



APPENDIX **C**

OSS Use Cases and Client Development

This appendix lists interceptors and use cases to resynchronize the NMS from the EMS.

- [C.1 Use Cases to Resynchronize the NMS from the EMS, page C-1](#)
- [C.2 Developing a CTM GateWay/CORBA Client, page C-4](#)

C.1 Use Cases to Resynchronize the NMS from the EMS

[Table C-1](#), [Table C-2](#), and [Table C-3](#) list use cases to resynchronize the NMS from the EMS.

Table C-1 *Use Cases to Resynchronize the NMS from the EMS*

Information	Detail
Name	The NMS retrieves CTM information.
Summary	The NMS retrieves all EMS and ManagedElement alarms and information for existing MultiLayerSubnetworks, ManagedElements, PTPs, CTPs, SNCs, and protection groups from CTM GateWay/CORBA interfaces.
Actor(s)	NMS.
Pre-Conditions	The NMS locates the EmsSessionFactory object and obtains references to EMSMgr_I, MultiLayerSubnetworkMgr_I, ManagedElementMgr_I, EquipmentInventoryMgr_I, MaintenanceMgr_I, ProtectionMgr_I, and PerformanceManagementMgr_I interfaces.
Begins When	The NMS sends a request to retrieve all of the CTM information.

Table C-1 Use Cases to Resynchronize the NMS from the EMS (continued)

Information	Detail
Description	<p>The NMS requests the following information from the CTM GateWay/CORBA interface:</p> <ol style="list-style-type: none"> 1. The NMS uses the <code>emsMgr::EMSMgr_I::getAllEMSSystemActiveAlarms</code> interface to request all EMS alarms. 2. The NMS uses the <code>emsMgr::EMSMgr_I::getAllTopLevelSubnetworks</code> interface to request all multilayer subnetworks in CTM. CTM returns a list of <code>MultiLayerSubnetwork_T</code> objects with detailed information about a <code>MultiLayerSubnetwork</code>. 3. For each <code>MultiLayerSubnetwork_T</code> object, the NMS uses the <code>multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::getAllManagedElements</code> interface to request all MEs that belong to the same multilayer subnetwork. CTM returns a list of <code>ManagedElement_T</code> objects with detailed information about a <code>ManagedElement</code>. For each <code>ManagedElement</code> object, the NMS executes the use case “NMS Resynchronizes Information Specific to a Managed Element.” 4. The NMS uses the <code>emsMgr::EMSMgr_I::getAllTopLevelTopologicalLinks</code> interface to request all topological links associated with the multilayer subnetwork. CTM returns a list of <code>TopologicalLink_T</code> objects with detailed information about a topological link. 5. The NMS uses the <code>multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::getAllVLANs</code> interface to request all VLANs associated with the multilayer subnetwork.
Ends When	The NMS retrieves all CTM information.
Exceptions	Refer to the exceptions thrown by the individual interface method.
Post-Conditions	<ul style="list-style-type: none"> • The NMS synchronizes with CTM. • The NMS registers with CTM to retrieve notifications related to changes in the managed object on CTM, new alarms, and TCAs.
Traceability	—

Table C-2 NMS Resynchronizes Information Specific to a Managed Element

Information	Detail
Name	The NMS retrieves CTM information specific to the managed element.
Summary	The NMS retrieves all PTP, CTP, SNC, and protection group information for the specific ME.
Actor(s)	NMS.
Pre-Conditions	The NMS obtains references to <code>ManagedElementMgr_I</code> , <code>EquipmentInventoryMgr_I</code> , <code>MaintenanceMgr_I</code> , <code>ProtectionMgr_I</code> , and <code>PerformanceManagementMgr_I</code> interfaces. The NMS also obtains the ME name.
Begins When	The NMS sends a request to retrieve all information about an ME. The NMS identifies the name of the ME for which information is to be retrieved.

