iManager U2000 Unified Network Management System V200R018C60

CORBA NBI Programming Guide

Issue	**01**
Date	**2018-10-31**

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Contents

1 Interface Operation Principles 1
1.1 Technology Description 1
1.2 Principles 1
1.3 Compliant Protocol 3
2 Management Model 4
2.1 Object Model 4
2.2 Mapping Description of Managed Objects 6
2.3 Object Naming Rules 7
2.3.1 Naming Rules and References of Objects Defined by TMF Standards 7
2.3.2 Naming Rules of Objects of the U2000 CORBA NBI 10
2.4 Equipment Model 14
3 Code Implementation 15
3.1 Making Preparations for CORBA NBI Invoke 15
3.2 Selecting a Middleware 16
3.3 Installing the JacORB 16
3.4 Obtaining the IDL File 17
3.5 Compiling and Compressing the IDL File 17
3.6 A Programming Example of Client Codes in Java 19
3.6.1 Obtaining the EmsSession and Querying the Manager List Supported by the U2000 20
3.6.2 Subscribing to an Event Notification 25
3.6.3 Notification Subscription 31
3.6.4 Syntaxes for Notification Parameter Filtering 31
3.6.5 Filtered Parameters in Notifications 32
3.7 Running Codes 33
4 Test and Verification 36
4.1 Testing and Verifying the CORBA NBI by Using the CORBA Explorer 36
4.2 Testing and Verifying the CORBA NBI by Using the JacORB Notification Service 36
4.3 Precautions About CORBA NBI Verification 37
4.3.1 Checking the CORBA NBI Configuration 37
4.3.2 Checking the Communication Mode of the CORBA NBI 38
4.3.3 Checking the Configuration of the SSL Mode 38

1 Interface Operation Principles

1.1 Technology Description

Common Object Request Broker Architecture (CORBA) is a technology that incorporates the object-oriented programming (OOP), distributed computing, multi-tier architecture, and interface technologies. CORBA is an industry standard defining the communication between objects.

The CORBA technology involves the following concepts:

- An interface definition language (IDL)
- A mapping between the IDL and a specific advanced programming language
- The Internet Inter-ORB Protocol (IIOP)
- The object referencing format that is transmitted over networks

The basic features of the CORBA technology are:

- Interoperation that features cross-platform, cross-language, and cross-ORB
- Distributed data transfer that features plug-and-play

1.2 Principles

The U2000 CORBA northbound interface (NBI) provides the standard naming service and notification service. Only with the naming service can the U2000 provide the OSS with a unique entry to the U2000 CORBA NBI. The U2000 interconnects with the OSS in the multiple-to-multiple mode. Specifically, one U2000 can interconnect with multiple OSSs, and one OSS can also interconnect with multiple U2000s. You only need to ensure that the name of the managed U2000 is unique in the management domain of the OSS. With the notification service, the U2000 can notify the OSS of configuration changes in details when its configuration data is changed. Besides, the notification service serves to ensure data consistency between the OSS and the U2000.

The Naming Service maps names to object references. Name binding is a name-to-reference association. The same object reference can be stored several times under different names. A naming context object implements a table that maps names to object references. And a hierarchy of contexts and bindings is known as a naming graph.

Figure 1-1 and Figure 1-2 show the naming graphs in the TMF and the U2000 CORBA NBI.

Figure 1-1 Naming graph in the TMF

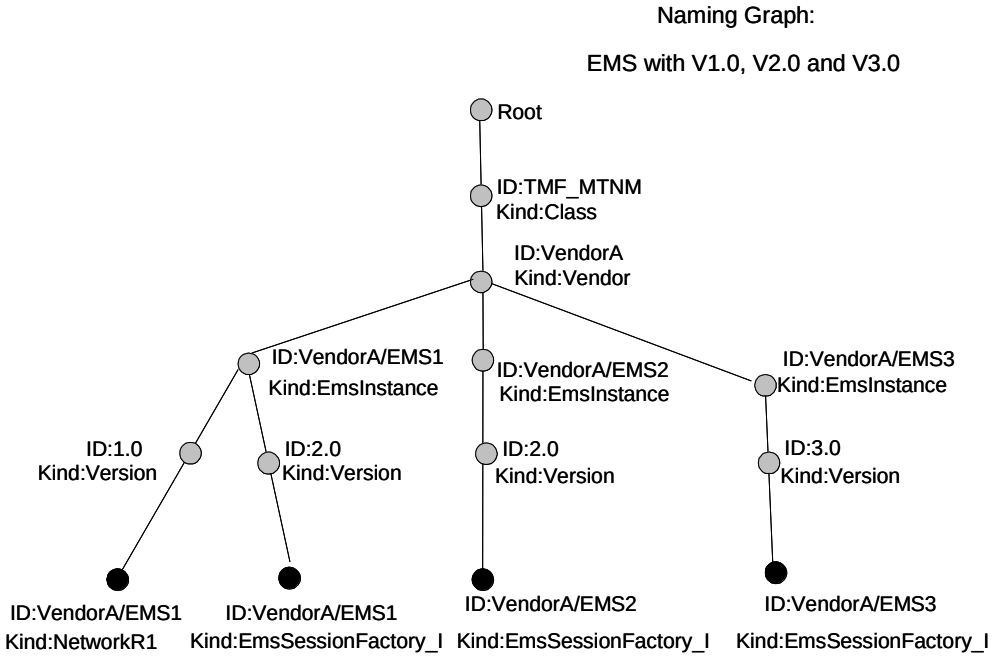
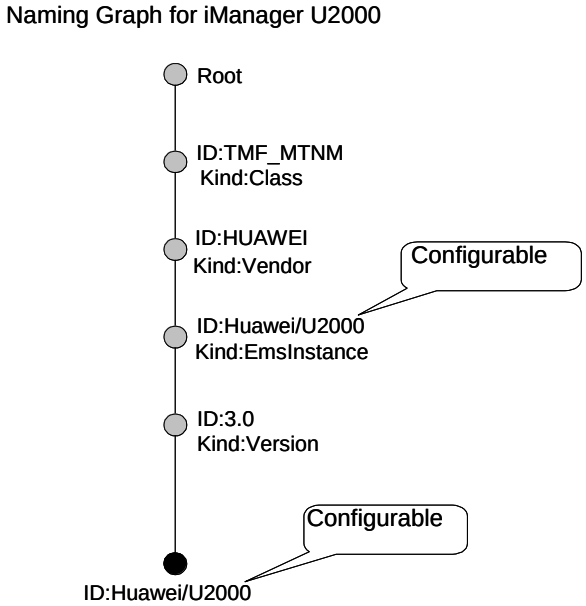


Figure 1-2 Naming graph of the U2000 CORBA NBI



Based on the naming graph of the U2000 CORBA NBI, the OSS can interconnect with the U2000 CORBA NBI by using the naming service. The name of the U2000 (Huawei/ U2000) can be set in the system configuration file. Multiple U2000s can be connected to one OSS at the same time.

1.3 Compliant Protocol

The U2000 complies with the TeleManagement Forum (TMF) multi technology network management (MTNM) series standards and provides the CORBA IDL interface between the network management layer (NML) and the element management layer (EML). Through the CORBA IDL interface, the upper-layer OSS can manage Huawei network management systems.

The U2000 CORBA IDL interface complies with the following TMF standards:

- TMF 513 MTNM Business Agreement
- TMF 608 MTNM Information Agreement
- TMF 814 MTNM CORBA Solution Set
- TMF 814A MTNM Implementation Statement

The U2000 CORBA IDL interface supports the functions of the following standard interfaces:

- Most MTNM V2.1-based interface
- Some MTNM V3.0-based interfaces
- Some MTNM V3.5-based interfaces

The U2000 CORBA NBI provides the following features:

