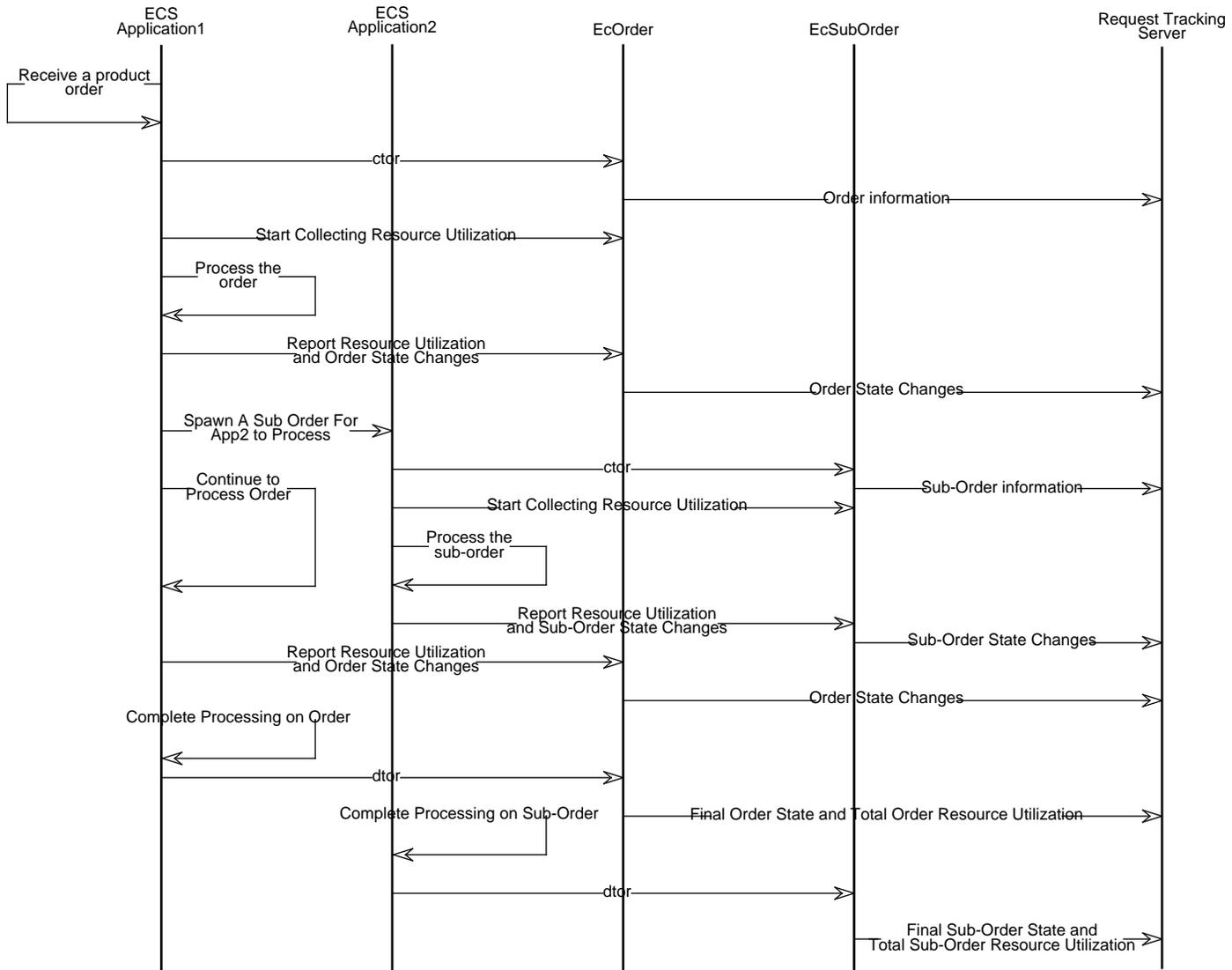


Figure 6.2-9. Request Tracking Overview

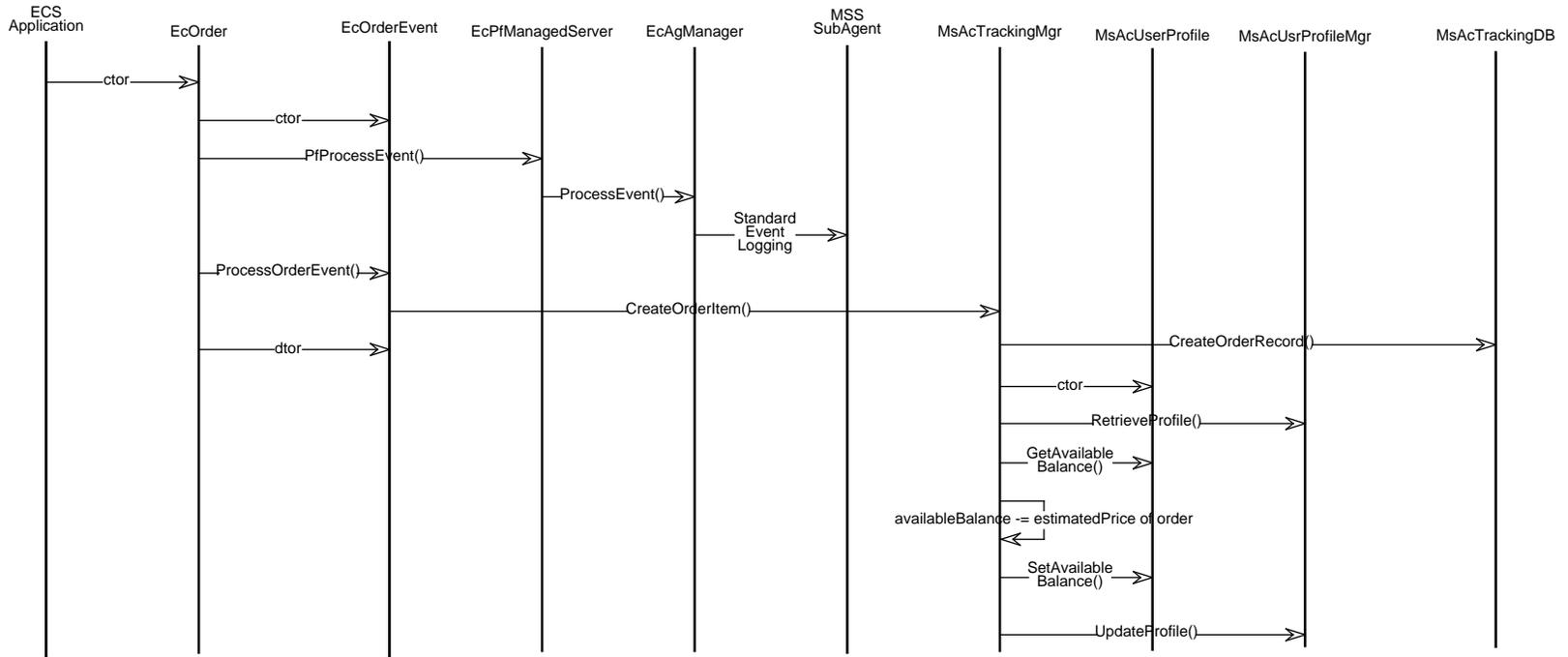


Figure 6.2-10. Request Tracking-Creating An Order

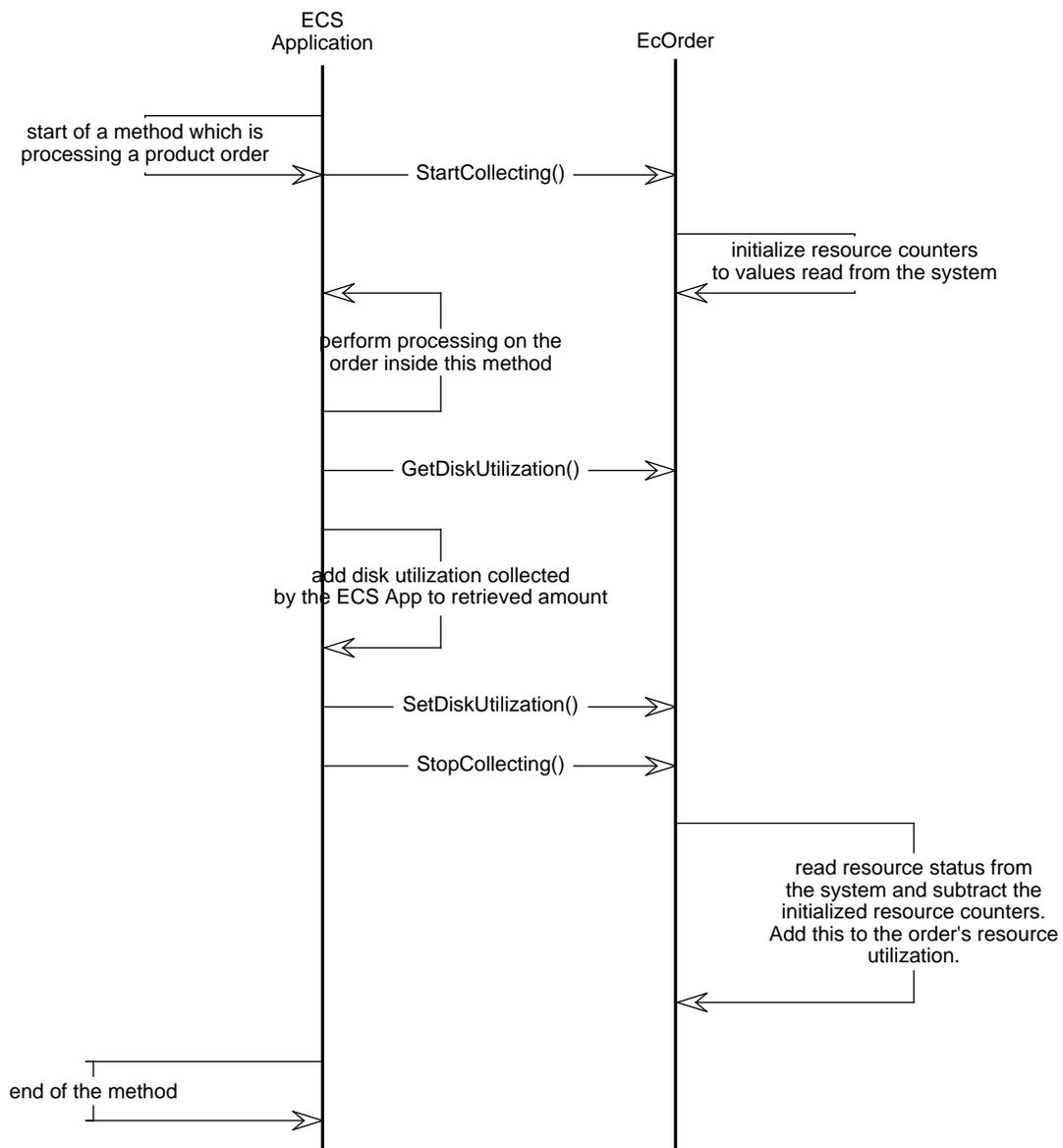


Figure 6.2-11. Request Tracking-Collecting Resource Utilization For An Order

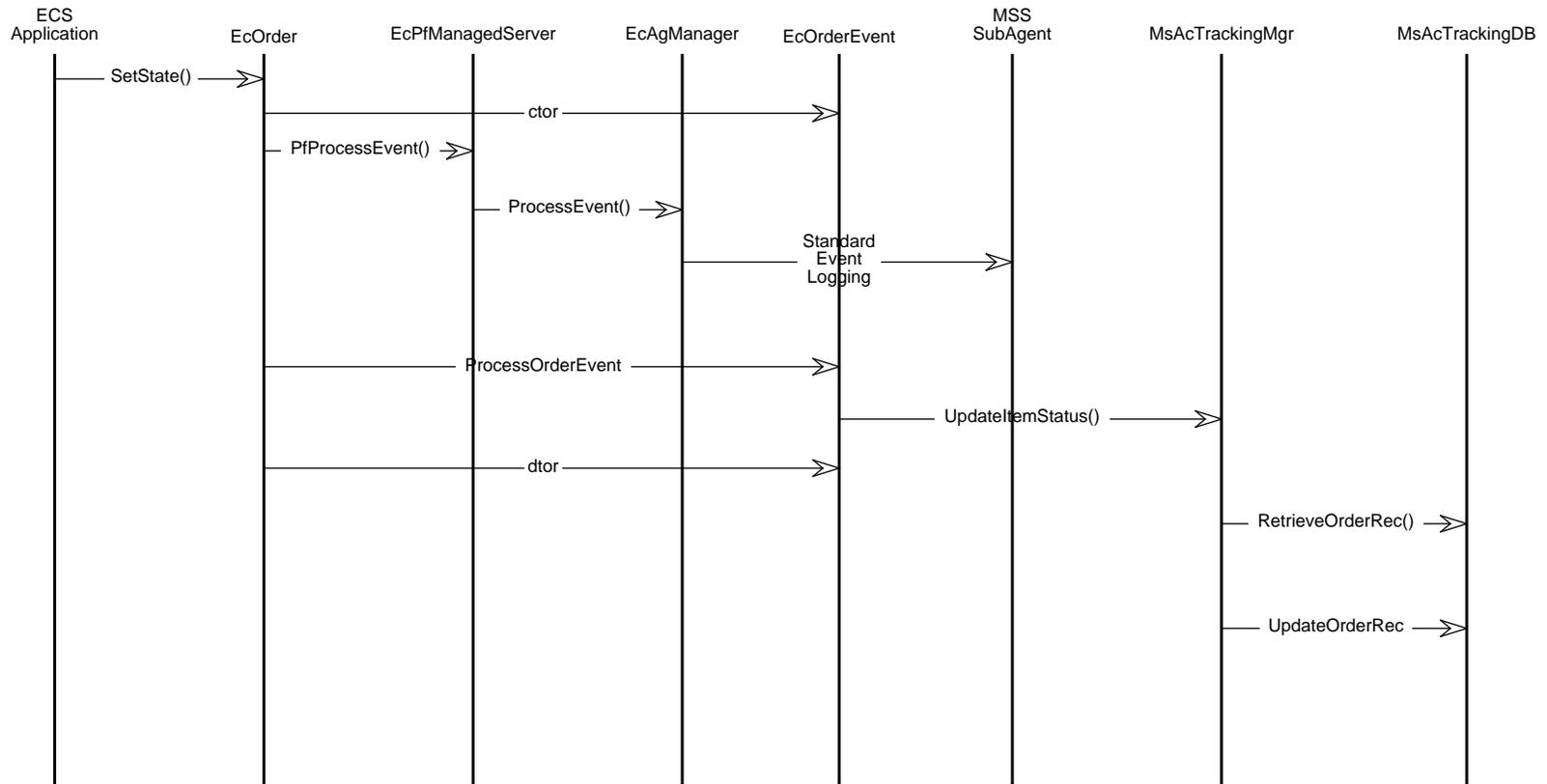


Figure 6.2-12. Request Tracking-Update The State of An Order

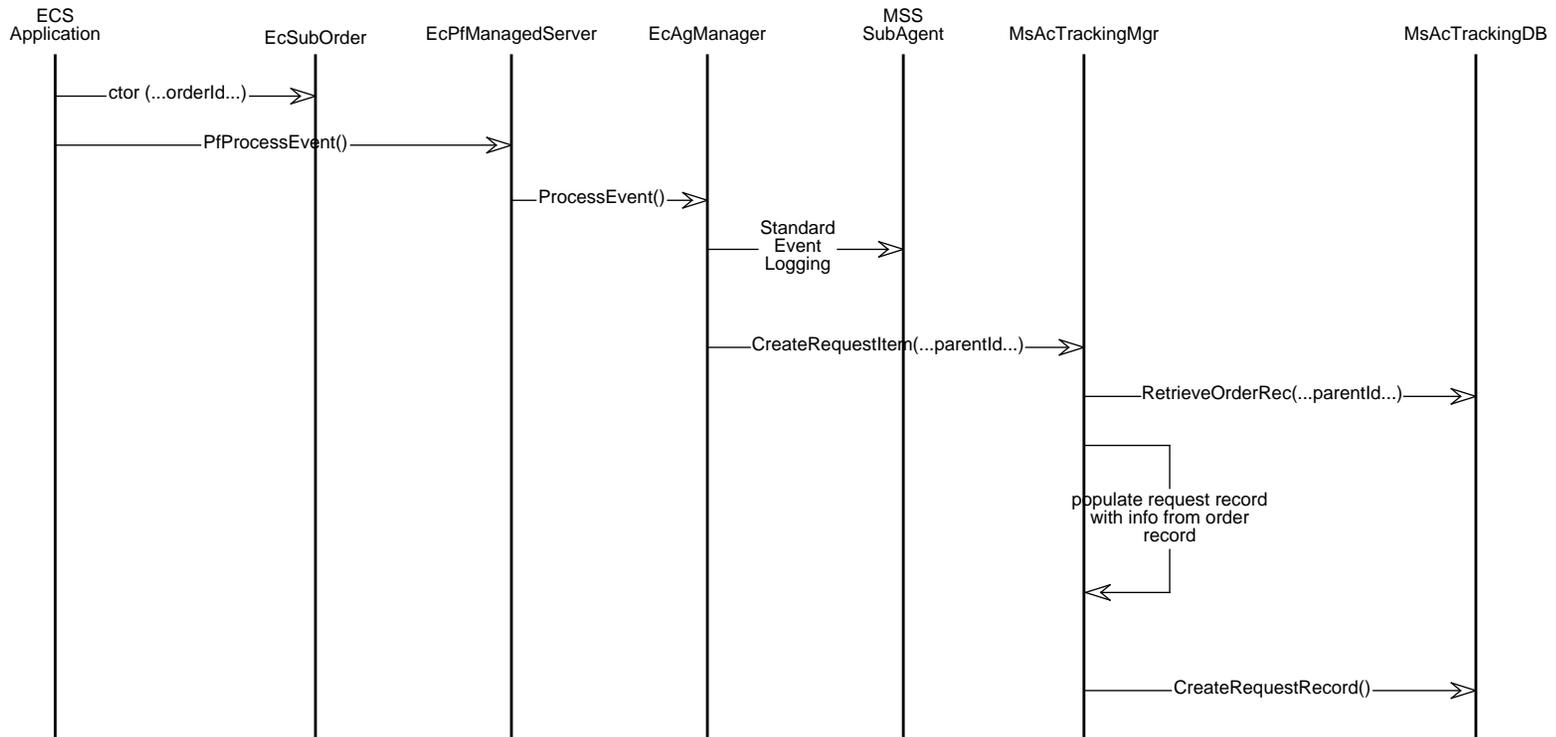


Figure 6.2-13. Request Tracking-Spawning A Sub-Order From An Order

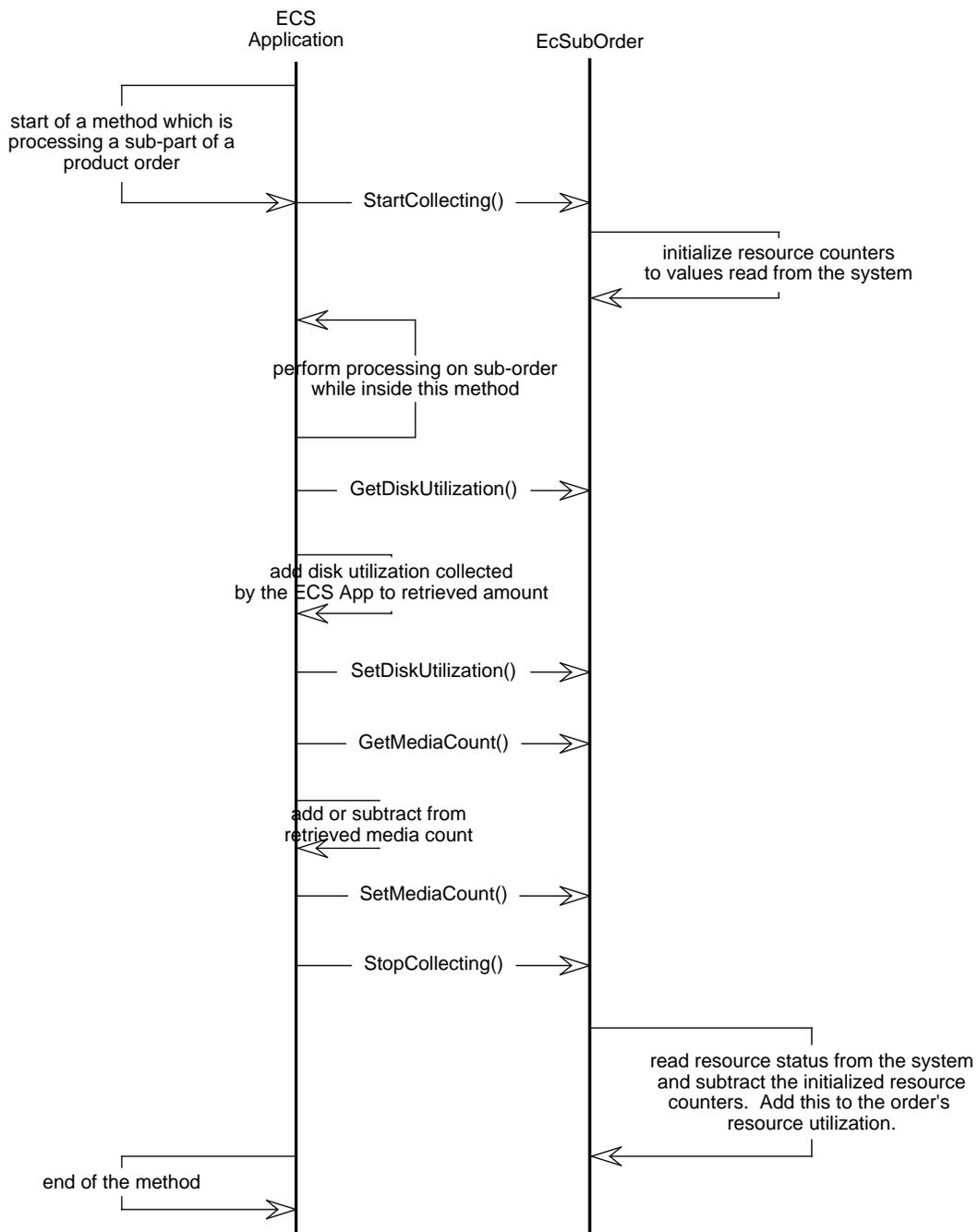


Figure 6.2-14. Request Tracking-Collecting Resource Utilization For A Sub-Order

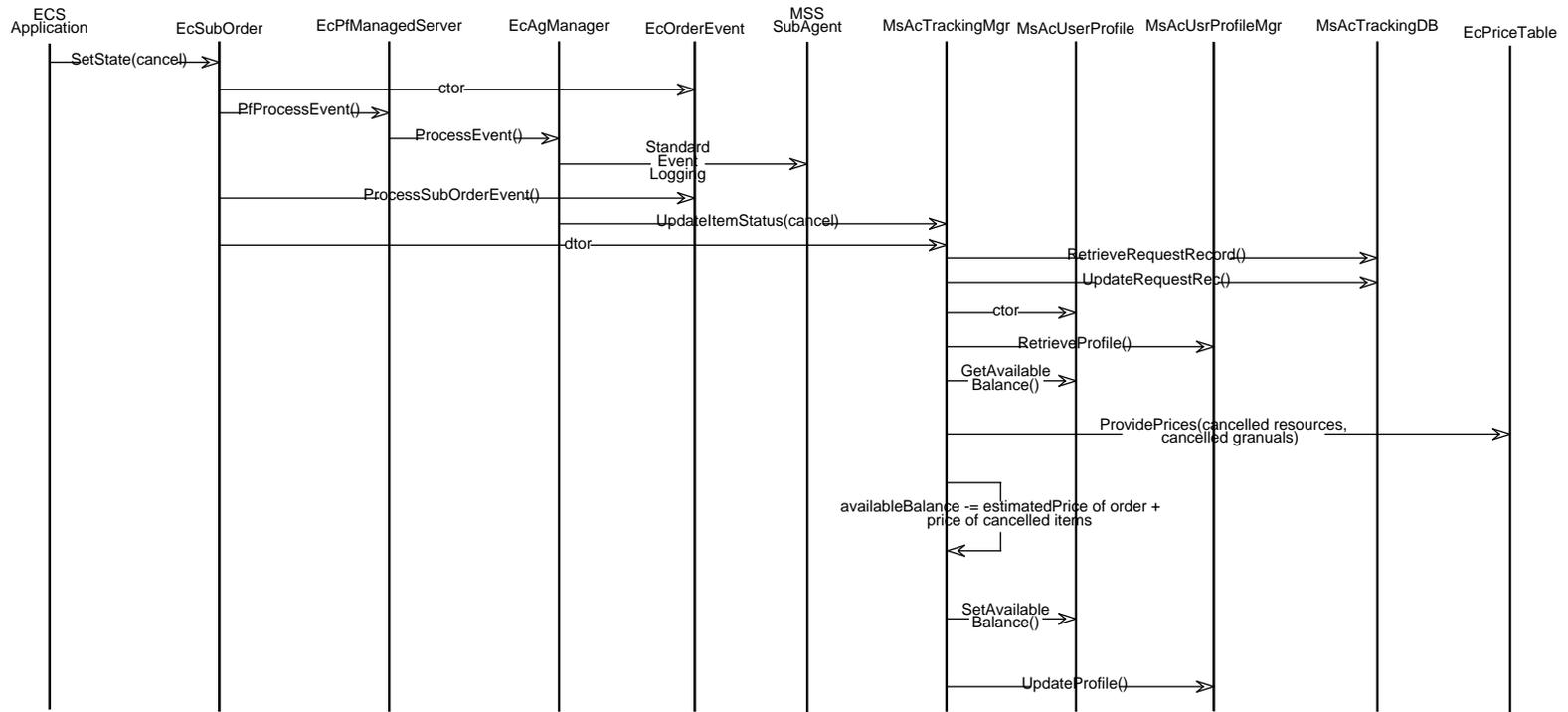


Figure 6.2-15. Request Tracking-Canceling A Sub-Order

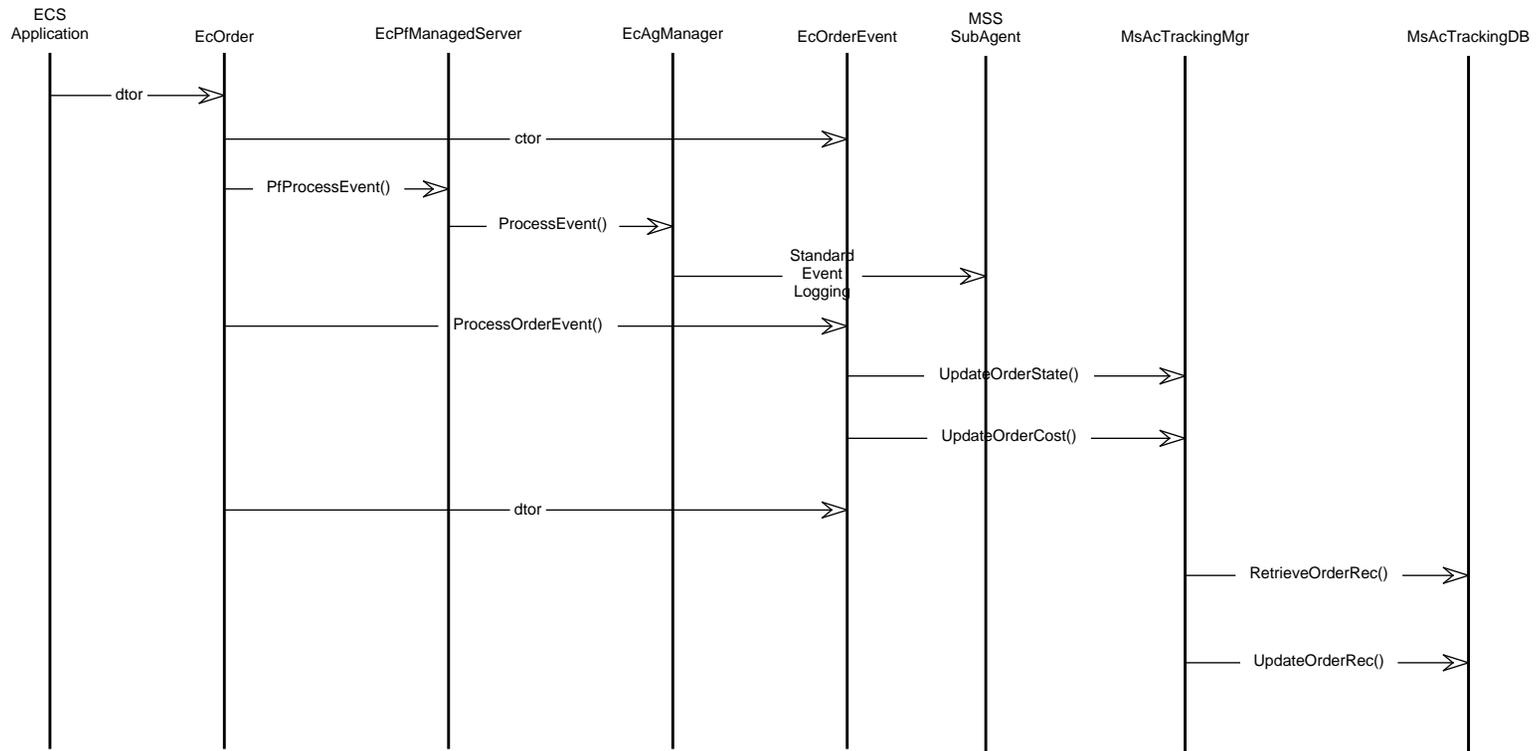


Figure 6.2-16. Request Tracking-Finished Processing Of An Order

4. Inside each method which ECS Application1 executes to process the order, a call is made to the start utilization collection method in EcOrder. At the end of the method, a call is made to EcOrder to stop collecting utilization data. This will enable EcOrder to automatically collect process-related utilization data associated with the request. This process is represented by the Start Collecting Resource Utilization event.
5. ECS Application1 performs appropriate processing on the Product Order.
6. During the processing, as ECS Application1 uses resources outside of process-related utilization, ECS Application1 reports the resources used to the EcOrder object. In addition, if the state of the Product Order request changes, those state changes are reported to EcOrder. EcOrder reports the state change to the Request Tracking Server at the MSS server to provide near real-time request status to the operator. The resource utilization (cost) is stored with the EcOrder object until the request has completed processing. This process is represented by the Report Resource Utilization and Order State Changes event.
7. State changes are reported to the Request Tracking Server at the MSS server by EcOrder, but resource utilization is collected and only reported at the end of the life of the order.
8. During the processing of the Product Order, ECS Application finds that it must spawn a sub-part of this Product Order and have another application (ECS Application2) process the sub-part.
9. When the sub-request of the Product Order is received by ECS Application2, the application creates a EcSubOrder object. As part of this creation, the ID of EcOrder is passed - which will enable the Request Tracking Server to relate the sub-order with its associated order.
10. EcSubOrder sends the information about the sub-request back to the Request Tracking Server at the MSS server
11. ECS Application1 and ECS Application2 then continue processing each of their parts of the Product Order independently. The state changes and cost information are reported and collected for each part independently as described above. The state changes are immediately sent to the Request Tracking Server at the MSS server to provide near real-time state tracking of requests and the resource utilization (cost) is stored with the EcSubOrder object until the request has completed processing.
12. When an application is finished processing their part of the Product Order, the application destroys the EcOrder or EcSubOrder. This causes the final state of the request as well as the total resource utilization of the request part to be reported back to the Request Tracking Server at the MSS server.

6.2.5 Accountability Management Structure

Table 6.2-1 lists the components of the Accountability Management Service.

Table 6.2-1. Accountability Management Components

Component Name	COTS/Custom
Accountability Manager	Custom
User Account User Interface	Custom
User Account User Interface	Custom
Account Creation Management	Custom
User Profile Access	Custom
Request Tracking Management	Custom
Request Tracking Collection	Custom

6.2.5.1 Accountability Manager CSC

Purpose and Description

The Accountability Manager CSC manages the operation of the service and provides the audit trail reporting functionality. This class inherits from EcPfManagedServer to provide the management framework for the service.

6.2.5.2 User Account User Interface

Purpose and Description

This CSC provides the graphical user interface for the operator to perform user account management.

6.2.5.3 User Account Management CSC

Purpose and Description

This CSC performs the account management functions on the registered user accounts based on inputs from the User Account User Interface. This includes account creation, deletion, and modification.

6.2.5.4 Account Creation Management CSC

Purpose and Description

This CSC receives user registration requests and keeps the requests as pending user accounts in a database until the User Account User Interface approves the pending account or deletes the pending account.

6.2.5.5 User Profile Access 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.

6.2.5.6 Request Tracking Management CSC

Purpose and Description

This CSC manages the request tracking information. The class receives request tracking information from the Request Tracking Collection CSC and updates the request tracking information in the database. This CSC also includes a user interface to the tracking database which enables the operator to see the state of requests executing or completed executing in the ECS system in near real-time and allows the operator to generate and output cost accounting type reports from the resource utilization data which is collected for each request.

6.2.5.6 Request Tracking Collection CSC

Purpose and Description

This CSC provides the Request Tracking Key Mechanism for ECS Applications to report request status changes back to a central database to be displayed to an operator in near real-time. This CSC contains the classes which the applications use to collect resource utilization. The classes are designed to support a hierarchy of requests and to allow each request in the hierarchy to be tracked independently. The Request Tracking Management CSC at the MSS server receives the state changes and the resource utilization for the requests to provide near real-time state tracking and off line resource cost reporting.

6.2.6 Accountability Management and Operation

6.2.6.1 System Management Strategy

The Accountability Management Service utilizes the public class exported by the process framework, represented by EcPfManagedServer. These classes facilitate the management of the service.

6.2.6.2 Operator Interfaces

The Accountability Management Service provides two graphical user interfaces. One user interface provides the operator the capability to view and delete pending user requests; to create registered user accounts from (approved) pending requests; and to modify and delete registered user accounts. The other user interface provides the operator the capability to view the current status of the requests (limited to those types mentioned in the overview) that are being or have been processed by the ECS system.

6.2.6.3 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

Using the Report Generation CSC associated with the Management Database, M&O staff will be able to generate a diverse range of reports such as user audit, data audit, and request tracking.

6.3 Billing and Accounting

6.3.1 Billing and Accounting Overview

ECS operations are supported by integrated and automated billing and accounting functions. The Enterprise Monitoring and Coordination (EMC) Billing and Accounting Application Service (BAAS) provides the mechanisms for ECS to price user data orders, invoice users for data and media, and meet ECS' needs to track and to provide financial data.

One of the BAAS' primary functions is to provide bill-back capabilities. The billing and invoicing functionality allows ECS to gather and track information on science user data orders, and to cost these orders based on different resources (e.g., disk utilization, CPU, media, connect time) or standard product ordered using pricing algorithms associated with each one. Policy will determine what prices are applied. A standard pricing policy for ECS products across sites is assumed.

The Data Processing Subsystem (DPS) and Data Server Subsystem (DSS) will provide the BAAS with accounting and resource data for science user orders which have been fulfilled so that the data may be priced. For purposes of estimating the price of a new product request, pricing algorithms maintained in pricing tables in BAAS will be made available to the DSS.

The billing and invoicing functionality allows ECS to inform accounts of their activity during a particular billing cycle and of the charges associated with such activity.

Policy may dictate that no charges be applied to any account, or to particular accounts; or that certain accounts be measured on resources consumed (e.g., number of tapes, number of images) rather than dollars. In such cases, the accounts would not receive a bill invoice but a statement of account does not anticipate payment. An account also would receive a statement of account instead of a bill invoice when the account has funds credited to it in advance of purchases of data. As charges are incurred by the account, these are deducted (debited) from the existing credits. The statements of account would show activity and balance remaining. An account's balance status also will be available for on-line consulting via the Client Subsystem (CLS).

Science user payments (made in the form of checks or purchase orders) will be credited to the appropriate accounts and forwarded to a designated NASA Financial Management Office (FMO) for processing and deposit. To track all financial data information gathered on science users there will be accounts set up and maintained by the BAAS. Science User's product orders and collected payments will be tracked by the BAAS COTS Accounts Receivables module. A current contract to purchase consumables resides with EDS, which precludes the need for the BAAS to report on payables until this contract ends.

The BAAS COTS will provide the following major functions: Billing and Invoicing, Accounts Receivable, Accounts Payable (dormant until the third party contract to purchase consumables ends), and General Ledger Reporting. A custom piece will be developed to provide cost accounting information gathered by the Accountability portion of MSS. The COTS will be capable of supporting the information tracking of 17,000 user accounts that are current accounts and are accessible on-line (not archived or historical). The Billing and Accounting Application package supports a minimum of 500 uniquely priced items or products. As part of the BAAS, it will provide an interface to other ECS Subsystems and Services to provide access to information such as pricing estimates and account balance data maintained within the Billing and Accounting Application package. The price estimate and account balance data supplied by the BAAS as part of a response

to a query (from DSS or CLS) will reflect the same pricing and invoicing options as would be presented to a client M&O user directly accessing the BAAS COTS Package. The interface to the COTS package can be accessed either through a Microsoft Windows client or a Unix server interface. Only registered users of the BAAS COTS will be able to access sensitive billing information. Access will be strictly controlled by the COTS, with only certain M&O user services personnel able to access those functions that have been explicitly granted to them. In addition, the COTS package is also capable of interfacing with the ECS Sybase Management Database directly.

6.3.2 Billing and Accounting Context

The Billing and Accounting Application Service (BAAS), as shown in the context diagram on figure 6.3-1, depends on other subsystems for the information it needs to price data orders, invoice the correct accounts, and to track financial data. Accounts are set up in the BAAS using User Profile Information received from the Accountability Management Service. Resource, data product, and accounting information are received from other subsystems, such as DSS, to which the BAAS provides information contained on pricing tables to be used in price estimation. The BAAS will be able to accept accounting data that may originate from DAAC unique accounting requirements. At the end of a billing cycle, statements are generated to be sent to the science user accounts. If payments are due from the science users, these payments are received at the SMC for posting to the appropriate account. Science user payments could be purchase orders as well. All science user payments are forwarded to the SMC to be recorded into the BAAS COTS and to be deposited in Federal Treasury accounts in accordance with GAO and OMB regulations and guidelines. In the case of overpayments, or if an account requests the return of funds already credited to it, refunds will be requested from the BAAS COTS. The CLS will be able to request and access account status information. Payment for consumable items provided by vendors will be issued by the agent authorized by the EDS contract which is out of scope of the BAAS, and so is not shown in the accompanying context diagram.

6.3.3 Billing and Accounting Object Model

The Object Model for the Billing and Accounting Application Service is shown on Figure 6.3-2.

6.3.3.1 EcAgCOTSManager Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

this abstract class embodies the characteristics and functionality of a manager object responsible for managing a single COTS process. It encapsulates all MSS management application functions into a single class. The COTS proxy agent developer is responsible for inheriting from this class and specializing it towards the COTS process to manage.