Table C-2 *NMS Resynchronizes Information Specific to a Managed Element (continued)*

Information	Detail
Description	<p>The NMS requests the following information from the CTM GateWay/CORBA interface:</p> <ol style="list-style-type: none"> 1. The NMS uses the managedElementManager::ManagedElementManager_I::getAllActiveAlarms interface to request all current alarms on the ME. 2. The NMS uses the equipment::EquipmentInventoryMgr_I::getAllEquipment interface to request all existing equipment on the ME. CTM returns a list of EquipmentHolder_T objects with detailed information about equipment holders and equipment on the ME. 3. For the equipment information retrieved in EquipmentHolder_T object, the NMS uses the equipment::EquipmentInventoryMgr_I::getAllSupportedPTPs interface to request all supported PTPs on the equipment. CTM returns a list of TerminationPoint_T objects with detailed PTP information, such as SDH_SONET port, AdminState, ServiceState, LineCode, FrameFormat, and so on. 4. For each TerminationPoint_T object, the NMS executes the “NMS Resynchronizes Information Specific to TerminationPoint” use case. 5. The NMS uses the managedElementManager::ManagedElementManager_I::getAllSNCs interface to request all SNCs originating, terminating, or passing through the ME. CTM returns a list of SubnetworkConnection_T objects with detailed SNC information, such as layer rate, SNC state, SNC protection state, aEnd CTP, zEnd CTP, and so on. 6. The NMS uses the protection::ProtectionMgr_I::getAllProtectionGroups interface to request all protection groups created on the ME. CTM returns a list of ProtectionGroup_T objects with detailed protection group information, such as 1:1, 1+1, or 2F-BLSR; revertive or nonrevertive, and so on.
Ends When	The NMS retrieves all ME-specific information.
Exceptions	Refer to the exceptions thrown by the individual interface method.
Post-Conditions	The NMS synchronizes information about an ME with CTM.
Traceability	—

Table C-3 *NMS Resynchronizes Information Specific to TerminationPoint*

Information	Detail
Name	The NMS retrieves CTM information specific to the termination point.
Summary	The NMS retrieves all loopback and threshold information for PTPs and CTPs in use by the SNC.
Actor(s)	NMS.
Pre-Conditions	The NMS obtains references to ManagedElementMgr_I, MaintenanceMgr_I, and PerformanceManagementMgr_I interfaces. The NMS also obtains a list of PTPs.
Begins When	The NMS sends a request to retrieve all threshold and loopback information on a PTP and on any in-use CTPs. The NMS identifies the name and type (PTP/CTP) of the TP for which the information is to be retrieved.

Table C-3 NMS Resynchronizes Information Specific to TerminationPoint (continued)

Information	Detail
Description	<p>The NMS requests the following information from the CTM GateWay/CORBA interface:</p> <ol style="list-style-type: none"> 1. The NMS uses the <code>maintenanceOps::MaintenanceMgr_I::getActiveMaintenanceOperations</code> interface to request all loopback information on the TP. <p>CTM returns a list of <code>CurrentMaintenanceOperation_T</code> objects with detailed information about the type of loopback set on the PTP, if any. CTM supports two types of loopback, <code>FACILITY_LOOPBACK</code> and <code>TERMINAL_LOOPBACK</code>. If there are no loopbacks set on the TP, the returned list is empty.</p> 2. The NMS uses the <code>performance::PerformanceManagementMgr_I::getTCATPPParameter</code> interface to request the threshold values set for all PM thresholds on the given TP object. <p>CTM returns the <code>TCAPParameter_T</code> object with detailed information about all PM threshold name/value pairs for the PTP.</p> 3. For each PTP specified by the NMS as the TP, the NMS uses the <code>managedElementManager::ManagedElementManager_I::getContainedInUseTPs</code> interface to request all CTPs contained in the PTP participating in an SNC. <p>CTM returns a list of <code>TerminationPoint_T</code> objects with detailed information about the CTPs participating in an SNC and the associated provisioning details, such as <code>IPPMonitor</code> status, <code>J1</code> path trace status, and so on.</p> 4. For each CTP object, the NMS uses the <code>performance::PerformanceManagementMgr_I::getTCATPPParameter</code> interface to request threshold information. <p>CTM returns a list of <code>TCAPParameter_T</code> objects with detailed information about the PM threshold name/value pairs for the CTP.</p>
Ends When	The NMS retrieves loopback and threshold information for all PTPs and CTPs.
Exceptions	Refer to the exceptions thrown by the individual interface method.
Post-Conditions	The NMS synchronizes information about a TP object with CTM.
Traceability	—

C.2 Developing a CTM GateWay/CORBA Client

CTM GateWay/CORBA implements Application Programming Interfaces (APIs) defined by the TeleManagement Forum's Multi-Technology Network Management (MTNM) group. These APIs are defined for communication between an NMS and the EMS. The NMS must develop a client application that uses these APIs. The following sections describe the tools required for developing a client application. Sample code is provided.