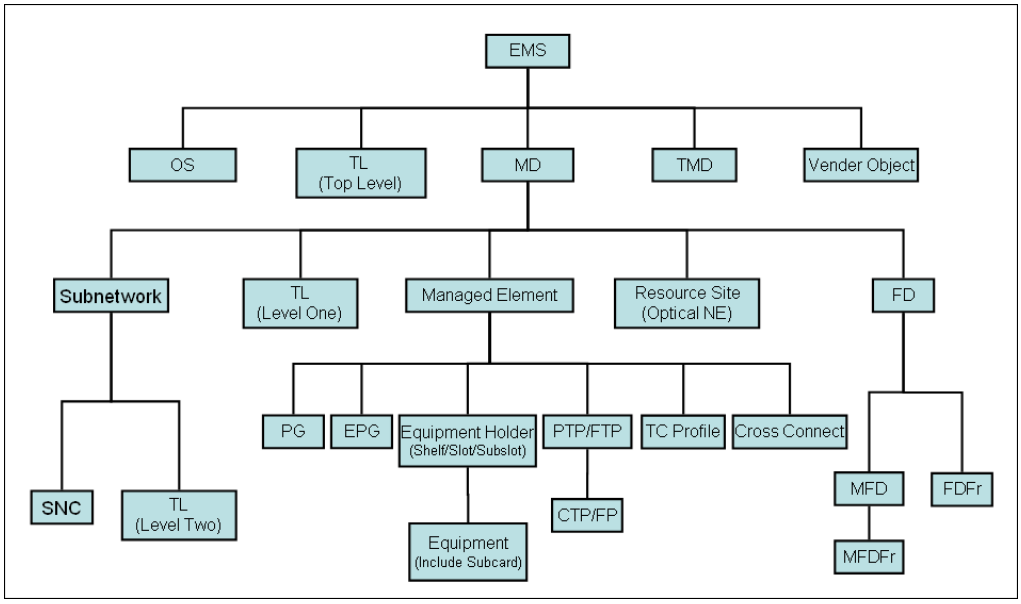
- Fully complying with the OMG CORBA 2.6 specifications and supporting the IIOP 1.1 and IIOP 1.2 protocols.
- Adopting the standard CORBA naming service 1.1 and notification service 1.0.
- Achieving highly efficient implementation based on the ACR ORB (TAO) 1.5.
- Supporting the intercommunications between different ORBs on the ORB platforms such as the IONA Orbix2000, IONA Orbix 6.1, InterBus, JacORB, Borland VisiBroker, and Borland BES protocols.
- Supporting cross-OS platforms, such as Windows, Solaris, and Linux OSs.
- Supporting the standard secure sockets layer (SSL) protocol to allow for different secure access control modes after a simple configuration. Currently, the IIOP and SSLIOP access modes are supported.

2 Management Model

2.1 Object Model

Figure 2-1 shows the model of the objects managed by the U2000.

Figure 2-1 Managed object model



The following uses a physical object as an example to describe the object model in detail. CORBA presents involved objects in the following sequence from the top layer: Element management system (EMS) > Managed element (ME) > Slot > Physical termination point (PTP) > Connection termination point (CTP). CORBA presents objects by using the name-value pairs described in section 2.3.1 in the preceding sequence.

1. EMS

   An NMS manages multiple EMSs and an EMS can be managed by multiple NMSs at a time. An EMS is identified by a unique name on one NMS. For example, the NMS identifies an EMS with the name-value "*name* EMS value Huawei/U2000."As defined by TMF standards, *name* indicates the name of an EMS. The names of most name-value

pairs have already been defined and vendors only need to customize *value*, as described in section 2.3.1)

2. Managed NE (ME)

   CORBA describes an NE with the following parameters: NE name, userLabel (user label), nativeEMSName (name of the local EMS), Owner, Location, Version, productName (product name), communicationState (communication status), emsInSyncState (synchronization status), supportedRates (layer rate supported by NEs), and additional information. The TMF-defined structure of an NE is as follows:

   ManagedElement_T

   ```
   struct ManagedElement_T {
       globaldefs::NamingAttributes_T name;
       string userLabel;
       string nativeEMSName;
       string owner;
       string location;
       string version;
       string productName;
       CommunicationState_T communicationState;
       boolean emsInSyncState;
       transmissionParameters::LayerRateList_T supportedRates;
       globaldefs::NVSList_T additionalInfo;
   };
   ```

3. Slot

   TMF defines the physical architecture of an NE such as racks, shelves, and slots. In addition, the TMF also defines the frame of slots on a subrack and the order of slot IDs. The NMS, under certain circumstances, does not concern about the board where a specific port locates.

   The U2000 forms a hierarchy for slot IDs (refer to the ID of slots where boards locate) only. No rack hierarchy is available on the U2000. For details, see the PTP description.

4. PTP

   PTP refers to the terminals of topology connections. According to CORBA, the PTPs of an NE are the optical ports inserted with optical fibers. Currently, CORBA identifies the location of a PTP in the format of "*rack ID*/*shelf ID*/*slot ID*/*port ID*." In this format, *slot ID* indicates the ID of the slot where a board locates and *port ID* indicates the ID of a physical port. Note that a port described in CORBA implementation corresponds to a pair of physical ports on a board. As shown in the topology view on the U2000, a user creates a fiber in the topology view, physically, there is a pair of fibers. Physical optical ports on a board appear in pairs. If one optical port is marked with **in**, the other port is **out** event if no mark exists.

   The TMF defines the architecture of TPs (represent points, PTPs, and CTPs) as follows:

TerminationPoint_T

```
struct TerminationPoint_T {
    globaldefs::NamingAttributes_T name;
    string userLabel;
    string nativeEMSName;
    string owner;
    globaldefs::NamingAttributes_T ingressTrafficDescriptorName;
    globaldefs::NamingAttributes_T egressTrafficDescriptorName;
    TPType_T type;
    TPConnectionState_T connectionState;
    TerminationMode_T tpMappingMode;
    Directionality_T direction;
    transmissionParameters::LayeredParameterList_T transmissionParams;
    TPProtectionAssociation_T tpProtectionAssociation;
    boolean edgePoint;
    globaldefs::NVSList_T additionalInfo;
};
```

5. CTP

   A PTP incorporates a CTP and a CTP is incorporated into a PTP. A PTP serves as the container of a CTP and a CTP functions as the logical port of a PTP. The application of the CTP depends on the actual service configuration. The CTP architecture is the same as the PTP architecture.

2.2 Mapping Description of Managed Objects

Managed Object	Standard Object	Standard
NMS	OS	TMF814/MTOSI
NE	ME	TMF814/MTOSI
Shelf	Shelf	TMF814/MTOSI
Slot	Slot	TMF814/MTOSI
Subslot	Sub_Slot	TMF814/MTOSI
Board	Equipment	TMF814/MTOSI
Subboard	Equipment	TMF814/MTOSI
Physical port	PTP	TMF814/MTOSI

Managed Object	Standard Object	Standard
Logical port	FTP	TMF814/MTOSI
Protection	PG/EPG	TMF 864/MTOSI
Trail management	SNC	TMF 864/MTOSI
PWE3 service	MFDFr	TMF814/MTOSI
VSI service	MFDFr	TMF814/MTOSI
PW service	FTP	TMF814/MTOSI
VLAN managed object	FTP	TMF814/MTOSI
PW switch	IPCrossConnection	Extended by HUAWEI
MPLS Static Tunnel	IPCrossConnection	Extended by HUAWEI
IP tunnel	TrafficTrunk	Extended by HUAWEI
MPLS Dynamic Tunnel	TrafficTrunk	Extended by HUAWEI
Maintenance domain of OAM management	MD	Extended by HUAWEI
Maintenance association of OAM management	MA	Extended by HUAWEI
Remote maintenance point	MEP	Extended by HUAWEI
Maintenance intermediate point	MIP	Extended by HUAWEI

2.3 Object Naming Rules

This section provides an example to describe rules for naming various objects of the U2000 CORBA NBI according to the naming rules defined by TMF standards.

2.3.1 Naming Rules and References of Objects Defined by TMF Standards

Naming Convention

Description	Remarks
The strings used for the value field of the name-value pairs will be at most 1024 characters	N/A
White-space character allowed in the value but with no leading or trailing spaces.	N/A
All name and value strings are case-sensitive.	N/A

Description	Remarks
The value field is a free format string assigned by each EMS and is not standardized across this interface except for the EMS, CTP, EquipmentHolder, and Equipment names	N/A

Object Naming Requirement

Managed Object	Standard Naming Format	Remarks
EMS	1. name="EMS"; value="CompanyName/EMSname"	TMF MTNM/MTOSI
Subnetwork	1. name="EMS"; value="CompanyName/EMSname" 2. name="MultiLayerSubnetwork"; value="SubnetworkName"	TMF MTNM/MTOSI
Subnetwork connection	1. name="EMS"; value="CompanyName/EMSname" 2. name="MultiLayerSubnetwork"; value="SubnetworkName" 3. name="SubnetworkConnection"; value="SubnetworkConnectionName"	TMF MTNM/MTOSI
ManagedElement	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName"	TMF MTNM/MTOSI
Topological Link	1. name="EMS"; value="CompanyName/EMSname" 2. name="TopologicalLink"; value="TopologicalLinkName"	TMF MTNM/MTOSI
PTP	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="PTP"; value="PTPName"	TMF MTNM/MTOSI
FTP	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="FTP"; value="FTPName"	TMF MTNM/MTOSI