Attributes:

None

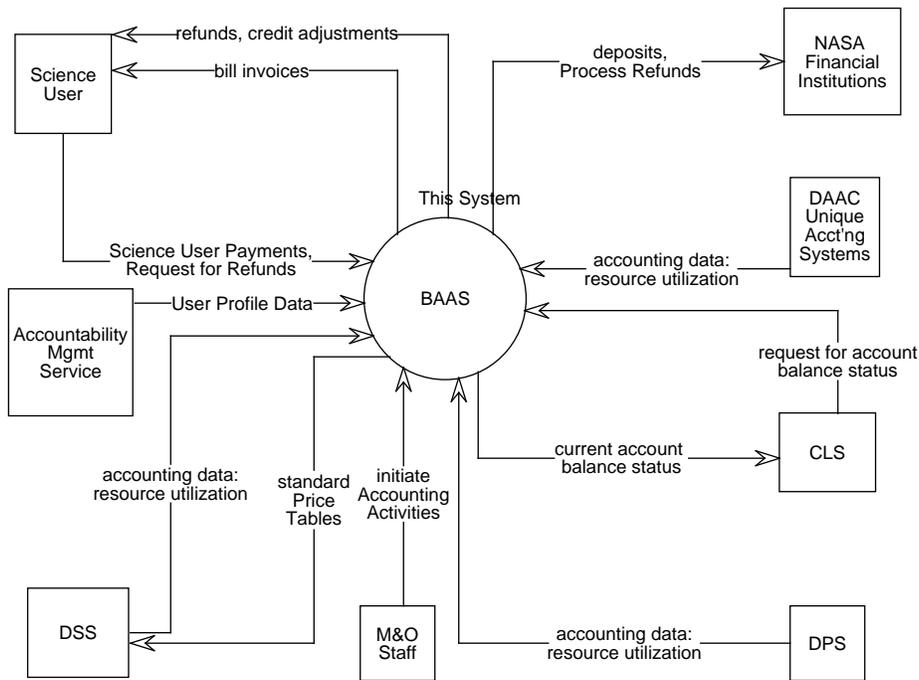


Figure 6.3-1. Billing and Accounting Context Diagram

Operations:

None

Associations:

The EcAgCOTSManger class has associations with the following classes:

None

6.3.3.2 EcPfManagedServer Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

This is the container class that starts up the event Manager, table Manager, monitor, port monitor, discoverer, subagent configuration, static buffer, and the deputy gate. This class also starts a thread that triggers scheduled events (i.e. polling ECS application's performance metrics).

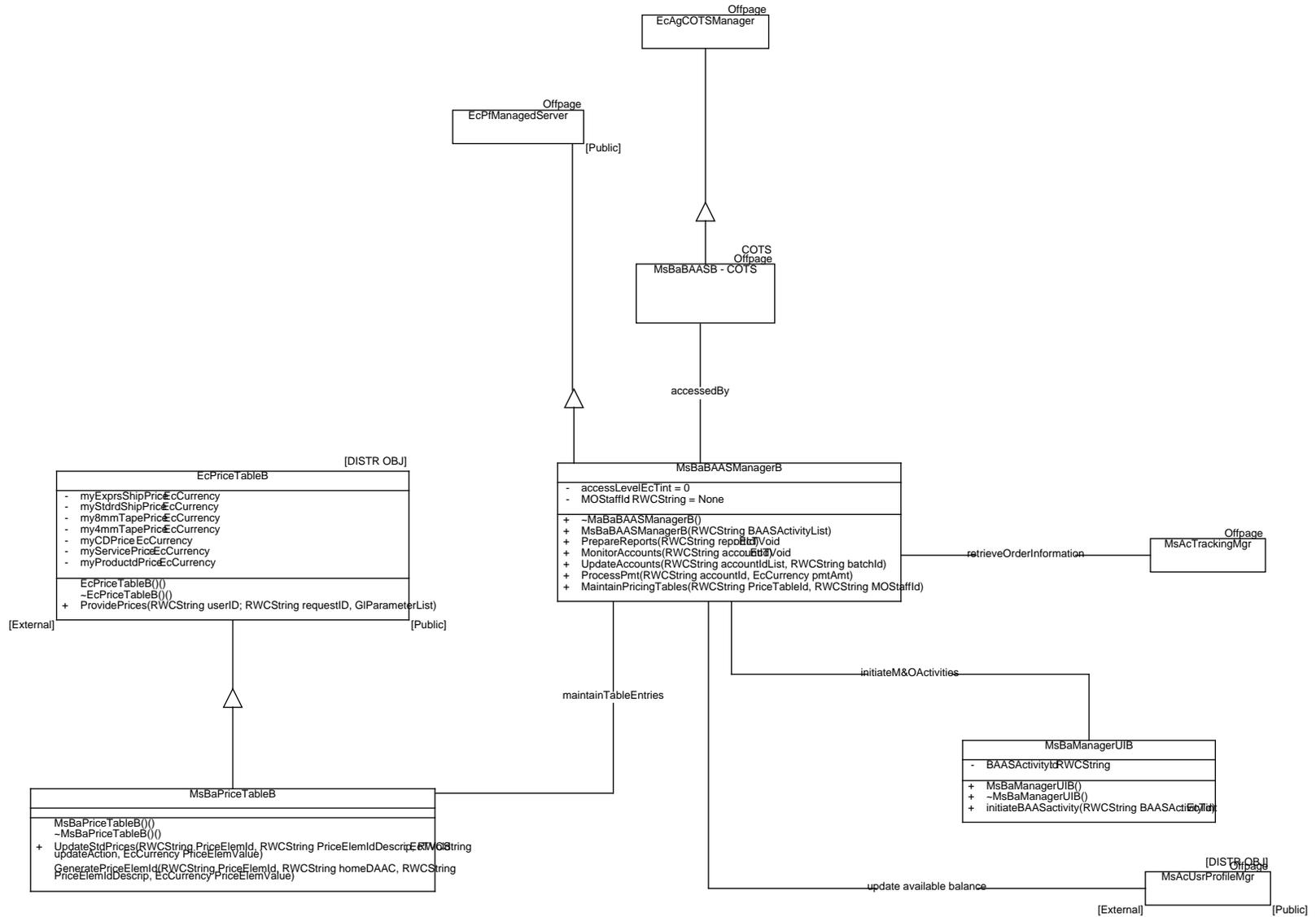


Figure 6.3-2. Billing and Accounting Object Model

Attributes:

None

Operations:

None

Associations:

The EcPfManagedServer class has associations with the following classes:

None

6.3.3.3 EcPriceTableB Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This class represents a public and distributed class that holds the prices of every billable item in the ECS inventory of products and services. Price of hard media and standard shipping costs are also maintained in this table.

Attributes:

my4mmTapePrice - This attribute represents the price charged for an 4mm tape that is part of an order for an ECS data product request.

Data Type:EcCurrency

Privilege:Private

Default Value:

my8mmTapePrice - This attribute represents the price charged for an 8mm tape that is part of an order for an ECS data product request.

Data Type:EcCurrency

Privilege:Private

Default Value:

myCDPrice - This attribute represents the price charged for one Compact Disc (CD) that will be used to store data that is part of an order for an ECS data product request.

Data Type:EcCurrency

Privilege:Private

Default Value:

myExprsShipPrice - This attribute represents the price charged for shipping an ECS data

product request by the express mail method contained in the user's profile information.

Data Type:EcCurrency

Privilege:Private

Default Value:

myProductdPrice - This attribute represents the price charged for a chargeable and identifiable ECS data product. The format and content of existing DAACs pricing lists of products, media and services will be incorporated into the structure of the EcPriceTableB as much as possible. Identifying products by a granule Id, size of granule and other price related factors will also be considered by the ECS system with the actual price guidelines for such attributes determined by an EOSDIS Pricing Policy committee.

Data Type:EcCurrency

Privilege:Private

Default Value:

myServicePrice - This attribute represents the price charged for a service (such as dataset subsetting) that is required in the process of fulfilling an order for an ECS data product request.

Data Type:EcCurrency

Privilege:Private

Default Value:

myStdrdShipPrice - This attribute represents the price charged for shipping an ECS data product request by the normal mail method contained in the user's profile information.

Data Type:EcCurrency

Privilege:Private

Default Value:

Operations:

EcPriceTableB

Arguments:

ProvidePrices - This method represents the summation of all the parameters in a user's request for a data product request including shipping charges. The global parameter list, GIParameterList passed in will contain the price element IDs corresponding to the type of product, type of service(s) required to satisfy the request, the type and number of media, and shipping method. This method will reference standard price entries for each of these parameters and arrive at a total price for the given request.

Arguments:RWCString userID; RWCString requestID, GIParameterList

Return Type:Void

Privilege:Public

~EcPriceTableB

Arguments:

Associations:

The EcPriceTableB class has associations with the following classes:

None

6.3.3.4 MsAcTrackingMgr Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents the manager class that collects order, request and service resource utilization statistics and status for ECS processes. This object is the interface that the request tracking event reporting objects (EcRequestEvent and its subclasses) as well as other ECS applications have to the request tracking database. The database will have near-real time status information about the requests as well as the final resource utilization of each request.

Attributes:

None

Operations:

None

Associations:

The MsAcTrackingMgr class has associations with the following classes:

Class: MsBaBAASManagerB retrieveOrderInformation

6.3.3.5 MsAcUsrProfileMgr Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This class represents the User Profile Manager class that governs the update and maintenance of information in the MsAcUsrProfile class. An ECS science user's available balance will be retrieved using this class and be debited by the amount of each data product request received by MSS.

Attributes:

None

Operations:

None

Associations:

The MsAcUsrProfileMgr class has associations with the following classes:

Class: MsBaBAASManagerB updateavailablebalance

6.3.3.6 MsBaBAASB-COTS Class

Parent Class:EcAgCOTSManager

Public:No

Distributed Object:No

Purpose and Description:

This class represents the COTS that provides bill-back capabilities for data purchased from ECS by science users. The Billing and Accounting Application Service (BAAS) COTS will provide the following major functions: Billing and Invoicing, Accounting: Accounts Receivable, Accounts Payable (deferred), General Ledger, Reporting, which will be consistent with generally accepted accounting principles and standards for the Federal Government where appropriate including General Accounting Office (GAO) standards Title 2 (Accounting), Title 3 (Audit), OMB Circular A-127 on Financial Management Systems and the Federal Financial Management System Requirements issued by the Joint Financial Management Improvement Program (JFMIP).

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsBaBAASB-COTS class has associations with the following classes:

Class: MsBaBAASManagerB accessedBy

6.3.3.7 MsBaBAASManagerB Class

Parent Class:EcPfManagedServer

Public:No

Distributed Object:No

Purpose and Description:

This class manages all processes for the Billing and Accounting Application Service (BAAS) that includes controlling the COTS, updating the Standard Price table, initiate and the retrieval of accounting data from the ECS Management Database via the MsAcTrackingMgr class and the adjusting of user profile balances via the MsAcUsrProfileMgr class. In addition, the initiation and generation of reports by the COTS is controlled by this class.

Attributes:

MOSTaffId - This attribute represents a unique Id associated with a member of the Maintenance and Operations (M&O) staff. Combined with accessLevel attribute, these attributes will provide the proper authorization levels to both the BAAS COTS package and the BAAS Cost Accounting function contained in the MsBaCostAcctB class.

Data Type:RWCString

Privilege:Private

Default Value:None

accessLevel - This attribute represents the current access level the user of this class has, which will determine which of the management activities will be permitted by the user associated with an MOSTaffID.

Data Type:EcTint

Privilege:Private

Default Value:0

Operations:

MaintainPricingTables - This method represents the update and maintenance functions associated with an instance of the EcPriceTableB, that will be invoked as MsBaPriceTableB. Two methods contained in the MsBaPriceTableB will allow M&O staff members to add, change or delete entries in the ECS-wide price tables as new product and service charges are determined. This method manages that process.

Arguments:RWCString PriceTableId, RWCString MOSTaffId

Return Type:Void

Privilege:Public

MonitorAccounts - This method represents the monitoring of accounts in the BAAS COTS. Monitoring includes but is not limited to verifying account balances, requesting the aging of accounts information through the COTS package, and other information that does not result in changing of information either in the COTS accounting database or the ECS Management Database.

Arguments:RWCString accountId

Return Type:EcTvoid

Privilege:Public

MsBaBAASManagerB - This method represents the constructor for this class.

Arguments:RWCString BAASActivityList

Return Type:Void

Privilege:Public

PrepareReports - This method represents the initiation, generation, retrieval or transmission of reports that are created by the BAAS COTS.

Arguments:RWCString reportId

Return Type:EcTVoid

Privilege:Public

ProcessPmt - This method represents the initiation of the depositing of funds collected from ECS Science Users for payment on their accounts. Monies collected will be tracked by the COTS package and will be forwarded to the designated NASA Financial Management Office (FMO) which will then deposit these payments into the appropriate U.S. Treasury accounts. The Credit Management component of the COTS Accounts Receivables module will have the capability to set up credit accounts which can be configured to not only grant credit approval and determine credit worthiness for ECS registered users, but perform credit approval automatically based on the amount of credit available in a Science User's account and the outstanding balance applied to a given order. This method will also be considered to be "overloaded" and be able to initiate refund requests from the FMO as well as track payments received. The COTS will have the capability to handle a variety of electronic funds transfer formats and electronic data interchanges (EDI) to ensure monies collected are deposited and transferred between government accounts in a timely fashion, if the FMO wishes to use these COTS capabilities.

Arguments:RWCString accountId, EcCurrency pmtAmt

Return Type:Void

Privilege:Public

UpdateAccounts - This method represents the updating of accounts in the BAAS COTS. Updating includes but is not limited to processing batches of shipped data product requests received from an ECS Data Center (DAAC), credit or debit memos, and other information that does result in the changing of information either in the COTS accounting database or the ECS Management Database. The COTS package will have the capability to execute SQL statements directly on a SYBASE database such as the ECS Management Database.

Arguments:RWCString accountIdList, RWCString batchId

Return Type:Void

Privilege:Public

~MaBaBAASManagerB - This method represents the destructor of this class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsBaBAASManagerB class has associations with the following classes:

Class: MsBaBAASB-COTS accessedBy

Class: MsBaManagerUIB initiateM&OActivities

Class: MsBaPriceTableB maintainTableEntries

Class: MsAcTrackingMgr retrieveOrderInformation

Class: MsAcUsrProfileMgr updateavailablebalance

6.3.3.8 MsBaManagerUIB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents the user interface used to initiate BAAS activities that include accessing the COTS, the price table update function, retrieve order information and report generation.

Attributes:

BAASActivityId - This attribute represents which function of the BAAS will be selected by the user interface class. The range of BAAS activities that may be requested includes, but will not be limited to: enter test mode, invoke manager class for authorization, update MsBaPriceTable, invoke MsAcTrackingMgr to retrieve order and request information, invoke the cost accounting function, invoke the COTS package, invoke the custom report function, or terminate selected activity.

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

MsBaManagerUIB - This method represents the constructor for this class.

Arguments:

Return Type:Void

Privilege:Public

initiateBAASactivity - This method represents which function of the BAAS will be selected by the user interface class. The range of BAAS activities that may be requested, identified by BAASActivityId includes, but will not be limited to: enter test mode, invoke manager class for authorization, update MsBaPriceTable, invoke MsAcTrackingMgr to retrieve order and request information, invoke the cost accounting function, invoke

the COTS package, invoke the report function, terminate selected activity
Arguments:RWCString BAASActivityId
Return Type:EcTint
Privilege:Public

~MsBaManagerUIB - This method represents the destructor for this class.

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsBaManagerUIB class has associations with the following classes:
Class: MsBaBAASManagerB initiateM&OActivities

6.3.3.9 MsBaPriceTableB Class

Parent Class:EcPriceTableB

Public:No

Distributed Object:No

Purpose and Description:

This class inherits all the attributes from the public EcPriceTable class but adds methods to update the current prices in the table and to provide the capability to create new table entries via the MsBaBAASManagerB class.

Attributes:

All Attributes inherited from parent class

Operations:

GeneratePriceElemId

This method represents the capability for M&O staff members to generat a unique price element ID to be added to the instance of the standard price table. A price element can be either a data granule , a type of media, shipping method, file size or service type. Arguments:RWCString PriceElemId, RWCString homeDAAC, RWCString PriceElemIdDescrip, EcCurrency PriceElemValue

MsBaPriceTableB

Arguments:

UpdateStdPrices - This method represents the capability for M&O staff members to update the standard price table, which MsBaPriceTableB inherits all the attributes from, mapping each of the EcPriceTableB attributes to an indentifiable price element ID, followed by a description of that price element and further identified by which ECS Data Center (i.e. DAAC) caused the entry to be

placed in the table. The method will either add, change or delete the entry. The BAAS COTS package will use the same information contained in the standard price table when generating bill invoices for ECS accounts.

Arguments:RWCString PriceElemId, RWCString PriceElemIdDescrip, RWCString updateAction, EcCurrency PriceElemValue

Return Type:EcTVold

Privilege:Public

~MsBaPriceTableB

Arguments:

Associations:

The MsBaPriceTableB class has associations with the following classes:

Class: MsBaBAASManagerB maintainTableEntries

6.3.4 Billing and Accounting Dynamic Model

6.3.4.1 Billing and Invoicing a Science User

In this scenario, the science user will be billed and invoiced for requesting ECS data products or services. This scenario traces the events associated with gathering the cost information, generating statements, and posting the charges to the appropriate account. The scenario is depicted in Figure 6.3-3.

6.3.4.1.1 Beginning Assumptions

A valid Science User with an account in good standing has already generated a request. Details of the request, showing the products shipped and the price estimate, will have been captured by the methods described in the MSS Accountability class MsAcTrackingMgr.

6.3.4.1.2 Interfaces with Other Subsystems and Segments

DSS and DPS will provide data product order information which MsAcTrackingMgr will be able to store and retrieve from the ECS Management Database.

6.3.4.1.3 Stimulus

The Science User has submitted a request that initiates a product or service to be generated. Upon completion of the request, details on products shipped traceable to a user will be stored in the ECS Management Database area.

6.3.4.1.4 Participating Classes From the Object Model

MsAcUsrProfile

MsAcUsrProfileMgr

MsAcTrackingMgr

MsBaManagerUIB

MsBaBAASManagerB

EcPriceTableB

MsBaBAASB-COTS

Billing Clerk(Actor)

Science User (Actor)

6.3.4.1.5 Beginning System, Segment and Subsystem State(s)

The system can be in normal operational mode, or it may be in training mode. The mode will be taken into consideration when the cost resource data details on the request are gathered for pricing. Requests performed under training mode will not be priced.

6.3.4.1.6 Ending State

The Science User is sent a statement detailing the charges incurred as a result of the activities performed to fulfill a request.

6.3.4.1.7 Scenario Description

These steps describes the accompanying event trace diagram.

- 1) A Billing Clerk (Actor) requests that the user interface for the BAAS select Update Accounts as an option to execute.
- 2) The user interface class, MsBaManagerUIB, requests the MsBaManagerB controlling class to allow the current M&O user of the BAAS to update ECS accounts with new order information. The MsBaManagerB class will check the user's accessLevel to determine if access to the accounts is permitted by this M&O user (Billing Clerk Actor).
- 3) MsBaManagerB class will determine which order(s) for the Science User identified by userId exist to be processed. These orders will be retrieved by invoking the MsAcTrackingMgr public method GetOrdersByUser(userId).
- 4) The MsBaManagerB object will then request through the public method GetRequestInfo of the MsAcTrackingMgr class, all requests tied to the order that have been shipped.
- 5) The MsAcTrackingMgr will then search for and return all requests tied to the original Science User's order that have shipped.
- 6) MsBaManagerB invokes the EcPriceTableB public method, ProvidePrices given the list of requestIds and a global parameter list (GIParameterList) supplied by the information retrieved by MsAcTrackingMgr to price the total amount of the requests that have been shipped. The GIParameterList will contain the data product(s) ordered, the type and number of media used to fulfill the order, any services used to complete an order (i.e. subsetting), the shipping method and the userId associated with the shipped data product request(s). The price table information retrieved should match the price estimates provided to the science user when the original order was placed.
- 7) MsBaBAASManagerB provides the MsBaBAASB-COTS package with the order/request information for the current Science User and instructs the COTS to update this account.

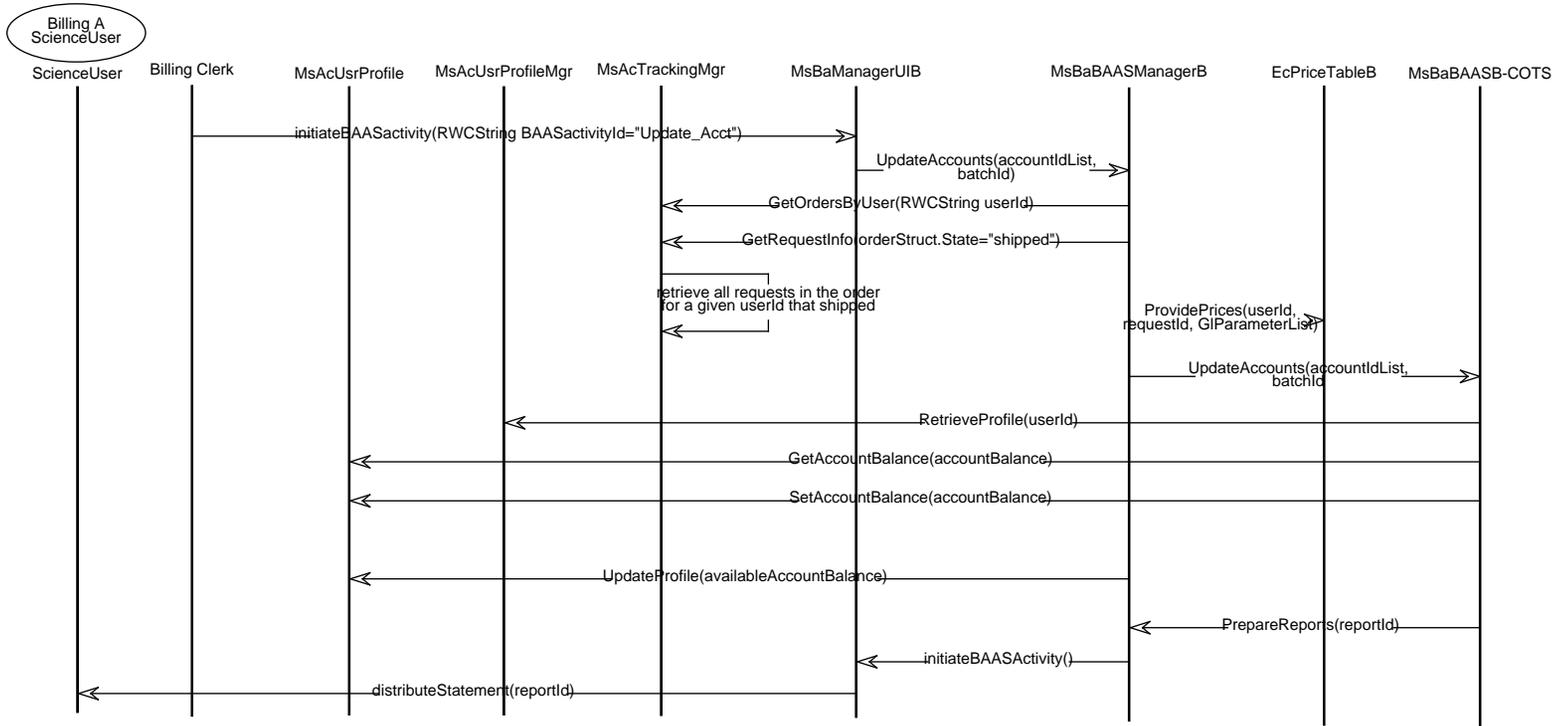


Figure 6.3-3. Billing and Invoicing a Science User Event Trace

- 8) MsBaBAASB-COTS updates the user's appropriate accounts receivable by the amount of the price of the fulfilled request. Normally, a batch of fulfilled requests that could be from multiple ECS user accounts would be processed at a time, but for this scenario only one Science User's account will be shown to be updated and billed.
- 9) As the BAAS COTS updates the Science User's account, the MsBaBAASB-COTS has the ability to retrieve the Science User's profile information, through the public method `RetrieveProfile` from the `MsAcUsrProfileMgr` class. This information will be used to prepare an accurate statement that will be sent to the account holder of record at the end of the billing cycle.
- 10) MsBaBAASB-COTS then retrieves and debit the Science User's available balance to take into account the fulfilled order that was being processed. The methods `GetAccountBalance` and `SetAccountBalance` from `MsAcUsrProfile` can be accessed by the MsBaBAASB-COTS which can interface with the ECS Sybase Management Database directly.
- 11) MsBaBAASManagerB takes the accountBalance computed by the MsBaBAASB-COTS and updates the Science User's information through the method `UpdateProfile` from the `MsAcUsrProfileMgr` class.
- 12) Each individual data request priced by MsBaBAASB-COTS provides input to the Billing and Invoicing function of the COTS, which will generate statements on a monthly basis, invoking the MsBaBAASManagerB class method `PrepareReports`. These statements will be distributed to the accounts, which will reflect all data order activity during the monthly billing cycle. For the purposes of this scenario, a statement will be considered a type of report.
- 13) MsBaBAASManagerB notifies the MsBaManagerUIB class that a report (statement) has been generated which can be mailed to the Science User.
- 14) An M&O staff member (Billing Clerk) accessing the MsBaManagerUIB interface class, distributes the statement to the appropriate Science User's account.
 - 14a) If the user's account is a type of pre-paid account, the statement issued will show the current activity on the account for the past billing cycle.
 - 14b) For non pre-paid accounts, the statement sent will be an actual bill for the services and products provided during this period plus any previous balance not paid. An actual bill will be generated in this case by the MsBaBAASB-COTS.

6.3.4.2 Receiving and Posting Science User Payments to Accounts Scenario

This scenario traces the events associated with receiving payments from a Science User and to process these payments to the proper account. The scenario is depicted in event trace diagram in Figure 6.3-4.

6.3.4.2.1 Beginning Assumptions

Assume the user's account does not have any amounts already credited to it (i.e. is not a pre-paid account) and that the user has received a billing invoice for purchases during the past billing cycle.

Receiving and Posting Science User Pmts

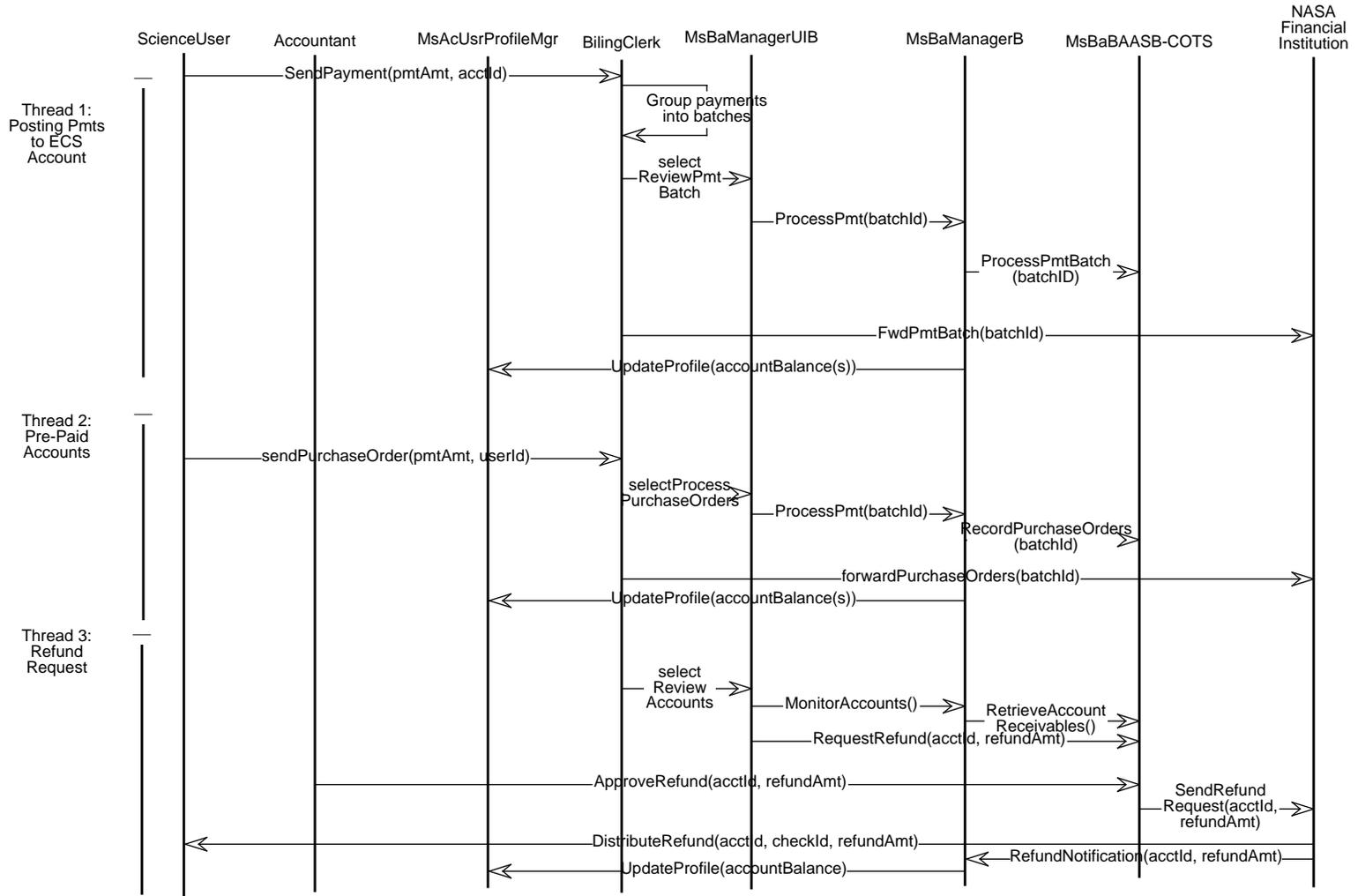


Figure 6.3-4. Receiving and Posting Science User Payments Event Trace

6.3.4.2.2 Interfaces with Other Subsystems and Segments

External interfaces with financial institutions will be required to eventually deposit payments to a U.S. Treasury account. This will be performed by the BAAS COTS functionality that is capable of supporting a variety of lockbox and Electronic Data Interchange (EDI) interfaces to accept payments at the banking institution that has been configured to accept ECS payments and accept electronic transfers.

6.3.4.2.3 Stimulus

A Science User has sent a payment to cover the outstanding balance on their account for ECS products or services ordered during the past billing period.

6.3.4.2.4 Participating Classes From the Object Model

Science User (Actor)

Billing Clerk (Actor)

MsAcUsrProfileMgr

MsBaManagerUIB

MsBaBAASManagerB

MsBaBAASB-COTS

Accountant (Actor)

Financial Institution (Actor)

6.3.4.2.5 Beginning System, Segment and Subsystem State(s)

The system is in a normal operational mode.

6.3.4.2.6 Ending State

The Science User's account is credited the proper amount and the payment is received at the financial institution's lockbox. A lockbox can be either electronic or a physical post office box established by a financial institution for receipt of payments to an agency or organization such as ECS. An alternate thread would involve overpayment by the Science User, triggering a refund check to be eventually sent to the Science User.

6.3.4.2.7 Scenario Description

Thread 1:

The following steps describe the accompanying event trace for receiving and posting Science User Payments to the proper account.

- 1) The Science User mails a check to the SMC; the check is received by a Billing Clerk at the SMC and the clerk groups the checks into a batch of payments to be processed by the BAAS COTS.
- 2) The Billing Clerk will review the batch of checks received, which represents Science User payments received during a given time period, and then prepares an Accounts Receivable

batch report. Upon input of the batch entries, the payments will be credited to the appropriate accounts receivables, represented by the COTS class MsBaBAASB-COTS method ProcessPmtBatch.

- 3) The Billing Clerk then sends the checks to a NASA designated financial institution that will deposit the payments into a Federal Treasury account.
- 4) Once the accounts reflect the payments received in the Accounts Receivable batch report, the MsBaManagerB class invokes the MsAcUsrProfileMgr method, UpdateProfile to adjust the accountBalance(s) affected in the LockBox Report of payments.

Thread 2: Pre-paid accounts, where funds are available in advance of purchases

- 1) The Science User sends a purchase order to the SMC; the purchase order is received by a Billing Clerk, which invokes the COTS package through the MsBaManagerUIB user interface class.
- 2) The SMC Billing Clerk will take purchase orders received during a given time period, and prepare an Accounts Receivable batch report. Upon input of the batch entry, amounts on the purchase orders will be credited to the appropriate accounts using the method RecordPurchaseOrders, contained in the MsBaBAASB-COTS class. As the Science User(s) of the account purchases ECS products, charges will be deducted from the account's existing balance.
- 3) The Billing Clerk will then forward the batch of purchase orders to the financial institution that has been set up to accept funds collected for deposit to a U.S. Treasury account.
- 4) Once the accounts reflect the purchase order amount(s), the MsBaManagerB class invokes the MsAcUsrProfileMgr method, UpdateProfile to adjust the accountBalance(s) affected in the list of purchase order "payments".

Thread 3: Overpayment of a Science User's account, triggering a refund request

- 1) The Science user sends a payment that exceeds the balance due; a Billing Clerk recognizes the overage in the account when the accounts are balanced at the end of a reporting period and invokes the RequestRefund method in the MsBaBAASB-COTS class.
- 2) The Billing Clerk, forwards a refund request for the proper amount to another actor, an M&O staff member, such as an Accountant to approve the request.
- 3) The refund request is approved by the Accountant and the MsBaBAASB-COTS packages generates a refund request that is forwarded to a NASA designated financial institution.
- 4) The NASA designated financial institution receives the refund request and distributes a refund check to the Science User.
- 5) Once the refund check has been issued, the MsBaManagerB class invokes the MsAcUsrProfileMgr method, UpdateProfile to adjust the accountBalance(s) by the amount of the refund check issued.

6.3.5 Billing and Accounting Structure

Table 6.3-2 lists the components of the Billing and Accounting Application Service.

Table 6.3-2. Billing and Accounting Components

Component Name	COTS/Custom
Billing and Accounting Service	COTS/Custom
BAAS Manager	Custom
Price Table	Custom (C++ code and scripts)

6.3.5.1 Billing and Accounting Service CSC

Purpose and Description

The Billing and Accounting Service CSC provides the COTS package and the Operator Interface which allows operators at both the SMC and the DAACs to review account statuses, input billing information, perform account queries, balance accounts, and generate statements and reports by the COTS package.

Mapping to objects implemented by this component

MsBaBAASB - COTS Billing and Accounting Package

MsBaManagerUIB

initiateBAASActivity - C++ code

6.3.5.2 Billing and Accounting Application Service (BAAS) Manager CSC

Purpose and Description

This class category provides services such as the startup and shutdown of the Billing and Accounting Application Service and other BAAS activities that related to the running of the COTS package. Such activities include both the manual and automatic processes that will update accounts, issue requests for refunds, and forward payments from ECS users to a NASA designated financial institution for deposit to a U.S. Treasury account. Other operations are inherited from the external class, EcAgCOTSManager.

Mapping to objects implemented by this component

EcAgCOTSManager

MsBaBAASManagerB

MaintainPricingTables - Scripts and C++ code

UpdateAccounts - Scripts and C++ code

MonitorAccounts - Scripts and C++ code

PrepareReports - Scripts and C++ code

ProcessPmt - Scripts and C++ code

6.3.5.3 Pricing Table CSC

Purpose and Description

This CSC provides a distributed object class to provide pricing information to external subsystems. It also provides the means to maintain the table that will be maintained in the ECS Management SYBASE database.

Mapping to objects implemented by this component

EcPriceTableB

ProvidePrices - C++

MsBaPriceTable

pdateStdPrices - C++

GeneratePriceElement - C++

6.3.6 Billing and Accounting Management and Operation

6.3.6.1 System Management Strategy

The Billing and Accounting Application Service is based on COTS product to be determined.

6.3.6.2 Operator Interfaces

The Operator Interface to Billing and Accounting is the graphical user interface provided by the COTS product to be selected. The human interface will be predominately through a Microsoft Windows client interface, with the capability to access a Unix server interface as well.

6.3.6.3 Reports

The following predefined Billing and Accounting reports will be provided by the BAAS COTS:
Chart of Accounts - list of all ledger accounts in the ECS. This report can be used to locate a specific account within a ledger account.

Invoices - shows invoices received by M & O Staff for a given period

Purchase Orders - will show purchase orders initiated by M & O Staff.

Summary Reports - summarizes billing and accounting activity for a given reporting period (day, month, fiscal year, year-to-date)

Inventory Receiving Reports - to track consumable items ordered from vendors that can be used during the invoice reconciliation process (dormant until the third party contract to supply consumables ends).

Other accounting statistics will be reportable via ad hoc reports provided by COTS product and the report generation capability associated with the custom portion of the Billing and Accounting Application Service and the MSS accountability CSCI.

Further information on billing and accounting reports is available in the Release B Overview Design Specification (305-CD-020-002).

6.4 Report Generation Service

6.4.1 Report Generation Service Overview

The Report Generation Service provides M&O staff with access to management information across all areas of the ECS enterprise and DAAC operations. The service is implemented through the collective reporting capabilities offered by the MSS Report Writer associated with the management database, the report generation capabilities of specialized MSS management applications, and the management reporting capabilities provided by other ECS application subsystems. The service provides for the generation of both routine and adhoc reports and queries. Adhoc reports and queries are supported through the COTS reporting/query tools associated with the management database and through management application-specific COTS tools. An HTML-based user interface supports convenient browse access to routinely generated reports by non-specialists.

The Report Generation Service is for the exclusive use of ECS/DAAC management and M&O staff responsible for monitoring system performance, workload, capacity utilization, security, reliability, accountability, and user satisfaction. Access to all reporting tools is restricted to registered M&O personnel. The Report Generation Service provides for the generation of a range of standard management reports. A standard report, also referred to as a canned report, is one for which a template specifying format and content has been previously defined and saved. These standard reports are maintained by M&O database specialists. Standard reports can be run automatically on a periodic basis (e.g., daily, monthly, quarterly) based on setup parameters associated with the report. ECS management and M&O personnel can access these reports for viewing from their desktop through the HTML-based user interface. Optionally, they can apply time and domain scope to the standard reports to generate adhoc reports. Data underlying reports can be saved in text format for import into an analysis tool such as a spreadsheet.

In support of the data specialist on the M&O staff, the Report Generation Service provides a workbench for use in constructing adhoc reports/queries and for maintaining the complement of standard management report templates. This workbench consists of a report writer COTS package associated with the management RDBMS and a statistical analysis COTS package providing tools for analyzing performance trends.

Report Generation Services are accessible only by M&O personnel. In general, the default scope of reports at the SMC include all of the ECS enterprise whereas the scope at an LSM is the local management domain.