C.2.1 Recommended Tools

You can develop the CORBA client on UNIX or PC platforms. Cisco recommends the following development tools:

- Sun Microsystems Java Development Kit (JDK) 1.4.2 (available on <http://java.sun.com/javase/downloads/index.jsp>)

- JacORB 2.x

C.2.2 Sample Code in Java

A typical CORBA client application involves the following steps:

-
- Step 1** Initialize the client connection to the object request broker (ORB).
 - Step 2** Obtain a reference to the naming service.
 - Step 3** Resolve the reference to EMSSessionFactory. See [C.2.2.3 Get Reference to EMSSessionFactory](#), page C-7.
 - Step 4** Implement an NmsSession_IOperations. See [C.2.2.4 Implement NmsSession_IOperations](#), page C-7.
 - Step 5** Retrieve an EMSSession by supplying the username and password. See [C.2.2.5 Log In and Retrieve EmsSession](#), page C-8.
 - Step 6** Query the EMSSession and obtain a list of managers available for operations. See [C.2.2.6 Retrieve List of Managers](#), page C-9.
 - Step 7** Invoke the desired method on that manager. See [C.2.2.7 getEMS Operation on EMS Manager](#), page C-9.
If you want your client to receive notifications from CTM GateWay/CORBA, the following additional steps are required:
 - Step 8** Obtain a reference to an EventChannel object in the notification server in [Step 2](#).
 - Step 9** Obtain a ConsumerAdmin object reference.
 - Step 10** Invoke obtain_notification_push_consumer() on the SupplierAdmin object, specifying CosNotifyChannelAdmin::STRUCTURED_EVENT as a parameter. This operation returns a reference to the StructuredProxyPushSupplier.
 - Step 11** Implement an instance of StructuredPushConsumer (defined by the OMG).
 - Step 12** Invoke the connect_structured_push_consumer() operation on the StructuredProxyPushSupplier object to connect the StructuredPushConsumer implementation object to the notification channel.
 - Step 13** Monitor incoming notifications.
-

The following sections show sample Java code for the steps required to develop a client application.



Note

The sample code shown is for example purposes only; the code might not compile as shown. The sample code does not handle all exceptions.

C.2.2.1 Initialize the Client Connection

```
// Import OMG packages
import org.omg.CORBA.IntHolder;
import org.omg.CORBA.ORB;
import org.omg.CORBA.Policy;

// Import naming context packages.
import org.omg.CosNaming.NameComponent;
import org.omg.CosNaming.NamingContextExt;
import org.omg.CosNaming.NamingContextExtHelper;
```

```

// Import notify channel and event service packages.
import org.omg.CosEventChannelAdmin.AlreadyConnected;
import org.omg.CosEventChannelAdmin.TypeError;
import org.omg.CosNotifyChannelAdmin.AdminLimitExceeded;
import org.omg.CosNotifyChannelAdmin.AdminNotFound;
import org.omg.CosNotifyChannelAdmin.ClientType;
import org.omg.CosNotifyChannelAdmin.ConsumerAdmin;
import org.omg.CosNotifyChannelAdmin.EventChannel;
import org.omg.CosNotifyChannelAdmin.EventChannelHolder;
import org.omg.CosNotifyChannelAdmin.ProxySupplier;
import org.omg.CosNotifyChannelAdmin.StructuredProxyPushSupplier;
import org.omg.CosNotifyChannelAdmin.StructuredProxyPushSupplierHelper;
import org.omg.CosNotifyComm.StructuredPushConsumer;
import org.omg.CosNotifyComm.StructuredPushConsumerPOATie;

// Import POA packages
import org.omg.PortableServer.POA;
import org.omg.PortableServer.POAHelper;
import org.omg.PortableServer.POAPackage.ServantAlreadyActive;
import org.omg.PortableServer.POAPackage.WrongPolicy;

// Import TMF packages
import org.tmforum.mtnm.emsSession.EmsSession_I;
import org.tmforum.mtnm.emsSession.EmsSession_IHolder;
import org.tmforum.mtnm.emsSession.EmsSession_IPackage.managerNames_THolder;
import org.tmforum.mtnm.emsSessionFactory.EmsSessionFactory_I;
import org.tmforum.mtnm.emsSessionFactory.EmsSessionFactory_IHelper;
import org.tmforum.mtnm.nmsSession.NmsSession_I;
import org.tmforum.mtnm.nmsSession.NmsSession_IPOATie;
public static void main(String[] args)
{
    try {
        // Optional: set up ORB properties
        // Properties sys_properties = System.getProperties();
        // For Orbix 6.2 one may want to define the system properties
        // sys_properties.put("org.omg.CORBA.ORBClass",
        "com.ionacorba.art.artimpl.ORBImpl");
        // sys_properties.put("org.omg.CORBA.ORBSingletonClass",
        "com.ionacorba.art.artimpl.ORBSingleton");

        // Step 1: Initialize the ORB and obtain ROOT POA reference
        // Note: ORB_init will process any -ORB arguments
        // be called before any other argument processing.
        //
        global_orb = ORB.init(args, null);
        org.omg.CORBA.Object root_poa = global_orb.resolve_initial_references("RootPOA");
        POA rpoa = POAHelper.narrow(root_poa);
        POA poa = rpoa.create_POA("myPolicy",null, new Policy[0]);
        poa.the_POAManager().activate();
        orb = ORB.init(args, null);
    }
    catch (SystemException ex) {
        // Exception handling
    }
}