Managed Object	Standard Naming Format	Remarks
CTP	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="PTP"; value="PTPName" 4. name="CTP"; value="CTPName"	TMF MTNM/MTOSI
Equipment Holder	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="EquipmentHolder"; value="EquipmentHolderName"	TMF MTNM/MTOSI
Equipment	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="EquipmentHolder"; value="EquipmentHolderName" 4. name="Equipment"; value="EquipmentName"	TMF MTNM/MTOSI
ProtectionGroup	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="PGP"; value="ProtectionGroupName"	TMF MTNM/MTOSI
EquipmentProtectionGroup	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="EPGP"; value="EquipmentProtectionGroupName"	TMF MTNM/MTOSI
FlowDomain	1. name="EMS"; value="CompanyName/EMSname" 2. name="FlowDomain"; value=" FlowDomainName"	TMF MTNM/MTOSI
MatrixFlowDomain	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="MatrixFlowDomain"; value="MatrixFlowDomainName"	TMF MTNM/MTOSI

Managed Object	Standard Naming Format	Remarks
FlowDomainFragment	1. name="EMS"; value="CompanyName/EMSname" 2. name="FlowDomain"; value=" FlowDomainName" 3. name="FlowDomainFragment"; value="FlowDomainFragmentName"	TMF MTNM/MTOSI
AID	1. name="EMS"; value="CompanyName/EMSname" 2. name="ManagedElement"; value="ManagedElementName" 3. name="AID"; value="NameOfEntity"	TMF MTNM/MTOSI

2.3.2 Naming Rules of Objects of the U2000 CORBA NBI

Managed Object	Naming Format	Remarks
EMS	1. name="EMS"; value="Huawei/U2000"	Modify values in the configuration file.
Subnetwork	1. name="EMS"; value="Huawei/U2000" 2. name="MultiLayerSubnetwork"; value="1"	A managed domain maps a subnet. For a multilayer subnetwork (MLSN) object, modify its identifier in the configuration file.
Flow domain	1. name="EMS"; value="Huawei/U2000" 2. name="Flowdomain"; value="1"	A managed domain maps a flow domain (FD). For an FD object, modify its identifier in the configuration file.
Managed Element	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID"	N/A
Equipment holder	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="EquipmentHolder"; value="/rack=xx/shelf=xx[sub_slot=xx]/slot=xx"	In "/rack=xx/shelf=xx/sub_slot=xx /slot=xx ", *xx* indicates the rack ID, shelf ID, slot ID, and subslot ID respectively.

Managed Object	Naming Format	Remarks
Equipment	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="EquipmentHolder"; value="/rack=xx/shelf=xx[sub_slot=xx]/slot=xx" 4. name="Equipment"; value="1"	N/A
PTP	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="PTP"; value="/rack=xx/shelf=xx/slot=xx[sub_slot=xx]/type=xx/port=xx"	*xx* in **type=xx** indicates a port type. The values are **physical**, **lag**, **iptrunk**, **atmtrunk**, **serial**, **vctrunk**, **virtual**, **pw**, and **tunnel**.
FTP	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="PTP"; value="/rack=xx/shelf=xx/slot=xx[sub_slot=xx]/type=xx/port=xx"	N/A

Managed Object	Naming Format	Remarks
CTP/FTP	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="PTP/FTP"; value="[/shelf=xx][/slot=xx][/sub_slot=xx]/type=xx[/port=xx]" 4. name="CTP"; value="Ethernet CTP:[/eth=1][/ethvid=xx][/ethsvid=xx/ethcvid=xx] ATM CTP:[/atminterface=1][/vpi=xx][/vci=xx] MPLS CTP:[/tunnel=xx][/vc=xx] Channelized E1 CTP:[/ds0=1..31] SDH VC4 CTP:/sts3c_au4-j=1..256 SDH VC3 CTP:/sts3c_au4-j=1..256/tu3_vc3-k=1..3 SDH VC12 CTP:/sts3c_au4-j=1..256/vt2_tu12-k=1..3-l=1..7-m=1..3 PDH E4/E3/E1 CTP:[/sts3c_au4=1][/tu3_vc3=1][/vt2_tu2=1] WDM CTP:[/oms=1..n][/os=1][/och=1..192][/otuk=1..4][/oduk=1..4]/[dsr=1..n]"	**[/eth=1]**, **[/ethvid=xx]**, and **[/ethsvid=xx/ethcvid=xx]** indicate the scenarios wherein a VLAN, a single-layer VLAN tag, and a QinQ VLAN do not exit respectively. *xx* in **[/vpi=xx]** indicates a VPI value and *xx* in **[/vci=xx]** indicate a VCI value. *xx* in **[tunnel=xx]** indicates a tunnel tag. xx in **[vc=xx]** indicates a PW ID. *k* in **[/otuk=1..4]** and **[/oduk=1..4]** indicates the level of OTU and ODU respectively. It can be set to **1**, **2**, **3**, **5**, and **5G**.
SNC	1. name="EMS"; value="Huawei/U2000" 2. name="MultiLayerSubnetwork"; value="1" 3. name="SubnetworkConnection"; value=" SNC ID "	**SNID** indicates the unique ID of a subnetwork connection (SNC). This ID cannot be modified in the object life cycle.
TL	1. name="EMS"; value="Huawei/U2000" 2. name="TopologicalLink"; value="TL ID"	**TL ID** indicates the unique ID of a topological link (TL). This ID cannot be modified in the object life cycle.
FDFr	1. name="EMS"; value="Huawei/U2000" 2. name="Flowdomain"; value="1" 3. name="FlowdomainFragment"; value="FlowDomainFragmentID"	**FlowDomainFragmentID** indicates the unique ID of a flow domain fragment (FDFr). This ID cannot be modified in the object life cycle.

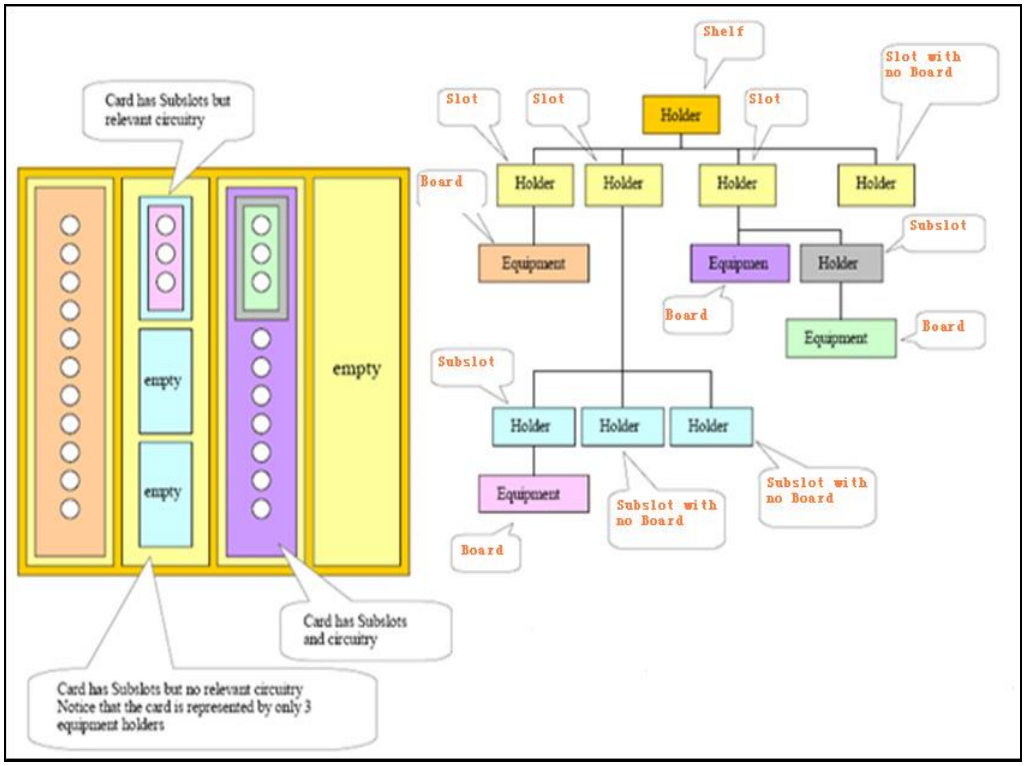
Managed Object	Naming Format	Remarks
Traffic Trunk	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="TrafficTrunk"; value="TT ID"	**TT ID** indicates the unique ID of a TT. This ID cannot be modified in the object life cycle.
IPCrossConnection	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="IPCrossConnection"; value="IPCrossConnection ID"	**IPCrossConnection ID** indicates the unique ID of a IPCrossConnection. This ID cannot be modified in the object life cycle.
MFDFr	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="MatrixFlowdomainFragment"; value="MatrixFlowdomainFragment ID"	N/A
PG	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="PGP"; value="PG ID"	PG ID indicates the unique ID of a PG. This ID cannot be modified in the object life cycle.
EPGP	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="EPGP"; value="EPGP ID"	EPGP ID indicates the unique ID of an EPGP. This ID cannot be modified in the object life cycle.
TrafficDescriptor	1. name="EMS"; value="Huawei/U2000" 2. name="ManagedElement"; value="ME ID" 3. name="TrafficDescriptor"; value=" TrafficDescriptor ID"	N/A
TC profile	1. name="EMS"; value="Huawei/U2000" 2. name="TCPROFILE"; value="TC Profile ID"	N/A

Managed Object	Naming Format	Remarks
AID	1. name="EMS"; value="Huawei /U2000" 2. name="ManagedElement"; value="ME ID" 3. name="AID"; value="[/shelf=xx][/slot=xx][/sub_slot=xx][/port=xx][/type=xx][/location1=xx][/location2=xx][/location3=xx][location4=xx]"	AID indicates an object that is not managed by the CORBA NBI.