Management reports fall in the following major categories:

Performance/Compliance - these reports are designed to reveal short and long term trends in system operation relative to relevant benchmark requirement or performance goal. An example is a report trending production operations adherence to schedule comparing planned versus actual product generation completion times. Another example would be the average turnaround time to resolve trouble tickets. A third example is the reliability of system components using outage times obtained from the system fault management application.

Workload - these reports are designed to reveal short and long term trends in the system workload such as the number of products generated and the number of problem reports submitted to the User Services group.

Resource Utilization - these reports are designed to reveal short and long term trends in the utilization of resources for correlation with workload. These reports include resources such as CPU/disk utilization in hosts, network utilization, as well as personnel resources such as User Services. These reports present utilization at the host-site level and across sites.

User Satisfaction - these reports depict short and long term trends in the degree of satisfaction with ECS products and services from the external user standpoint.

Profiles/Characterizations - these reports provide finer insight into diverse areas such as product generation workload, fault occurrences, user service activities, and the like by showing the distribution of basic measured elements in these areas according to subtypes. An example is the volume distribution of product types generated over a specified interval.

Accountability - these reports provide an audit/trace of significant events associated with users and their access of system resources and services, security, data integrity, fault management, billing and accounting, and configuration management.

Section 6.4.2 itemizes specific reporting areas.

6.4.2 Report Generation Context

Figure 6.4.1 is the context diagram for the Report Generation Service.

The Report Generation service interfaces with the user (ECS management and M&O staff personnel) for report generation requests, with system management agent services for lifecycle commands, status reporting and event logging, and with the User Profile support within MSS accountability to validate a requester is a member of the M&O staff.

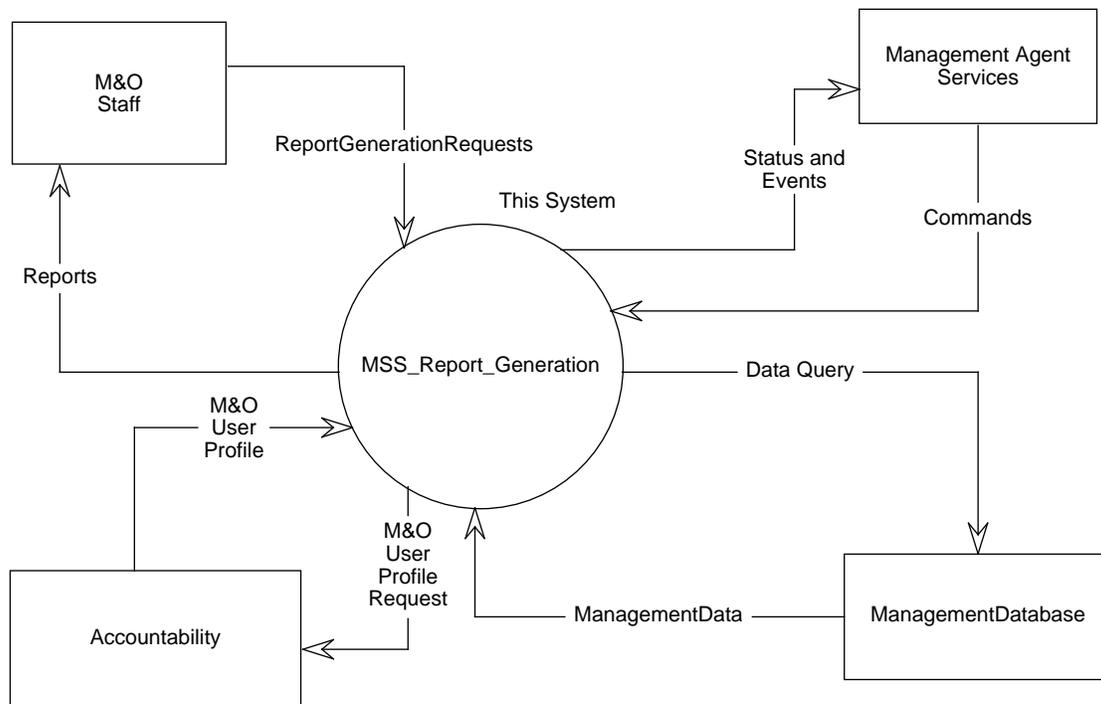


Figure 6.4-1. Report Generation Context Diagram

6.4.3 Report Generation Object Model

The Report Generation object model is shown in Figure 6.4-2.

6.4.3.1 CGI_Vars Class

Parent Class:RWHashDictionary

Attributes:

All Attributes inherited from parent class

Operations:

CGI_Vars

Arguments:

Return Type:Void

Privilege:Public

LoadEnvironmentVariables

Arguments:

Return Type:Void

Privilege:Private

LoadGetElements

Arguments:

Return Type:Void

Privilege:Private

LoadPostElements

Arguments:

Return Type:Void

Privilege:Private

get

Arguments:char * szName

Return Type:CGI_Element *

Privilege:Public

get

Arguments:RWCString &rsName

Return Type:CGI_Element *

Privilege:Public

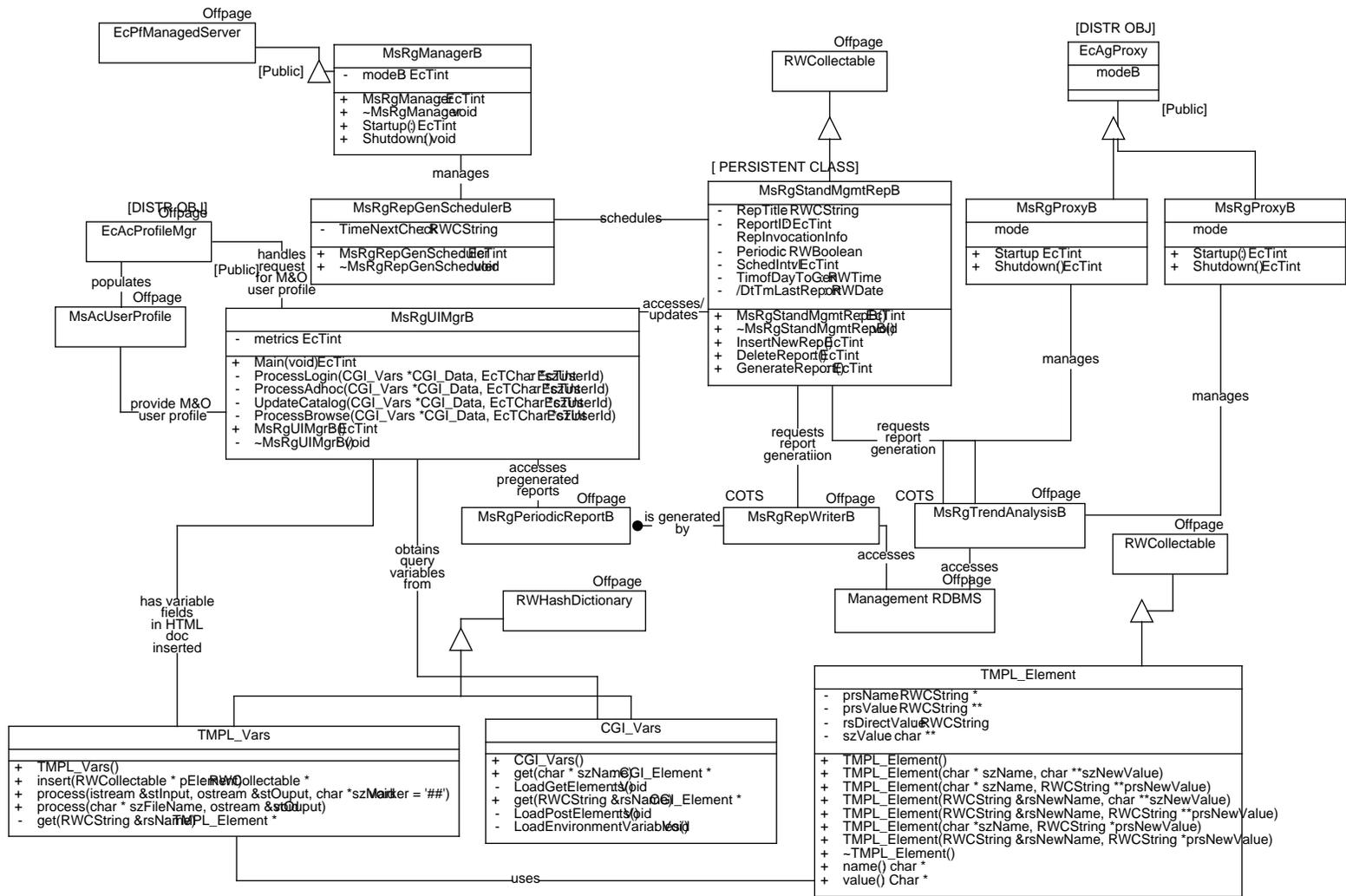


Figure 6.4-2. Report Generation Object Model

Associations:

The CGI_Vars class has associations with the following classes:

Class: MsRgUIMgrB obtainsqueryvariablesfrom

6.4.3.2 EcAcProfileMgr 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 EcAcProfileMgr class has associations with the following classes:

Class: MsRgUIMgrB handlesrequestforM&Ouserprofile

Class: MsAcUserProfile populates

6.4.3.3 EcAgProxy Class

Parent Class:Not Applicable

Public:Yes

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 a security event occurs, this object has to detect the incident and send out an event notification to the MsAgSubagent.

Attributes:

modeB - This attribute contains the mode in which the application is executing under. It identifies functional activity(operational, testing, training).

Operations:

None

Associations:

The EcAgProxy class has associations with the following classes:

None

6.4.3.4 EcPfManagedServer Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

This is the container class that starts up the event Manager, table Manager, monitor, port monitor, discoverer, subagent configuration, static buffer, and the deputy gate. This class also starts a thread that triggers scheduled events (i.e. polling ECS application's performance metrics).

Attributes:

None

Operations:

None

Associations:

The EcPfManagedServer class has associations with the following classes:

None

6.4.3.5 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: MsRgRepWriterB accesses

Class: MsRgTrendAnalysisB accesses

6.4.3.6 MsAcUserProfile Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The MsAcUserProfile class has associations with the following classes:

Class: EcAcProfileMgr populates

Class: MsRgUIMgrB provideM&Ouserprofile

6.4.3.7 MsRgManagerB Class

Parent Class:EcPfManagedServer

Public:No

Distributed Object:No

Purpose and Description:

MsRgManager provides the interface between the Managed Process Framework and the report generation scheduler. It allow the report generator to be controlled and monitored by from the system management position.

Attributes:

modeB

Data Type:EcTint

Privilege:Private

Default Value:

Operations:

MsRgManager - Class constructor.

Arguments:

Return Type:EcTint

Privilege:Public

Shutdown - Shutdown the report scheduler.

Arguments:

Return Type:void

Privilege:Public

Startup - Startup the report scheduler.

Arguments:

Return Type:EcTint

Privilege:Public

~MsRgManager - Class destructor.

Arguments:

Return Type:void

Privilege:Public

Associations:

The MsRgManagerB class has associations with the following classes:

Class: MsRgRepGenSchedulerB manages

6.4.3.8 MsRgPeriodicReportB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class contains browsable periodic (daily,weekly,etc) report output automatically generated per the standard report schedule.

Attributes:

None

Operations:

None

Associations:

The MsRgPeriodicReportB class has associations with the following classes:

Class: MsRgUIMgrB accessespregeneratedreports

Class: MsRgRepWriterB isgeneratedby

6.4.3.9 MsRgProxyB Class

Parent Class:EcAgProxy

Public:No

Distributed Object:No

Purpose and Description:

This class provides the interface between the ECS Management Agent Services and the COTS report writer (MsRgReportWriter object) allowing ECS to issue lifecycle commands to the COTS and receive processing events and status.

Attributes:

mode

Operations:

Shutdown - This method shuts down the report writer.

Arguments:

Return Type:EcTint

Privilege:Public

Startup - This method starts up the report writer.

Arguments:

Return Type:EcTint

Privilege:Public

Associations:

The MsRgProxyB class has associations with the following classes:

Class: MsRgTrendAnalysisB manages

6.4.3.10 MsRgRepGenSchedulerB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

MsRgRepGenScheduler initiates the generation of routine periodic standard reports based on scheduling information and generation methods provided in the MsRgStandMgmtRep catalog object.

Attributes:

TimeNextCheck - Time interval in hours to wait before checking for generation of a periodic report.

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

MsRgRepGenScheduler - Class constructor.

Arguments:

Return Type:EcTint

Privilege:Public

~MsRgRepGenScheduler - Class destructor.

Arguments:

Return Type:void

Privilege:Public

Associations:

The MsRgRepGenSchedulerB class has associations with the following classes:

Class: MsRgManagerB manages

Class: MsRgStandMgmtRepB schedules

6.4.3.11 MsRgRepWriterB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class is the COTS report writer product associated with the management DBMS.

Attributes:

None

Operations:

None

Associations:

The MsRgRepWriterB class has associations with the following classes:

Class: ManagementRDBMS accesses

Class: MsRgPeriodicReportB isgeneratedby

Class: MsRgStandMgmtRepB requestsreportgeneratiion

6.4.3.12 MsRgStandMgmtRepB Class

Parent Class:RWCollectable

Public:No

Distributed Object:No

Persistent Class:True

Purpose and Description:

This class represents a registered ECS standard report. A standard report is one for which a report template defining the format and data type content has been prebuilt and saved using a management reporting tool. A subset of standard ECS reports are generated according to a defined schedule which is characterized in this class (e.g., daily/weekly/etc). Other reports are generated strictly on an ad-hoc basis per user request.

Attributes:

/DtTmLastReport - This is the data/time at which the last generation of this report was performed.

Data Type:RWDate

Privilege:Private

Default Value:

Periodic - This flag is set to indicate a report which is to be produced on a periodic basis.

Data Type:RWBoolean

Privilege:Private

Default Value:

RepInvocationInfo - This attribute provides information on invoking the associated COTS report writer to generate the report.

RepTitle - Report Title attribute.

Data Type:RWCString

Privilege:Private

Default Value:

ReportID - A unique report identification code for this report.

Data Type:EcTint

Privilege:Private
Default Value:

SchedIntvl - The interval in days between generations of this report.

Data Type:EcTint
Privilege:Private
Default Value:

TimofDayToGen - Time of day to initiate generation of a periodic report.

Data Type:RWTime
Privilege:Private
Default Value:

Operations:

DeleteReport - This method deletes an existing report entry in the standard management report catalog.

Arguments:
Return Type:EcTint
Privilege:Public

GenerateReport - This method generates the report invoking the associated management reporting tool.

Arguments:
Return Type:EcTint
Privilege:Public

InsertNewRep - This method inserts a new report in the catalog of standard management reports.

Arguments:
Return Type:EcTint
Privilege:Public

MsRgStandMgmtRepB - Constructor for this class.

Arguments:
Return Type:EcTint
Privilege:Public

~MsRgStandMgmtRepB - Destructor for the class.

Arguments:
Return Type:void
Privilege:Public

Associations:

The MsRgStandMgmtRepB class has associations with the following classes:

Class: MsRgUIMgrB accesses/updates

Class: MsRgRepWriterB requestsreportgeneratiion

Class: MsRgTrendAnalysisB requestsreportgeneration

Class: MsRgRepGenSchedulerB schedules

6.4.3.13 MsRgTrendAnalysisB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

The MsPmTrendAnalysis class generates trend data for a specified parameter over a specified time period.

Attributes:

None

Operations:

None

Associations:

The MsRgTrendAnalysisB class has associations with the following classes:

Class: ManagementRDBMS accesses

Class: MsRgProxyB manages

Class: MsRgStandMgmtRepB requestsreportgeneration

6.4.3.14 MsRgUIMgrB Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This is the gateway interface between the M&O HTML client and the Report Generation service.

Attributes:

metrics

Data Type:EcTint

Privilege:Private

Default Value:

Operations:

Main

Arguments: void
Return Type: EcTint
Privilege: Public

MsRgUIMgrB - Class constructor.

Arguments:
Return Type: EcTint
Privilege: Public

ProcessAdhoc - This method processes an M&O user's request for an adhoc management report.

Arguments: CGI_Vars *CGI_Data, EcTChar *szUserId
Return Type: EcTint
Privilege: Private

ProcessBrowse

Arguments: CGI_Vars *CGI_Data, EcTChar *szUserId
Return Type: EcTint
Privilege: Private

ProcessLogin

Arguments: CGI_Vars *CGI_Data, EcTChar *szUserId
Return Type: EcTint
Privilege: Private

UpdateCatalog -

Arguments: CGI_Vars *CGI_Data, EcTChar *szUserId
Return Type: EcTint
Privilege: Private

~MsRgUIMgrB - Class destructor.

Arguments:
Return Type: void
Privilege: Private

Associations:

The MsRgUIMgrB class has associations with the following classes:

Class: MsRgStandMgmtRepB accesses/updates

Class: MsRgPeriodicReportB accesses pregenerated reports

Class: EcAcProfileMgr handles request for M&O user profile

Class: TMPL_Vars has variable fields in HTML doc inserted

Class: CGI_Vars obtainsqueryvariablesfrom
Class: MsAcUserProfile provideM&Ouserprofile

6.4.3.15 RWCollectable Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWCollectable class has associations with the following classes:
None

6.4.3.16 RWHashDictionary Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWHashDictionary class has associations with the following classes:
None

6.4.3.17 TMPL_Element Class

Parent Class:RWCollectable

Attributes:

prsName

Data Type:RWCString *

Privilege:Private

Default Value:

prsValue

Data Type:RWCString **

Privilege:Private

Default Value:

rsDirectValue

Data Type:RWCString

Privilege:Private

Default Value:

szValue

Data Type:char **

Privilege:Private

Default Value:

Operations:**TMPL_Element**

Arguments:char * szName, char **szNewValue

Return Type:Void

Privilege:Public

TMPL_Element

Arguments:char * szName, RWCString **prsNewValue

Return Type:Void

Privilege:Public

TMPL_Element

Arguments:RWCString &rsNewName, char **szNewValue

Return Type:Void

Privilege:Public

TMPL_Element

Arguments:RWCString &rsNewName, RWCString **prsNewValue

Return Type:Void

Privilege:Public

TMPL_Element

Arguments:char *szName, RWCString *prsNewValue

Return Type:Void
Privilege:Public

TMPL_Element

Arguments:RWCString &rsNewName, RWCString *prsNewValue
Return Type:Void
Privilege:Public

TMPL_Element

Arguments:
Return Type:Void
Privilege:Public

name

Arguments:
Return Type:char *
Privilege:Public

value

Arguments:
Return Type:Char *
Privilege:Public

~TMPL_Element

Arguments:
Return Type:Void
Privilege:Public

Associations:

The TMPL_Element class has associations with the following classes:

Class: TMPL_Vars uses

6.4.3.18 TMPL_Vars Class

Parent Class:RWHashDictionary

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:

All Attributes inherited from parent class

Operations:

TMPL_Vars

Arguments:

Return Type:Void

Privilege:Public

get

Arguments:RWCString &rsName

Return Type:TMPL_Element *

Privilege:Private

insert

Arguments:RWCollectable * pElement

Return Type:RWCollectable *

Privilege:Public

process

Arguments:istream &stInput, ostream &stOuput, char *szMarker = '###'

Return Type:void

Privilege:Public

process

Arguments:char * szFileName, ostream &stOuput

Return Type:void

Privilege:Public

Associations:

The TMPL_Vars class has associations with the following classes:

Class: MsRgUIMgrB hasvariablefieldsinHTMLdocinserted

Class: TMPL_Element uses

6.4.4 Report Generation Dynamic Model

6.4.4.1 Request to Browse a Report

This scenario traces the processing events associated with a user request through the Report Generation HTML interface to browse a standard management report. Figure 6.4-3 contains the scenario event trace diagram.

Scenario: M&O user selects pregenerated standard report for browsing from HTML UI.

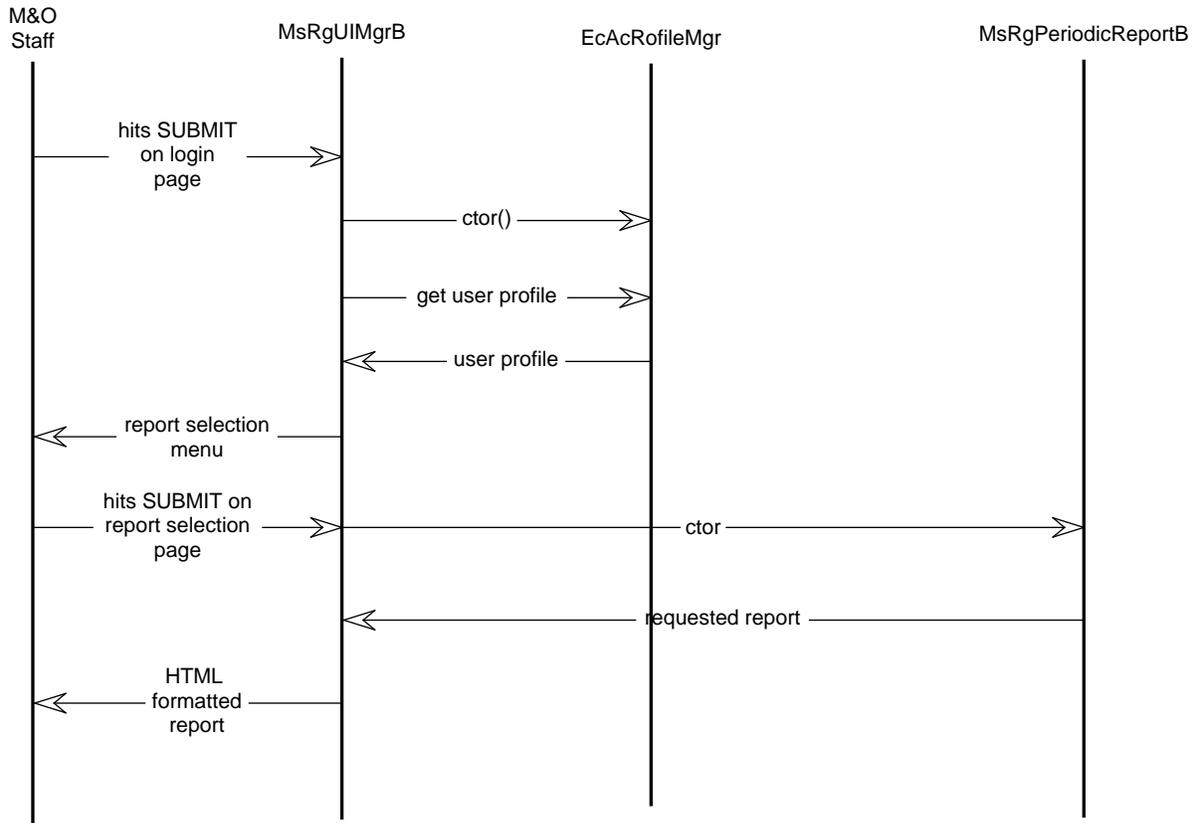


Figure 6.4-3. Request to Browse a Management Report

6.4.4.1.1 Beginning Assumptions

None

6.4.4.1.2 Interfaces with Other Subsystems and Segments

None.

6.4.4.1.3 Stimulus

User logs in and selects a management report for browsing.

6.4.4.1.4 Participating Classes From the Object Model

MsRgUIMgrB

MsRgPeriodicReportB

6.4.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal operational state.

6.4.4.1.6 Ending State

The requested report is displayed at the user's terminal through an HTML browser.

6.4.4.1.7 Scenario Description

1. A member of the M&O staff at the SMC or LSM brings up the Management Reports home page from a desktop HTML browser, enters login information and hits the SUBMIT button.
2. The reporting gateway object, MsRgUIMgrB, receives the login request, validates the user is a registered member of the M&O staff through the EcAcProfileMgr class, and returns the Report Selection page.
3. The M&O user selects a particular report from the report selection list and hits the SUBMIT button.
4. The MsRgUIMgrB gateway class receives the selection request, obtains the indicated pre generated report from the MsRgPeriodicReportB container class, and returns it to the user.

6.4.4.2 Request to Generate an Adhoc Report

This scenario traces the events associated with a user request through the Report Generation HTML interface to generate an adhoc report using a standard management report template but specifying a particular time range/domain scope. Figure 6.4-4 contains the scenario event trace diagram.

6.4.4.2.1 Beginning Assumptions

None.

6.4.4.2.2 Interfaces with Other Subsystems and Segments

None.

6.4.4.2.3 Stimulus

User selects a management report for browsing.

6.4.4.2.4 Participating Classes From the Object Model

MsRgUIB

MsRgStandardMgmtRepB

MsRgRepWriterB

6.4.4.2.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal operational state.

Scenario: M&O user requests adhoc report

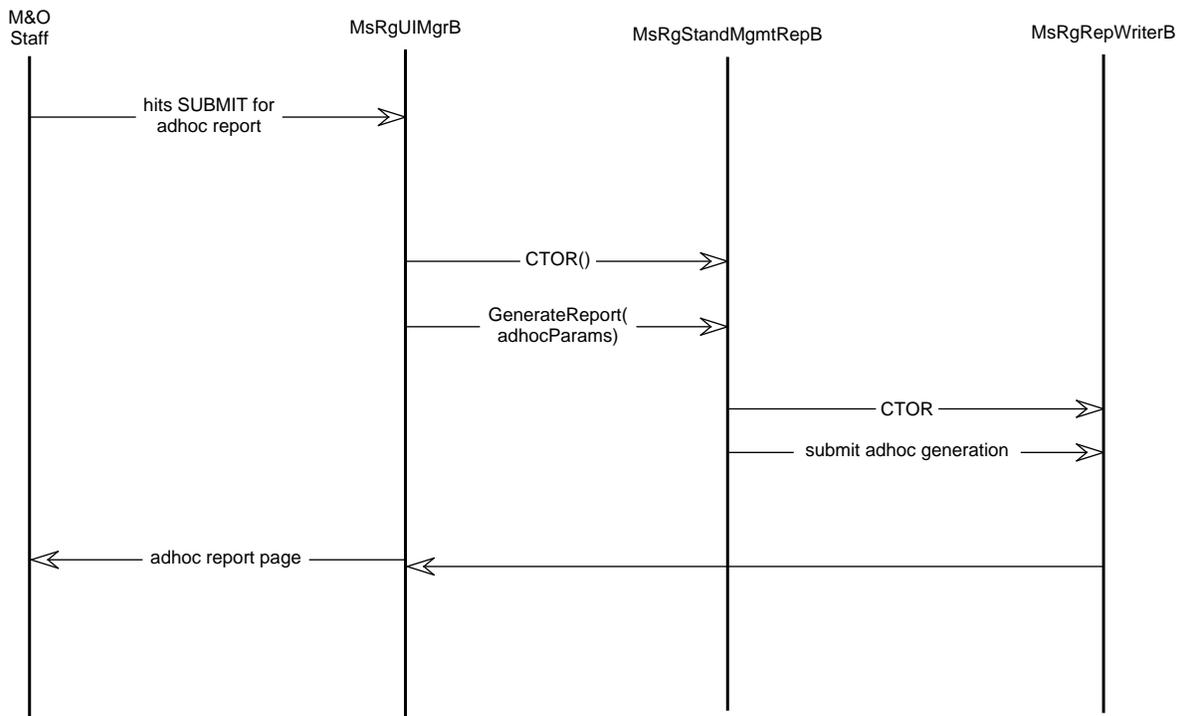


Figure 6.4-4. Request to Generate an Adhoc Report

6.4.4.2.6 Ending State

The requested report is displayed at the user's terminal through the web browser.

6.4.4.2.7 Scenario Description

1. A member of the M&O staff who has already logged in through the Management Reports page, selects a standard report from the Report Selection page, enters a start/end time, and hits the SUBMIT button.
2. The MsRgUIMgrB class receives the report request with the specified report ID and adhoc time scope and, through the MsRgStandMgmtRepB class, submits a request to the MsRgRepWriterB class to generate the report but with the user specified time scope applied.
4. The MsRgRepWriterB COTS report writer generates the report.
5. MsRgUIMgrB receives indication of report completion from the MsRgStandMgmtRepB generate report method and formats the report for return to the requesting user.

6.4.4.3 Request to Add a New Report to the Service

This scenario traces the events associated with an M&O database specialist creating a new report and adding it to the complement of standard periodically generated reports which can be browsed through the web interface. Figure 6.4-5 contains the scenario event trace diagram.

Scenario: M&O data specialist creates a new standard report.

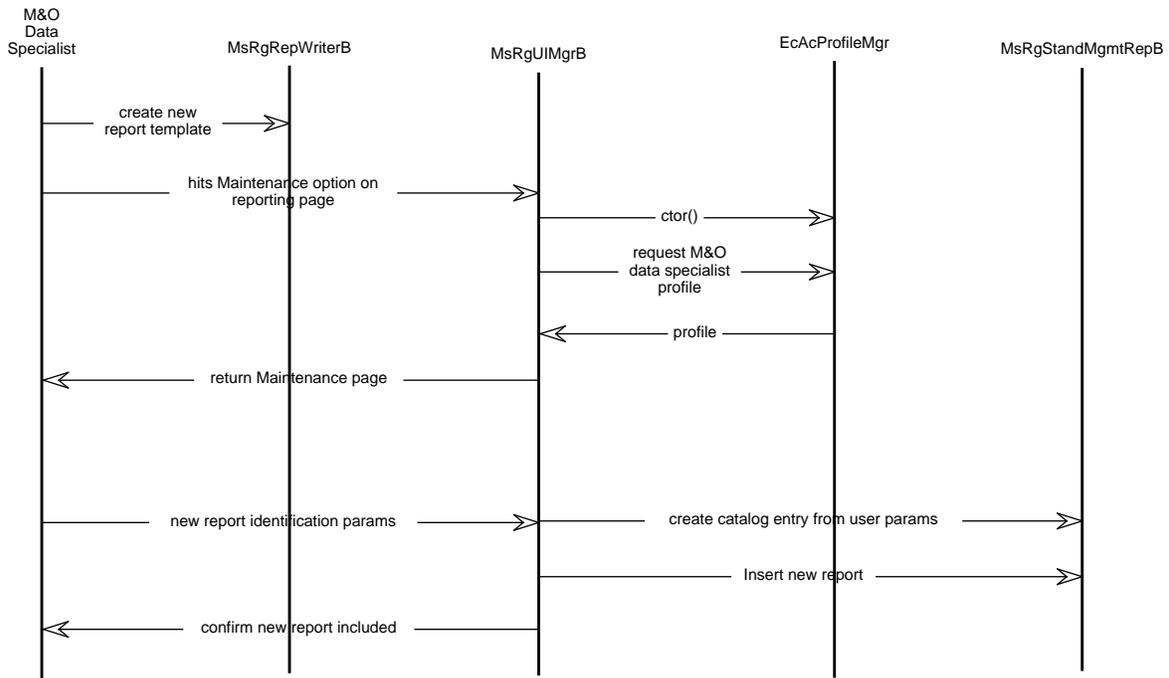


Figure 6.4-5. Request to Add a New Report to the Service

6.4.4.3.1 Beginning Assumptions

None.

6.4.4.3.2 Interfaces with Other Subsystems and Segments

None.

6.4.4.3.3 Stimulus

User selects a management report template and enters a particular time/domain scope.

6.4.4.3.4 Participating Classes From the Object Model

MsRgWebUIB

MsRgUIMgrB

MsRgPeriodicReportB

MsRgRepWriterB

MsRgStandMgmtRepB

6.4.4.3.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal operational state. The M&O data specialist has already logged in to the Report Writer COTS tool and the Management Reports home page.

6.4.4.3.6 Ending State

The new report title is added to the complement of standard reports. It can be viewed through the web browser interface.

6.4.4.3.7 Scenario Description

1. The M&O database specialist creates a new report using the MsRgRepWriterB COTS class and saves it as a report template.
2. The M&O database specialist then selects the Maintenance option on the Management Reports main menu.
3. The MsRgUIMgrB reporting gateway class authenticates that the user is an M&O data specialist with privilege to update the standard report catalog and returns the maintenance page.
4. The specialist enters the title of the newly created report, the resource name of the report template, and optional scheduling parameters if the report is to be automatically generated on a periodic basis. The specialist then hits the SUBMIT button on the page.
5. The MsRgUIMgrB gateway builds a new RgStandMgmtRepB entry from the user's parameters and inserts it in the catalog of standard reports. A confirmation is then sent back to the M&O data specialist that the report has been added.

6.4.5 Report Generation Service Structure

Table 6.4-1 lists the components of the Report Generation Service.

Table 6.4-1. Report Generation Service Components

Component Name	COTS/Custom
Report Generation Manager	Custom (C++ code)
Report Generator	COTS
Report Generation Proxy Agent	Custom (C++ code)

6.4.5.1 Report Generation Manager CSC

Purpose and Description

This CSC provides the custom software support for interfacing with the ECS managed process framework, for supporting M&O user interactions through an HTML interface, for maintaining a catalog of standard management reports, and for automatically generating selected reports on a periodic basis.

Mapping to objects implemented by this component

MsRgManagerB

MsRgUIMgrB

MsRgRepGenSchedulerB

MsRgStandMgmtRepB

6.4.5.2 Report Generator CSC

Purpose and Description

This COTS component is the report writer accompaniment to the Management DBMS. It supports the generation of reports and report templates as well as adhoc queries.

Mapping to object implemented by this component

MsRgRepWriterB

6.4.5.3 Report Generator Proxy Agent

Purpose and Description

This proxy (exported by the Management Agent Services) facilitates the management (startup and shutdown) of Report Generation, and the logging of events.

Mapping to objects implemented by this component

MsRgProxyB

6.4.6 Report Generation Management and Operation

6.4.6.1 System Management Strategy

The Report Generation Service utilizes the public class exported by the Management Agent Services, represented by MsRgProxyB for management of the Report Writer COTS package and the EcPfManagedServer, part of the ECS process framework for management of the custom report generation software. These classes facilitate the management of the overall service.

6.4.6.2 Operator Interfaces

The Report Generation Service provides an HTML based web interface allowing users to browse automatically generated periodic reports and to generate limited adhoc reports. This interface also allows an M&O Database specialist to add/delete/update the catalog of standard management reports.

The Report Generation Service provides the COTS unique user interface provided by the report writers to generate adhoc queries against the management database and to create/modify standard report templates.

6.4.6.3 Reports

Table 6.4-2 lists the management reports provided by the Release B Report Generation Service.

Table 6.4-2 Release B Management Reports (1 of 2)

Report Title
Enhancement Proposal Status Report
Routine Data Production Performance Detail Report
Routine Data Production Performance Summary Report
User-Requested Data Production Performance Detail Report
User-Requested Data Production Performance Summary Report
Ground Operations Activity Performance Detail Report
Ground Operations Activity Performance Summary Report
Product Generation Status Detail Report
Product Generation Status Summary Report
Resource Performance Report
CPU Load Report
Interface Traffic Report
Ethernet Traffic Report
SNMP Traffic Report
SNMP Operations Report
Site Host Resource Utilization Report
SMC Host Resource Utilization Report
Disk Space Report
User Service Performance Report
Data Distribution Performance Report
Media Distribution Profile Report
Data Orders Tracking Summary Report
Data Products Tracking Summary Report
Returned Product Summary Report
Fault Management Report
Trouble Status Report
Ethernet Errors Report
SNMP Errors Report
SNMP Errors Report
SNMP Authentication Failures Report
SNMP Event Log Report
Site Host Errors Report
EMC Host Errors Report
Ground Resource Availability Audit Report
Data Accountability Audit Report
Pending Service Request Audit Report

Table 6.4-2 Release B Management Reports (2 of 2)

Report Title
User Activity Audit Report
Security Audit Report
User Characterization Report
System Access Profile Report
Utilization of User Services Personnel Summary Report
Storage Management Activity Report
Storage Management Inventory Update Report
Ingest History Report
Ingest Error Report
Processing Log Report
Production and Data Processing Request Status Report
Planning Workload Processing Turn-around Report
Planning Management Report
Account Authorization Report
Service Cost Schedule Report
Standard Product Cost Schedule Report
Accounts Payable Report
Accounts Receivable Report
Functional Allocation Report
Configuration Status Report
System Information Report
SNMP Event Notification Report
Indented Level of Assembly List Report
Document Configuration Status Report
System Configuration Tracking Report
Maintenance Schedule Report
Training Program Report
Inventory Status Report
Security Compromise Report
Security Compromise Statistics Report
Virus Detection Report

6.5 Fault Management

6.5.1 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 correlation of recorded symptom using HP Openview and Tivoli and where necessary, through the use of diagnostic tests. 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 within the ECS Process Framework and the Management Agent Service. Management Agent Services is described in detail in the section four. The Overview Design Specification (305-CD-020-001) provides the context and the criteria for ECS applications to use the Process Framework to report their faults. Table 6.5-1 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 and the Process Framework.

**Table 6.5-1. Examples of Faults and Events Reported by ECS Managed Objects
(1 of 2)**

Managed Object	Fault/Event
Standard SNMP Traps	Link Down
	Link Up
	Authentication Failure
Network Device	Node Added
	Node Down
	Node Unknown
	Node Up
	Node Deleted
	Node Marginal
Interface Card	Interface Added
	Interface Disconnected
	Interface Marginal
	Interface Deleted
	Interface Down
	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 warming
	Printer idle

**Table 6.5-1. Examples of Faults and Events Reported by ECS Managed Objects
(2 of 2)**

Managed Object	Fault/Event
Tape drive	Tape drive on-line
	Tape drive off-line
ECS Application	Application missing
	Application failed
	Application startup
	Application discovered
	Application shutdown
	Cold Startup
	Warm Startup
	ECS Database
	Database down
	Database update error
	Database access error
	Database lock table full
	Database media failure

6.5.1.1 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, Figure 6.5-2, HP OpenView Network Node Manager is represented by the object labeled ManagementFramework. Sentry and The Enterprise Console from Tivoli have been selected to perform rules based fault correlation. The Enterprise Console receives events from HP Openview and Sentry agents and performs event correlation using a configurable set of rules. The HP OpenView Network Node Manager provides capabilities and features to allow customization for fault and for the configuration management of the network. This customization, represented as MsFIConfig 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.

6.5.2 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 time of resources due to the fault, results of fault analysis, fault resolution information, and summary fault and performance information. Figure 6.5-1 contains the Fault Management context diagram.

6.5.3 Fault Management Object Model

Figure 6.5-2, the Object Model for Fault Management depicts the major classes and their associations with one another. These are described below:

6.5.3.1 EcDAAC Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

This public class provides methods to respond to requests from other subsystems for the status of resources (processors and their associated disks).