```

C.2.2.2 Get Reference to the Naming Service

```

// Get Nameservice reference
NamingContext nsRootContext = null;

```

```

try {
    // Step 2: Get reference to the name service
    // Option 1: Resolve initial reference (RIR)
    // org.omg.CORBA.Object obj = global_orb.resolve_initial_references("NameService");
    // or

    // Option 2: corbaloc URL
    String objRef = "corbaloc:iiop:gatewayserver.cisco.com:14005/NameServiceGWC";
    org.omg.CORBA.Object obj = global_orb.string_to_object(objRef);
    /* NOTE: Please replace "gatewayserver.cisco.com" with the name of the server on
    which CORBA Naming service is running. */
    // Narrow to root naming context
    NamingContextExt root_context = NamingContextExtHelper.narrow(obj);

}
catch (org.omg.CORBA.ORBPackage.InvalidName inEx) {
    // Exception handling
}

```

C.2.2.3 Get Reference to EMSSessionFactory

Follow the example in [C.2.2.2 Get Reference to the Naming Service](#), page C-6 to obtain a reference to the naming service.

```

NameComponent name = new NameComponent[6];
name[0] = new NameComponent("TMF_MTNM", "Class");
name[1] = new NameComponent("Cisco Systems", "Vendor");
name[2] = new NameComponent("Cisco Transport Manager", "EMSInstance");
name[3] = new NameComponent(version, "Version"); //where version = "8_0" for CTM 8.0
name[4] = new NameComponent(ctm_sys_id, "EMS"); // ctm_sys_id = "CTM"
name[5] = new NameComponent("SessionFactory", "EmsSessionFactory");
try {
    org.omg.CORBA.Object emsSessionI = root_context.resolve(name);
}
catch (InvalidName inEx) {
    // Exception handling
}
catch (NotFound nfEx) {
    // Exception handling
}

```

C.2.2.4 Implement NmsSession_IOperations

```

import org.tmforum.mtnm.session.*;
import org.tmforum.mtnm.nmsSession.*;

public class SessionImpl implements NmsSession_IOperations {

    Session_I myAssociatedSession = null;

    public SessionImpl() {
        super();
        // TODO Auto-generated constructor stub
    }

    public void setAssociatedSession(Session_I emsSession) {
        myAssociatedSession = emsSession;
    }

    public Session_I getAssoicatedSession () { return myAssociatedSession;}
}

```

```

public void eventLossOccurred(String startTime, String notificationId) {
    // TODO Auto-generated method stub

}

public void eventLossCleared(String endTime) {
    // TODO Auto-generated method stub

}

public void historyPMDDataCompleted(String fileName) {
    // TODO Auto-generated method stub

}

public void historyPMDDataFailed(String errorReason) {
    // TODO Auto-generated method stub

}

public Session_I associatedSession() {
    // TODO Auto-generated method stub
    return null;
}

public void ping() {
    // TODO Auto-generated method stub

}

public void endSession() {
    // TODO Auto-generated method stub

}

}
}

```

C.2.2.5 Log In and Retrieve EmsSession

To perform operations from CTM GateWay/CORBA, your client must log in using a username and password created on the CTM client. See [B.1 Creating an OSS Client Profile for CTM GateWay/CORBA, page B-1](#).

```

EmsSession_I m_emsSession = null;
SessionImpl mySessionImpl = new SessionImpl();
try {
    EmsSessionFactory_I ems_ref = EmsSessionFactory_IHelper.narrow(emsSessionI);
    EmsSession_IHolder emsSessionHldr = new EmsSession_IHolder();
    NmsSession_IPOATie tieobj = new NmsSession_IPOATie(mySessionImpl, poa);
    poa.activate_object(tieobj);
    NmsSession_I nmsSession_ref = tieobj._this();
    if (ems_ref != null) {
        ems_ref.getEmsSession(user, password, nmsSession_ref, emsSessionHldr);
        m_emsSession = emsSessionHldr.value;
    }

} catch (Exception ex) {
    // System.out.println("Could not narrow");
    ex.printStackTrace();
}