2.4 Equipment Model

Figure 2-2 shows a TMF standard-compliant reference equipment model, each part of which corresponds to one layer of a physical board.

Figure 2-2 TMF standard-compliant equipment model



The U2000 equipment model is the same as that specified in the TMF standards. That is, holders in the model are subracks, slots, and subslots from top to bottom, and equipment in the model refers to boards.

3 Code Implementation

This chapter describes how to make preparations for CORBA NBI development and provides a code implementation example.

3.1 Making Preparations for CORBA NBI Invoke

Figure 3-1 shows the dependency between the U2000 client, U2000 server, and naming service. U2000 CORBA NBI is implemented with its object reference status being notified using the naming service. An external program can invoke the CORBA NBI only after obtaining the CORBA NBI's object reference by accessing the naming service's naming tree. Figure 3-2 shows the entire process of developing the CORBA client.

Figure 3-1 Relationships between the client, server, and naming service

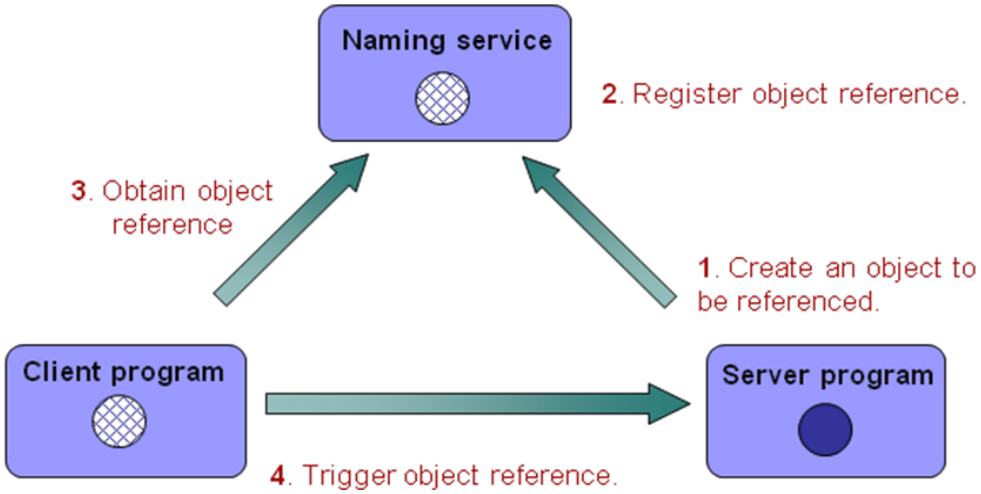
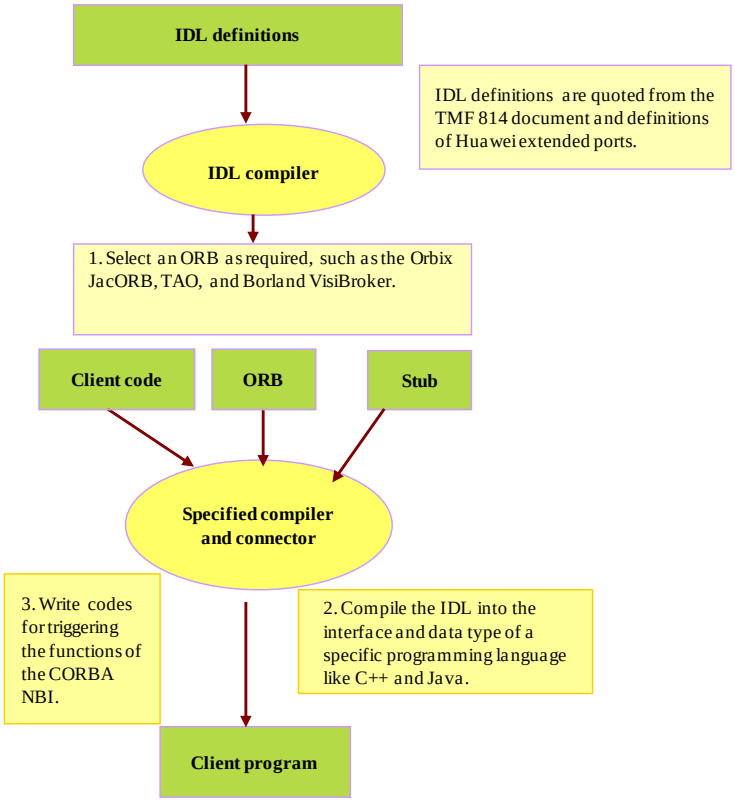


Figure 3-2 Process of developing the CORBA client



3.2 Selecting a Middleware

Select an appropriate ORB to develop the CORBA client.

Open-source ORBs are as follows:

- JacORB
- TAO

Commercial ORBs are as follows:

- Orbix
- VisiBroker

Select a middleware to develop the CORBA client. This section uses the JacORB as an example to describe how to develop the CORBA client. Download the JacORB from http://www.jacorb.org.

3.3 Installing the JacORB

Perform the following operations to install the JacORB.

Step 1 Decompress the JacORB package into **C:\JacORB-2.2.4**.

Step 2 Set the **JACORB_HOME** environment variable to **C:\JacORB-2.2.4**.

Step 3 Set the **PATH** environment variable to **%PATH%;%JACORB_HOME%\bin**.

Step 4 In the CLI, run the **idl –h** command to query commands for using the IDL file. Ensure that the IDL compiler of the JacORB is available. If the IDL compiler is not available, compile the **idl.bat** file in **%JACORB_HOME%\bin** as required.

----End

3.4 Obtaining the IDL File

Use the IDL file in the released document package.

3.5 Compiling and Compressing the IDL File

Perform the following operations to compile and compress the IDL file.

Step 1 Decompress the IDL file package into **D:\workspace\u2000_gen_idl**.

Step 2 Open the CLI and navigate to **D:\workspace\u2000_gen_idl**.

Step 3 Run the following command:

idl -IC:\JacORB-2.2.4\idl\omg -d java -i2jpackagefile format.txt D:\workspace\u2000_gen_idl\idl*.idl

For details and usage of the IDL compiler, see *JacORB Programming Guide*.

The **fomat.txt** file describes path rules for saving Java class packages that are generated. The **fomat.txt** file contains the following:

circuitCutMgr:mtnm.tmforum.org.circuitCutMgr

common:mtnm.tmforum.org.common

emsMgr:mtnm.tmforum.org.emsMgr

emsSession:mtnm.tmforum.org.emsSession

emsSessionFactory:mtnm.tmforum.org.emsSessionFactory

encapsulationLayerLink:mtnm.tmforum.org.encapsulationLayerLink

equipment:mtnm.tmforum.org.equipment

flowDomain:mtnm.tmforum.org.flowDomain

flowDomainFragment:mtnm.tmforum.org.flowDomainFragment

globaldefs:mtnm.tmforum.org.globaldefs

guiCutThrough:mtnm.tmforum.org.guiCutThrough

HW_controlPlane:mtnm.huawei.com.HW_controlPlane

HW_mstpInventory:mtnm.huawei.com.HW_mstpInventoryMgr

HW_mstpProtection:mtnm.huawei.com.HW_mstpProtection

HW_mstpService:mtnm.huawei.com.HW_mstpService

HW_securityMgr:mtnm.huawei.com.HW_securityMgr

HW_vpnManager:mtnm.huawei.com.HW_vpnManager

maintenanceOps:mtnm.tmforum.org.maintenanceOps

managedElement:mtnm.tmforum.org.managedElement

managedElementManager:mtnm.tmforum.org.managedElementManager

mtnmVersion:mtnm.tmforum.org.mtnmVersion

multiLayerSubnetwork:mtnm.tmforum.org.multiLayerSubnetwork

nmsSession:mtnm.tmforum.org.nmsSession

notifications:mtnm.tmforum.org.notifications

performance:mtnm.tmforum.org.performance

protection:mtnm.tmforum.org.protection

session:mtnm.tmforum.org.session

subnetworkConnection:mtnm.tmforum.org.subnetworkConnection

terminationPoint:mtnm.tmforum.org.terminationPoint

topologicalLink:mtnm.tmforum.org.topologicalLink

TopoManagementManager:mtnm.tmforum.org.TopoManagementManager

trafficConditioningProfile:mtnm.tmforum.org.trafficConditioningProfile

trafficDescriptor:mtnm.tmforum.org.trafficDescriptor

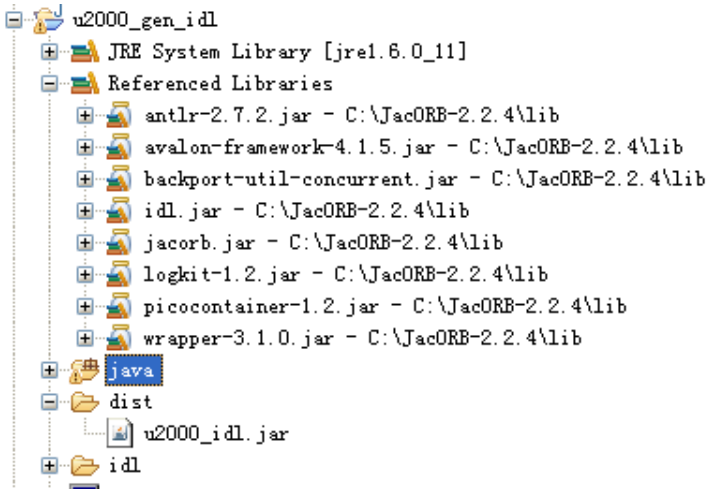
transmissionDescriptor:mtnm.tmforum.org.transmissionDescriptor

transmissionParameters:mtnm.tmforum.org.transmissionParameters

Step 4 After the preceding command is run successfully, the java file compiled by using the IDL compiler is saved in **D:\workspace\u2000_gen_idl\java**.

Compile and compress the java file in **D:\workspace\u2000_gen_idl\java** to **u2000_idl.jar** by using the **.jar** package in **C:\JacORB-2.2.4\lib**.

Figure 3-3 Compiling and compressing the **java** file using the Eclipse

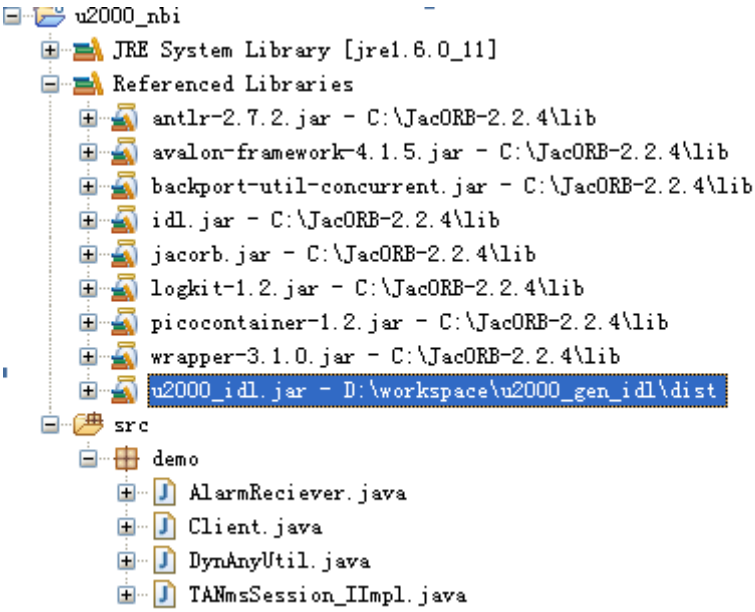


----End

3.6 A Programming Example of Client Codes in Java

Develop client codes by using the **u2000_idl.jar** file and the **.jar** package in **C:\JacORB-2.2.4\lib**.Environment for developing client codes

Figure 3-4 Eclipse code demo environment



The following sections provide programming examples for four classes: Client, AlarmReciever, TANmsSession_IImpl, and Util.

3.6.1 Obtaining the EmsSession and Querying the Manager List Supported by the U2000

Example codes are as follows:

```
package demo;
/**
 * iManager U2000 CORBA interface client program demo
 */
//Common java data type import
import mtnm.tmforum.org.emsSession.EmsSession_I;
import mtnm.tmforum.org.emsSession.EmsSession_IHolder;
import mtnm.tmforum.org.emsSession.EmsSession_IPackage.managerNames_THolder;
import mtnm.tmforum.org.emsSessionFactory.EmsSessionFactory_I;
import mtnm.tmforum.org.emsSessionFactory.EmsSessionFactory_IHelper;
import mtnm.tmforum.org.nmsSession.NmsSession_I;
import mtnm.tmforum.org.nmsSession.NmsSession_IPOA;

import org.omg.CosNaming.NameComponent;
import org.omg.DynamicAny.DynAnyFactory;
import org.omg.DynamicAny.DynAnyFactoryHelper;

/************************************************************
 Type            : client demo program code
 Description    : U2000 CORBA Program development demo to get emsSession object
 ************************************************************/
public class Client {
    private Client() {
    }

    private static Client instance = new Client();

    public static Client getInstance() {
        return instance;
    }
    //ORB object
    private static org.omg.CORBA.ORB orb = null;

    //POA object reference
    private static org.omg.PortableServer.POA rootPOA = null;

    //emsSeesion object
    private static EmsSession_I emsSession = null;

    //dynamic any factory,  for translating any type to dynamic any type
    private static DynAnyFactory dynAnyFactory = null;;

    public static void log(String str) {
        System.out.println(str);
    }
    //
    public static void main(String[] args) {
        if(args.length<4){
            log("usage: java demo.Client 10.71.227.152 12001 admin test1234");
            return;
```

```
            }
            log("-----------------------------------------------------");
            log("Naming service IP: " + args[0]);
            log("Naming service port: " + args[1]);
            log("EMS user name : " + args[2]);
            log("Password for user " + args[2] + " : " + args[3]);
            log("-----------------------------------------------------");
            Client.getInstance().connect(args[0], args[1], args[2], args[3]);
        }

        /*
         * NSIP: naming service IP
         * NSPort: naming service port
         * userName: an EMS user name
         * passWord: the password for a specified EMS user
         */
        public  int connect(String NSIP, String NSPort,String userName, String passWord)
    {
            try {
                //initialize ORB parameters
                String argv[] = new String[2];
                argv[0] = "-ORBInitRef";
                argv[1] = "NameService=corbaloc::" + NSIP + ":" + NSPort+ "/NameService";

                //The step 1: initialize ORB
                orb = org.omg.CORBA.ORB.init(argv, null);
                log("The step 1: initialize ORB successfully!");

                //The step 2: get RootPOA
                rootPOA =
    org.omg.PortableServer.POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
                rootPOA.the_POAManager().activate();
                log("The step 2: get RootPOA successfully!");

                //the step 3: construct dynamic any factory
                dynAnyFactory = DynAnyFactoryHelper.narrow(orb
                        .resolve_initial_references("DynAnyFactory"));