Attributes:

None

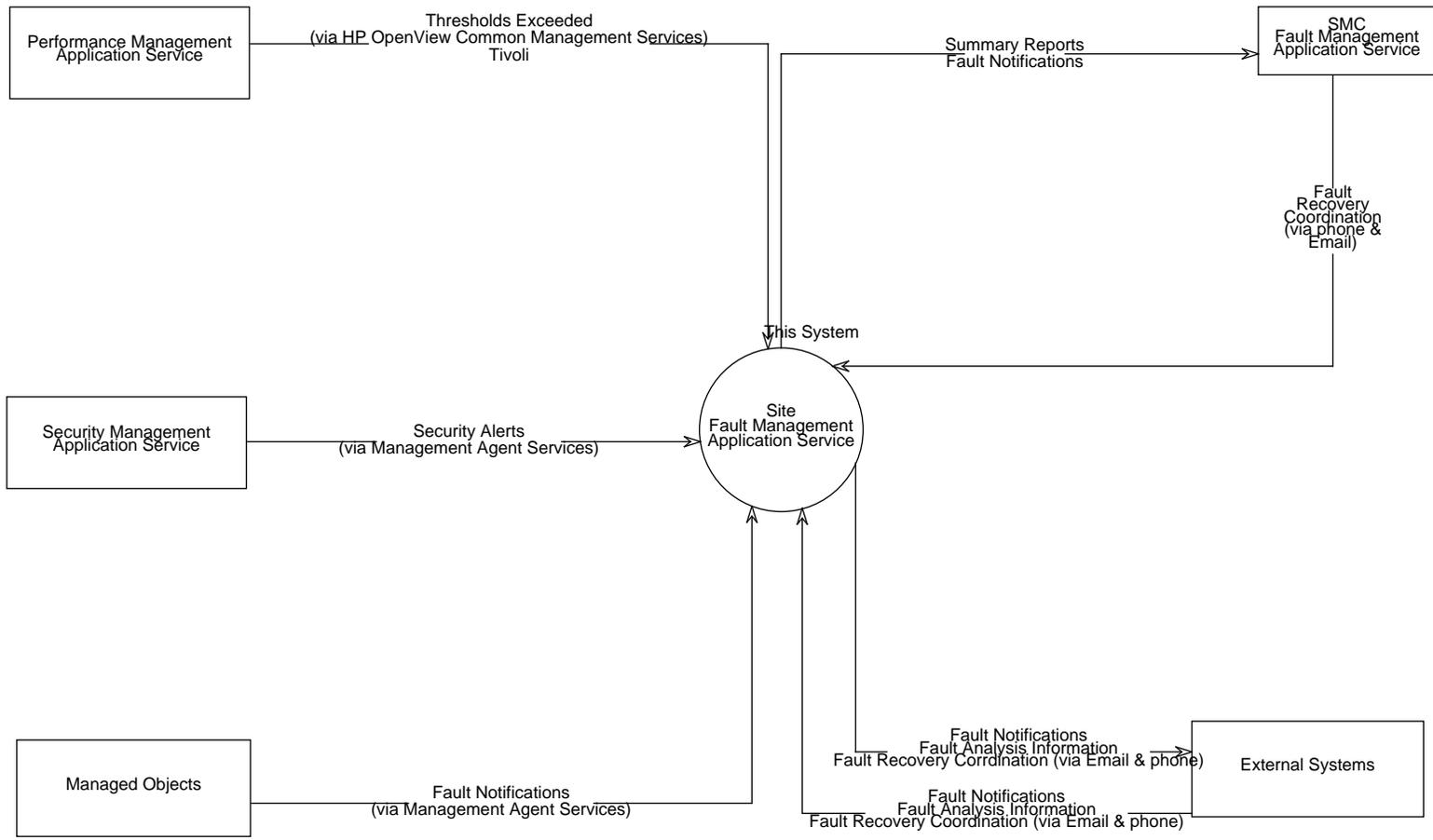


Figure 6.5-1. Fault Management Context Diagram

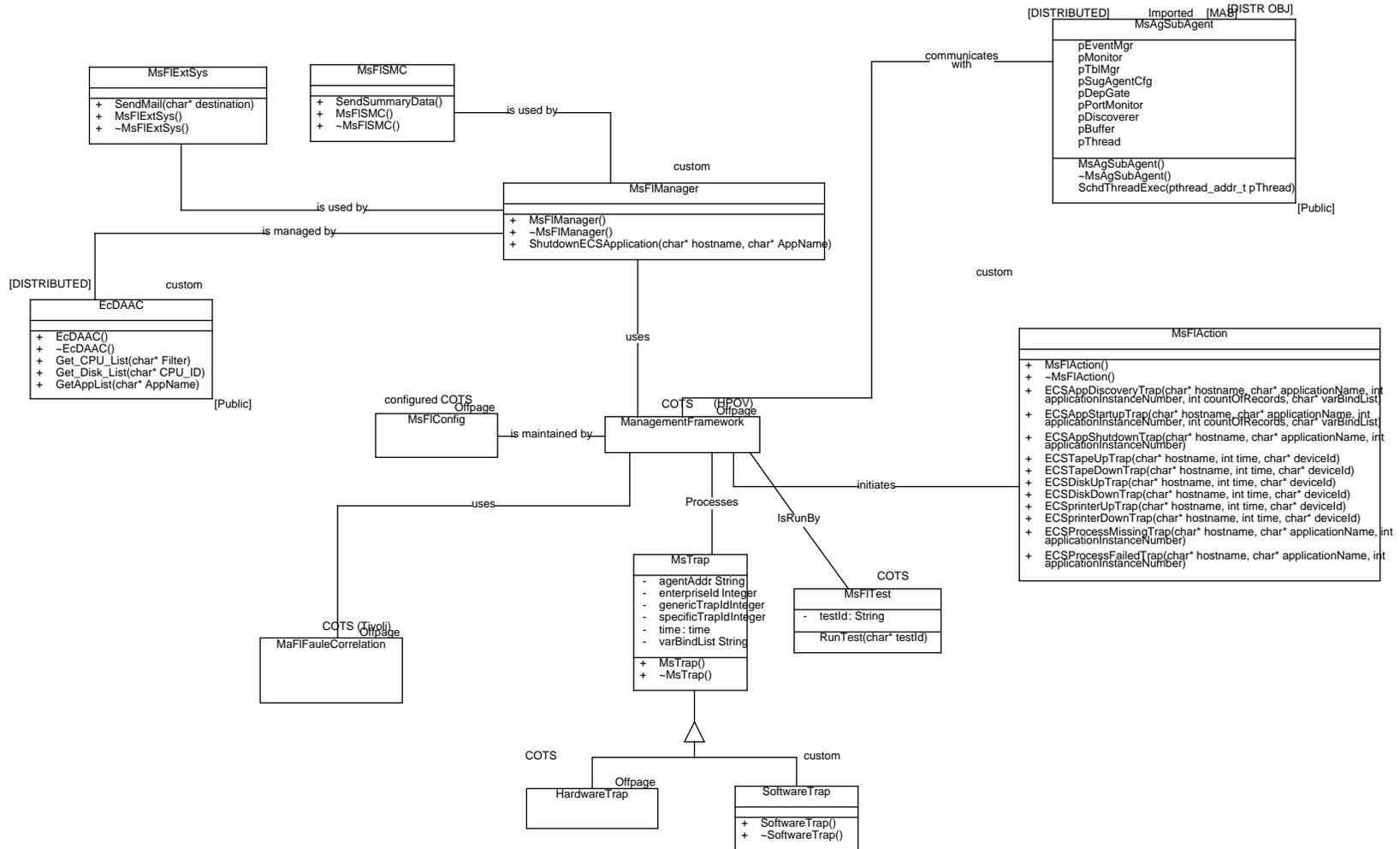


Figure 6.5-2. Fault Management Object Model

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: MsFIManager ismanagedby

6.5.3.2 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

6.5.3.3 MaFIFauleCorrelation Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

MsFIFaultCorrecation is a COTS product from Tivoli. It consists of the Enterprise Console and Sentry components. It communicates with HP Openview via a COTS adaptor and with agents supplied by Sentry. It performs rules based fault correlation of fault event.

Attributes:

None

Operations:

None

Associations:

The MaFIFauleCorrelation class has associations with the following classes:

Class: ManagementFramework uses

6.5.3.4 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: MsFITest IsRunBy

Class: MsTrap Processes

Class: MsAgSubAgent communicateswith

Class: MsFlAction initiates

Class: MsFlConfig ismaintainedby

Class: MaFlFauleCorrelation uses

Class: MsFlManager uses

6.5.3.5 MsAgSubAgent Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

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:

pBuffer - This attribute represents a pointer to a StaticBuffer.

pDepGate - This attribute represents a pointer to a deputy gate.

pDiscoverer - This attribute represents a pointer to a discoverer.

pEventManager - This attribute represents a pointer to an event manager.

pMonitor - This attribute represents a pointer to a monitor.

pPortMonitor - This attribute represents a pointer to a port monitor.

pSugAgentCfg - This attribute represents a pointer to the subagent configuration.

pTblMgr - This attribute represents a pointer to a table manager.

pThread - This attribute represents a pointer to a thread.

Operations:

MsAgSubAgent - This method represents the constructor of the object.

Arguments:

SchdThreadExec - This method spawns a DCE thread.

Arguments: pthread_addr_t pThread

~MsAgSubAgent - This method represents the destructor of the object.

Arguments:

Associations:

The MsAgSubAgent class has associations with the following classes:

Class: ManagementFramework communicateswith

6.5.3.6 MsFIAction Class

Parent Class: Not Applicable

Public: No

Distributed Object: No

Purpose and Description:

The class MsFIAction 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_startup, 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: create an object for each process in the var bind list, set the shutdown UUID for the process, assign the object a selection name based on the naming convention, create a symbol corresponding to the process object, put the symbol on the appropriate submap, set 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

ECSPProcessMissingTrap - 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

ECSSprinterDownTrap - 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

ECSSprinterUpTrap - 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

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

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

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsFlAction class has associations with the following classes:

Class: ManagementFramework initiates

6.5.3.7 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 is maintained by

6.5.3.8 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:

MsFIExtSys

Arguments:

Return Type:Void

Privilege:Public

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

Arguments:char* destination

Return Type:Void

Privilege:Public

~MsFIExtSys - This is the destructor for this class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsFIExtSys class has associations with the following classes:

Class: MsFIManager isusedby

6.5.3.9 MsFIManager 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:

Return Type:Void

Privilege:Public

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 deregeistered.

Arguments:char* hostname, char* AppName

Return Type:Void

Privilege:Public

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

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsFlManager class has associations with the following classes:

Class: EcDAAC ismanagedby

Class: MsFlExtSys isusedby

Class: MsFlSMC isusedby

Class: ManagementFramework uses

6.5.3.10 MsFlSMC Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents the interafce between the Fault Management Application Service at a site and the Fault Management Application Service at the SMC.

Attributes:

None

Operations:

MsFISMC - This is the default constructor for this class.

Arguments:

Return Type:Void

Privilege:Public

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

Arguments:

Return Type:Void

Privilege:Public

~MsFISMC - This is the destructor for this class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsFISMC class has associations with the following classes:

Class: MsFIManager isusedby

6.5.3.11 MsFITest 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 MsFITest class has associations with the following classes:
Class: ManagementFramework IsRunBy

6.5.3.12 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 MsFIAction to be executed based on the configuration information represented by MsFIConfig. 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 MsFIAction 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 varuable bindings sent with the trap.

Data Type:String

Privilege:Private

Default Value:

Operations:

MsTrap - This is the default constructor for this class.

Arguments:

Return Type:Void

Privilege:Public

~MsTrap - This is the destructor for this class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsTrap class has associations with the following classes:

Class: ManagementFramework Processes

6.5.3.13 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:

Return Type:Void

Privilege:Public

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

Arguments:

Return Type:Void

Privilege:Public

Associations:

The SoftwareTrap class has associations with the following classes:

None

6.5.4 Fault Management Dynamic Model

6.5.4.1 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 6.5-3.

6.5.4.1.1 Beginning Assumptions

None.

6.5.4.1.2 Interfaces with Other Subsystems and Segments

Management Agent Services

6.5.4.1.3 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.

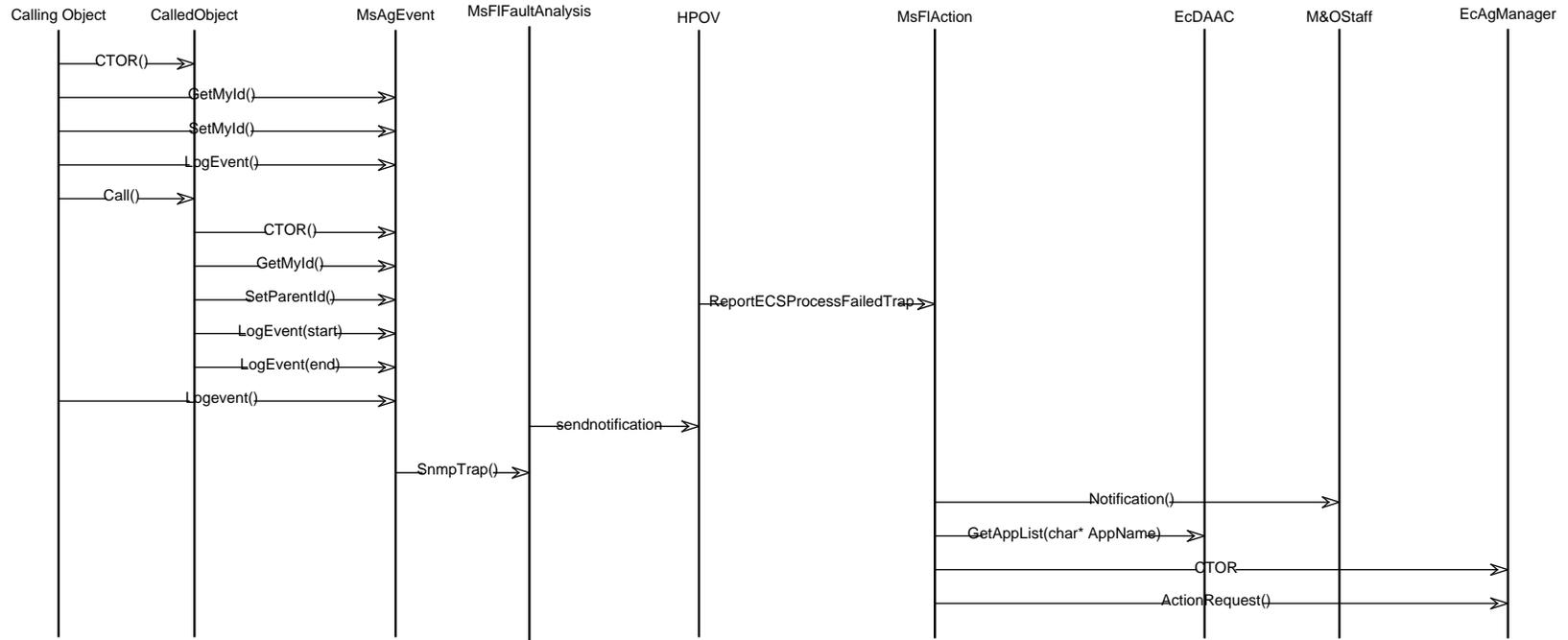


Figure 6.5-3. Fault Notification by an ECS Application Event Trace

6.5.4.1.4 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

6.5.4.1.5 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.

6.5.4.1.6 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.

6.5.4.1.7 Scenario Description

This description describes the accompanying event trace.

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 Data 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.

6.5.5 Fault Management Structure

Table 6.5-2 lists the components of the Fault Management Application Service.

Table 6.5-2. Fault Management Components

Component Name	COTS/Custom
MsFIManager	Custom (C++ code)
HPOV	COTS
MsFIConfig	Configuration of COTS (HPOV)
MsFIAction	Custom (C++ code and scripts)
MsManager	C++ code imported from Management Agent Services
EcDAAC	External C++ Class category
MsFITest	Vendor-provided diagnostic tests

6.5.5.1 Network Management Framework CSC

Purpose and Description

The Network Management Framework provides the framework for network 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 - HP Openview COTS

MsFIConfig

HP OpenView Network Node Manager

6.5.5.2 System Management Framework CSC

Purpose and Description

The System Management Framework provides the framework for system management . It consist of Tivoli COTS products: Enterprise Console, Courier, Sentry and application and event adapters.

Courier provides ECS wide software management and distribution capabilities including remote instillation of software from a central location. Sentry provides performance monitoring of ECS host and client software. It collects both Tivoli defined and user defined performance metrics and provides for the setting of thresholds to generate alerts to the management system. The Enterprise Console consists of a graphical GUI to view and control the ECS enterprise. It performs rules based fault correlation and response to events received from HP Openview and Sentry.

6.5.5.3 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

MsFITest

6.5.5.4 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

MsFIManager

- DiscoverECSApplications - C++ code

- StartupECS Applications - C++ code

- ShutdownECSApplications - C++ code

6.5.5.5 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, the correlation of faults, automatic responses to well known faults.

Mapping to objects implemented by this component

MsFIAction

- ECSAppDiscoveryTrap - C++ code

- ECSAppStartupTrap - C++ code

- ECSAppShutdownTrap - C++ code

Scripts:

- ECSTapeUpTrap

- ECSTapeDownTrap

- ECSDiskUpTrap

- ECSDiskDownTrap

- ECSPrinterUpTrap

- ECSPrinterDownTrap

- ECSProcessMissingTrap

- ECSProcessFailedTrap

- MsFIFaultCorrelation - COTS (Tivoli)

6.5.5.6 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

6.5.6 Fault Management Management and Operation

6.5.6.1 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, the CSS server will provide a backup platform to run MSS management software.

6.5.6.2 Operator Interfaces

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

6.5.6.3 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 B Overview Design Specification (305-CD-020-002).

6.6 Performance Management

6.6.1 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 B, the Performance Management Application Service will consist mainly of two COTS applications, HP OpenView and Tivoli/EC . Table 6.6-1 provides an indication of responsibility between HP OpenView and Tivoli for providing performance management of the various ECS managed objects.

Table 6.6-1. 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	Tivoli & HP Openview
Operating systems	Tivoli
File systems	Tivoli

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 6.6-2 through 6.6-4, 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 6.6-2. 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 6.6-3. 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 6.6-4. 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, Tivoli, provides similar monitoring capabilities for applications and operating systems. The following tables, Table 6.6-5 through 6.6-9, provide a representative sample of the performance attributes that will be monitored using the Tivoli COTS product. These attributes are not inclusive. Tivoli allows the collection of user defined metrics returned by user provided scripts and programs. The operator will have the capability to collect information on additional supported attributes or to stop collecting information on listed attributes.

Table 6.6-5. 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
	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

Table 6.6-5. Global System Performance Metrics Table (2 of 2)

Attribute type	Attribute Description
	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 6.6-6. 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 6.6-7. Process Performance Metrics Table

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
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 6.6-8. 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 6.6-9. 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 6.6-10. 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 6.6-10. 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)

6.6.2 Performance Management Context

The performance management context diagram is shown in Figure 6.6-1. 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.

6.6.3 Performance Management Object Model

The performance management object model is shown in Figure 6.6-2.

6.6.3.1 EcAgProxy Class

Parent Class:Not Applicable

Public:Yes

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:

modeB - This attribute contains the mode in which the application is executing under. It identifies functional activity(operational, testing, training).

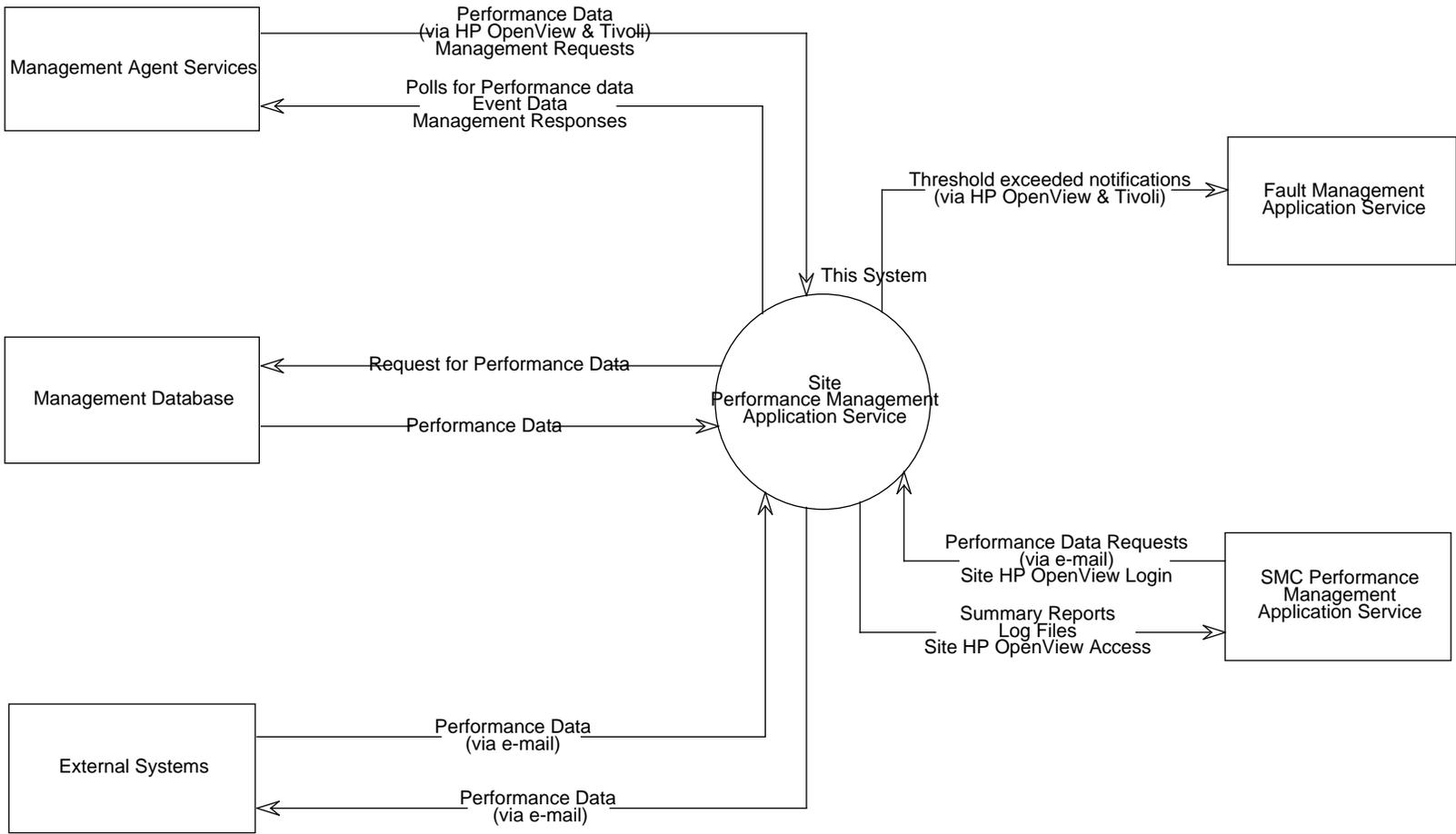


Figure 6.6-1. Performance Management Context Diagram

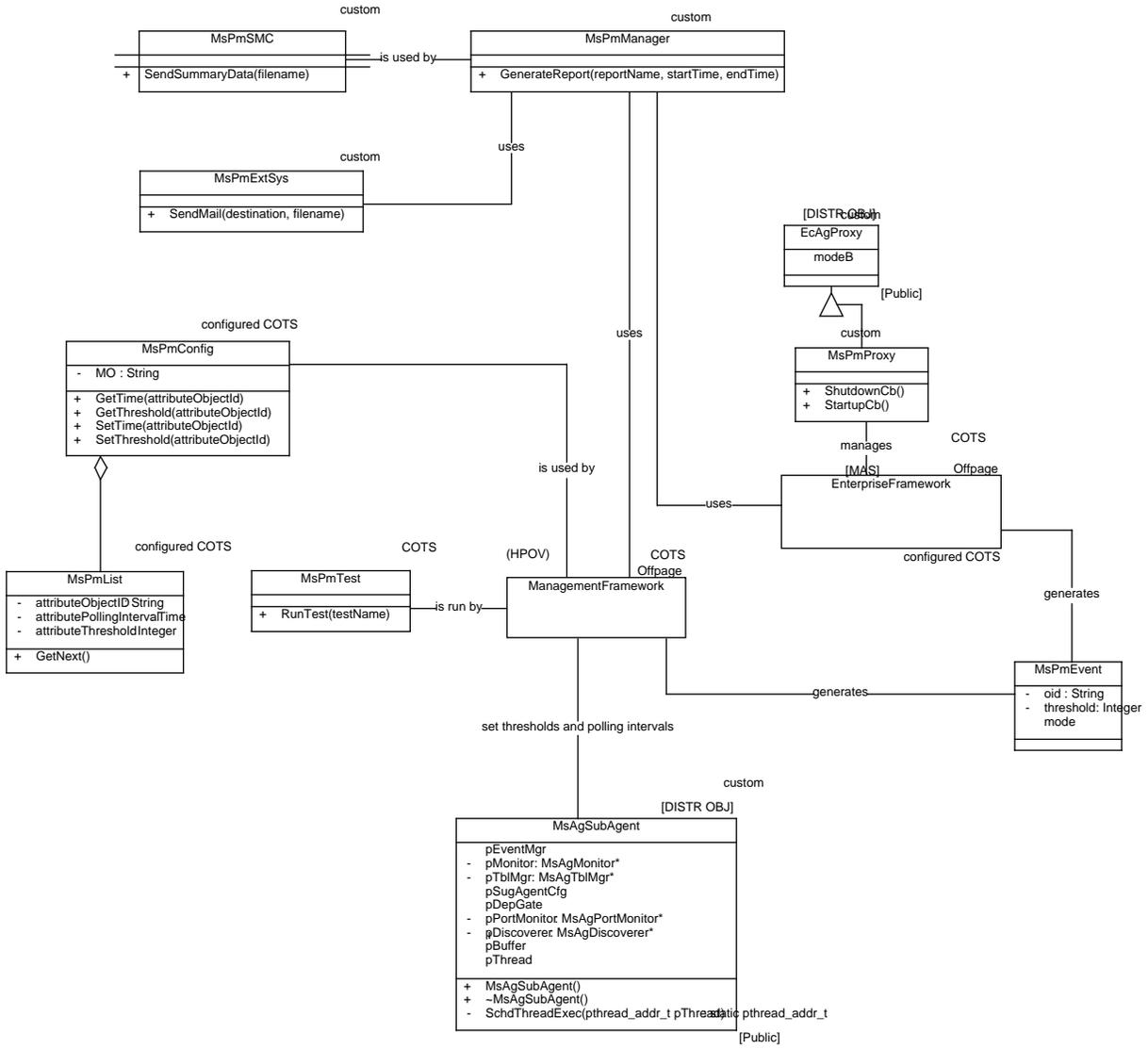


Figure 6.6-2. Performance Management Object Model

Operations:

None

Associations:

The EcAgProxy class has associations with the following classes:

None

6.6.3.2 EnterpriseFramework Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

EnterpriseFramework is the Tivoli COTS product that performs enterprise wide services: System Administration (Tivoli/Admin), Software distribution (Tivoli/Courier), performance monitoring (Tivoli/Sentry) and fault correlation (Tivoli/Enterprise Console). The framework also acts as the integrated desktop for Maintenance and Operations, integrating other administrative functions such as Sybase database administration, system backup/restore, and DCE Cell administration.

Attributes:

None

Operations:

None

Associations:

The EnterpriseFramework class has associations with the following classes:

Class: MsPmEvent generates

Class: MsPmProxy manages

Class: MsPmManager uses

6.6.3.3 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: MsAgSubAgent setthresholdsandpollingintervals

Class: MsPmManager uses

6.6.3.4 MsAgSubAgent Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

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:

pBuffer - This attribute represents a pointer to a StaticBuffer.

pDepGate - This attribute represents a pointer to a deputy gate.

pDiscoverer - This attribute represents a pointer to a discoverer.

Data Type:MsAgDiscoverer*

Privilege:Private

Default Value:

pEventManager - This attribute represents a pointer to an event manager.

pMonitor - This attribute represents a pointer to a monitor.

Data Type:MsAgMonitor*

Privilege:Private

Default Value:

pPortMonitor - This attribute represents a pointer to a port monitor.

Data Type:MsAgPortMonitor*

Privilege:Private

Default Value:

pSugAgentCfg - This attribute represents a pointer to the subagent configuration.

pTblMgr - This attribute represents a pointer to a table manager.

Data Type:MsAgTblMgr*

Privilege:Private

Default Value:

pThread - This attribute represents a pointer to a thread.

Operations:

MsAgSubAgent - This method represents the constructor of the object.

Arguments:

Return Type:Void

Privilege:Public

SchdThreadExec - This method spawns a DCE thread.

Arguments:pthread_addr_t pThread

Return Type:static pthread_addr_t

Privilege:Private

~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: MsAgSubAgent

Class: ManagementFramework setthresholdsandpollingintervals

6.6.3.5 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 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

Return Type:Void

Privilege:Public

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

Return Type:Void

Privilege:Public

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

Return Type:Void

Privilege:Public

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

Return Type:Void

Privilege:Public

Associations:

The MsPmConfig class has associations with the following classes:

Class: ManagementFramework isusedby

6.6.3.6 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:

mode - Defines the mode of the process that caused the event: Operational, test, simulation

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: EnterpriseFramework generates

Class: ManagementFramework generates

6.6.3.7 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

Return Type:Void

Privilege:Public

Associations:

The MsPmExtSys class has associations with the following classes:

Class: MsPmManager uses

6.6.3.8 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)

6.6.3.9 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

Return Type:Void

Privilege:Public

Associations:

The MsPmManager class has associations with the following classes:

Class: MsPmSMC is used by

Class: EnterpriseFramework uses

Class: ManagementFramework uses

Class: MsPmExtSys uses

6.6.3.10 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:

Return Type:Void

Privilege:Public

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:
Return Type:Void
Privilege:Public

Associations:

The MsPmProxy class has associations with the following classes:
Class: EnterpriseFramework manages

6.6.3.11 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
Return Type:Void
Privilege:Public

Associations:

The MsPmSMC class has associations with the following classes:
Class: MsPmManager isusedby

6.6.3.12 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

Return Type: Void

Privilege: Public

Associations:

The MsPmTest class has associations with the following classes:

Class: ManagementFramework isrunby

6.6.4 Performance Management Dynamic Model

6.6.4.1 Degradation of Performance Alert

The degradation of performance alert scenario is depicted in Figure 6.6-3.

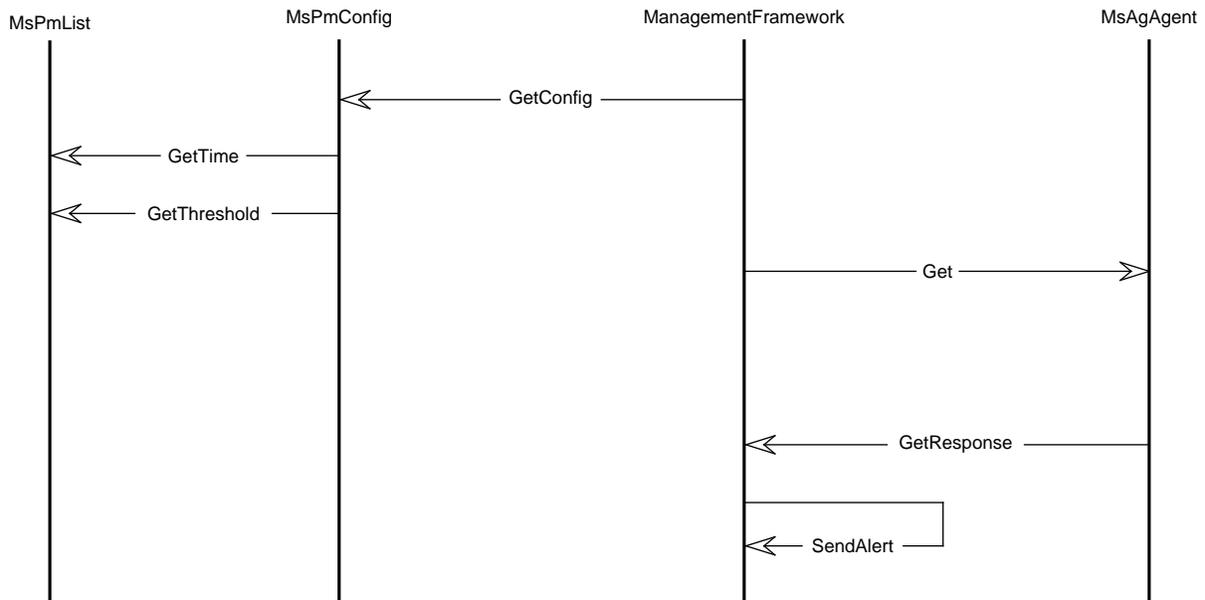


Figure 6.6-3. Degradation of Performance Alert

6.6.4.1.1 Beginning Assumptions

Performance thresholds have been stored for system managed objects.

6.6.4.1.2 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.

6.6.4.1.3 Stimulus

A measured managed object attribute exceeds the set threshold.

6.6.4.1.4 Participating Classes From the Object Model

ManagementFramework

MsPmConfig

MsPmList

6.6.4.1.5 Beginning System, Segment and Subsystem State(s)

The performance management system is in the normal operational state.

6.6.4.1.6 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.

6.6.4.1.7 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.

6.6.4.2 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 6.6-4.

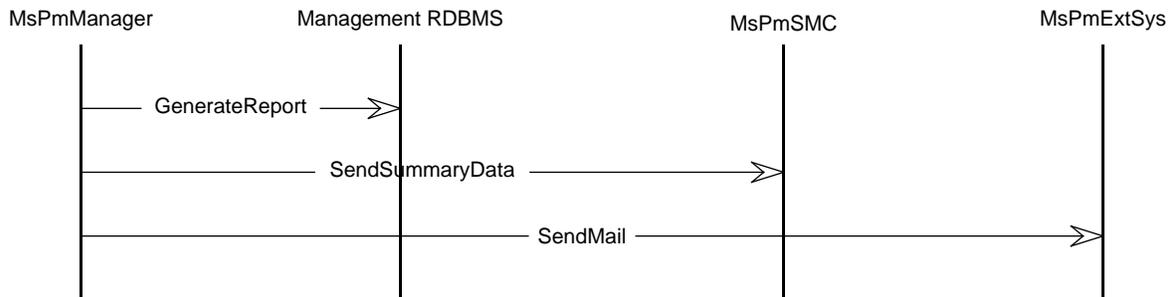


Figure 6.6-4. Providing Performance Summary to SMC

6.6.4.2.1 Beginning Assumptions

None.

6.6.4.2.2 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.

6.6.4.2.3 Stimulus

A daily site performance report is required to be delivered to the SMC and to an external system.

6.6.4.2.4 Participating Classes From the Object Model

MsPmExtSys

MsPmManager

MsPmSMC

6.6.4.2.5 Beginning System, Segment and Subsystem State(s)

The system is operating in the normal operational mode.

6.6.4.2.6 Ending State

The system is operating in the normal operational mode with the performance data sent to the SMC and the external system.

6.6.4.2.7 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.

6.6.5 Performance Management Structure

Table 6.6-11 identifies the components of the Performance Management Application Service.

Table 6.6-11. 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

6.6.5.1 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

6.6.5.2 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

6.6.5.3 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

6.6.5.4 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

6.6.6 Performance Management Management and Operation

6.6.6.1 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.

6.6.6.2 Operator Interfaces

All operator interfaces will be through HP OpenView or through a COTS performance management package.

6.6.6.3 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).

6.7 Physical Configuration Management Service

6.7.1 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.

6.7.2 Physical Configuration Management Service Context

The physical configuration management service context diagram is shown in Figure 6.7-1.

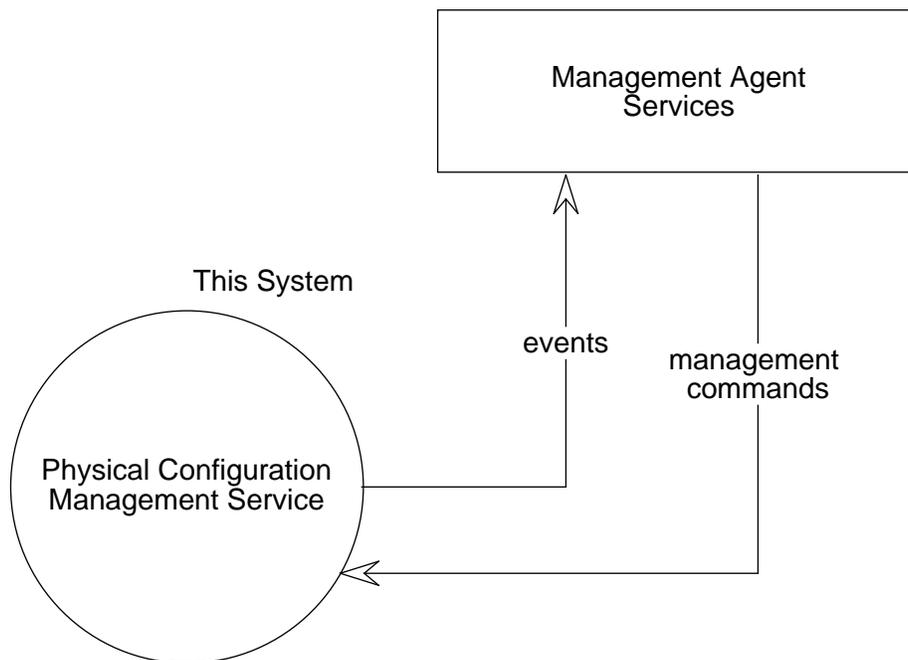


Figure 6.7-1. Physical Configuration Management Service Context Diagram

6.7.3 Physical Configuration Management Service Object Model

The Physical Configuration Management Object Model, Figure 6.7-2, 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.