```

```
    }
```

C.2.2.6 Retrieve List of Managers

```
managerNames_THolder names = new managerNames_THolder();
m_emsSession.getSupportedManagers(names);
managers = names.value;
for (i = 0; i < managers.length; i++ )
{
    System.out.print("Manager ");
    System.out.print(i);
    System.out.println(" " + managers[i]);
}
}
```

C.2.2.7 getEMS Operation on EMS Manager

```
EMS_T m_ems;
EMS_THolder m_emsHolder = new EMS_THolder();
try {
    Common_IHolder mgrHolder = new Common_IHolder();
    m_emsSession.getManager("EMS", mgrHolder);
    EMSMgr_I emsMgr = EMSMgr_IHelper.narrow(mgrHolder.value);
    emsMgr.getEMS(m_emsHolder);
}
catch (ProcessingFailureException pfe) {
    System.out.println("Processing Exception" + pfe.getMessage());
    pfe.printStackTrace();
}
m_ems = m_emsHolder.value;
System.out.println("Native EMS Name" + m_ems.nativeEMSName);
```

C.2.2.8 Get Reference to EventChannel

```
EventChannel notifChannel;
EventChannelHolder chanHolder = new EventChannelHolder();
try {
    ...
    emsSession.getEventChannel(chanHolder);
}
catch (Exception ex){
    // handle exceptions
}
notifChannel = chanHolder.value;
```

C.2.2.9 Obtain ConsumerAdmin Reference

```
//retrieve default consumer admin
try {
    ConsumerAdmin cadmin = notifChannel.get_consumeradmin(0);
}
catch (AdminNotFound anfSe) {
    // Exception handling
}
}
```

C.2.2.10 Obtain ProxyPushSupplier

```
IntHolder id = new IntHolder();
```

```

try {
    ProxySupplier baseSupplier =
        cadmin.obtain_notification_push_supplier(
            ClientType.STRUCTURED_EVENT, id);
    structuredProxyPushSupplier =
        StructuredProxyPushSupplierHelper.narrow(baseSupplier);
}
catch (AdminLimitExceeded aleEx) {
    // Exception handling
}

```

C.2.2.11 Implement StructuredPushConsumer

```

class StructuredPushConsumerImpl extends _StructuredPushConsumerImplBase
{
    StructuredPushConsumerImpl() {
        super();
        System.out.println("StructuredPushConsumerImpl created.");
    }
    public void disconnect_structured_push_consumer() {
        System.out.println("Disconnect structured push consumer.");
    }
    public void push_structured_event(StructuredEvent notification) {
        System.out.println("Received notification.");
    }
    public void offer_change(EventType[] added,
        EventType[] removed)
        throws InvalidEventType
    {
        System.out.println("Offer changed.");
    }
}

```

C.2.2.12 Connect StructuredPushConsumerImpl

```

try {
    StructuredPushConsumerImpl structProxyPushConsumer = new StructuredPushConsumerImpl();
    StructuredPushConsumerPOATie structuredPushConsumerTieObj = new
    StructuredPushConsumerPOATie (structProxyPushConsumer, poa);
    poa.activate_object(structuredPushConsumerTieObj);
    StructuredPushConsumer pushCon = structuredPushConsumerTieObj._this();
    structuredProxyPushSupplier.connect_structured_push_consumer(pushCon);

    global_orb.run();
}
catch (ServantAlreadyActive sae) {
    // Exception handling
}
catch (WrongPolicy wrongPolicyEx) {
    // Exception handling
}

```

C.2.3 Running the Client

If the initial naming context is not resolved using the option 2 method shown in [C.2.2.2 Get Reference to the Naming Service, page C-6](#), the following JVM flags must be used for CORBA clients written for Orbix 6.2, Visibroker, or jacORB:

Orbix 6.2:

```
-ORBInitRef NameService=corbaloc:iiop:1.2@<hostname>:<port>/NameServiceGWC
```

Visibroker:

```
-DORBInitRef NameService=corbaloc::<hostname>:<port>/NameServiceGWC or  
-DORBInitRef NameService=corbaname::<hostname>:<port>/NameServiceGWC
```

JacORB:

```
-DORBInitRef.NameService=corbaname::<hostname>:<port>/NameServiceGWC
```