                log("the step 3: construct dynamic any factory."
                        + (dynAnyFactory!= null ? "successfully!" : "failed!"));

                //The step 4: get naming service root object reference
                org.omg.CosNaming.NamingContextExt nc =
    org.omg.CosNaming.NamingContextExtHelper
                        .narrow(orb.resolve_initial_references("NameService"));
                log("The step 4: get naming service root object refrence successfully!");

                //The step 5: get EmsSessionFactory_I object reference from naming service

                //construct EmsSessionFactory_I NameComponent,please get it's parameters
    from HuaWei
                org.omg.CosNaming.NameComponent[] name;
                name = new NameComponent[5];
                name[0] = new NameComponent("TMF_MTNM", "Class");
                name[1] = new NameComponent("HUAWEI", "Vendor");
                name[2] = new NameComponent("Huawei/U2000", "EmsInstance");
```

```
            name[3] = new NameComponent("2.0", "Version");
            name[4] = new NameComponent("Huawei/U2000", "EmsSessionFactory_I");

            EmsSessionFactory_I EmsSessionFactory = null;
            try {
                EmsSessionFactory =
EmsSessionFactory_IHelper.narrow(nc.resolve(name));
                log("The step 5: get EmsSessionFactory_I object refrence from naming
service successfully!");
            } catch (Exception ex) {
                log("The step 5: get EmsSessionFactory_I object refrence from naming
service failed! \n please confirm U2000 CORBA is running and EMS name!");
                ex.printStackTrace();
                return 0;
            }

            //get emsSession
            NmsSession_IPOA pNmsSessionServant = new TANmsSession_IImpl();
            NmsSession_I NmsSession =  pNmsSessionServant._this(orb);

            //The step 6: invoke getEmsSession interface to login U2000 CORBA
            EmsSession_IHolder emsSessionInterfaceHolder = new EmsSession_IHolder();
            try {
                EmsSessionFactory.getEmsSession(userName, passWord,
NmsSession,emsSessionInterfaceHolder);
                emsSession = emsSessionInterfaceHolder.value;
                log("The step 6: invoke getEmsSession interface to login U2000 CORBA
successfully!");
            } catch (Exception ex) {
                log("The step 6: invoke getEmsSession interface to login U2000 CORBA
failed!");
                log(ex.getLocalizedMessage());
                ex.printStackTrace();
                return 0;
            }
            //get and list all supported Managers
            log("The step 7: list all supported Managers:");
            managerNames_THolder managerNames_Holder = new managerNames_THolder();
            emsSession.getSupportedManagers(managerNames_Holder);
            for (int i = 0; i < managerNames_Holder.value.length; i++) {
                log("EmsManagerName[" + i + "] " + managerNames_Holder.value[i]);
            }
            log("The step 8: try to connect structuredPushConsumer to the event
channel.");
            //get alarms from EventChannel
            new AlarmReciever(orb,emsSession).activate();
            orb.run();

        } catch (Exception e) {
            e.printStackTrace();

        }
        return 0;
    }
    public static void listen(){
```

```
        //get alarms from EventChannel
        new AlarmReciever(orb,emsSession).activate();
        orb.run();
    }

    /**
     *  Getter function, return dynAnyFactory
     */
    public static DynAnyFactory getDynAnyFactory() {
        return dynAnyFactory;
    }

    /**
     *  Setter function, set dynAnyFactory
     */
    public static void setDynAnyFactory(DynAnyFactory dynAnyFactory) {
        Client.dynAnyFactory = dynAnyFactory;
    }

    /**
     *  Getter function, return emsSession
     */
    public static EmsSession_I getEmsSession() {
        return emsSession;
    }

    /**
     *  Setter function, set emsSession
     */
    public static void setEmsSession(EmsSession_I emsSession) {
        Client.emsSession = emsSession;
    }

    /**
     *  Getter function, return orb
     */
    public static org.omg.CORBA.ORB getOrb() {
        return orb;
    }

    /**
     *  Setter function,set orb
     */
    public static void setOrb(org.omg.CORBA.ORB orb) {
        Client.orb = orb;
    }

    /**
     *  Getter function, return rootPOA
     */
    public static org.omg.PortableServer.POA getRootPOA() {
        return rootPOA;
    }

    /**
```

```
     *   Setter function,set rootPOA
     */
    public static void setRootPOA(org.omg.PortableServer.POA rootPOA) {
        Client.rootPOA = rootPOA;
    }

    /**
     *   Setter function, set instance
     */
    public static void setInstance(Client instance) {
        Client.instance = instance;
    }
}
package demo;
import mtnm.tmforum.org.nmsSession.NmsSession_IPOA;
import mtnm.tmforum.org.session.Session_I;
/**
 * NmsSession_IPOA for EMS(U2000) invoking.
 * @author
 *
 */
public class TANmsSession_IImpl extends NmsSession_IPOA {

    public void eventLossCleared(String endTime) {
        log("TANmsSession_IImpl.eventLossCleared(String endTime) is invoked by
EMS(U2000).");
        log("endTime:"+endTime);

    }

    public void eventLossOccurred(String startTime, String notificationId) {
        log("TANmsSession_IImpl.eventLossOccurred(String startTime, String
notificationId) is invoked by EMS.");
        log("startTime:"+startTime+", notificationId:"+notificationId);
    }

    public Session_I associatedSession() {
        log("TANmsSession_IImpl.associatedSession() is invoked by EMS(U2000).");
        return null;
    }

    public void endSession() {
        log("TANmsSession_IImpl.endSession() is invoked by EMS(U2000).");

    }

    public void ping() {
        log("TANmsSession_IImpl.ping() is invoked by EMS(U2000).");

    }
    private static void log(String str){
        System.out.println(str);
    }
}
```