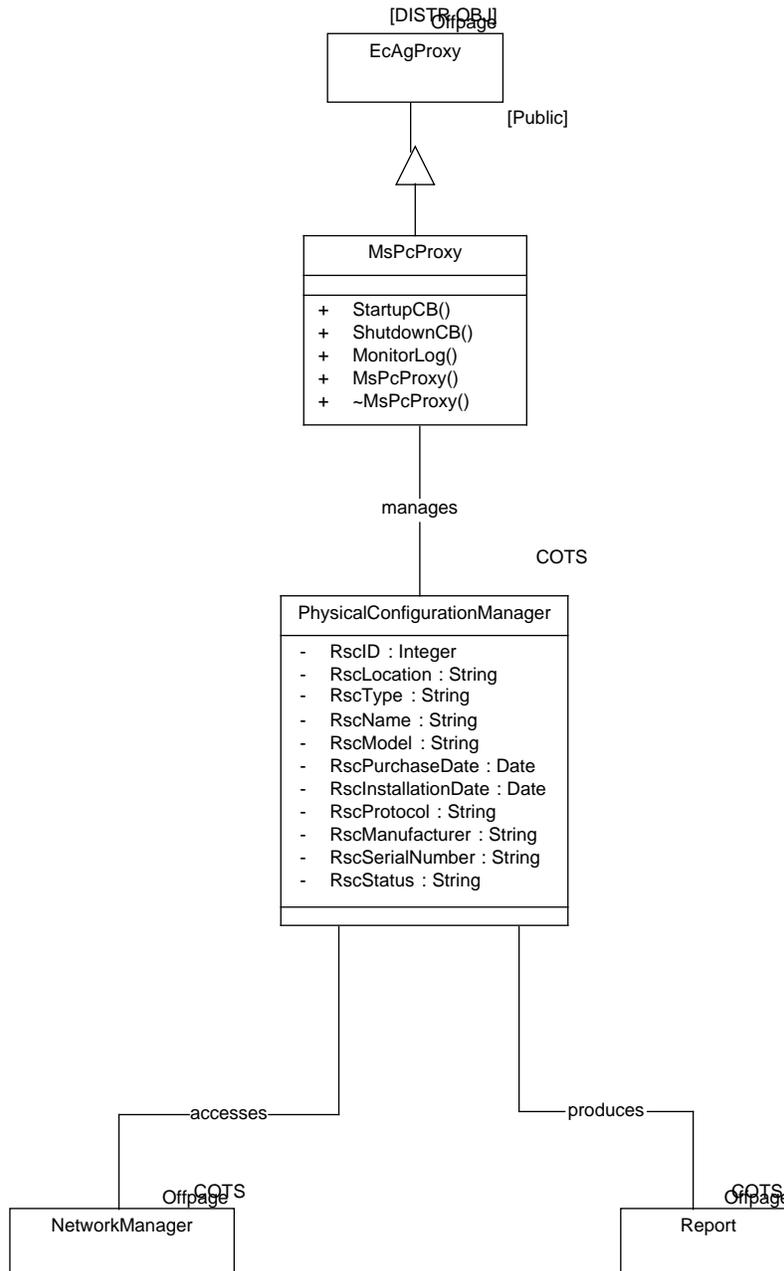


Figure 6.7-2. Physical Configuration Management Service Object Model

6.7.3.1 EcAgProxy Class

Parent Class:Not Applicable

Public:Yes

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 a 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

6.7.3.2 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

6.7.3.3 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

6.7.3.4 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

6.7.3.5 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

6.7.4 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.

6.7.4.1 Add A New Node Scenario

This scenario is depicted in Figure 6.7-3.

6.7.4.1.1 Beginning Assumptions

The network is operating normally.

6.7.4.1.2 Interfaces with Other Subsystems and Segments

None.

6.7.4.1.3 Stimulus

A new node is added to the network.

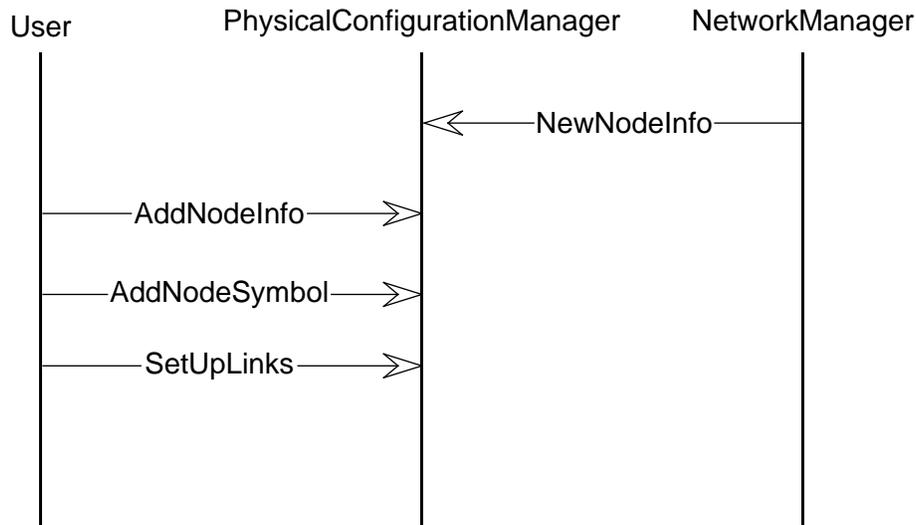


Figure 6.7-3. Add A New Node

6.7.4.1.4 Participating Classes From the Object Model

NetworkManager

6.7.4.1.5 Beginning System, Segment and Subsystem State(s)

The Physical Configuration Manager and the Network Manager are operating normally.

6.7.4.1.6 Ending State

All the relevant information concerning the new node becomes part of the system.

6.7.4.1.7 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.

6.7.4.2 Move An Existing Node Scenario

This scenario is depicted in Figure 6.7-4.

6.7.4.2.1 Beginning Assumptions

The network is operating normally.

6.7.4.2.2 Interfaces with Other Subsystems and Segments

None.

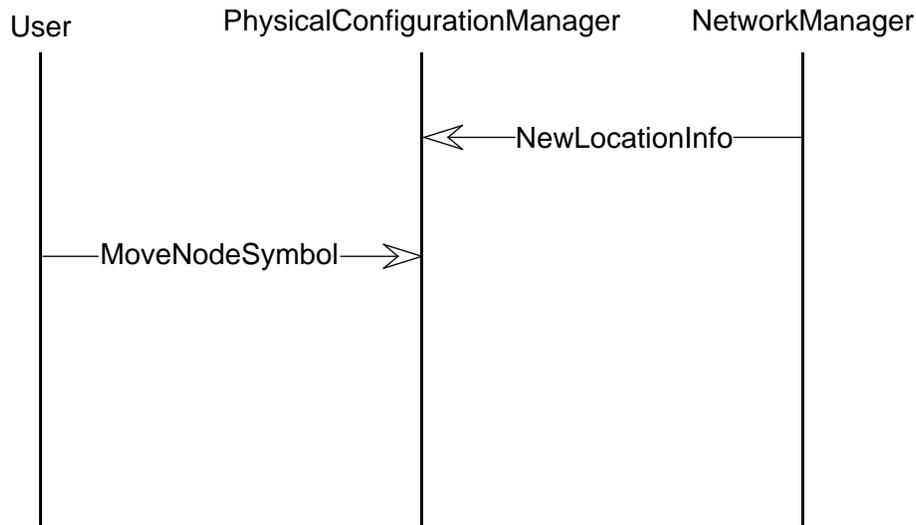


Figure 6.7-4. Move An Existing Node

6.7.4.2.3 Stimulus

An existing node is moved to a new location in the network.

6.7.4.2.4 Participating Classes From the Object Model

NetworkManager

6.7.4.2.5 Beginning System, Segment and Subsystem State(s)

The Physical Configuration Manager and the Network Manager are operating normally.

6.7.4.2.6 Ending State

The information concerning the new location is configured into the system.

6.7.4.2.7 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.

6.7.4.3 Delete An Existing Node Scenario

This scenario is depicted in Figure 6.7-5.

6.7.4.3.1 Beginning Assumptions

The network is operating normally.

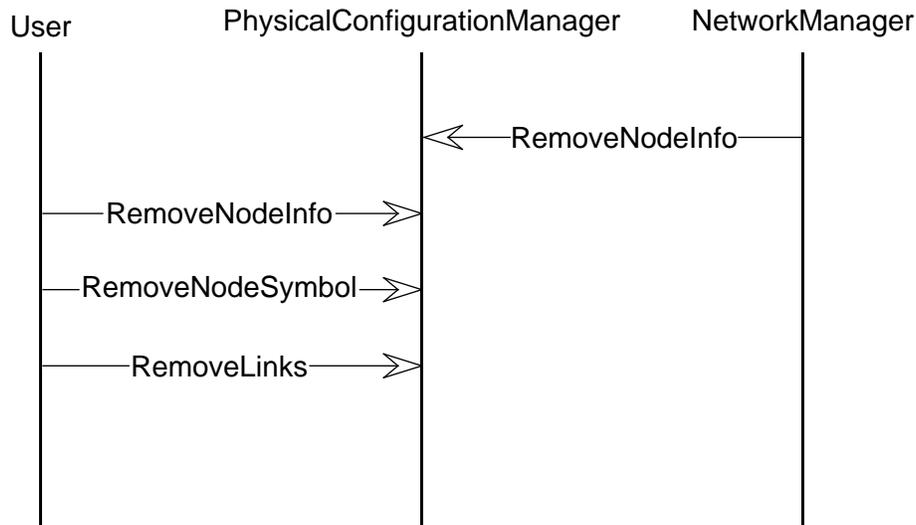


Figure 6.7-5. Delete An Existing Node

6.7.4.3.2 Interfaces with Other Subsystems and Segments

None.

6.7.4.3.3 Stimulus

An existing node is deleted from the network.

6.7.4.3.4 Participating Classes From the Object Model

NetworkManager

6.7.4.3.5 Beginning System, Segment and Subsystem State(s)

The Physical Configuration Manager and the Network Manager are operating normally.

6.7.4.3.6 Ending State

The node symbol and information related to the deleted node are removed from the system.

6.7.4.3.7 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.

6.7.5 Physical Configuration Management Service Structure

Table 6.7-1 lists the components of the Physical Configuration Management Service.

Table 6.7-1. Physical Configuration Management Service Components

Component Name	COTS/Custom
Physical Configuration Manager	COTS
Network Manager	COTS
Physical Configuration Proxy Agent	Custom

6.7.5.1 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

6.7.5.2 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

6.7.5.3 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

6.7.6 Physical Configuration Management Service Management and Operation

6.7.6.1 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.

6.7.6.2 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.

6.7.6.3 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. Additional 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.

6.8 Security Management

6.8.1 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) and the authorization database (DCE Access Control List Managers). This is managed through the use of Hewlett Packard's acctmgr tool. The acctmgr tool 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). The acctmgr is part of HP's DCE Core Services.

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, provides the capability to analyze security audit trails, and provides 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, receiving security reports from the sites, and receiving notifications of and coordinating the recovery from detected security breaches at the sites and external systems.

6.8.2 Security Management Context

The Security Management Application Service, as shown in the context diagram, Figure 6.8-1, interfaces with the SMC, the Fault Management Application Service, the Management Database and with systems external to ECS, namely NSI, IP, NCC, ASTER GDS, MMO, and NOLAN. 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.

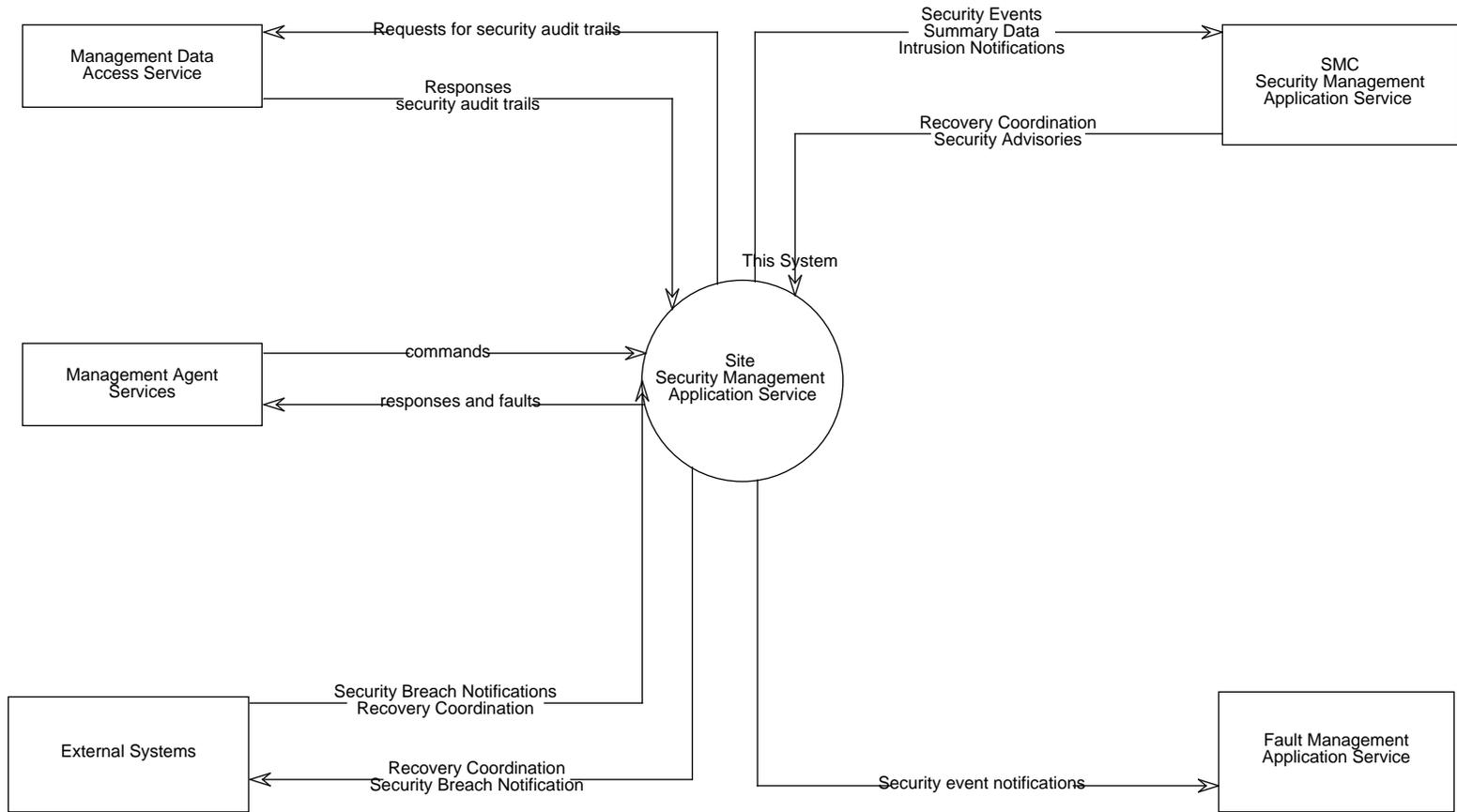


Figure 6.8-1. Security Management Context Diagram

6.8.3 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)). HP's DCE Cell Administration Tools are a collection of GUI tools that provide the capability for DCE Cell Management, to include the management of the authentication database (Registry database) for the management of DCE principals.

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). HP's acctmgr tool, 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 6.8-2.

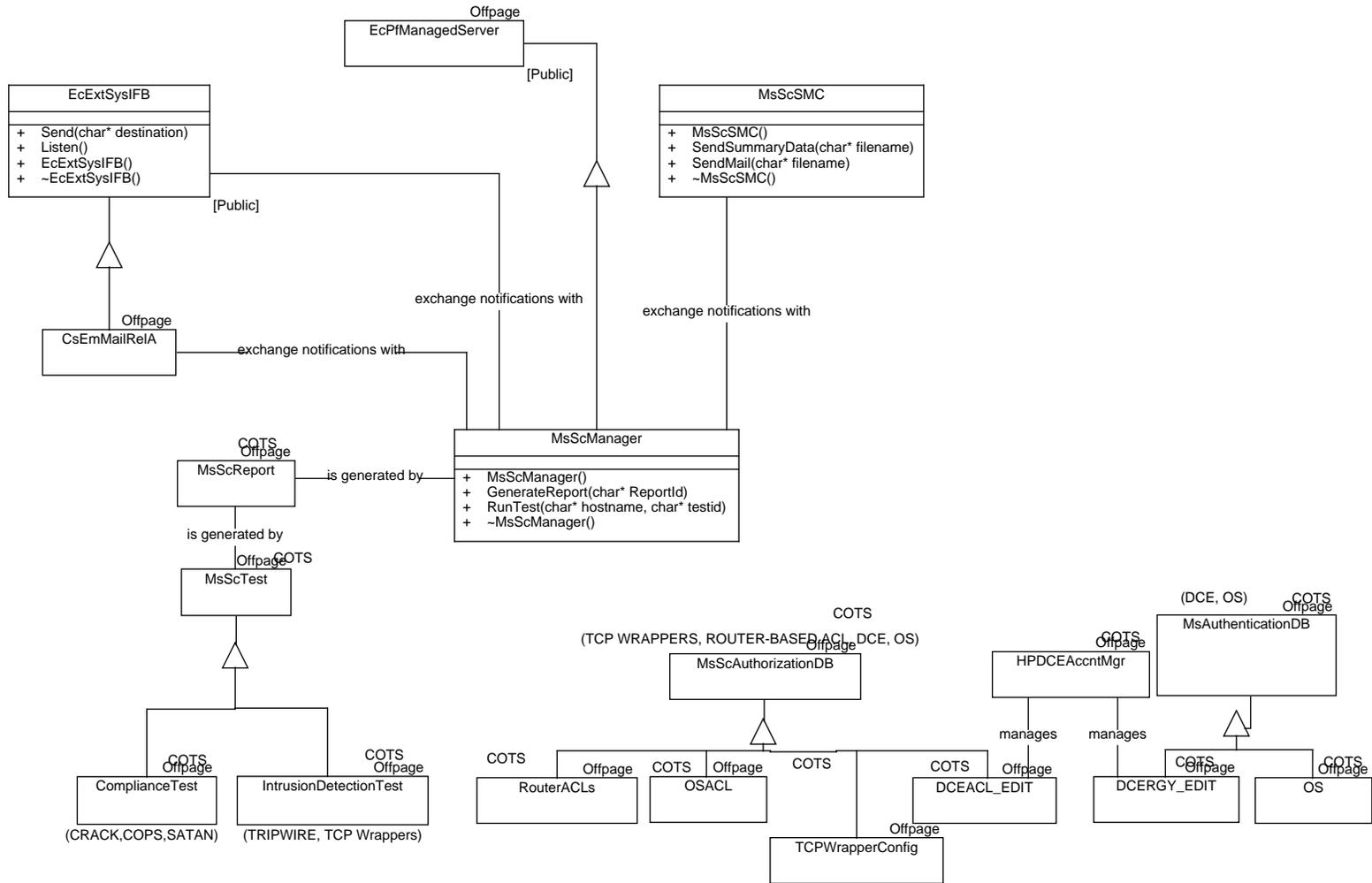


Figure 6.8-2. Security Management Object Model

6.8.3.1 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

6.8.3.2 CsEmMailRelA Class

Parent Class:EcExtSysIFB

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The CsEmMailRelA class has associations with the following classes:

Class: MsScManager exchangenotificationswith

6.8.3.3 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: HPDCEAcctMgr manages

6.8.3.4 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: HPDCEAcctMgr manages

6.8.3.5 EcExtSysIFB Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

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

Attributes:

None

Operations:

EcExtSysIFB - This is the default constructor for this class.

Arguments:

Return Type:Void

Privilege:Public

Listen - This method listens for an SNMP trap or a TCP socket, from an external subsystem.

Arguments:

Return Type:Void

Privilege:Public

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

Arguments:char* destination

Return Type:Void

Privilege:Public

~EcExtSysIFB - This is the destructor for this class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The EcExtSysIFB class has associations with the following classes:

Class: MsScManager exchangefunctionswith

6.8.3.6 EcPfManagedServer Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

This is the container class that starts up the event Manager, table Manager, monitor, port monitor, discoverer, subagent configuration, static buffer, and the deputy gate. This class also starts a thread that triggers scheduled events (i.e. polling ECS application's performance metrics).

Attributes:

None

Operations:

None

Associations:

The EcPfManagedServer class has associations with the following classes:

None

6.8.3.7 HPDCEAcctMgr Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents the COTS product acctmgr which is one of the DCE Cell Management GUI Tools that come with the DCE Core Service from Hewlet Packard. This product provides the capability to manage the DCE authentication database (the Registry) and access control for cell resources (authorization). It is through this interface that an ECS Registered user will be assigned to security groups that will define what information and services the user is authorized to access. 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 HPDCEAcctMgr class has associations with the following classes:

Class: DCEACL_EDIT manages

Class: DCERGY_EDIT manages

6.8.3.8 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

6.8.3.9 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

6.8.3.10 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

6.8.3.11 MsScManager Class

Parent Class:EcPfManagedServer

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:

All Attributes inherited from parent class

Operations:

GenerateReport

Arguments:char* ReportId

Return Type:Void

Privilege:Public

MsScManager - This is the default constructor for this class.

Arguments:

Return Type:Void

Privilege:Public

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:

Return Type:Void

Privilege:Public

Associations:

The MsScManager class has associations with the following classes:

Class: CsEmMailRelA exchangenotificationswith

Class: EcExtSysIFB exchangenotificationswith

Class: MsScSMC exchangenotificationswith

Class: MsScReport isgeneratedby

6.8.3.12 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

6.8.3.13 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 exchangefunctionswith

6.8.3.14 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

6.8.3.15 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

6.8.3.16 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

6.8.3.17 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 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

6.8.3.18 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:
Non

6.8.4 Security Management Dynamic Model

6.8.4.1 Executing a Compliance Test

This scenario is depicted in Figure 6.8-3.

6.8.4.1.1 Beginning Assumptions

None.

6.8.4.1.2 Interfaces with Other Subsystems and Segments

Management Agent Services

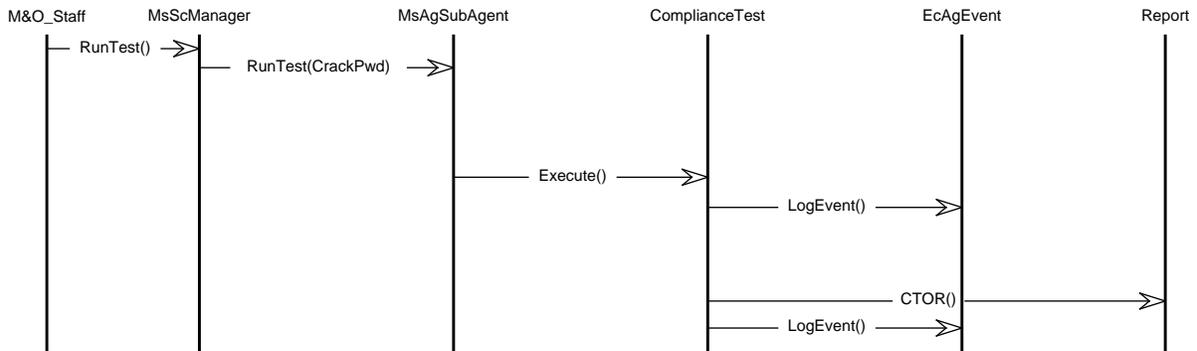


Figure 6.8-3. Executing a Compliance Test

6.8.4.1.3 Stimulus

An operator initiates the execution of a password guessing test (a compliance test).

6.8.4.1.4 Participating Classes From the Object Model

MsScManager

ComplianceTest

EcAgEvent

Report

6.8.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and subsystem are in a steady state.

6.8.4.1.6 Ending State

The system, segment and subsystem are in a steady state.

6.8.4.1.7 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.

6.8.4.2 Reporting a Security Intrusion

This scenario is depicted in Figure 6.8-4.

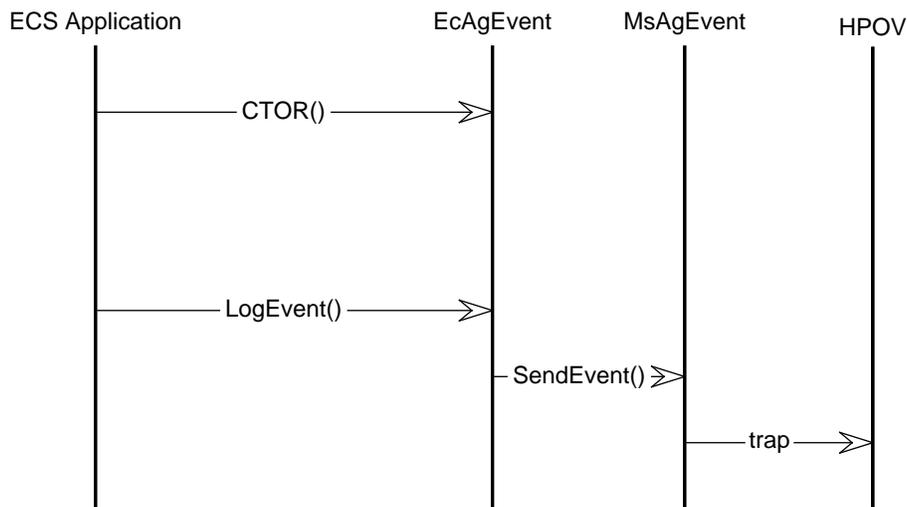


Figure 6.8-4. Reporting a Security Intrusion

6.8.4.2.1 Beginning Assumptions

None.

6.8.4.2.2 Interfaces with Other Subsystems and Segments

Management Agent Services

6.8.4.2.3 Stimulus

An application receives a request from an unauthorized client.

6.8.4.2.4 Participating Classes From the Object Model

An ECS application

EcAgEvent

MsAgEvent

UUID

ManagementFramework (HPOV)

6.8.4.2.5 Beginning System, Segment and Subsystem State(s)

The system, segment and subsystem are in a steady state.

6.8.4.2.6 Ending State

The system, segment and subsystem are in a steady state.

6.8.4.2.7 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).

6.8.5 Security Management Structure

Table 6.8-1 lists the components of the Security Management Application Service.

Table 6.8-1. Security Management Components

Component Name	COTS/Custom
MsScManager	Custom(C++ code)
MsScSMC	Custom
MsScExtSys	Custom
MsScReport	Custom
Authentication Databases	COTS (Operating system, DCE Registry database) HP's acctmgr tool is 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). HP's acctmgr tool is selected for the management of DCE ACLs.
ComplianceTest	COTS (SATAN, Crack, COPS)
IntrusionDetectionTest	COTS (Tripwire, TCP Wrappers)

6.8.5.1 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.

6.8.5.2 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.

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

6.8.5.3 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.

Candidate products

CRACK, COPS, SATAN, TRIPWIRE

6.8.5.4 DCE Cell Management CSC

Purpose and Description

This class provides the capability to manage DCE services including security services.

Candidate products

HP's acctmgr tool

6.8.6 Security Management Management and Operation

6.8.6.1 System Management Strategy

The Security Management Service Management Strategy utilizes the ECS Process Framework for its management.

6.8.6.2 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 HP's acctmgr tool has both interfaces.

6.8.6.3 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.

6.9 Trouble Ticketing

6.9.1 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 of 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 “Solution Proposed”, 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).

In addition to tracking Trouble Tickets, the Remedy ARS will also function as the User Contact Log. Remedy will be configured to have a separate schema that will contain the entries that User Services personnel enter for each contact that they receive from a user. The User Contact Log will also allow a trouble ticket to be initiated from a log entry with the push of a button - the trouble ticket will be populated with information from the contact log.

6.9.2 Trouble Ticketing Context

The Trouble Ticketing Service context diagram is shown in Figure 6.9-1. 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. To aid in the submission of trouble tickets, TTS requests and receives user profile information to populate submitter information in the HTML interface. The Trouble Ticketing Service receives updates and ticket submissions and sends notifications and displays to the Support Staff operators. The ECS users can submit Trouble Tickets and receive displays of Trouble Ticket status and e-mail notifications of status changes.

6.9.3 Trouble Ticketing Object Model

Figure 6.9-2 represents the classes which model the Trouble Ticketing Service.

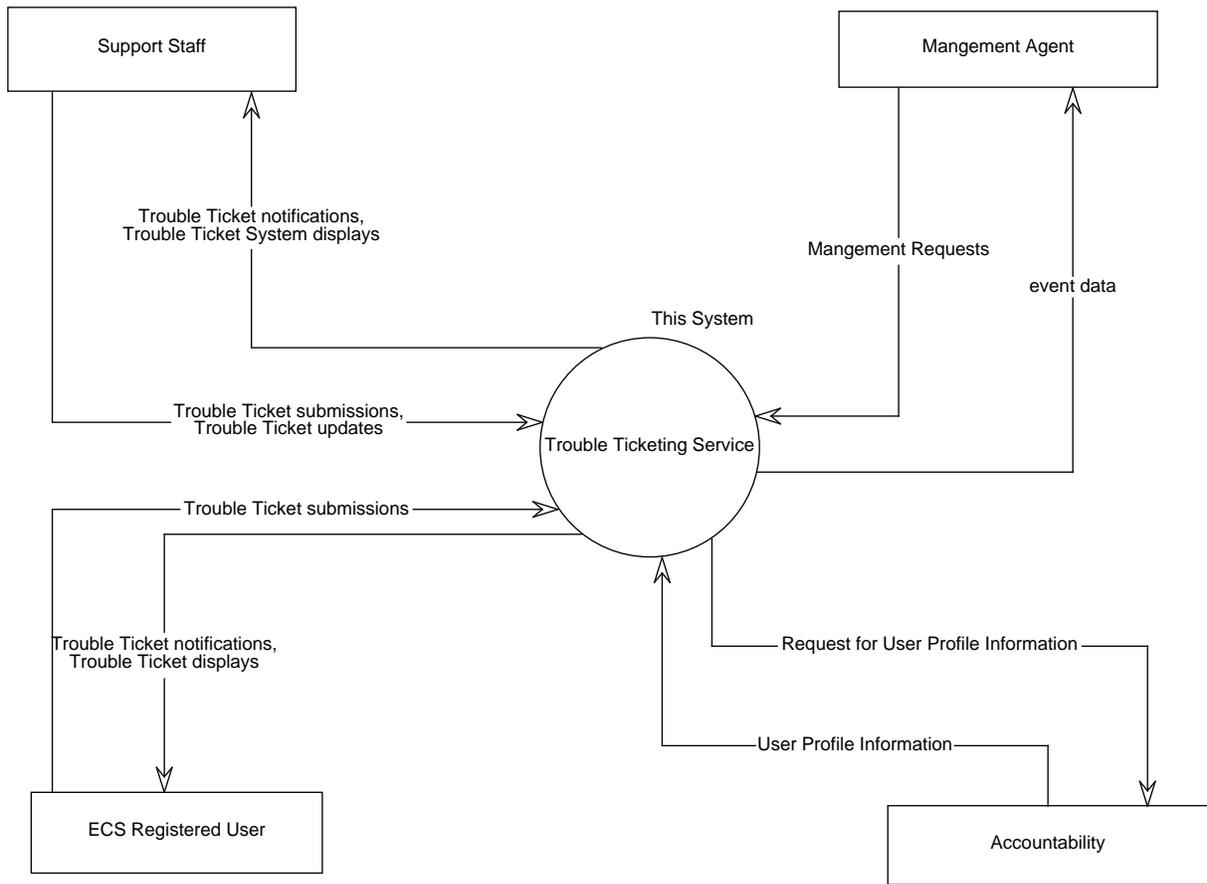


Figure 6.9-1. Trouble Ticketing Context Diagram

6.9.3.1 CGI_Vars Class

Parent Class:RWHashDictionary

Attributes:

All Attributes inherited from parent class

Operations:

CGI_Vars

Arguments:

Return Type:Void

Privilege:Public

LoadEnvironmentVariables

Arguments:

Return Type:Void

Privilege:Private

LoadGetElements

Arguments:

Return Type:Void

Privilege:Private

LoadPostElements

Arguments:

Return Type:Void

Privilege:Private

get

Arguments:char * szName

Return Type:CGI_Element *

Privilege:Public

get

Arguments:RWCString &rsName

Return Type:CGI_Element *

Privilege:Public

Associations:

The CGI_Vars class has associations with the following classes:

Class: MsTtHTMLItems providesHTMLinterface

Class: MsTtHTMLMenu providesHTMLinterface

6.9.3.2 CsEmMailRelA Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The CsEmMailRelA class has associations with the following classes:

Class: MsTtServiceRequestor sendTTsubmite-mail

6.9.3.3 EcAgCOTSManager Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

this abstract class embodies the characteristics and functionality of a manager object responsible for managing a single COTS process. It encapsulates all MSS management application functions into a single class. The COTS proxy agent developer is responsible for inheriting from this class and specializing it towards the COTS process to manage.

Attributes:

None

Operations:

None

Associations:

The EcAgCOTSMgr class has associations with the following classes:

None

6.9.3.4 MsAcUsrProfile Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

Attributes:

None

Operations:

None

Associations:

The MsAcUsrProfile class has associations with the following classes:

Class: MsTtHTMLItems Createsandreads

Class: MsAcUsrProfileMgr Populates

Class: MsTtHTMLMenu createsandreads

6.9.3.5 MsAcUsrProfileMgr Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:Yes

Purpose and Description:

This class represents the User Profile Manager class that governs the update and maintenance of information in the MsAcUsrProfile class. An ECS science user's available balance will be retrieved using this class and be debited by the amount of each data product request received by MSS.

Attributes:

None

Operations:

None

Associations:

The MsAcUsrProfileMgr class has associations with the following classes:

Class: MsAcUsrProfile Populates

Class: MsTtHTMLItems Requestuserprofilefrom

Class: MsTtHTMLMenu requestuserprofilefrom

6.9.3.6 MsTtEntry Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

The MsTtEntry class models a request for action (Trouble Ticket) on a particular problem and the subsequent actions performed on it. This class contains the fields which are accessible to a user through the Web interface. This class will contain a trouble ticket object to be submitted to the Remedy Action Request System or a trouble ticket object retrieved from Remedy to be displayed to the user through the Web interface.

Attributes:

entryId - The unique identifier of a TT (generated by Remedy Action Request, has unique prefix for each DAAC)

Data Type:RWCString

Privilege:Private

Default Value:

problemLongDescription - A detailed description of the problem reported in the TT.

Data Type:RWCString

Privilege:Private

Default Value:

problemShortDescription - A brief (1 line) description of the problem reported in the TT.

Data Type:RWCString

Privilege:Private

Default Value:

resolutionLog - A running diary of the resolution process for the TT.

Data Type:RWCString

Privilege:Private

Default Value:

status - The current status of the TT, 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, Solution Proposed : indicates the trouble ticket has been

proposed a solution, Implement Solution : indicates the proposed solution of the trouble ticket has been approved to implement, Solution Implemented : indicates the proposed solution of the trouble ticket has been implemented, Closed : indicates the trouble ticket has been closed, Forwarded : indicates the trouble ticket has been forwarded to another site Work Around : indicates the trouble ticket has been temporarily addressed Not Repeatable : indicates the trouble ticket problem is not repeatable.

Data Type:EcTInt

Privilege:Private

Default Value:

submitterEmail - The e-mail address of the TT submitter.

Data Type:RWCString

Privilege:Private

Default Value:

submitterId - User Id of the TT submitter.

Data Type:RWCString

Privilege:Private

Default Value:

submitterImpact - Indicator of the impact of the problem reported in the TT as seen by the submitter.

Data Type:EcTInt

Privilege:Private

Default Value:

submitterName - the name of the submitter

Data Type:RWCString

Privilege:Private

Default Value:

submitterPhone - the phone number of the submitter

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

GetEntryId - This function is a simple get method for getting the unique entry ID of the TT.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

GetProblemLongDesc - This function is a simple get method for getting the long problem description of the TT.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

GetProblemShortDesc - This function is a simple get method for getting the short problem description of the TT.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

GetResolutionLog - This function is a simple get method for getting the resolution log of the TT.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

GetStatus - This function is a simple get method for getting the status of the TT.

Arguments:

Return Type:EcTInt

Privilege:Public

PDL: No PDL

GetSubmitterEmail - This function is a simple get method for getting the Email address of the TT submitter.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

GetSubmitterId - This function is a simple get method for getting the user ID of the TT submitter.

Arguments:

PDL: No PDL

GetSubmitterImpact - This function is a simple get method for getting the submitter impact of the TT.

Arguments:

Return Type:EcTInt

Privilege:Public

PDL: No PDL

GetSubmitterName - This function is a simple get method for getting the name of the TT submitter.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

GetSubmitterPhone - This function is a simple get method for getting the phone number of the TT submitter.

Arguments:

Return Type:const EctChar *

Privilege:Public

PDL: No PDL

MsTtEntry - This is the constructor for the class.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

SetEntryId - This function is a simple set method for setting the entry ID of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * newSubmitterId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

SetProblemLongDesc - This function is a simple set method for setting the problem long description of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * newProblemLongDesc

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

SetProblemShortDesc - This function is a simple set method for setting the problem short description of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * newProblemShortDesc

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

setStatus - This function is a simple set method for setting the status of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:EcTInt newStatus

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

setSubmitterEmail - This function is a simple set method for setting the submitter email of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * setSubmitterEmail

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

setSubmitterId - This function is a simple set method for setting the submitter id of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * newSubmitterId

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

setSubmitterImpact - This function is a simple set method for setting the submitter impact of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:EcTInt newSubmitterImpact

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

setSubmitterName - This function is a simple set method for setting the submitter name of the TT. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * newSubmitterName

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

setSubmitterPhone - This function is a simple set method for setting the phone number of the TT submitter. If the passed string exceeds the maximum length allowed, an exception is thrown.

Arguments:const EcTChar * newSubmitterPhone

Return Type:EcTVoid

Privilege:Public

PDL: No PDL

Associations:

The MsTtEntry class has associations with the following classes:

Class: MsTtHTMLItems createsandpopulates

Class: MsTtServiceRequestor submitstoremedyARS

6.9.3.7 MsTtEntryList Class

Parent Class:RWSlistCollectables

Public:No

Distributed Object:No

Purpose and Description:

This class represents a linked list of trouble ticket entries. All attributes and functions are inherited from its parent class. This class was created for possible future extensibility.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsTtEntryList class has associations with the following classes:

Class: MsTtHTMLItems displayonWebpage

Class: MsTtServiceRequestor populatefromRemedy

MsTtEntry (Aggregation)

6.9.3.8 MsTtHTMLItems Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

The MsTtHTMLItems class is a manager class which manages the HTML interface provided to the users to allow them to create or query the status of their trouble tickets.

Attributes:

None

Operations:

CheckNetscape - This routine checks to see if the user is using Netscape.

Arguments: CGI_Vars * cgiData

Return Type: EcTInt

Privilege: Private

PDL: No PDL

MsTtError - This routine displays a Web page to the user indicating an error which occurred during the processing.

Arguments: EcTChar * szErrMsg

Return Type: EcTinT

Privilege: Private

PDL: No PDL

MsTtListTroubleTicket - This routine retrieves the list of TTs submitted by the user via MsTtServiceRequetor and displays a Web page containing that list to the user.

Arguments: CGI_Vars CGIData, EcTChar * szUserId

Return Type: EcTInt

Privilege: Private

PDL: No PDL

MsTtReadConfig - This routine reads the configuration file.

Arguments: EcTChar * szServerUser, EcTChar * szMachName

Return Type: EcTInt

Privilege: Private

PDL: No PDL

MsTtShowDetailedTroubleTicket - This routine retrieves the TT, via MsTtServiceRequestor, that the user selected from the list and displays the TT to the user on a Web page.

Arguments: CGI_Vars CGIData, EcTChar * szUserId

Return Type: EcTInt

Privilege: Private

PDL: No PDL

MsTtShowSubmitPage - This routine displays the TT submit form to the user, filling in information about the user retrieved from their user profile.

Arguments: CGI_Vars * CGIData, EcTChar *szUserId

Return Type: EcTInt

Privilege: Private

PDL: No PDL

MsTtSubmitTroubleTicket - This routine takes the information from the TT submit page that the user entered, builds a TT object from the information, and submits the TT via MsTtServiceRequestor.

Arguments: CGI_Vars CGIData, EcTChar * szUserId

Return Type: EcTInt

Privilege: Private

PDL: No PDL

main - This routine receives HTML input from the user selecting options on the Web page and calls the appropriate routines to perform the actions.