3.6.2 Subscribing to an Event Notification

Code example is as follows:

```
package demo;

import mtnm.tmforum.org.emsSession.EmsSession_I;
import mtnm.tmforum.org.globaldefs.ProcessingFailureException;

import org.omg.CORBA.IntHolder;
import org.omg.CosEventChannelAdmin.AlreadyConnected;
import org.omg.CosEventChannelAdmin.TypeError;
import org.omg.CosEventComm.Disconnected;
import org.omg.CosNotification.EventType;
import org.omg.CosNotification.StructuredEvent;
import org.omg.CosNotifyChannelAdmin.AdminLimitExceeded;
import org.omg.CosNotifyChannelAdmin.ClientType;
import org.omg.CosNotifyChannelAdmin.ConsumerAdmin;
import org.omg.CosNotifyChannelAdmin.EventChannel;
import org.omg.CosNotifyChannelAdmin.EventChannelHolder;
import org.omg.CosNotifyChannelAdmin.InterFilterGroupOperator;
import org.omg.CosNotifyChannelAdmin.ProxySupplier;
import org.omg.CosNotifyChannelAdmin.StructuredProxyPushSupplier;
import org.omg.CosNotifyChannelAdmin.StructuredProxyPushSupplierHelper;
import org.omg.CosNotifyComm.InvalidEventType;
import org.omg.CosNotifyComm.StructuredPushConsumer;
import org.omg.CosNotifyComm.StructuredPushConsumerPOA;
import org.omg.CosNotifyFilter.ConstraintExp;
import org.omg.CosNotifyFilter.Filter;
import org.omg.CosNotifyFilter.InvalidConstraint;
import org.omg.CosNotifyFilter.InvalidGrammar;

/**
 *
 * get events from event channel
 *
 */
public class AlarmReciever {
    private StructuredPushConsumer structuredPushConsumer;
    private EventChannel eventChannel;
    private ConsumerAdmin consumerAdmin;
    private StructuredProxyPushSupplier proxyPushSupplier;
    private EmsSession_I emsSession_I;
    // ORB object
    private org.omg.CORBA.ORB orb = null;

    public AlarmReciever(org.omg.CORBA.ORB orb, EmsSession_I emsSession) {
        this.orb = orb;
        this.emsSession_I = emsSession;
    }

    public void activate() {
        EventChannelHolder eventChannelHolder = new EventChannelHolder();
        try {
            emsSession_I.getEventChannel(eventChannelHolder);
        } catch (ProcessingFailureException e) {
```

```
        e.printStackTrace();
        return;
    }
    eventChannel = eventChannelHolder.value;

    // get consumer consumerAdmin
    try {
        IntHolder adminID = new IntHolder();
        consumerAdmin = eventChannel.new_for_consumers(
                InterFilterGroupOperator.AND_OP, adminID);
    } catch (Exception e) {
        consumerAdmin = null;
        e.printStackTrace();
        return;
    }

    EventType eventTypeAdd[] = new EventType[2];
    eventTypeAdd[0] = new EventType();
    eventTypeAdd[1] = new EventType();

    // subscript only NT_ALARM and NT_HEARTBEAT type events
    eventTypeAdd[0].domain_name = "tmf_mtnm";
    eventTypeAdd[0].type_name = "NT_ALARM";
    eventTypeAdd[1].domain_name = "tmf_mtnm";
    eventTypeAdd[1].type_name = "NT_HEARTBEAT";
    EventType eventTypeRemove[] = new EventType[0];
    try {
        log("subscript alarm");
        consumerAdmin.subscription_change(eventTypeAdd, eventTypeRemove);
    } catch (InvalidEventType e) {
        e.printStackTrace();
        log("Failed to set subscription_change.");
        return;
    }

    consumerAdmin.remove_all_filters();
    try {

        EventType alarmEventTypes[] = new EventType[1];
        alarmEventTypes[0] = new EventType();
        alarmEventTypes[0].domain_name = "tmf_mtnm";
        alarmEventTypes[0].type_name = "NT_ALARM";

        ConstraintExp[] alarmExps = new ConstraintExp[2];

        // filter the alarm which perceivedSeverity is PS_CRITICAL or
        // PS_CLEARED.
        alarmExps[0] = new ConstraintExp(alarmEventTypes,
                "($perceivedSeverity == 2) or ($perceivedSeverity == 5)");

        // filter the alarm which serviceAffecting is not
        // SA_NON_SERVICE_AFFECTING.
        alarmExps[1] = new ConstraintExp(alarmEventTypes,
                "$X733_EventType == 'equipmentAlarm'");
```

```
        Filter alarmFilter = eventChannel.default_filter_factory()
                .create_filter("TCL");
        alarmFilter.add_constraints(alarmExps);
        consumerAdmin.add_filter(alarmFilter);

        EventType htEventTypes[] = new EventType[1];
        htEventTypes[0] = new EventType();
        htEventTypes[0].domain_name = "tmf_mtnm";
        htEventTypes[0].type_name = "NT_HEARTBEAT";

        ConstraintExp[] htExps = new ConstraintExp[1];

        // filter all heart beat
        htExps[0] = new ConstraintExp(htEventTypes, "1 == 1");
        Filter heartbeatFilter = eventChannel.default_filter_factory()
                .create_filter("TCL");
        heartbeatFilter.add_constraints(htExps);
        consumerAdmin.add_filter(heartbeatFilter);

    } catch (InvalidGrammar e1) {
        e1.printStackTrace();
    } catch (InvalidConstraint e) {
        e.printStackTrace();
    }

    // create and implicitly activate the client
    structuredPushConsumer = (StructuredPushConsumer) new Consumer()._this(orb);

    // get the structured proxy push supplier
    ClientType clientType = ClientType.STRUCTURED_EVENT;
    org.omg.CORBA.IntHolder proxyId = new org.omg.CORBA.IntHolder(0);
    ProxySupplier proxySupplier = null;
    try {
        proxySupplier = consumerAdmin.obtain_notification_push_supplier(
                clientType, proxyId);

    } catch (AdminLimitExceeded e) {
        e.printStackTrace();
        log(" Failed to obtain notification_push_supplier.");
        return;
    }
    proxyPushSupplier = StructuredProxyPushSupplierHelper
            .narrow(proxySupplier);

    // connect structuredPushConsumer to the event channel
    try {
        proxyPushSupplier
                .connect_structured_push_consumer(structuredPushConsumer);
        log("Succeed to connect structuredPushConsumer to the event channel.");
    } catch (AlreadyConnected e) {
        e.printStackTrace();
        log("Failed to connect to structuredPushConsumer to the event channel.");
    } catch (TypeError e) {
        e.printStackTrace();
        log("Failed to connect to structuredPushConsumer to the event channel.");
```



```
            }
        }

        public static void log(String log) {
            System.out.println(log);
        }

        class Consumer extends StructuredPushConsumerPOA {

            public void disconnect_structured_push_consumer() {
                log("Consumer disconnect_structured_push_consumer.");

            }
            public void push_structured_event(StructuredEvent event)
                    throws Disconnected {
                // print received event
                log("<+++>");
                log("event_type=" + event.header.fixed_header.event_type.type_name);
                for (int i = 0; i < event.filterable_data.length; i++) {
                    if (null == event.filterable_data[i]) {
                        continue;
                    }
                    log(event.filterable_data[i].name + "="
                            + Util.parseAny(event.filterable_data[i].value));
                }
                log("<--->");
            }

            public void offer_change(EventType[] eventTypeArray,
                    EventType[] eventTypeArray1) throws InvalidEventType {
                for(int i=0;i<eventTypeArray.length;i++){
                    log(eventTypeArray[i].type_name);
                }
            }
        }
    }

    package demo;
    import org.omg.CORBA.Any;
    import org.omg.CORBA.TCKind;
    import org.omg.DynamicAny.DynAnyFactory;
    import org.omg.DynamicAny.DynArray;
    import org.omg.DynamicAny.DynEnum;
    import org.omg.DynamicAny.DynSequence;
    import org.omg.DynamicAny.DynStruct;
    import org.omg.DynamicAny.DynUnion;
    /**
     * parse any type to string.
     * @author
     *
     */
    public class Util
    {
       private static DynAnyFactory factory = Client.getDynAnyFactory();
```

```
/**
 * parseAny
 *
 * @param any Any
 * @return String
 */
public static  String parseAny( Any any )
{
   if( null==any ){
      return null;
   }
   StringBuffer result = new StringBuffer();
   try {
      switch (any.type().kind().value()) {
      case TCKind._tk_char:
         result.append(any.extract_char());
         break;
      case TCKind._tk_null:
         break;
      case TCKind._tk_boolean:
         result.append(any.extract_boolean());
         break;
      case TCKind._tk_short:
         result.append(any.extract_short());
         break;
      case TCKind._tk_long:
         result.append(any.extract_long());
         break;
      case TCKind._tk_double:
         result.append(any.extract_double());
         break;
      case TCKind._tk_float:
         result.append(any.extract_float());
         break;
      case TCKind._tk_octet:
         result.append(any.extract_octet());
         break;
      case TCKind._tk_ulong:
         result.append(any.extract_ulong());
         break;
      case TCKind._tk_string:
         result.append(any.extract_string());
         break;
      case TCKind._tk_enum:
      {
         DynEnum dynEnum = (DynEnum) factory.create_dyn_any(any);
         result.append(dynEnum.get_as_string());
         break;
      }
      case TCKind._tk_any:
      {
         any=factory.create_dyn_any(any).get_any();
         result.append(any);
         break;
      }
```

```
                case TCKind._tk_objref:
                {
                    result.append(any.extract_Object());
                    break;
                }

                case TCKind._tk_struct:
                case TCKind._tk_except:
                {
                    DynStruct dynstruct = (DynStruct) factory.create_dyn_any(any);
                    org.omg.DynamicAny.NameValuePair[] members = dynstruct.get_members();
                    result.append("{");
                    for (int i = 0; i < members.length; i++) {
                        //result.append("[" + members[i].id + "=");
                         if(i>0){
                              result.append(" ");
                         }
                        result.append(members[i].id).append("
").append(parseAny(members[i].value));
                    }
                    result.append("}");
                    break;
                }
                case TCKind._tk_union:
                    DynUnion dynunion = (DynUnion) factory.create_dyn_any(any);
                    result.append(dynunion.member_name()).append(" ");
                    result.append(parseAny(dynunion.member().to_any()));
                    break;
                case TCKind._tk_sequence:
                    DynSequence dynseq = (DynSequence) factory.create_dyn_any(any);
                    Any[] contents = dynseq.get_elements();
                    result.append("{");
                    for (int i = 0; i < contents.length; i++){
                        result.append(parseAny(contents[i]));
                    }
                    result.append("}");
                    break;
                case TCKind._tk_array:
                    DynArray dynarray = (DynArray) factory.create_dyn_any(any);
                    Any[] array_contents = dynarray.get_elements();
                    result.append("{");
                    for (int i = 0; i < array_contents.length; i++){
                        result.append(parseAny(array_contents[i]) ).append(" ");
                    }
                    result.append("}");
                    break;
                default:
                    result.append(any.type().kind().value());

                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }
            return result.toString();
```

```
        }
    }
```

3.6.3 Notification Subscription

Table 3-1 lists the notifications that the OSS can subscribe to.

Table 3-1 Notifications that the OSS can subscribe to

Notification	Description
NT_ALARM	Alarm notification
NT_ALARM_UPDATED	Alarm update notification
NT_TCA	Performance TCA notification
NT_OBJECT_CREATION	Object creation notification
NT_OBJECT_DELETION	Object deletion notification
NT_ATTRIBUTE_VALUE_CHANGE	Attribute change notification
NT_STATE_CHANGE	Status change notification
NT_ROUTE_CHANGE	Route change notification
NT_PROTECTION_SWITCH	Protection switching notification
NT_FILE_TRANSFER_STATUS	File transfer status notification
NT_EPROTECTION_SWITCH	Equipment protection switching notification
NT_ASON_RESOURCE_CHANGE	ASON resource change notification
NT_PRBSTEST_STATUS	Notification for pseudo random binary sequence (PRBS) test status
NT_WDMPROTECTION_SWITCH	WDM protection switching notification
NT_ATMPROTECTION_SWITCH	ATM protection switching notification
NT_RPRPROTECTION_SWITCH	Notification for resilient packet ring (RPR) protection switching
NT_IPPROTECTION_SWITCH	Notification for resilient packet ring (Tunnel) protection switching

3.6.4 Syntaxes for Notification Parameter Filtering

Notification parameter filtering filters all filterable parameters in a notification. Table 3-2 lists the Tool Command Language (TCL) syntaxes that are used during the filtering.

Table 3-2 TCL syntaxes

Syntax	Representation	Example
Variable	$var_name	$perceivedSeverity
Numerical value	Numbers	2
String	'string'	'str'
Brackets	()	($perceivedSeverity == 5)
Greater than	>	$perceivedSeverity > 1
Greater than or equal to	>=	perceivedSeverity >= 1
Smaller than	<	$perceivedSeverity < 1
Smaller than or equal to	<=	$perceivedSeverity <= 1
Equal to	==	$perceivedSeverity == 1
Not equal to	!=	$perceivedSeverity != 1
And	And	($perceivedSeverity == 1) and ($title == 'testAlarm')
Or	Or	($perceivedSeverity == 1) or ($perceivedSeverity == 5)
Not	not	not ($perceivedSeverity == 1)

3.6.5 Filtered Parameters in Notifications

This section uses alarm filtering as an example to describe what parameters in notifications can be filtered and the filtering expressions.