Arguments:

Return Type: EcTInt

Privilege: Public

PDL: No PDL

Associations:

The MsTtHTMLItems class has associations with the following classes:

Class: MsAcUsrProfile Createsandreads

Class: MsAcUsrProfileMgr Requestuserprofilefrom

Class: TMPL_Element createsWebpageelements

Class: MsTtEntry createsandpopulates

Class: MsTtEntryList displayonWebpage

Class: TMPL_Vars displaysitemsonWebpage

Class: CGI_Vars providesHTMLinterface

Class: MsTtServiceRequestor provideswebinteface

6.9.3.9 MsTtHTMLMenu Class

Parent Class: Not Applicable

Public: No

Distributed Object: No

Purpose and Description:

This class manages the initial HTML Web page that presents the options that the user can perform on TTs.

Attributes:

None

Operations:

MsTtError - This routine displays an error message in a Web page to the user.

Arguments:EcTChar *szErrMsg

Return Type:EcTInt

Privilege:Private

PDL: No PDL

main - This routine verifies that the user is registered in ECS and displays the TT menu Web page. If the user is not registered or any error occurs, an error Web page is displayed.

Arguments:

Return Type:EcTInt

Privilege:Public

PDL: No PDL

Associations:

The MsTtHTMLMenu class has associations with the following classes:

Class: MsAcUsrProfile createsandreads

Class: TMPL_Vars displaysitemsonWebpage

Class: CGI_Vars providesHTMLinterface

Class: MsAcUsrProfileMgr requestuserprofilefrom

6.9.3.10 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 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 accessedby

Class: MsTtProxy manages

6.9.3.11 MsTtProxy Class

Parent Class:EcAgCOTSManger

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:

MsTtProxy - default constructor

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

Shutdown - This method will shutdown the MsTtManager (Remedy) software.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

Startup - This method will startup the MsTtManager (Remedy) software.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

~MsTtProxy - default destructor

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

Associations:

The MsTtProxy class has associations with the following classes:

Class: MsTtManager manages

6.9.3.12 MsTtServiceRequestor Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

The MsTtServiceRequestor class is responsible for processing requests from MsTtHTMLItems (the user interface) and fulfilling them using the functionality provided by the MsTtManager (Remedy).

Attributes:

control - Remedy AR Control Structure

Data Type:ARControlStruct

Privilege:Private

Default Value:

Operations:

MsTtServiceRequestor - This function serves as the constructor of the MsTtServiceRequestor class. It establishes a connection with the specified Remedy Action Request System server.

Arguments:EcTChar * userID, EcTChar * server

Return Type:Void

Privilege:Public

PDL: No PDL

PrintStatusList - This function is meant for development level debugging only. It will give a formatted printout of a linked list of Remedy status structures.

Arguments:requestARStatusList * statusList

Return Type:EcTVoid

Privilege:Private

PDL: No PDL

PrintStatusStruct - This function is meant for development level debugging only. It will give a formatted printout of a single Remedy status structure.

Arguments:requestARStatusStruct * statusStruct

Return Type:EcTVoid
Privilege:Private
PDL: No PDL

Retrieve - The function retrieves a single entry from the Remedy Action Request System based on the entry ID that is passed.

Arguments:const EcTChar * entryId, MsTtEntry * entry
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

RetrieveList - This function retrieves a linked list of entries for a particular user from the Remedy action Request System.

Arguments:const EcTChar * submitterId, MsTtEntryList * entryList
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

Submit - This function submits the passed entry on to the Remedy Action Request System.

Arguments:MsTtEntry * entry
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

SubmitEMail - This function builds a formatted e-mail message from the information in the TT and sends the message to the Remedy Action Request system via e-mail.

Arguments:MsTtEntry * entry
Return Type:EcTVoid
Privilege:Private
PDL: No PDL

Associations:

The MsTtServiceRequestor class has associations with the following classes:

Class: MsTtManager accessedby
Class: MsTtEntryList populatefromRemedy
Class: MsTtHTMLItems provideswebinteface
Class: CsEmMailRelA sendTTsubmite-mail
Class: MsTtEntry submitstoremedyARS

6.9.3.13 RWCollectable Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWCollectable class has associations with the following classes:

None

6.9.3.14 RWHashDictionary Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWHashDictionary class has associations with the following classes:

None

6.9.3.15 RWSlistCollectables Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWSlistCollectables class has associations with the following classes:

None

6.9.3.16 TMPL_Element Class

Parent Class:RWCollectable

Attributes:

prsName

Data Type:RWCString *

Privilege:Private

Default Value:

prsValue

Data Type:RWCString **

Privilege:Private

Default Value:

rsDirectValue

Data Type:RWCString

Privilege:Private

Default Value:

szValue

Data Type:char **

Privilege:Private

Default Value:

Operations:

TMPL_Element

Arguments:char * szName, char **szNewValue

Return Type:Void

Privilege:Public

TMPL_Element

Arguments:char * szName, RWCString **prsNewValue

Return Type:Void

Privilege:Public

TMPL_Element

Arguments:RWCString &rsNewName, char **szNewValue
Return Type:Void
Privilege:Public

TMPL_Element

Arguments:RWCString &rsNewName, RWCString **prsNewValue
Return Type:Void
Privilege:Public

TMPL_Element

Arguments:char *szName, RWCString *prsNewValue
Return Type:Void
Privilege:Public

TMPL_Element

Arguments:RWCString &rsNewName, RWCString *prsNewValue
Return Type:Void
Privilege:Public

TMPL_Element

Arguments:
Return Type:Void
Privilege:Public

name

Arguments:
Return Type:char *
Privilege:Public

value

Arguments:
Return Type:Char *
Privilege:Public

~TMPL_Element

Arguments:
Return Type:Void
Privilege:Public

Associations:

The TMPL_Element class has associations with the following classes:

Class: MsTtHTMLItems createsWebpageelements

Class: TMPL_Vars displaysonWebpage

6.9.3.17 TMPL_Vars Class

Parent Class:RWHashDictionary

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:

All Attributes inherited from parent class

Operations:

TMPL_Vars

Arguments:

Return Type:Void

Privilege:Public

get

Arguments:RWCString &rsName

Return Type:TMPL_Element *

Privilege:Private

insert

Arguments:RWCollectable * pElement

Return Type:RWCollectable *

Privilege:Public

process

Arguments:istream &stInput, ostream &stOutput, char *szMarker = '##'

Return Type:void

Privilege:Public

process

Arguments:char * szFileName, ostream &stOutput

Return Type:void

Privilege:Public

Associations:

The TMPL_Vars class has associations with the following classes:

Class: MsTtHTMLItems displaysitemsonWebpage

Class: MsTtHTMLMenu displaysitemsonWebpage

Class: TMPL_Element displaysonWebpage

6.9.4 Trouble Ticketing Dynamic Model

The following scenarios demonstrate typical TTS interaction as it relates to both users and support staff members.

6.9.4.1 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 6.9-3.

6.9.4.1.1 Beginning Assumptions

none

6.9.4.1.2 Interfaces with Other Subsystems and Segments

MsAcUsrProfile

MsAcProfileMgr

6.9.4.1.3 Stimulus

A user encounters a problem.

6.9.4.1.4 Participating Classes From the Object Model

MsTtHTMLItems

MsAcUsrProfile

MsAcProfileMgr

TMPL_Vars

MsTtServiceRequestor

MsTtManager

MsTtEntry

6.9.4.1.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is in normal operational state.

6.9.4.1.6 Ending State

A new trouble ticket is created. The MsTtManager is in normal operational state.

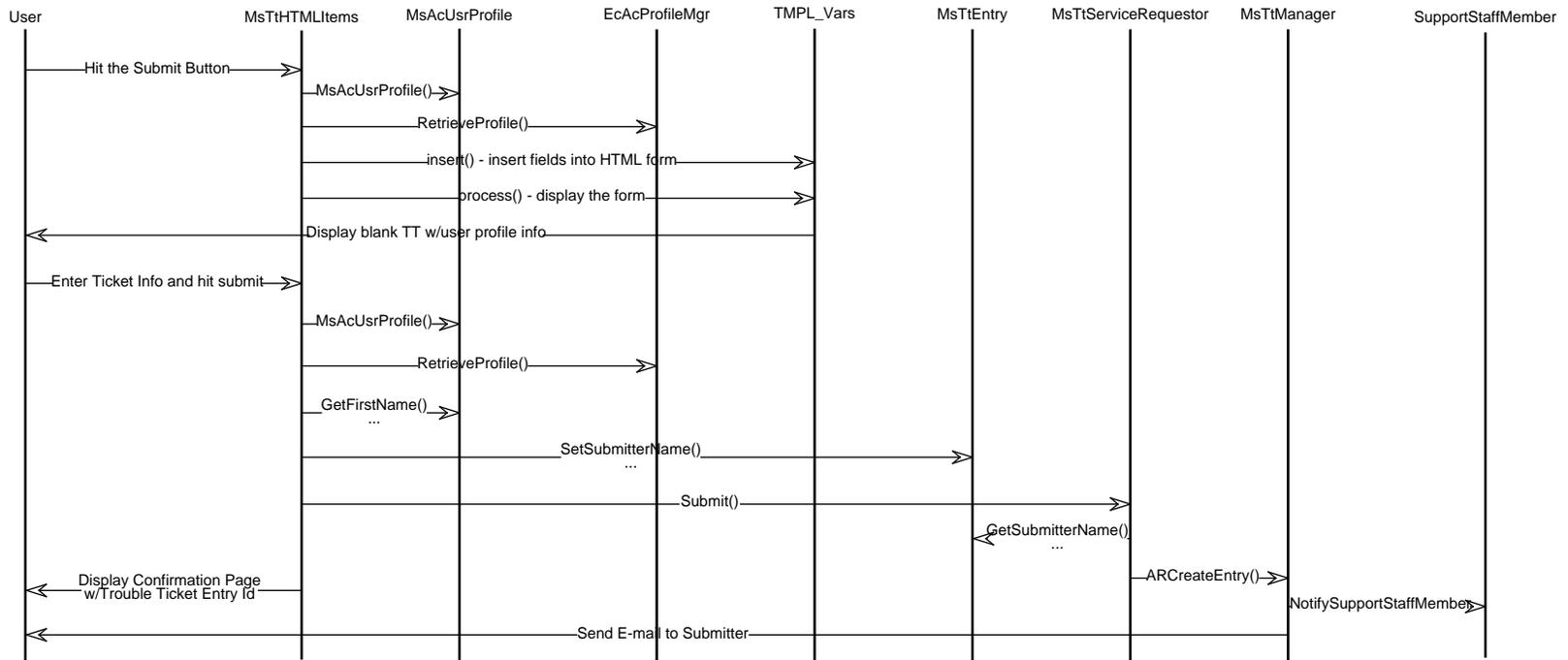


Figure 6.9-3. User Submits Trouble Ticket

6.9.4.1.7 Scenario Description

A user of the system encounters a problem. This problem may be with either hardware or software. The user then invokes the Web interface to the Trouble Tickets via the MsTtHTMLItems class. The user indicates that a new trouble ticket is to be created. The MsTtHTMLItems retrieves information about the user from the user profile database (MsAcUsrProfile, MsAcProfileMgr) and populates the Trouble Ticket form. MsTtHTMLItems displays the initialized 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 MsTtHTMLItems again retrieves the user profile information populates a MsttEntry object and submits the trouble ticket entry to MsTtServiceRequestor. Retrieving the user detail information, the MsTtServiceRequestor submits the Trouble Ticket 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 MsTtHTMLItems . The MsTtHTMLItems indicates a successful trouble ticket submission to user.

6.9.4.2 User Submits a Trouble Ticket When Remedy is Down

This scenario is much like the previous, except in this scenario, the Remedy Action Request System is not currently running. The scenario is shown in Figure 6.9-4.

6.9.4.2.1 Beginning Assumptions

none

6.9.4.2.2 Interfaces with Other Subsystems and Segments

MsAcUsrProfile

MsAcProfileMgr

6.9.4.2.3 Stimulus

A user encounters a problem.

6.9.4.2.4 Participating Classes From the Object Model

MsTtHTMLItems

MsAcUsrProfile

MsAcProfileMgr

TMPL_Vars

MsTtServiceRequestor

MsTtManager

MsTtEntry

CsEmMailRelA

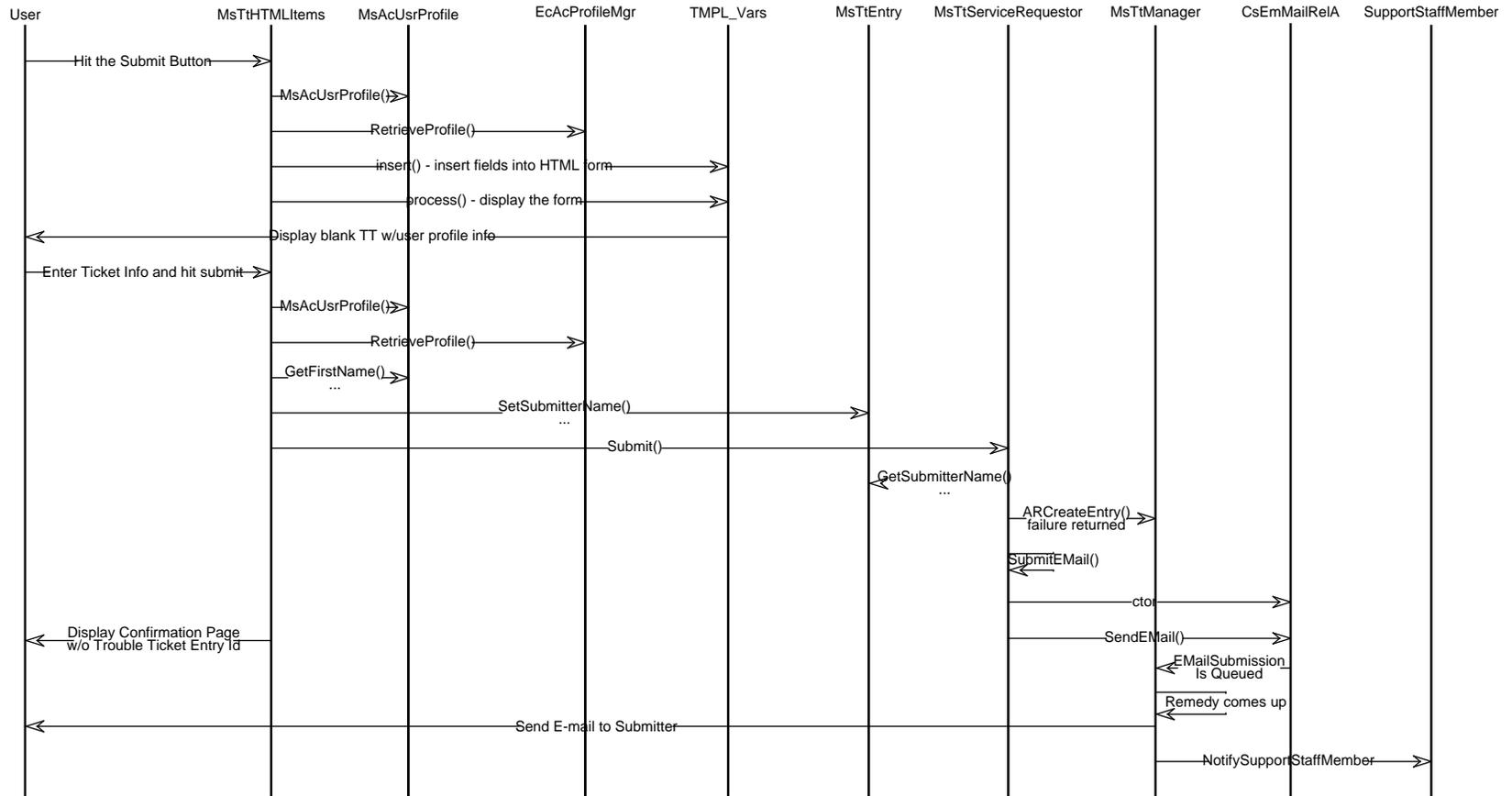


Figure 6.9-4. User Submits Trouble Ticket When Remedy is Down

6.9.4.2.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is not operational.

6.9.4.2.6 Ending State

A new trouble ticket is created. The MsTtManager is back in normal operational state.

6.9.4.2.7 Scenario Description

The following portion is just the same as the above scenario:

A user of the system encounters a problem. This problem may be with either hardware or software. The user then invokes the Web interface to the Trouble Tickets via the MsTtHTMLItems class. The user indicates that a new trouble ticket is to be created. The MsTtHTMLItems retrieves information about the user from the user profile database (MsAcUsrProfile, MsAcProfileMgr) and populates the Trouble Ticket form. MsTtHTMLItems displays the initialized 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 MsTtHTMLItems again retrieves the user profile information populates a MsTtEntry object and submits the trouble ticket entry to MsTtServiceRequestor. Retrieving the user detail information, the MsTtServiceRequestor submits the Trouble Ticket to the MsTtManager for entry into the Remedy System.

Here is where the scenario changes from the previous:

The MsTtServiceRequestor receives a failure indication when trying to access MsTtManager. MsTtServiceRequestor creates a CsEmMailRelA object and uses the object to create a formatted e-mail message addressed to the MsTtManager (Remedy ARS). MsTtServiceRequestor tells CsEmMailRelA to send the e-mail. A confirmation page is displayed to the user indicating that the Trouble Ticketing system is down, but the new Trouble Ticket will be submitted as soon as it becomes available.

At some later time, MsTtManager (Remedy) comes back up. Remedy will read its incoming mail and create a Trouble Ticket with a unique entry id. The MsTtManger notifies the originating user and the support staff member responsible for assigning new trouble tickets of the new trouble ticket.

6.9.4.3 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 6.9-5.

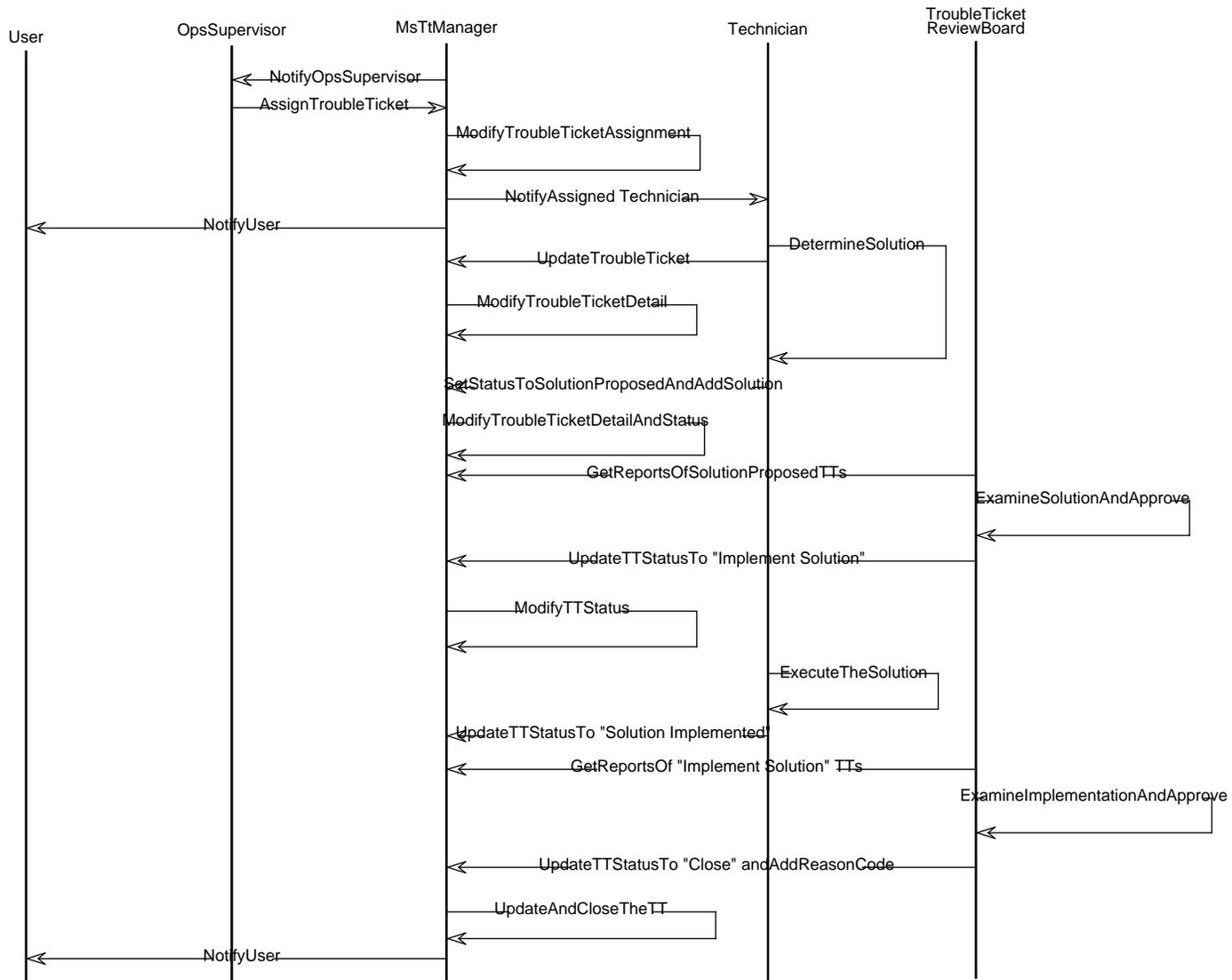


Figure 6.9-5. A Trouble Ticket is Worked

6.9.4.3.1 Beginning Assumptions

none

6.9.4.3.2 Interfaces with Other Subsystems and Segments

none

6.9.4.3.3 Stimulus

A new trouble ticket enters the system.

6.9.4.3.4 Participating Classes From the Object Model

MsTtManager

6.9.4.3.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is in normal operational state.

6.9.4.3.6 Ending State

A trouble ticket is closed. The MsTtManager is in normal operational state.

6.9.4.3.7 Scenario Description

The support staff member responsible for assigning trouble tickets (Ops Supervisor) receives notification that a new trouble ticket has entered the system. On examining the detail information, the support staff member assigns the trouble ticket to a Technician. The trouble ticket status is then updated to reflect the assignment and the assignee is notified (via e-mail). Additionally (configurable at each site), the submitter of the trouble ticket is sent an e-mail indicating that their trouble ticket has been assigned.

Assessing the trouble ticket, the Technician forms a plan to resolve the issue. Adding this information to the trouble ticket, it is then updated.

When the issue has been resolved, the assignee updates the trouble ticket with any subsequent information regarding the solution and changes the status to "Solution Proposed." The Trouble Ticket Review Board will examine the solution and its potential impact on the system. If the solution is approved, the trouble ticket status is changed to "Implement Solution." The assigned Technician executes the solution and sets the trouble ticket status to "Solution Implemented." The Review Board examines the implemented solution and after finding it acceptable, sets the status to "Closed" while assigning 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.

6.9.4.4 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 6.9-6.

6.9.4.4.1 Beginning Assumptions

none

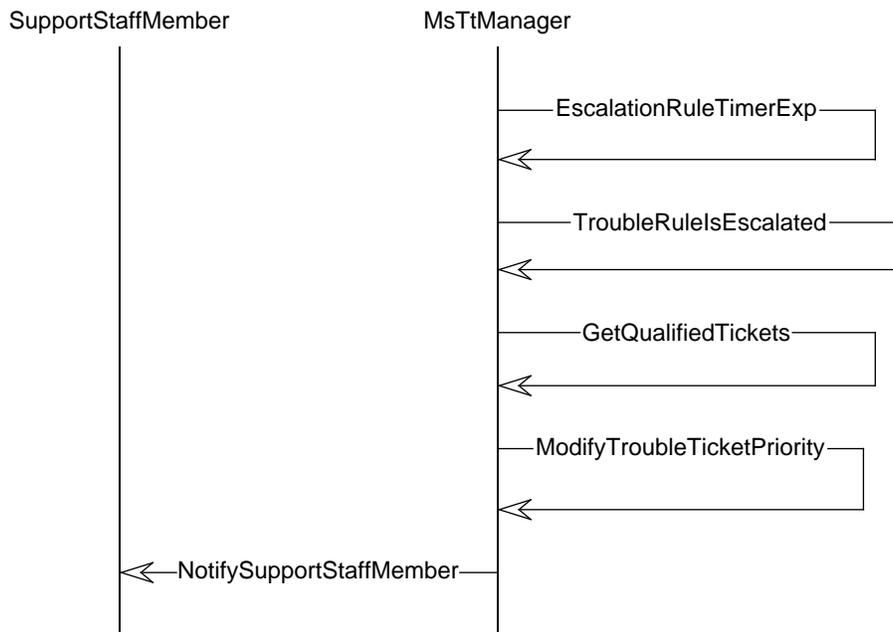


Figure 6.9-6. A Trouble Ticket is Escalated

6.9.4.4.2 Interfaces with Other Subsystems and Segments

none

6.9.4.4.3 Stimulus

A defined escalation rule is hit for a trouble ticket.

6.9.4.4.4 Participating Classes From the Object Model

MsTtManager

6.9.4.4.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager is in normal operational state.

6.9.4.4.6 Ending State

A trouble ticket is escalated. The MsTtManager is in normal operational state.

6.9.4.4.7 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.

6.9.4.5 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 6.9-7.

6.9.4.5.1 Beginning Assumptions

none

6.9.4.5.2 Interfaces with Other Subsystems and Segments

none

6.9.4.5.3 Stimulus

A user reports a problem to a DAAC via a trouble ticket.

6.9.4.5.4 Participating Classes From the Object Model

MsTtManager

6.9.4.5.5 Beginning System, Segment and Subsystem State(s)

The MsTtManager at both the DAAC and the SMC is normal operational state.

6.9.4.5.6 Ending State

The user is notified of closure of the trouble ticket. The MsTtManager is in normal operational state.

6.9.4.5.7 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.

At 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.

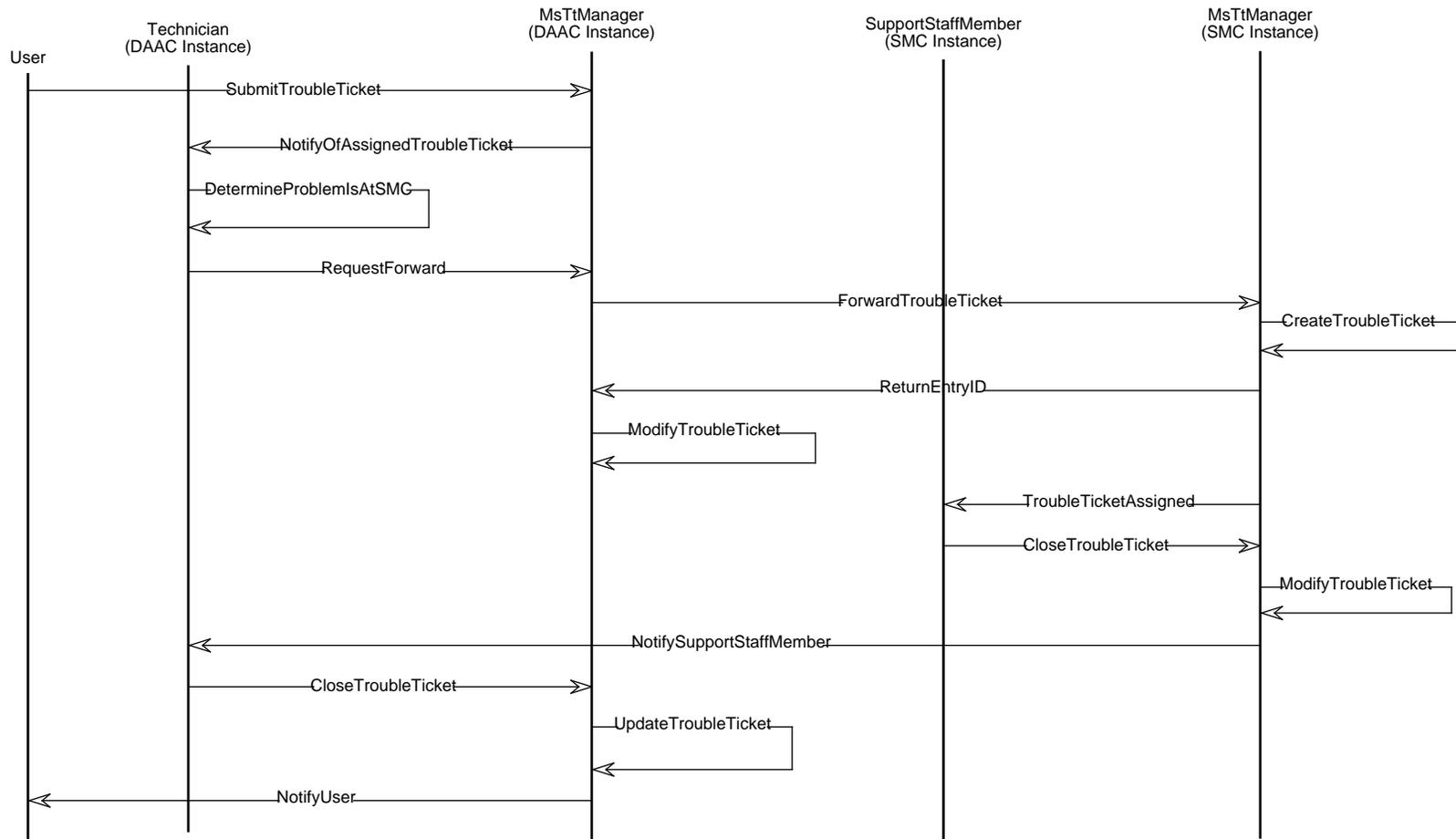


Figure 6.9-7. A Trouble Ticket is Forwarded

6.9.5 Trouble Ticketing Structure

Table 6.9-1. Trouble Ticketing Components

Component Name	COTS/Custom
Trouble Ticketing Management Services	COTS (Remedy Action Request System)
Trouble Ticketing HTML Menu	Custom
Trouble Ticketing HTML Submission/List	Custom
Trouble Ticketing Proxy Agent	Custom

6.9.5.1 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.

6.9.5.2 Trouble Ticketing HTML Menu CSC

Purpose and Description

This CSC provides an HTML Web Page to registered ECS users that lists the actions which can be performed on the Trouble Ticketing system through the Web interface. These actions include submitting a new Trouble Ticket and listing trouble tickets previously opened by the user.

6.9.5.3 Trouble Ticketing HTML Submission/List CSC

Purpose and Description

This CSC provides the interface between the users (those who submit trouble tickets) and the MsTtManger. 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. In addition, this CSC provides the interface between the MsTtManager (Remedy) and the MsTtHTMLItems. It translates requests from the HTTPD server and fulfills them using the MsTtManager API functions.

6.9.5.4 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.

6.9.6 Trouble Ticketing Management and Operation

6.9.6.1 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.

6.9.6.2 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.

MsTtHTMLMenu/MsTtHTMLItems - This set of HTML documents provide the primary interface for users to submit and query trouble tickets.

6.9.6.3 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.

6.10 Management Data Access

6.10.1 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, by one of three means. The transfer will occur as established by a predefined and configurable schedule (time interval or absolute time), by a log size threshold exceeded event, or on command by the M&O staff. The management log files for each managed host maintain a common collection of log file records without regard to the mode of the event. Log file records are transferred to the appropriate mode-identified management database (established upon inception of the mode), based on the event's mode identifier. The MsMdProcessEvent class handles the filtering of log file records, ensuring that they are transferred to the appropriate management

database. Non-mode specific log file records will be logged to all mode specific management databases. The log files are processed and retained for a period of time before being transferred by MDA to the ECS Data Archives. 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. Additionally, an event chaining capability is provided to enable the user to select an event and retrieve a list of its ancestor events(transactions). Each event contains a TransactionID and a ParentTransactionID field to enable a parent-child transaction linking. Event chaining will link the specified event to its ancestor transaction events for the time period specified. 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.

6.10.2 Management Data Access Context

The Management Data Access context diagram is shown in Figure 6.10-1. 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.

6.10.3 Management Data Access Object Model

The Object Model for the Management Data Access Service is shown in Figure 6.10-2.

6.10.3.1 CsFtFTPReIA Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class is imported from CSS

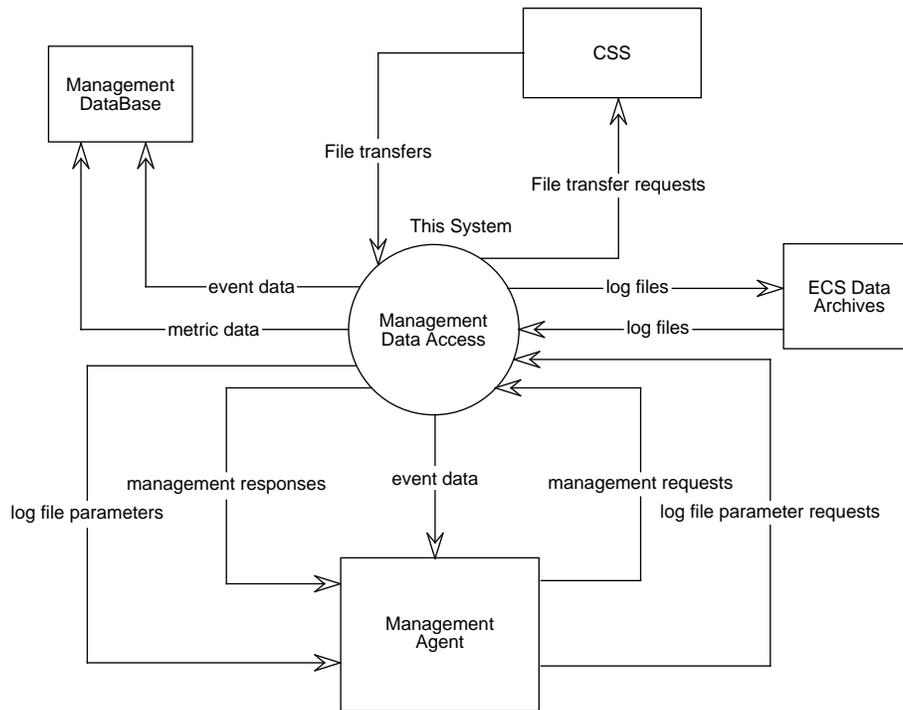


Figure 6.10-1. Management Data Access Context Diagram

Attributes:

None

Operations:

None

Associations:

The CsFtFTPReIA class has associations with the following classes:

Class: MsMdManager uses

6.10.3.2 EcAgEvent Class

Parent Class:Not Applicable

Public:Yes

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

6.10.3.3 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: MsMdProcessEvent isloadedby

6.10.3.4 MsMdAggregateLogEntry 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 processed for the Management RDBMS and archived.

Attributes:

myCompression - This attribute represents the flag to indicate if files have been compressed prior to archival.

Data Type:RWCString

Privilege:Private

Default Value:

myDirectoryId - This attribute represents the path and address of the directory where the files are located.

Data Type:RWCString

Privilege:Private

Default Value:

myEndTime - This attribute represents the end of the time period covered by the aggregate.

Data Type:RWTime

Privilege:Private

Default Value:

myFileId - This attribute represents the name of the files that are to archived (repeat for each file to be archived).

Data Type:RWCString

Privilege:Private

Default Value:

myFileSize - This attribute represents the size of the individual file to be archived, in bytes.
Data Type:RWInteger
Privilege:Private
Default Value:

myFileTimeStamp - This attribute represents the date and time the file was created.
Data Type:RWDate
Privilege:Private
Default Value:

myStartTime - This attribute represents the starting time of the time period covered by the aggregate.
Data Type:RWTime
Privilege:Private
Default Value:

myTotalFileCount - This attribute represents the total number of files to be archived.
Data Type:RWInteger
Privilege:Private
Default Value:

myTotalFileSize - This attribute represents the sum of the file sizes in bytes.
Data Type:RWInteger
Privilege:Private
Default Value:

Operations:

GetCompression - This operation returns whether the file is compressed or not.
Arguments:
Return Type:RWCString
Privilege:Public

GetDirectoryId - This operation returns the string value of the directory ID.
Arguments:
Return Type:RWCString
Privilege:Public

GetEndTime - This operation returns the ending time.
Arguments:
Return Type:RWTime
Privilege:Public

GetFileId - This operation returns the value of the file ID.

Arguments:

Return Type:RWCString

Privilege:Public

GetFileSize - This operation returns the size of the file.

Arguments:

Return Type:RWInteger

Privilege:Public

GetFileTimeStamp - This operation returns the file time stamp.

Arguments:

Return Type:RWDate

Privilege:Public

GetStartTime - This operation returns the starting time.

Arguments:

Return Type:RWTime

Privilege:Public

MsMdAggregateLogEntry - This operation serves as the copy constructor for the MsMdAggregateLogEntry.

Arguments:MsMdAggregateLogEntry&

Return Type:Void

Privilege:Public

MsMdAggregateLogEntry - This operation serves as the default constructor for the MsMdAggregateLogEntry class.

Arguments:

Return Type:Void

Privilege:Public

MsMdAggregateLogEntry - This operation is a set constructor.

Arguments:RWTime,RWTime,RWCString, RWCString, RWCString, RWInteger, RWDate

Return Type:Void

Privilege:Public

MsMdAggregateLogEntry - This operation serves as a constructor which accepts start and end time parameters.

Arguments:RWTime, RWTime

Return Type:Void

Privilege:Public

binaryStoreSize - This operation returns the number of bytes used by the virtual operation

saveGuts(RWFile&) to store an object.

Arguments:

Return Type:RWspace

Privilege:Public

isEqual - This operation returns TRUE if the collectable object matches the starting time.

Arguments:RWCollectable*

Return Type:RWBoolean

Privilege:Public

restoreGuts - This operation reads an object's state from a binary file, using class RWFile, replacing the previous state.

Arguments:RWFile&

Return Type:EcTVoid

Privilege:Public

restoreGuts - This operation reads an object's state from a virtual stream using class RWFile, replacing the previous state.

Arguments:RWvistream&

Return Type:EcTVoid

Privilege:Public

saveGuts - This operation writes an object's state to a binary file, using class RWFile.

Arguments:RWFile&

Return Type:EcTVoid

Privilege:Public

saveGuts - This operation writes an object's state to a virtual stream using class RWFile.

Arguments:RWvostream&

Return Type:EcTVoid

Privilege:Public

~MsMdAggregateLogEntry - This operation serves as the default destructor for the MsMdAggregateLogEntry class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdAggregateLogEntry class has associations with the following classes:

MsMdAggregateLogFileList (Aggregation)

6.10.3.5 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:

GetHostList - This operation gets the host log.

Arguments:RWCString&

Return Type:RWCollectable*

Privilege:Public

GetLogEntry - This operation gets the log entry.

Arguments:RWCString&, RWTime, RWTime

Return Type:MsMdAggregateLogEntry*

Privilege:Public

InsertHostList - This operation inserts the new log entry for the new host.

Arguments:RWCString&, MsMdAggregateLogEntry*

Return Type:EcTVoid

Privilege:Public

MsMdAggregateLogFileList

Arguments:

Return Type:Void

Privilege:Public

RemoveLogEntry - This operation removes the log entry from the found host log.

Arguments:RWCString&, MsMdAggregateLogEntry*

Return Type:EcTVoid

Privilege:Public

UpdateHostList - This operation updates the log entry by adding a log entry to the found host log.

Arguments:RWCString&, MsMdAggregateLogEntry*

Return Type:EcTVoid

Privilege:Public

~MsMdAggregateLogFileList

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdAggregateLogFileList class has associations with the following classes:

Class: MsMdManager manages

6.10.3.6 MsMdArchiveLog Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The MsMdArchiveLog class has associations with the following classes:

Class: MsMdProcessEvent uses

6.10.3.7 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:

myAbsoluteTimeList - the list of absolute time at which the log file is to be transferred to the MSS Server.

Data Type:RWSortedVector

Privilege:Private

Default Value:

myHostName - the name of the host to which this entry applies

Data Type:RWCString

Privilege:Private

Default Value:

myInterval - the interval in minutes the log file is to be transferred

Data Type:RWInteger

Privilege:Private

Default Value:

mySizeLimit - the size threshold at which the log file is to be transferred

Data Type:RWInteger

Privilege:Private

Default Value:

Operations:

GetAbsoluteTimeList - This operation returns the absoluteTimeList.

Arguments:

Return Type:RWSortedVector

Privilege:Public

GetHostName - This operation returns the hostName.

Arguments:

Return Type:RWCString

Privilege:Public

GetInterval - This operation returns the interval.

Arguments:

Return Type:RWInteger

Privilege:Public

GetSizeLimit - This operation returns the sizeLimit.

Arguments:

Return Type:RWInteger

Privilege:Public

GetSortBy

Arguments:

Return Type:Void

Privilege:Public

GetSortOrder

Arguments:

Return Type:Void
Privilege:Public

GetTimeStamp

Arguments:
Return Type:Void
Privilege:Public

MsMdConfigurationEntry - constructor with only host name defined.

Arguments:RWCString&
Return Type:Void
Privilege:Public

MsMdConfigurationEntry - copy constructor.

Arguments:MsMdConfigurationEntry&
Return Type:Void
Privilege:Public

MsMdConfigurationEntry - default constructor.

Arguments:
Return Type:Void
Privilege:Public

ProcessEvent

Arguments:
Return Type:Void
Privilege:Public

binaryStoreSize - This operation returns the number of bytes used by the virtual function saveGuts(RWFile&) to store an object.

Arguments:
Return Type:RWSpace
Privilege:Public

compareTo

Arguments:RWCollectable*
Return Type:Void
Privilege:Public

isEqual - This operation will return TRUE if collectable object "matches" given objects.

Arguments:RWCollectable*
Return Type:RWBoolean
Privilege:Public

restoreGuts - This operation reads an object's state from a binary file, using class RWFile,

replacing the previous state.

Arguments:RWFile&
Return Type:EcTVoid
Privilege:Public

restoreGuts - his operation reads an object's state from a virtual stream.

Arguments:RWvistream&
Return Type:EcTVoid
Privilege:Public

saveGuts - This operation writes an object's state to a binary file, using class RWFile.

Arguments:RWFile&
Return Type:EcTVoid
Privilege:Public

saveGuts - This operation writes an object's state to a virtual stream.

Arguments:RWvostream&
Return Type:EcTVoid
Privilege:Public

~MsMdConfigurationEntry - destructor.

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsMdConfigurationEntry class has associations with the following classes:

MsMdConfigurationList (Aggregation)

6.10.3.8 MsMdConfigurationList Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents the configuration of the MsMdManager.

Attributes:

myArchiveInterval - the interval when the log files are to be transferred to the Management Database.

Data Type:RWInteger
Privilege:Public

Default Value:

myArchiveLoadTime - the time of day the aggregate log files should be processed and load to ECS Data Archive.

Data Type:RWSortedVector

Privilege:Public

Default Value:

myDBInterval - the interval when the log files are to be transferred.

Data Type:RWInteger

Privilege:Public

Default Value:

myDBLoadTime

Data Type:RWSortedVector

Privilege:Public

Default Value:

myMssServerSize - the size limit of the MSS Server.

Data Type:RWInteger

Privilege:Public

Default Value:

Operations:

GetEntry - This operation will get a MsMdConfigurationEntry for a given host name.

Arguments:RWCSrintg&

Return Type:RWCollectable*

Privilege:Public

InsertEntry - This operation will add a MsMdConfigurationEntry to the current configuration list.

Arguments:MsMdConfigurationEntry*

Return Type:EcTVoid

Privilege:Public

MsMdConfigurationList - default constructor.

Arguments:

Return Type:Void

Privilege:Public

RemoveEntry - This operation will remove a MsMdConfigurationEntry from the current configuration list.

Arguments:RWCSrintg&

Return Type:EcTVoid
Privilege:Public

UpdateEntry - This operation will change a MsMdConfigurationEntry to the current configuration.

Arguments:MsMdConfigurationEntry*
Return Type:EcTVoid
Privilege:Public

binaryStoreSize - This operation returns the number of bytes used by the virtual function saveGuts(RWFile&) to store an object.

Arguments:
Return Type:RWspace
Privilege:Public

restoreGuts - This operation reads an object's state from a binary file, using class RWFile, replacing the previous state.

Arguments:RWFile&
Return Type:EcTVoid
Privilege:Public

restoreGuts - This operation reads an object's state from a virtual stream.

Arguments:RWvistream&
Return Type:EcTVoid
Privilege:Public

saveGuts - This operation writes an object's state to a binary file, using class RWFile.

Arguments:RWFile&
Return Type:EcTVoid
Privilege:Public

saveGuts - This operation writes an object's state to a virtual stream.

Arguments:RWvostream&
Return Type:EcTVoid
Privilege:Public

~MsMdConfigurationList - destructor.

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsMdConfigurationList class has associations with the following classes:

Class: MsMdManager manages

6.10.3.9 MsMdEventField Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents a description of an event. It also recognizes the different types of events and their filtered conditions.

Attributes:

myAppID - This attribute represents the application ID.

Data Type:EcTInt

Privilege:Private

Default Value:

myCategory - This attribute represents the category of the event.

Data Type:EcTInt

Privilege:Private

Default Value:

myCsci - This attribute represents the CSCI of the event.

Data Type:EcTInt

Privilege:Private

Default Value:

myEndTime - This attribute represents the end time.

Data Type:RWTime

Privilege:Private

Default Value:

myEventType - This attribute is an enumerated type for events.

Data Type:MsTAgEventDelimiter

Privilege:Private

Default Value:

myGenEvent - This attribute defines a general event.

Data Type:EcAgEvent

Privilege:Private

Default Value:

myMode - This attribute represents the mode of the event.

Data Type:RWCString
Privilege:Private
Default Value:

myPerfEvent - This attribute represents a performance event.

Data Type:MsAgPerfEvent
Privilege:Private
Default Value:

myProcID - This attribute represents the process ID.

Data Type:EcTInt
Privilege:Private
Default Value:

myProgID - This attribute represents the program ID.

Data Type:EcTInt
Privilege:Private
Default Value:

mySeverity - This attribute represents the event severity level.

Data Type:EcTInt
Privilege:Private
Default Value:

mySortBy - This attribute represents the value decides sorting.

Data Type:EcTInt
Privilege:Private
Default Value:

mySortOrder - This attribute represents a value that determines sorting order.

Data Type:EcTInt
Privilege:Private
Default Value:

myStartTime - This attribute represents the start time.

Data Type:RWTime
Privilege:Private
Default Value:

mySubSys - This attribute represents the subsystem of the event.

Data Type:EcTInt
Privilege:Private
Default Value:

myTimeStamp - This attribute represents the time stamp of the event.

Data Type:RWTime
Privilege:Private
Default Value:

myType - This attribute represents the type of the event.

Data Type:EcTInt
Privilege:Private
Default Value:

Operations:

CompareToEvent - This operation compares two event attributes and sorts them.

Arguments:EcTInt&, EcTInt&, EcTInt&
Return Type:EcTInt
Privilege:Public

DisplayEvent - This operation displays an event.

Arguments:
Return Type:EcTVoid
Privilege:Public

GetEventType - This operation returns the event type.

Arguments:
Return Type:MsTAgEventDelimiter&
Privilege:Public

GetGenEvent - This operation returns the general event.

Arguments:
Return Type:EcAgEvent&
Privilege:Public

GetPerfEvent - This operation returns the performance event.

Arguments:
Return Type:MsAgPerfEvent&
Privilege:Public

GetSortBy - This operation returns the sort by field specified.

Arguments:
Return Type:EcTInt&
Privilege:Public

GetSortOrder - The operation returns the sort order.

Arguments:
Return Type:EcTInt&

Privilege:Public

GetTimeStamp - This operation returns the time stamp.

Arguments:

Return Type:RWTime&

Privilege:Public

MatchEvent - This operation matches an event.

Arguments:MsMdEventField&

Return Type:RWBoolean

Privilege:Public

MsMdEventField - This operation serves as the default constructor for the MsMdEventField

Arguments:

Return Type:Void

Privilege:Public

MsMdEventField - This operation serves as the copy constructor for the MsMdEventField class.

Arguments:MsMdEventField&

Return Type:Void

Privilege:Public

MsMdEventField - This operation serves as the general event constructor for MsMdEventField class.

Arguments:MsTAgEventDelimiter&, EcAgEvent&

Return Type:Void

Privilege:Public

MsMdEventField - This operation serves as the performance event constructor.

Arguments:MsTAgEventDelimiter&, MsAgPerfEvent&

Return Type:Void

Privilege:Public

ParentChildMatch - This operation compares the transactionID of the current event to the parentTransactioID of the event to be chained. A boolean value is returned to indicate whether or not a match exists.

Arguments:MsMdEventField&

Return Type:RWBoolean

Privilege:Public

SetSortCondition - This operation sets the sorting condition.

Arguments:EcTInt, EcTInt

Return Type:EcTVoid

Privilege:Public

SetTimeRange - This operation sets the start and stop time.

Arguments:RWTime&, RWTime&

Return Type:EcTVoid

Privilege:Public

compareTo - This operation compares two events and sorts them.

Arguments:RWCollectable*

Return Type:EcTInt

Privilege:Public

isEqual - This operation returns TRUE if collectable object "matches" the given objects.

Arguments:RWCollectable*

Return Type:EcTInt

Privilege:Public

isEqualEvent - This operation returns true if event matches given objects.

Arguments:MsMdEventField*, EcTInt

Return Type:RWBoolean

Privilege:Public

~MsMdEventField - this operation serves as the default destructor for the MsMdEventField class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdEventField class has associations with the following classes:

MsMdEventList (Aggregation)

6.10.3.10 MsMdEventList Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This operation represents the collection of conditional events for metrics that are to be loaded to the ManagementRDBMS.

Attributes:

None

Operations:

Insert - This operation inserts an event into an event list.

Arguments:MsMdEventField*

Return Type:EcTVoid

Privilege:Public

MsMdEventList - This operation serves as the default constructor for the MsMdEventList class.

Arguments:

Return Type:Void

Privilege:Public

~MsMdEventList - This operation serves as the default destructor for the MsMdEventList class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdEventList class has associations with the following classes:

Class: MsMdLogBrowser

Class: MsMdProcessEvent

Class: MsMdManager manages

6.10.3.11 MsMdLogBrowser Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents a collection of conditional events for metrics that are to be loaded to the ManagementRDBMS.

Attributes:

myCondition - This attribute is the filter condition for an event.

Data Type:MsMdEventField

Privilege:Private

Default Value:

myCurrentList - This attribute represents the top of the stack.

Data Type:RWSlistCollectables

Privilege:Private

Default Value:

myEndTime - This attribute represents the end time of the data being browsed.

Data Type:RWTime

Privilege:Private

Default Value:

myEventCount - This attribute represents the total number of events.

Data Type:EcTInt

Privilege:Private

Default Value:

myHostInfo - This attribute contains information about the host.

Data Type:EcAgHostInfo

Privilege:Private

Default Value:

myMode - This attribute represents the event's mode. It is used for filtering or sorting.

Data Type:RWCString

Privilege:Private

Default Value:

myOriginalList - This attribute represents the original event list.

Data Type:RWSlistCollectables

Privilege:Private

Default Value:

myParentTransactionID - This attribute represents the ParentTransactionID of the event which is being chained.

Data Type:RWCString

Privilege:Private

Default Value:

myPreviousList - This attribute represents the previous event list.

Data Type:RWSlistCollectables

Privilege:Private

Default Value:

mySortByList - This attribute defines the event field to be sorted.

Data Type:RWSlistCollectables

Privilege:Private

Default Value:

mySortOrderList

Data Type:RWSlistCollectables

Privilege:Private

Default Value:

myStartTime - This attribute is the start time of the data being browsed.

Data Type:RWTime

Privilege:Private

Default Value:

Operations:

Chain - This operation will traverse a list of events to determine if a parent-child relationship exists. If a match exists, the event is inserted into a list of chained events, and a retraversal occurs to seek additional ancestral transaction events. When no parent-child match is found, the operation halts.

Arguments:

Return Type:EcTVoid

Privilege:Public

Filter - This operation filters the event list based on user defined criteria.

Arguments:

Return Type:EcTVoid

Privilege:Public

GetCondition - This operation returns the event condition.

Arguments:

Return Type:MsMdEventField&

Privilege:Public

GetCurrentList - This operation returns the top of a stack.

Arguments:

Return Type:RWSlistCollectables&

Privilege:Public

GetEndTime - This operation returns the end time.

Arguments:

Return Type:RWTime&

Privilege:Public

GetEventCount - This operation returns the number of events in the list.

Arguments:

Return Type:EcTInt&

Privilege:Public

GetHostInfo - This operation returns the host information.

Arguments:

Return Type:EcAgHostInfo&

Privilege:Public

GetHostName - This operation returns the host name.

Arguments:

Return Type:RWCString&

Privilege:Public

GetMode - This operation will return the events of a given mode.

Arguments:RWSlistCollectables, RWCString

Return Type:EcTVoid

Privilege:Public

GetStartTime - This operation returns the start time.

Arguments:

Return Type:RWTime&

Privilege:Public

MsMdLogBrowser - This operation serves as the constructor for the MsMdLogBrowser class.

Arguments:

Return Type:Void

Privilege:Public

MsMdLogBrowser - This operation serves as the copy constructor for the MsMdLogBrowser class.

Arguments:MsMdLogBrowser&

Return Type:Void

Privilege:Public

SetEventList - This operation returns an event list.

Arguments:

Return Type:EcTVoid

Privilege:Public

SetFilter - This operation sets filtering attributes.

Arguments:MsMdEventField&

Return Type:EcTVoid

Privilege:Public

SetSort - This operation sets sorting attributes.

Arguments:RWSlistCollectables, RWSlistCollectables
Return Type:EcTVoid
Privilege:Public

SetTime - This operation sets the starting time attribute.

Arguments:MsMdEventField*, RWSlistCollectables
Return Type:EcTVoid
Privilege:Public

Sort - This operation sorts the event list.

Arguments:RWSlistCollectables, RWSlistCollectables, MsMdEventList
Return Type:EcTVoid
Privilege:Public

SortList

Arguments:
Return Type:EcTVoid
Privilege:Public

pHostInfo - This operation returns host information such as cell name, hostname, IP address, OS name, OS major version, OS minor version, Os rev, and OS maint level.

Arguments:
Return Type:EcTVoid
Privilege:Public

pList - This operation displays the events in a list.

Arguments:MsMdEventList*
Return Type:EcTVoid
Privilege:Public

~MsMdLogBrowser - this operation serves as the default destructor for the MsMdLogBrowser class.

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsMdLogBrowser class has associations with the following classes:

Class: MsMdEventList

Class: MsMdUserInterface

6.10.3.12 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:

FTPLogFile

Arguments:MsMdAggregateLogFileList*, CsFtFTPReIA*, RWCString&, RWCString&, RWCString&, RWCString

Return Type:Void

Privilege:Public

FindNextLoadCall

Arguments:MsMdScheduleList*, MsMdConfigurationList*, MsMdAggregatedLogFileList*

Return Type:EcTVoid

Privilege:Public

GetLogFileData

Arguments:RWCString, RWTime, RWTime, RWCString

Return Type:Void

Privilege:Public

GetLogFileMaxSize

Arguments:RWCString

Return Type:Void

Privilege:Public

MsMdManager - default constructor.

Arguments:

Return Type:Void

Privilege:Public

ProcessLogFile

Arguments:RWCString, RWCString

Return Type:Void
Privilege:Public

TransferLogFile - This operaton transfers all the log files from a managed host to the MSS Server at a given pacific time, appending it to the correct MsMdAggregateLogFile.

Arguments:MsMdScheduleList*, MsMdConfigurationList*,
MsMdAggregatedLogFileList*
Return Type:EcTVoid
Privilege:Public

~MsMdManager

Arguments:
Return Type:Void
Privilege:Public

Associations:

The MsMdManager class has associations with the following classes:

Class: MsMdProcessEvent
Class: MsMdUserInterface accesses
Class: EcAgEvent communicateswith
Class: MsMdAggregateLogFileList manages
Class: MsMdConfigurationList manages
Class: MsMdEventList manages
Class: MsMdSchedule manages
Class: CsFtFTPRelA uses

6.10.3.13 MsMdProcessEvent Class

Parent Class:Not Applicable

Attributes:

myDBs

Data Type:RWCString&
Privilege:Private
Default Value:

myModes

Data Type:RWCString&
Privilege:Private
Default Value:

Operations:

ArchiveLogFile

Arguments:RWCString, RWTime, RWTime, RWCString

GetActiveModes

Arguments:

Return Type:RWString*

Privilege:Public

OpenMgmtDBs

Arguments:RWCString&, RWCString&

Return Type:EcTVoid

Privilege:Public

ProcessEvent

Arguments:EcAgEvent&

Return Type:EcTVoid

Privilege:Public

RetrieveArchivedLogFile

Arguments:RWCString, RWTime, RWTime, RWCString

Associations:

The MsMdProcessEvent class has associations with the following classes:

Class: MsMdEventList

Class: MsMdManager

Class: ManagementRDBMS isloadedby

Class: MsMdArchiveLog uses

6.10.3.14 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:

myNextArchiveLoadTime - the list of time at which the log file is to be transferred to the ECS Data Archive.

Data Type:RWCollectableTime

Privilege:Private
Default Value:

myNextDBLoadTime - the list of time at which the log file is to be transferred to the Management Data Base.

Data Type:RWCollectableTime
Privilege:Private
Default Value:

myNextEntryLoadTime - the list of time at which the log file is to be transferred to the MSS Server.

Data Type:RWCollectableTime
Privilege:Private
Default Value:

Operations:

GetNextArchiveLoadTime - This operation will update a scheduled time of loading to the ECS Data Archive.

Arguments:
Return Type:RWCollectableTime
Privilege:Public

GetNextDBLoadTime - This operation will update a scheduled time of loading to the Management Data Base.

Arguments:
Return Type:RWCollectableTime
Privilege:Public

GetNextEntryLoadTime - This operation will return a scheduled time of loading to the MSS Server.

Arguments:
Return Type:RWCollectableTime
Privilege:Public

InsertEntry - This operation will add a MsMdConfigurationEntry to the current configuration list.

Arguments:MsMdScheduleEntry*
Return Type:EcTVoid
Privilege:Public

MsMdScheduleList - This operation serves as the default constructor.

Arguments:
Return Type:Void

Privilege:Public

MsMdScheduleList - This operation serves as the copy constructor.

Arguments:MsMdScheduleList&

Return Type:Void

Privilege:Public

RemoveEntry - This operation will remove a MsMdConfigurationEntry from the current configuration list.

Arguments:MsMdScheduleEntry*

Return Type:EcTVoid

Privilege:Public

UpdateArchiveLoadTime - This operation will update a scheduled time of loading to the ECS Data Archive.

Arguments:RWCollectableTime&, MsMdConfigurationList*

Return Type:EcTVoid

Privilege:Public

UpdateDBLoadTime - This operation will update a scheduled time of loading to the Management Data base.

Arguments:RWCollectableTime&, MsMdConfigurationList*

Return Type:EcTVoid

Privilege:Public

UpdateEntryLoadTime - This operation will update a next scheduled time for a host.

Arguments:MsMdScheduleEntry*, MsMdConfigurationList*

Return Type:EcTVoid

Privilege:Public

UpdateTime - This operation will return an updated time.

Arguments:RWSortedVector&, RWCollectableTime&, RWInteger&

Return Type:RWCollectableTime

Privilege:Public

~MsMdScheduleList - This operation serves as the default destructor.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdSchedule class has associations with the following classes:

Class: MsMdManager manages

6.10.3.15 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:

myHostName - This attribute represents the hostname to which the schedule entry applies.

Data Type:RWCString

Privilege:Private

Default Value:

myNextEntryTime - the next scheduled time for this log file to be transferred.

Data Type:RWCollectableTime

Privilege:Private

Default Value:

Operations:

GetHostName - This operation will return the name of the host.

Arguments:

Return Type:RWCString

Privilege:Public

GetNextEntryTime - This operation will return the next time of the applied host.

Arguments:

Return Type:RWCollectableTime

Privilege:Public

MsMdScheduleEntry - this operation serves as the constructor for the MsMdScheduleEntry class..

Arguments:RWCSring, RWCollectableTime

Return Type:Void

Privilege:Public

MsMdScheduleEntry - This operation serves as the constructor with only the time.

Arguments:RWCollectableTime&

Return Type:Void

Privilege:Public

MsMdScheduleEntry - This operation serves as the default constructor for the

MsMdScheduleEntry class.

Arguments:

Return Type:Void

Privilege:Public

MsMsScheduleEntry - This operation serves as the copy constructor for the MsMdScheduleEntry class.

Arguments:MsMdScheduleEntry&

Return Type:Void

Privilege:Public

compareTo - This operation compare two times to sort the time in a collection.

Arguments:RWCollectable*

Return Type:EcTInt

Privilege:Public

~MsMdScheduleEntry - This operation serves as the default destructor for the MsMdScheduleEntry class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdScheduleEntry class has associations with the following classes:

MsMdSchedule (Aggregation)

6.10.3.16 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:

Return Type:Void

Privilege:Public

PDL:MsMdUserInterface::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:

Return Type:Void

Privilege:Public

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:

Return Type:Void

Privilege:Public

PDL:MsMdUserInterface::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 - This operation allows the configuration of log transfers.

Arguments:MsMdConfigurationEntry

Return Type:Void

Privilege:Public

MsMdUserInterface - This operation represents the default constructor for the class MsMdUserInterface.

Arguments:

SetChainTransID - This operation allows an event transactionID to be entered via the user interface for the purpose of event chaining.

Arguments:childID

SetFilter - This operation allows a user to set a filter.

Arguments:filter

Return Type:Void

Privilege:Public

SortLogFile -

Arguments:sortMethod

Return Type:Void

Privilege:Public

~MsMdUserInterface - This operation represents the default constructor for the MsMdUserInterface class.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsMdUserInterface class has associations with the following classes:

Class: MsMdLogBrowser

Class: MsMdManager accesses

6.10.4 Management Data Access Dynamic Model

6.10.4.1 User Browses Logfile Data

Figure 6.10-3 contains the event trace diagram for the browse log scenario.

6.10.4.1.1 Beginning Assumptions

None.

6.10.4.1.2 Interfaces with Other Subsystems and Segments

CSS (via CsFtFTPReIA)

6.10.4.1.3 Stimulus

The user requests to browse log file data from the MDA user interface.

6.10.4.1.4 Participating Classes From the Object Model

MsMdManagerInterface

MsMdManager

MsMdAggregateLogFileList

MsMdAggregateLogEntry

CsFtFTPReIA

MsMdArchiveLog

6.10.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.10.4.1.6 Ending State

The user is provided the log file browser and the requested data.

6.10.4.1.7 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.



Figure 6.10-3. User Browses Log File Data

6.10.5 Management Data Access Dynamic Model

6.10.5.1 User Chains Events

Figure 6.10-4 contains the event trace diagram for the User Chains Logfile Data scenario.

6.10.5.1.1 Beginning Assumptions

None.

6.10.5.1.2 Interfaces with Other Subsystems and Segments

CSS (via CsFtFTPReIA)

6.10.5.1.3 Stimulus

The user requests to chain log file data from the MDA user interface.

6.10.5.1.4 Participating Classes From the Object Model

MsMdManagerInterface

MsMdManager

MsMdAggregateLogFileList

MsMdAggregateLogEntry

MsMdLogBrowser

CsFtFTPReIA

MsMdArchiveLog

6.10.5.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.10.5.1.6 Ending State

The user is provided the log file browser and the requested data.

6.10.5.1.7 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.

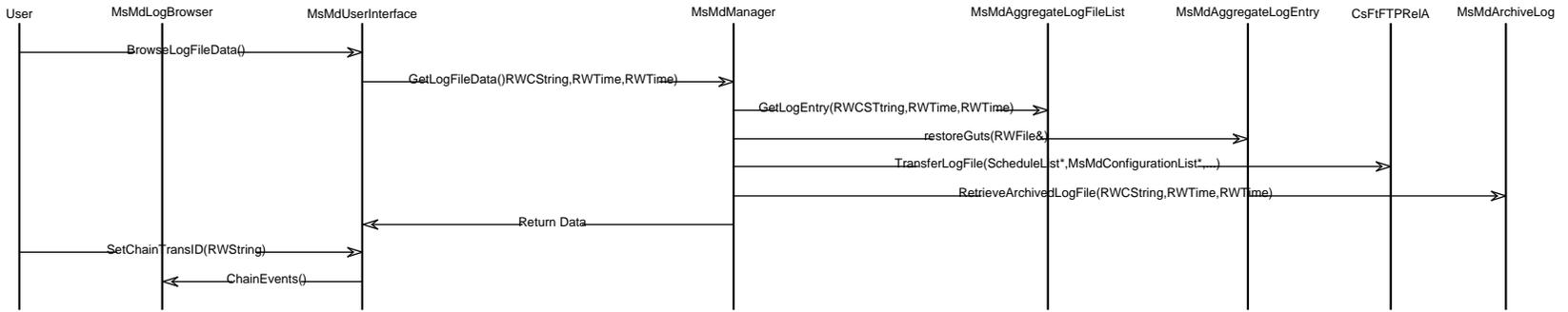


Figure 6.10-4. User Chains Logfile Data

At this point the interface allows the user to chain events by inputting an event TransactionID. The event chain sequence is computed and returned to the user interface.

6.10.6 Management Data Access Dynamic Model

6.10.6.1 MSS Logfile is Processed

Figure 6.10-5 contains the event trace diagram for the Process Logfile scenario..

6.10.6.1.1 Beginning Assumptions

None.

6.10.6.1.2 Interfaces with Other Subsystems and Segments

CSS (via CsFtFTPReIA)

6.10.6.1.3 Stimulus

An MSS logfile requires processing and transfer to the management database.

6.10.6.1.4 Participating Classes From the Object Model

MsMdManager

MsMdAggregateLogFileList

MsMdAggregateLogEntry

MsMdEventList

CsFtFTPReIA

MsMdArchiveLog

ManagementRDBMS

6.10.6.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and the subsystem are in a normal, steady state.

6.10.6.1.6 Ending State

The MSS logfile on a managed host is processed and transferred to the management database..

6.10.6.1.7 Scenario Description

An MSS logfile is scheduled to be processed and transferred to the management database. The MsMdManager utilizes the MsMdAggregateLogfileList to determine location of the data. The MsMdManager then initiates the processing by retrieving the logfile data. Once the data has been retrieved, MsMdProcessEvent determines the active modes of the managed host from which the processing is occurring by reading that host's active mode-configuration file. This information is then used by MsMdProcessEvent to open the appropriate management databases (a management database will exist for each active mode). MsMdProcessEvent will process, then transfer the logfile records to their appropriate management database according to the event's mode. When all of the logfile records have been transferred, the open management databases will be closed.

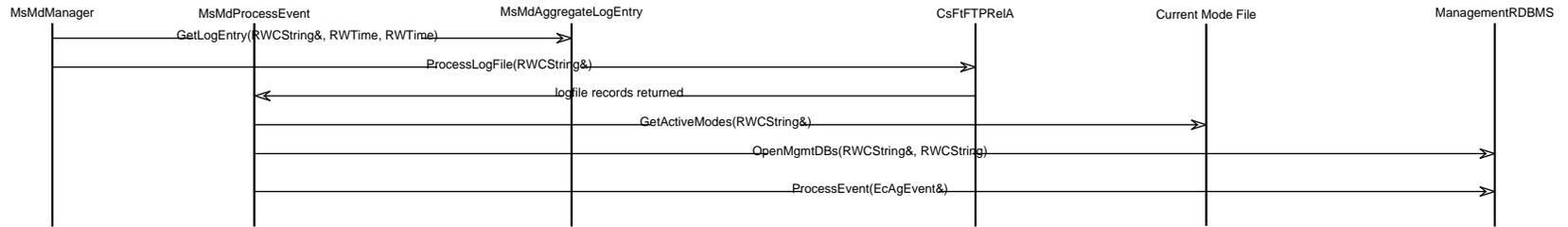


Figure 6.10-5. Process Logfile Scenario

At this point, the remote logfile has been transferred to the MDA, logfile records processed, and the records written to the management database. The MsMdManager is then ready to process other remote logfiles, as necessary.

6.10.7 Management Data Access Structure

Table 6.10-1. Management Data Access Components

Component Name	COTS/Custom
Management Data Access Services	Custom
Management Data Access User Interface	Custom

6.10.7.1 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)
 MsMdConfigurationList (C++ code)
 MsMdConfigurationEntry (C++ code)
 MsMdSchedule (C++ code)
 MsMdScheduleEntry (C++ code)
 MsMdAggregateLogFileList (C++ code)
 MsMdAggregateLogEntry (C++ code)
 MsMdEventList (C++ code)
 MsMdEventField (C++ code)
 MsMdProcessEvent (C++ code)
 MsMdLogBrowser (C++ code)

6.10.7.2 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)

6.10.8 Management Data Access Management and Operation

6.10.8.1 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.)

6.10.8.2 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, chained 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.

6.10.8.3 Reports

The Management Data Access Service does not provide reports.

6.11 Management DBMS and Database

6.11.1 Overview

A COTS Database Management System (DBMS), for which Sybase has been selected, provides the data storage and retrieval functions a database that is designed to store 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 use to perform ad hoc statistical analysis and generate ad hoc reports to satisfy the ECS report generation requirements. A COTS Report Generation product will provide the user interface procedures that construct queries in Sybase SQL format, compute query based results, and format reports for printing and display.

6.11.2 Implementation

The Management Database architecture is an implementation of the ANSI Three-Schema Architecture. The Internal schema is implemented using the a-client/server paradigm. 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 Structured 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.

6.12 User Comment Survey

6.12.1 User Comment Survey Overview

The User Comment Survey is a server which manages the user comment surveys which are made available to any user who wishes to provide comments and suggestions about the ECS system.

The surveys are organized by categories. A category may represent a capability of the ECS system. For each category there is a survey which is a list of questions and one general comment field. The user can enter an answer for any or all of the questions and the user may enter any general comment in the comment field. The User Comment Survey server stores all responses in a database which can be accessed to generate reports.

When a survey is used by a registered ECS user, the user's existing responses to the survey will be retrieved from the database and displayed to the user. The user can then modify his responses, but whatever is in the survey response area when the user exits the survey will replace what is currently stored in the database.

If a guest user accesses the survey, his previous responses are not filled in to the survey and when responses are entered for the survey, new entries are made in the database for the responses.

6.12.2 User Comment Survey Context

The User Comment Survey Service, as shown in the context diagram, Figure 6.12-1, interfaces with the Client and the Management Agent. The information exchanged across these interfaces, as shown in the diagram, is described here.

The Client Subsystem requests the comment survey information to display the survey categories, the survey questions and the user's current answers in response to the user's actions. The User Comment Survey server provides the requested information. The Client Subsystem also sends updated answers and comments to the User Comment Survey server to be updated in the database.

The User Comment Survey server performs management requests received from the Management Agent, such as startup and shutdown and sends performance information as well as error information to the Management Agent.

6.12.3 User Comment Survey Object Model

The MsCsSurveyMgr class is the manager class for the server and that class performs most of the significant functions. The MsCsSurveyMgr class receives and responds to inputs received from the Client Subsystem. The MsCsSurveyMgr is also considered an interface class to the Sybase Database which is where the comment survey data is stored.

The server is managed by the ECS Process Framework (EcPfManagedServer) which provides the communication framework as well as the interface to the Management Agent. As part of the server startup, the McCsProcessingTimeMetric performance metric is registered with the framework so that the Management Agent can request and set the value of the metric. The MsCsTimer is the class which provides the server the mechanism for collecting the processing time.

The User Comment Survey object model is shown in Figure 6.12-2.

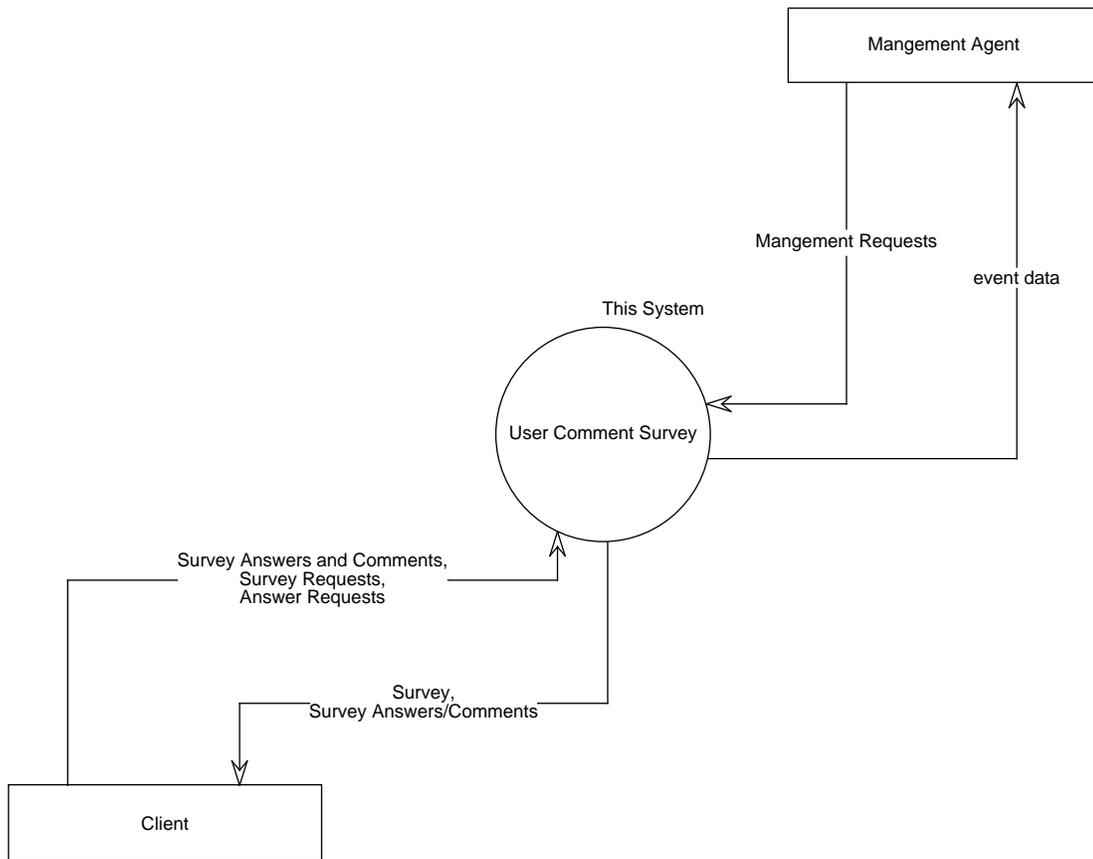


Figure 6.12-1. User Comment Survey Context Diagram

6.12.3.1 EcAgPerfMetric Class

Parent Class:Not Applicable
 Public:No
 Distributed Object:No
 Purpose and Description:
 This metric contains performance data.

Attributes:

None

Operations:

None

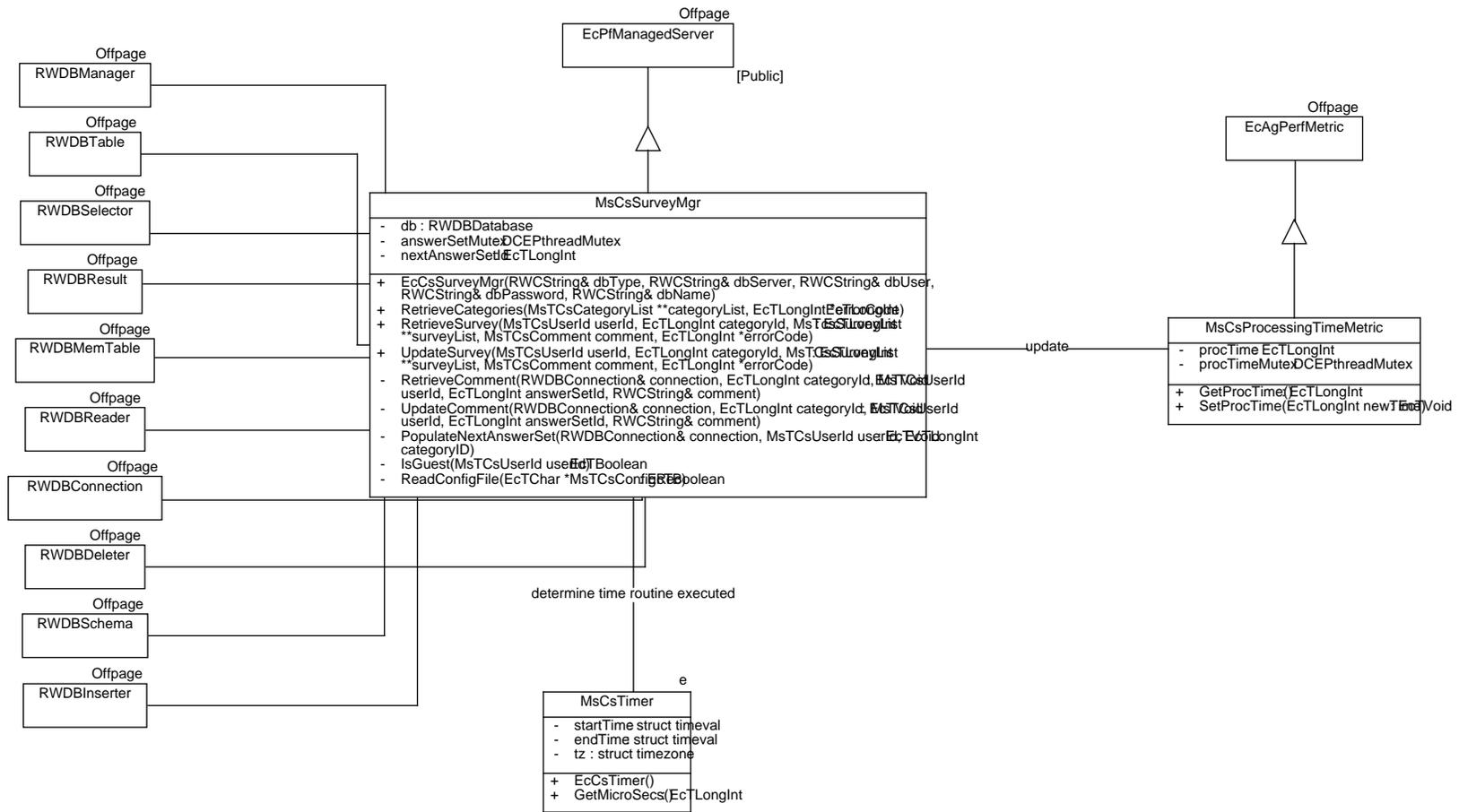


Figure 6.12-2. User Comment Survey Object Model

Associations:

The EcAgPerfMetric class has associations with the following classes:

None

6.12.3.2 EcPfManagedServer Class

Parent Class:Not Applicable

Public:Yes

Distributed Object:No

Purpose and Description:

This is the container class that starts up the event Manager, table Manager, monitor, port monitor, discoverer, subagent configuration, static buffer, and the deputy gate. This class also starts a thread that triggers scheduled events (i.e. polling ECS application's performance metrics).

Attributes:

None

Operations:

None

Associations:

The EcPfManagedServer class has associations with the following classes:

None

6.12.3.3 MsCsProcessingTimeMetric Class

Parent Class:EcAgPerfMetric

Public:No

Distributed Object:No

Purpose and Description:

This is a performance collecting metric reporting class. This class uses the performance data collecting metric key mechanism. This class will be registered with the ECS Process Framework which enables the Management Agent to get the value of the metric and to set the value of the metric. The metric is the amount of real-time that this process executes.

Attributes:

procTime - This is the metric value which can be set or retrieved. This process sets the value as it executes and the Management Agent will read this value.

Data Type:EcTLongInt
Privilege:Private
Default Value:

procTimeMutex - This attribute is used to lock the metric value while it is being read and written to, in order to prevent the Management Agent from accessing the value when this process is accessing the value.

Data Type:DCEPthreadMutex
Privilege:Private
Default Value:

Operations:

GetProcTime - Returns the current value of the performance metric.

Arguments:
Return Type:EcTLongInt
Privilege:Public
PDL: No PDL

SetProcTime

Arguments:EcTLongInt newTime
Return Type:EcTVoid
Privilege:Public

Associations:

The MsCsProcessingTimeMetric class has associations with the following classes:

Class: MsCsSurveyMgr update

6.12.3.4 MsCsSurveyMgr Class

Parent Class:EcPfManagedServer

Public:No

Distributed Object:No

Purpose and Description:

This is the Manager class for this process. This class is responsible for taking all of the inputs to the process and performing the necessary actions. This class also acts as an interface class for the Sybase database which stores the user survey results. This class provides methods to read and update the user survey categories, surveys, and comments.

Attributes:

answerSetMutex - Mutex to protect the next answer set ID.

Data Type:DCEPthreadMutex

Privilege:Private
Default Value:

db - The survey database.
Data Type:RWDBDatabase
Privilege:Private
Default Value:

nextAnswerSetId - The next available answer set ID (for guest survey).
Data Type:EcTLongInt
Privilege:Private
Default Value:

Operations:

EcCsSurveyMgr - This is the constructor for this class. This class initializes the connection to the specified Sybase database.
Arguments:RWCString& dbType, RWCString& dbServer, RWCString& dbUser, RWCString& dbPassword, RWCString& dbName
Return Type:Void
Privilege:Public
PDL: No PDL

IsGuest - This method determines if the passed user ID represents a guest in the system.
Arguments:MsTCsUserId userId
Return Type:EcTBoolean
Privilege:Private
PDL: No PDL

PopulateNextAnswerSet - This method retrieves and populates to the nextAnswerSetId attribute, the next available answerSetId.
Arguments:RWDBConnection& connection, MsTCsUserId userId, EcTLongInt categoryID
Return Type:EcTVoid
Privilege:Private
PDL: No PDL

ReadConfigFile - This method reads the configuration parameters from the configuration file.
Arguments:EcTChar *MsTCsConfigRec
Return Type:EcTBoolean
Privilege:Private
PDL: No PDL

RetrieveCategories - This method retrieves the descriptions and icons for all available survey categories.

Arguments:MsTCsCategoryList **categoryList, EcTLongInt *errorCode

Return Type:EcTLongInt

Privilege:Public

PDL: No PDL

RetrieveComment - This method is used internally to obtain the current comment for a category and user.

Arguments:RWDBConnection& connection, EcTLongInt categoryId, MsTCsUserId userId, EcTLongInt answerSetId, RWCString& comment

Return Type:EcTVoid

Privilege:Private

PDL: No PDL

RetrieveSurvey - Based on a user id and category, this method returns a structure containing all the elements of a survey, including questions, current answers and the comment for the category. For a guest user, the answers returned will always be zero.

Arguments:MsTCsUserId userId, EcTLongInt categoryId, MsTcsSurveyList **surveyList, MsTCsComment comment, EcTLongInt *errorCode

Return Type:EcTLongInt

Privilege:Public

PDL: No PDL

UpdateComment - This method is used internally to update the current comment for a category and user.

Arguments:RWDBConnection& connection, EcTLongInt categoryId, MsTCsUserId userId, EcTLongInt answerSetId, RWCString& comment

Return Type:EcTVoid

Privilege:Private

PDL: No PDL

UpdateSurvey - Based on a user id and category, this method updates this given survey answers for the appropriate criteria. For a guest user, a new set of answers is created. For registered users, the old answers and comments are overwritten.

Arguments:MsTCsUserId userId, EcTLongInt categoryId, MsTCsSurveyList **surveyList, MsTCsComment comment, EcTLongInt *errorCode

Return Type:EcTLongInt

Privilege:Public

PDL: No PDL

Associations:

The MsCsSurveyMgr class has associations with the following classes:

Class: RWDBConnection
Class: RWDBDeleter
Class: RWDBInserter
Class: RWDBManager
Class: RWDBMemTable
Class: RWDBReader
Class: RWDBResult
Class: RWDBSchema
Class: RWDBSelector
Class: RWDBTable
Class: MsCsTimer determinetimeroutineexecuted
Class: MsCsProcessingTimeMetric update

6.12.3.5 MsCsTimer Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class is used to collect the amount of time (real-time, not CPU) that this process spends executing. In each method of this process, an object of this class is constructed and at the end of the method, the GetMicroSecs method is called. This results in the duration that the method was executed.

Attributes:

endTime - This attribute is set to the current time when the process temporarily stops processing.

Data Type:struct timeval

Privilege:Private

Default Value:

startTime - This attribute is set to the current time when this class is constructed.

Data Type:struct timeval

Privilege:Private

Default Value:

tz - A default timezone which is used when the current time is recorded at the beginning and the end of processing.

Data Type:struct timezone

Privilege:Private

Default Value:

Operations:

EcCsTimer - The default constructor of this class. The current time is recorded when the object is constructed.

Arguments:

Return Type:Void

Privilege:Public

PDL: No PDL

GetMicroSecs - This method gets the current time of day and subtracts that value from the time recorded when the class was constructed. This results in the amount of real-time that was expended since the class was constructed.

Arguments:

Return Type:EcTLongInt

Privilege:Public

PDL: No PDL

Associations:

The MsCsTimer class has associations with the following classes:

Class: MsCsSurveyMgr determinetimeroutineexecuted

6.12.3.6 RWDBConnection Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBConnection class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.7 RWDBDeleter Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBDeleter class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.8 RWDBInserter Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBInserter class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.9 RWDBManager Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBManager class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.10 RWDBMemTable Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBMemTable class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.11 RWDBReader Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBReader class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.12 RWDBResult Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBResult class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.13 RWDBSchema Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBSchema class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.14 RWDBSelector Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBSelector class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.3.15 RWDBTable Class

Parent Class:Not Applicable

Attributes:

None

Operations:

None

Associations:

The RWDBTable class has associations with the following classes:

Class: MsCsSurveyMgr

6.12.4 User Comment Survey Dynamic Model

6.12.4.1 User Fills Out A Survey

This scenario is depicted in Figure 6.13-3.

6.12.4.1.1 Beginning Assumptions

None.

6.12.4.1.2 Interfaces with Other Subsystems and Segments

Client

6.12.4.1.3 Stimulus

A user selects the user survey tool to update or enter answers to survey questions for a survey or enter a comment for a survey.

6.12.4.1.4 Participating Classes From the Object Model

MsCsSurveyMgr

6.12.4.1.5 Beginning System, Segment and Subsystem State(s)

The system, segment and subsystem are in a steady state.

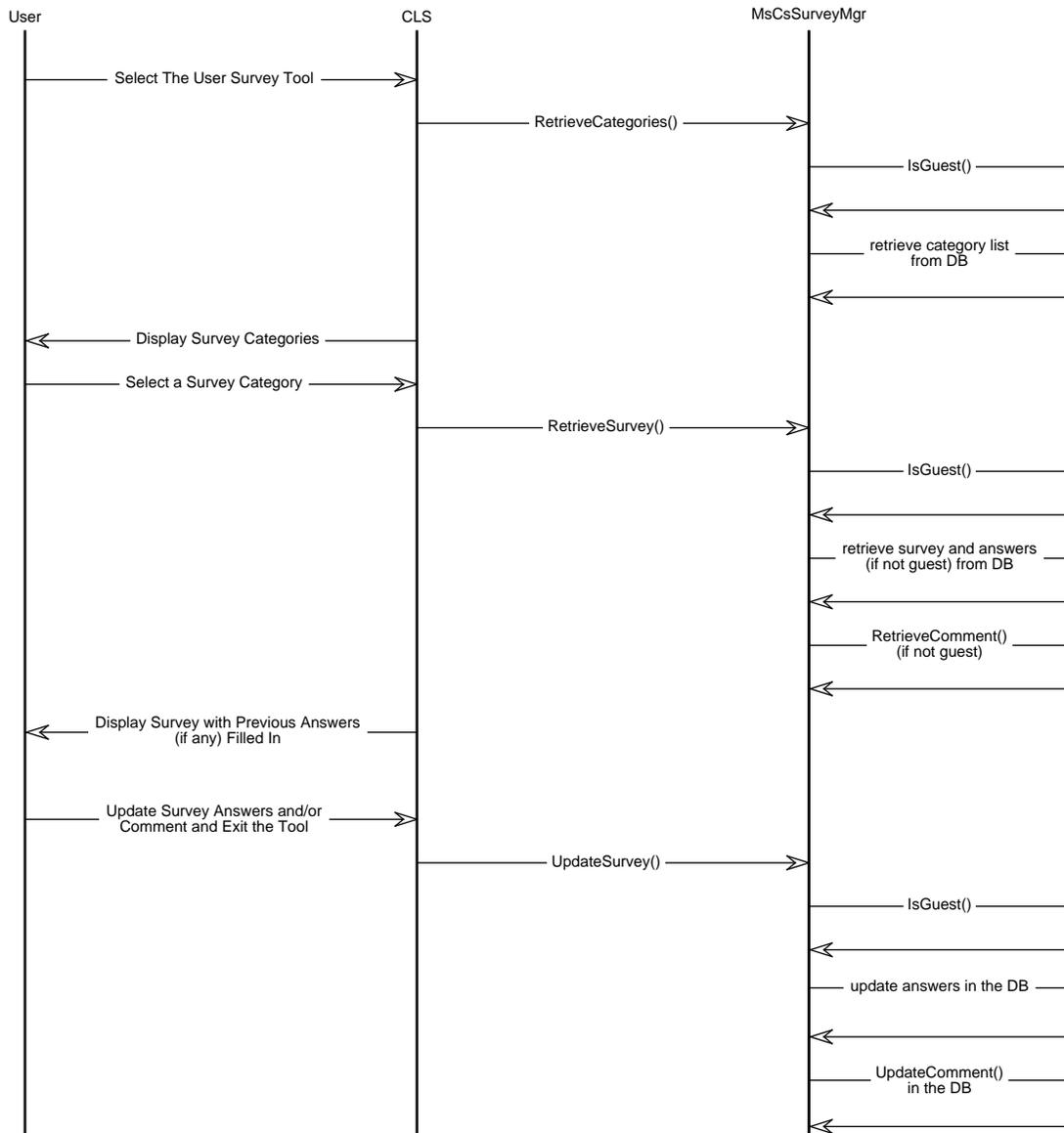


Figure 6.12-3. User Fills Out A Survey

6.12.4.1.6 Ending State

The system, segment and subsystem are in a steady state.

6.12.4.1.7 Scenario Description

When the user starts the User Comment Survey tool, the client subsystem issues a request for the survey categories from MsCsSurveyMgr. MsCsSurveyMgr checks to see if the user is a guest user and performs guest user unique initialization. The survey categories are retrieved from the Database and returned to CLS to be displayed to the user.

When the user selects a survey category to display, CLS issues a request for the survey from MsCsSurveyMgr. MsCsSurveyMgr again checks to see if the user is a guest user. The survey questions are retrieved from the database. If the user is not a guest, the user's current survey responses and the user's current comment are retrieved from the Database. The questions, answers, and comment are returned to CLS to be displayed to the user.

The user will then update or add their responses to the survey questions and will also update the comment field of the survey. When the user exits the survey, CLS sends the answers and comment to MsCsSurveyMgr. MsCsSurveyMgr replaces the user's current responses with the received responses and replaces the comment field if the user is not a guest. For guest users, new entries are made in the database to store the answers and comment.

6.12.5 User Comment Survey Structure

Table 6.12-1 lists the components of the User Comment Survey Service.

Table 6.12-1. User Comment Survey Components

Component Name	COTS/Custom
MsCsSurveyMgr	Custom(C++ code)
MsCsTimer	Custom(C++ code)
MsCsProcessingTimeMetric	Custom(C++ code)

6.12.5.1 Survey Manager CSC

Purpose and Description

The survey manager CSC includes all of the classes for this CI.

6.12.6 User Comment Survey Management and Operation

6.12.6.1 System Management Strategy

The User Comment Survey Management Strategy utilizes the MSS Management Agent Services and the ECS Process Framework (EcPfManagedServer) for its management.

6.12.6.2 Operator Interfaces

The Client Subsystem is providing the user interface for this server.

6.12.6.3 Reports

The following predefined User Comment Survey reports will be available:

Survey Answers By User -- report which lists the answers stored for a specified user for a specified survey.

Survey Answers By Question -- report which lists all of the answers for a particular survey question.

Survey Answers By Category -- report which lists all of the answers for all questions for a particular category.

Comments by User -- report which lists all of the comments a user had (for each category).

Comments by Category -- report which lists all of the comments from all of the users for a specified category.

Other user comment survey reports will be generated on an ad hoc basis.

6.13 Enterprise Framework Management Service

6.13.1 Enterprise Framework Management Overview

The Enterprise Framework Management Service is collection of system administration tools which are integrated using the Tivoli Management Environment. The Enterprise Framework Management tools provide the operator the ability to administer the underlying framework of the ECS system.

The Enterprise Framework Management Service consists of the following administrative capabilities: Software Distribution (described in detail in Software Distribution Management, Section 5.7 of this document - 305-CD-029-002), Event Monitoring (described in detail in Fault Management, Section 6.5 of this document - 305-CD-029-002), and System Administration.

System Administration consists of the tools required to administer the underlying system framework of ECS. System Administration consists of Unix Administration, Database Administration, DCE Cell Administration, and System Backup.

Unix Administration will be performed using the Tivoli Admin product. Tivoli Admin is a tool which provides a Graphical User Interface for the administration of a distributed heterogeneous Unix system. Tivoli Admin manages the following resources: Unix Host, NIS Maps, Host Namespace, Unix Users, Unix Groups.

Database Administration will be performed for the Sybase DBMS using the ESSM Tivoli Plus Module. ESSM is an extension to Tivoli's administrative toolset that provides a Graphical User Interface for Sybase Database Administration. ESSM provides the subset of Sybase DBA capabilities which are necessary to perform on a regular basis.

DCE Cell Administration will be performed by Hewlett Packard's DCE Cell Management tools that are provided as part of HP's DCE Core Services. The HP tool set includes the following: CDS Browser to administer the Cell Directory Service, acctmgr to administer the DCE Security Server, DCE Cell Configurator (integrated with HP's SAM tool) to administer the DCE cell configuration, and CellMon to monitor the status of the DCE Cell. The HP DCE Administration tool set is not integrated with Tivoli, but Tivoli will be used to provide an integrated desktop which will present icons representing the tools to the operator. The operator will be able to launch the DCE tools from the Tivoli desktop.

In support of site-wide ECS backup, the DAAC configuration at EDC includes Legatto's Networker software for network storage management. It provides a suite of integrated tools for backup and recovery, archive and retrieval, hierarchical storage management, on-line database backup and system management tool integration. It works on multi-platform networks, is motif-based with on-line help, supports concurrent device support for parallel backup and recovery using up to 16 storage devices. Both scheduled and ad-hoc backups, recoveries and other data management services can be performed by authorized users.

Site-wide system backup will be performed by Legato Systems, Inc's NetWorker product. NetWorker provides a suite of integrated tools for backup and recovery, archive and retrieval, hierarchical storage management and on-line database backup. The product supports multi-platform networks, contains a motif-based GUI with on-line help, and supports concurrent device support for parallel backup and recovery using up to 16 storage devices. Both scheduled and ad-hoc backups, recoveries and other data management services can be performed by authorized users. NetWorker software consists of two components: a client portion, which runs on the systems to be backed up, and a server portion, which is the system to which the backup devices are connected. The client portions will send the data to be backed up to the server portion which then writes the data out to disk.

6.13.2 Enterprise Framework Management Context

The Enterprise Framework Management Service, as shown in the context diagram, Figure 6.13-1, interfaces with . The information exchanged across these interfaces, as shown in the diagram, is described here.

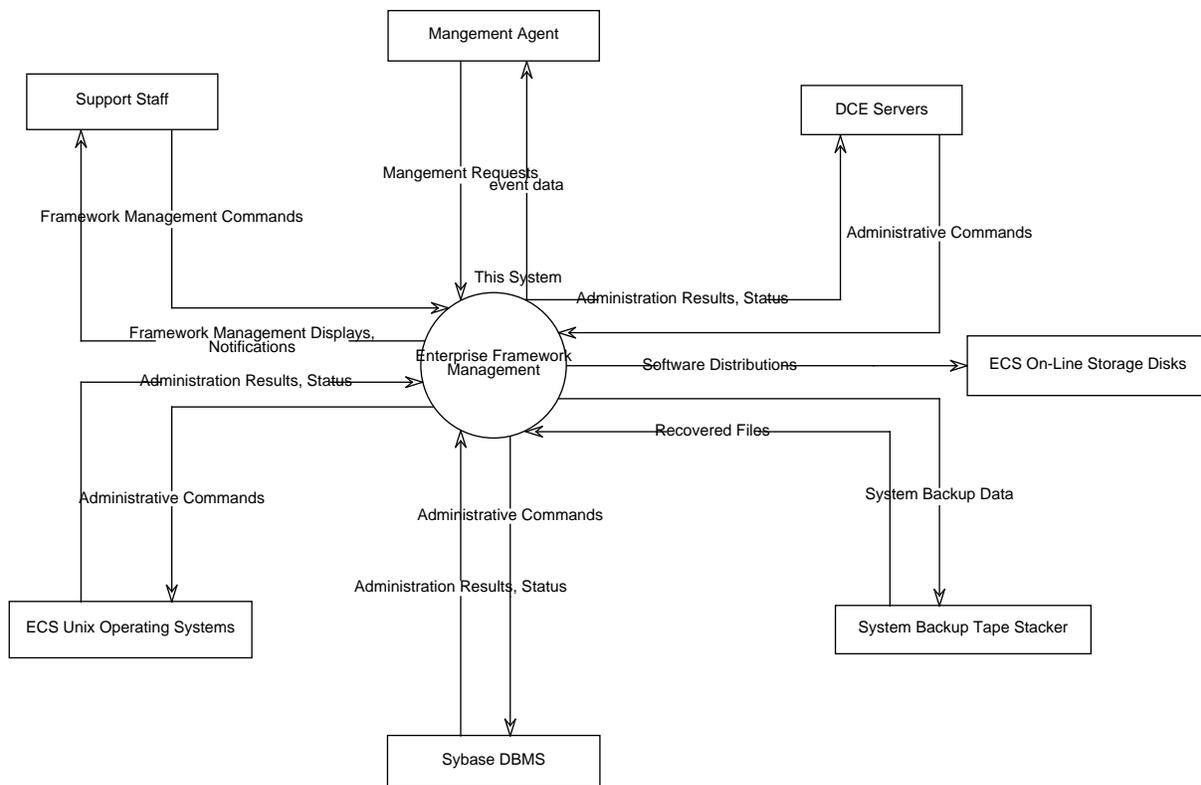


Figure 6.13-1. Enterprise Framework Context Diagram

The Enterprise Framework receives commands from the support staff, implements the commands on the selected systems or servers and displays the results back to the support staff. In addition, unsolicited notifications are also sent to the support staff. The Enterprise Framework interfaces with the objects which are being managed. This includes the Unix Operating Systems running on all of the ECS machines, the Sysbase DBMS servers running on the ECS machines, the DCE Servers (Security and CDS), and the hardware including on-line storage disks for software distributions and the tape stacker for files to be backed up and files to be recovered.

Portions of the Enterprise Framework Management Service perform management requests received from the Management Agent, such as startup and shutdown.

6.13.3 Enterprise Framework Management Object Model

The object model shows three different hierarchical trees. The primary hierarchy represents the breakdown of the EnterpriseFramework into its parts. At the first level of the EnterpriseFramework hierarchy, the Enterprise Framework consists of MsEfSoftwareDistribution for delivering software or other files to all ECS computers, MsEfEventMonitoring for receiving and correlating fault messages, and MsEfSystemAdmin which represents the collection of system administration tools for ECS. MsEfSystemAdmin consists of MsEfUnixAdmin for the administration of all of the Unix systems in ECS, MsEfDatabaseAdmin for the administration of all of the Sybase DBMSs in the ECS system, and MsEfSystemBackup for performing backup and recover actions on the ECS system.

The second hierarchy in the model represents the underlying distributed framework for the Tivoli product (MsTfTivoliFramework). This framework is what allows Tivoli products and products integrated into Tivoli to administer our distributed heterogeneous system. The Tivoli framework provides a CORBA compliant object and message passing mechanism which is used by all Tivoli products. The Tivoli Framework consists of one Tivoli Server (MsEfTivoliServer) and one Tivoli Client (MsEfTivoliClient) at each ECS machine.

The third hierarchy consists of the proxy agents to the ECS Management Agent which are used to allow those portions of the COTS products which are always running to be started up and shutdown as part of the entire ECS system.

The Enterprise Framework Management Service object model is shown in Figure 6.13-2.

6.13.3.1 EcAgCOTSManager Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

this abstract class embodies the characteristics and functionality of a manager object responsible for managing a single COTS process. It encapsulates all MSS management application functions into a single class. The COTS proxy agent developer is responsible for inheriting from this class and specializing it towards the COTS process to manage.

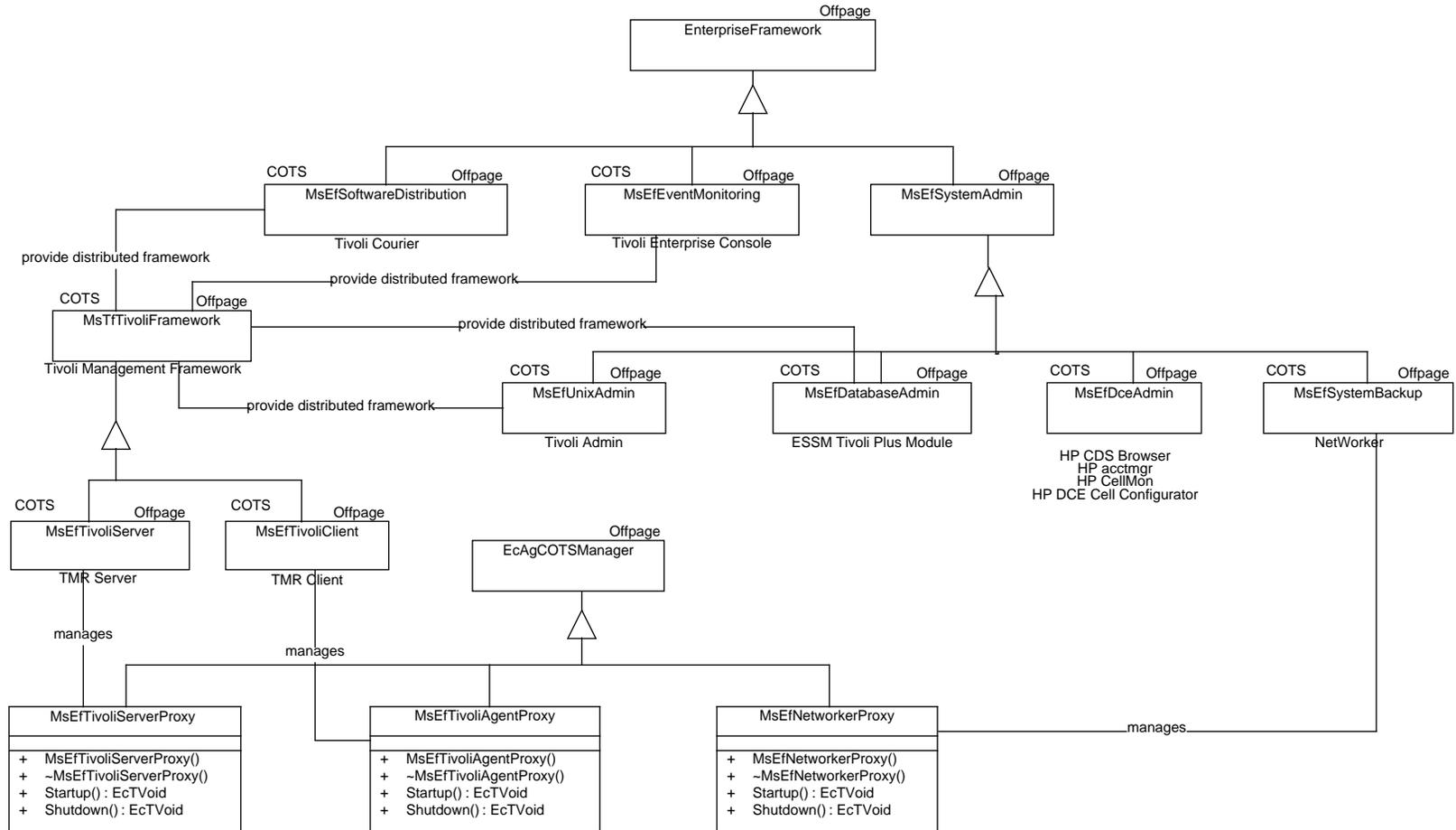


Figure 6.13-2. Enterprise Framework Management Service Object Model

Attributes:

None

Operations:

None

Associations:

The EcAgCOTSManger class has associations with the following classes:

None

6.13.3.2 EnterpriseFramework Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

EnterpriseFramework is the Tivoli COTS product that performs enterprise wide services: System Administration (Tivoli/Admin), Software distribution (Tivoli/Courier), performance monitoring (Tivoli/Sentry) and fault correlation (Tivoli/Enterprise Console). The framework also acts as the integrated desktop for Maintenance and Operations, integrating other administrative functions such as Sybase database administration, system backup/restore, and DCE Cell administration.

Attributes:

None

Operations:

None

Associations:

The EnterpriseFramework class has associations with the following classes:

None

6.13.3.3 MsEfDatabaseAdmin Class

Parent Class:MsEfSystemAdmin

Public:No

Distributed Object:No

Purpose and Description:

This class represents the ESSM Tivoli Plus COTS module. This product extends the Tivoli product by providing a Database Administration for Sybase databases. The product contains the subset of database administration features that are performed on a regular basis. The product runs on top of the Tivoli Framework which allows the administration tool to be run from the Tivoli desktop when the desktop is launched from any machine in ECS.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfDatabaseAdmin class has associations with the following classes:

Class: MsTfTivoliFramework providedistributedframework

6.13.3.4 MsEfDceAdmin Class

Parent Class:MsEfSystemAdmin

Public:No

Distributed Object:No

Purpose and Description:

This class represents the collection of DCE administration tools which are provided in Hewlett Packard's DCE Core Services product. The tools include: CDS Browser to administer the CDS Server, acctmgr to administer the Security Server, DCE Cell Configurator to administer the DCE cell configuration, and CellMon to monitor the status of the DCE Cell. While these tools are not integrated with the Tivoli product, the Tivoli desktop will be set up to allow the operator to launch the tools from the integrated desktop.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfDceAdmin class has associations with the following classes:

None

6.13.3.5 MsEfEventMonitoring Class

Parent Class:EnterpriseFramework

Public:No

Distributed Object:No

Purpose and Description:

This class represents the Tivoli/TEC (Tivoli Enterprise Console) COTS product. This product provides a Graphical User Interface fault reporting and fault correlation. Tivoli/TEC runs on top of the Tivoli Framework which allows the product to perform its functions on all of the machines in the network at the same time and from the same interface.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfEventMonitoring class has associations with the following classes:

Class: MsTfTivoliFramework providedistributedframework

6.13.3.6 MsEfNetworkerProxy Class

Parent Class:EcAgCOTSManger

Public:No

Distributed Object:No

Purpose and Description:

This is a proxy for the NetWorker Server which provides the interface to the ECS framework for the COTS product. This class interfaces with the management agent for startup and shutdown commands.

Attributes:

All Attributes inherited from parent class

Operations:

MsEfNetworkerProxy - This is the default constructor for the class. The startup and shutdown routines are registered.

Arguments:

Return Type:Void

Privilege:Public

Shutdown - This method performs a normal shutdown for the NetWorker server.

Arguments:

Return Type:EcTVoid

Privilege:Public

Startup - This method starts up the NetWorker server processes.

Arguments:

Return Type:EcTVoid

Privilege:Public

~MsEfNetworkerProxy - This is the default destructor for the class. Any resources allocated to the object will be returned to the system.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsEfNetworkerProxy class has associations with the following classes:

Class: MsEfSystemBackup manages

6.13.3.7 MsEfSoftwareDistribution Class

Parent Class:EnterpriseFramework

Public:No

Distributed Object:No

Purpose and Description:

This class represents the Tivoli/Courier COTS product. This product provides software and file distribution capabilities across the ECS herogenous network. Tivoli/Courier runs on top of the Tivoli Framework which allows the product to perform the software and file distribution on all of the machines in the network at the same time and from the same interface.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfSoftwareDistribution class has associations with the following classes:

Class: MsTfTivoliFramework providedistributedframework

6.13.3.8 MsEfSystemAdmin Class

Parent Class:EnterpriseFramework

Public:No

Distributed Object:No

Purpose and Description:

This is an abstract class that consists of the collection of tools which are used to perform system administration functions.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfSystemAdmin class has associations with the following classes:

None

6.13.3.9 MsEfSystemBackup Class

Parent Class:MsEfSystemAdmin

Public:No

Distributed Object:No

Purpose and Description:

This class represents the system backup and restore COTS product NetWorker, by Legato. While this tool is not integrated with the Tivoli product, the Tivoli desktop will be set up to allow the operator to launch the tool from the integrated desktop.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfSystemBackup class has associations with the following classes:

Class: MsEfNetworkerProxy manages

6.13.3.10 MsEfTivoliAgentProxy Class

Parent Class: EcAgCOTSManger

Public: No

Distributed Object: No

Purpose and Description:

This is a proxy for the Tivoli Agent which provides the interface to the ECS framework for the COTS product. This class interfaces with the management agent for startup and shutdown commands.

Attributes:

All Attributes inherited from parent class

Operations:

MsEfTivoliAgentProxy - This is the default constructor for the class. The startup and shutdown routines are registered.

Arguments:

Return Type: Void

Privilege: Public

Shutdown - This method performs a normal shutdown for the Tivoli agent.

Arguments:

Return Type: EcTVoid

Privilege: Public

Startup - This method starts up the Tivoli Agent processes.

Arguments:

Return Type: EcTVoid

Privilege:Public

~MsEfTivoliAgentProxy - This is the default destructor for the class. Any resources allocated to the object will be returned to the system.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsEfTivoliAgentProxy class has associations with the following classes:

Class: MsEfTivoliClient manages

6.13.3.11 MsEfTivoliClient Class

Parent Class:MsTfTivoliFramework

Public:No

Distributed Object:No

Purpose and Description:

This class represents the agent portion of Tivoli's framework. Tivoli uses the framework to provide system administration across the distributed heterogenous network. There is one copy of the agent running on every machine in the Tivoli Management Environment (in ECS each site will be one Tivoli Management Environment).

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfTivoliClient class has associations with the following classes:

Class: MsEfTivoliAgentProxy manages

6.13.3.12 MsEfTivoliServer Class

Parent Class:MsTfTivoliFramework

Public:No

Distributed Object:No

Purpose and Description:

This class represents the server portion of Tivoli's framework. Tivoli uses the framework

to provide system administration across the distributed heterogenous network. There is only one copy of the server running in the Tivoli Management Environment (which corresponds to one per site in ECS).

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfTivoliServer class has associations with the following classes:

Class: MsEfTivoliServerProxy manages

6.13.3.13 MsEfTivoliServerProxy Class

Parent Class:EcAgCOTSManger

Public:No

Distributed Object:No

Purpose and Description:

This is a proxy for the Tivoli Server which provides the interface to the ECS framework for the COTS product. This class interfaces with the management agent for startup and shutdown commands.

Attributes:

All Attributes inherited from parent class

Operations:

MsEfTivoliServerProxy - This is the default constructor for the class. The startup and shutdown routines are registered.

Arguments:

Return Type:Void

Privilege:Public

Shutdown - This method performs a normal shutdown for the Tivoli server.

Arguments:

Return Type:EcTVoid

Privilege:Public

Startup - This method starts up the Tivoli Server processes.

Arguments:

Return Type:EcTVoid

Privilege:Public

~MsEfTivoliServerProxy - This is the default destructor for the class. Any resources allocated to the object will be returned to the system.

Arguments:

Return Type:Void

Privilege:Public

Associations:

The MsEfTivoliServerProxy class has associations with the following classes:

Class: MsEfTivoliServer manages

6.13.3.14 MsEfUnixAdmin Class

Parent Class:MsEfSystemAdmin

Public:No

Distributed Object:No

Purpose and Description:

This class represents the Tivoli/Admin COTS product. This product provides a Graphical User Interface to Unix system administration tasks. Tivoli/Admin runs on top of the Tivoli Framework which allows the product to perform Unix administration on all of the machines in the network at the same time and from the same interface. For example, Tivoli/Admin will allow an operator to add a new user account to multiple Unix Machines and/or multiple NIS domains with one action.

Attributes:

All Attributes inherited from parent class

Operations:

All Operations inherited from parent class

Associations:

The MsEfUnixAdmin class has associations with the following classes:

Class: MsTfTivoliFramework providedistributedframework

6.13.3.15 MsTfTivoliFramework Class

Parent Class:Not Applicable

Public:No

Distributed Object:No

Purpose and Description:

This class represents Tivoli's distributed framework. Tivoli uses the framework to provide system administration across the distributed heterogenous network. The framework is used by all components of Tivoli to pass Tivoli objects around the system while performing their processing.

Attributes:

None

Operations:

None

Associations:

The MsTfTivoliFramework class has associations with the following classes:

Class: MsEfDatabaseAdmin providedistributedframework

Class: MsEfEventMonitoring providedistributedframework

Class: MsEfSoftwareDistribution providedistributedframework

Class: MsEfUnixAdmin providedistributedframework

6.13.4Enterprise Framework Management Dynamic Model

Enterprise Framework Management Service is a collection of tools which enable the operator to perform administration functions on the ECS system. The tools are all COTS products which operate independently, so the dynamic model is not applicable to this CSC.

6.13.5Enterprise Framework Management Structure

Table 6.13-1 lists the components of the Enterprise Framework Management Service.

Table 6.13-1. Enterprise Framework Management Service Components (1 of 2)

Object Class Name	COTS/Custom
EnterpriseFramework	Abstract Class
MsEfSoftwareDistribution	COTS (Tivoli Courier)
MsEfEventMonitoring	COTS (Tivoli Enterprise Console)
MsEfSystemAdmin	Abstract Class
MsEfUnixAdmin	COTS (Tivoli Admin)

Table 6.13-1. Enterprise Framework Management Service Components (2 of 2)

Object Class Name	COTS/Custom
MsEfDatabaseAdmin	COTS (ESSM Tivoli Plus Module)
MsEfDceAdmin	COTS (HP DCE Administration Tools)
MsEfSystemBackup	COTS (Legato Systems, Inc's NetWorker)
MsEfTivoliFramework	COTS (Tivoli Management Framework)
MsEfTivoliServer	COTS (Tivoli Server)
MsEfTivoliClient	COTS (Tivoli Client)
MsEfTivoliServerProxy	Custom
MsEfTivoliAgentProxy	Custom
MsEfNetWorkerProxy	Custom

6.13.5.1 Software Distribution CSC

Purpose and Description

This CSC manages the distribution of new and updated software to the ECS system. A detailed description of this functionality is contained in Software Distribution Management, Section 5.7 of this document - 305-CD-029-002.

6.13.5.2 Event Monitoring CSC

Purpose and Description

This CSC receives and displays to the operator significant events which occur in the system and provides some automatic correlation of events and automatic recovery from fault events. A detailed description of this functionality is contained in Fault Management, Section 6.5 of this document - 305-CD-029-002.

6.13.5.3 System Administration CSC

Purpose and Description

This CSC contains the tools which are used to manage the underlying framework for the ECS system. These tools include Unix Administration tools, DCE Cell Administration tools, DBMS Administration tools, and Backup/Restore Administration tools. This CSC also includes the Backup/Restore client/server software and the associated management of the server.

6.13.5.4 Tivoli Framework CSC

Purpose and Description

This CSC is the underlying framework for the Tivoli product. This CSC consists of a Tivoli server and many Tivoli clients. This framework provides the distributed environment that enables the Tivoli tools to administer across the heterogeneous Unix system transparently to the operator. This CSC also consists of the custom developed components to manage the startup and shutdown of the Tivoli environment within the ECS infrastructure.

6.13.6 Enterprise Framework Management Service Management and Operation

6.13.6.1 System Management Strategy

The Management Strategy for this CSC utilizes the MSS Management Agent Services (EcAgCOTSManger) for its management.

6.13.6.2 Operator Interfaces

The operator interface for this CSC is provided by the individual COTS products. The Tivoli Desktop will be used to present an integrated desktop environment for the operators. From the desktop, the operator will be able to access Tivoli functions directly and the operator will be able to launch the other COTS GUIs from icons on the desktop. The Tivoli Desktop also provides a mechanism to limit each operator's view into the management toolset. The Tivoli Administrator operator will setup each operator's desktop and assign operators privileges that will limit what tools the operator can access.

6.13.6.3 Reports

This CSC is a collection of COTS products. The reports which are available are documented in the vendor documentation for each product.