Table 3-3 Parameters that can be filtered in alarms

Parameter	Definition in IDL Files	Description	Value
$objectType	ObjectType_T	Indicates the type of an alarm object.	See the IDL files.
$X733_EventType	X_733_EventType_T	Indicates the alarm type.	See the IDL files.
$perceivedSeverity	PerceivedSeverity_T	Indicates the alarm severity.	See the IDL files.

Parameter	Definition in IDL Files	Description	Value
$probableCause	string	Indicates the alarm cause.	See the *iManager U2000 Northbound CORBA Interface Alarm List.*

For example, you can filter alarm parameters by the following criteria:

- Type of the alarm object: all objects except the U2000
- Alarm type: NE alarms
- Alarm severity: major and clear alarms
- Alarm cause: AIS

To filter alarm parameters by the preceding criteria, use the following filtering expressions:

```
($objectType > 0)
and ($X733_EventType == 'equipmentAlarm' )
and ( ($perceivedSeverity == 2) or ($perceivedSeverity == 5) )
and ($probableCause == 'AIS')
```

NOTE

For details about other notification types, see the iManager U2000 Northbound CORBA Interface Developer Guide (Alarm) and iManager U2000 Northbound CORBA Interface Developer Guide (Resource) for notification examples and see the IDL files for parameter values and definitions.

3.7 Running Codes

Perform the following operations to run the codes.

Step 1 Obtain the IP address and port ID of the U2000 naming service.

Step 2 Obtain the user name and password for logging in to the emsSession.

Step 3 Obtain the paths of EmsSessionFactory_I in the naming service. The default paths are:

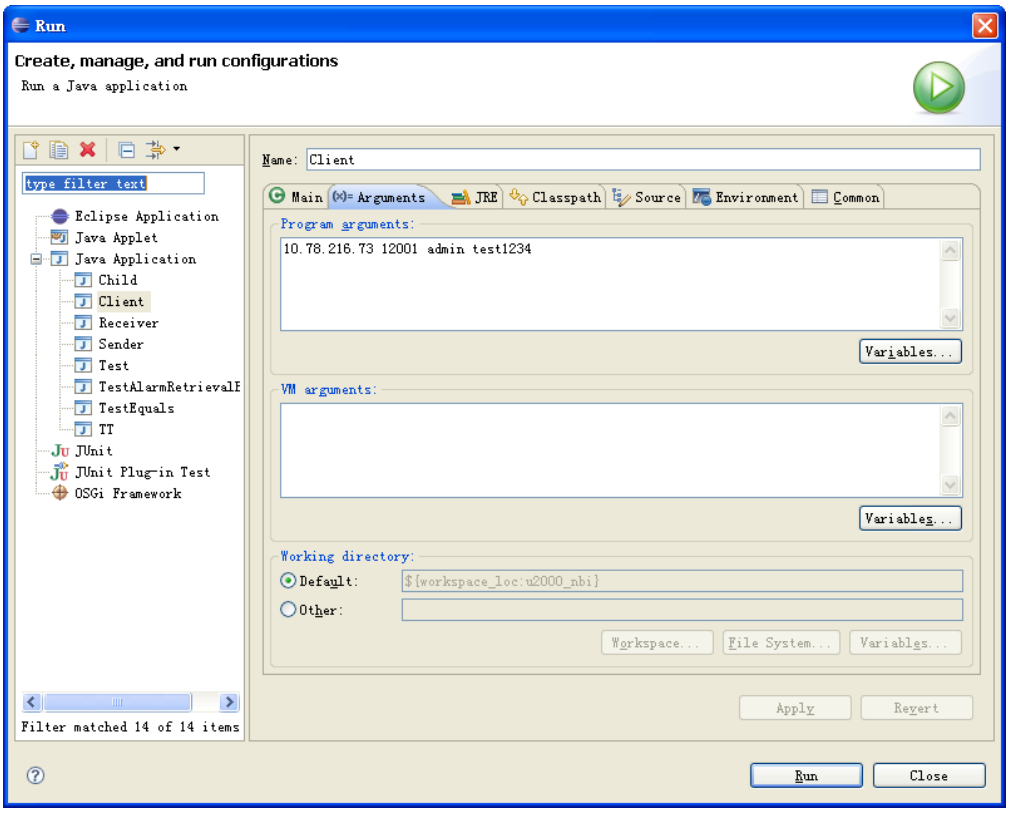
```
name[0] = new NameComponent("TMF_MTNM", "Class");
name[1] = new NameComponent("HUAWEI", "Vendor");
name[2] = new NameComponent("Huawei/U2000", "EmsInstance");
name[3] = new NameComponent("2.0", "Version");
name[4] = new NameComponent("Huawei/U2000", "EmsSessionFactory_I");
```

You can change the Client class parts in the paths (words marked in blue) based on the obtained paths.

Step 4 Make configurations in the Eclipse.

Enter parameters in the format of "*NameServiceIP NameServicePort username password*." in the **Program argument** area.

Figure 3-5 Making configurations in the Eclipse



Step 5 Run the following command that contains NameServiceIP NameServicePort username password:

java -cp "%CLASSPATH%;%JACORB_HOME%\lib\jacorb.jar;%JACORB_HOME%\lib\logkit-1.2.jar;%JACORB_HOME%\lib\avalon-framework-4.1.5.jar;%JACORB_HOME%\lib\idl.jar;.\lib\u2000_idl.jar;classes" demo.Client 10.78.216.73 12001 admin test123

A message similar to the following will be displayed.

----End

4 Test and Verification

4.1 Testing and Verifying the CORBA NBI by Using the CORBA Explorer

The CORBA Explorer simulates the upper-layer OSS to communicate with the CORBA NBI. The CORBA Explorer first connects to the naming service provided by the U2000 CORBA NBI to obtain the U2000 CORBA NBI's object references (such as EmsSessionFactory_I), and then invokes the getEmsSession operation provided by the object to set up a connection between the U2000 and the CORBA NBI.

The CORBA Explorer visits the naming service by any of the following ways to obtain the object referenced by the CORBA NBI:

- Mode 1: By obtaining the naming service's object reference file, that is, the **ns.ior** file. The **ns.ior** file is the naming service's object reference and is generated automatically after the naming service is started successfully. The default path to the **ns.ior** file is oss/server.
- By visiting the naming service process directly in EndPoint mode. For example, run the following command to visit and listen on the naming service at port 12001 with IP address 10.70.78.138 by specifying ORB parameters:

-ORBInitRef NameService=iioploc:/ 10.70.78.138: 12001 (NMS IP address: port ID)/NameService

4.2 Testing and Verifying the CORBA NBI by Using the JacORB Notification Service

The notification service plays an essential role in interconnection through the CORBA NBI. The upper-layer OSS can obtain notifications reported actively by the CORBA NBI only if the notification service is available. To configure a default notification service for the ORB, add a URL that points the service to the **.jacorb_properties** file. A valid URL can be obtained in the following ways:

By specifying the option –printCorbaloc to obtain a compact URL. You can use the URL in the form of "corbaloc::ip-address:port/NotificationService" to connect to a specific port. In this case, learn the IP address and port ID to obtain an object reference and the IP address and port ID of the notification to listen on.

The initial parameters for using this mode are: ./Notify_Service -ORBSvcConf notify.conf -ORBInitRef NameService=iioploc:// 10.70.78.138:12001 (IP address of the naming service: port ID of the naming service)/NameService -ORBEndpoint iiop:// 10.70.78.138:12003 (listening IP address:listening port ID)-ORBDottedDecimalAddresses 1

To check whether the notification service has been started, run the following command:

telnet 10.70.78.138 (listening IP address) 12003 (port ID)

If the connection is successful, the notification service has been started. Otherwise, the notification service has failed to start.

4.3 Precautions About CORBA NBI Verification

4.3.1 Checking the CORBA NBI Configuration

Before using the CORBA Explorer, configure the following CORBA configuration items in the U2000 installation directory.

1. Configuration items in **server/etc/oss_cfg/nbi/corba/cbb/nbi/nbicbb_3p/share/bin/ns.cfg**

   ```
   [NameService]
   # Host name of the naming service(default value is local host name)
   hostname=10.70.71.61
   ```

   Ensure that **hostname** is set to the IP address of the naming service.

   NOTE
   Do not set hostname to localhost or 127.0.0.1.

2. Configuration items in **server/etc/oss_cfg/nbi/corba/cbb/nbi/nbicbb_3p/share/bin/ntf.cfg**

   ```
   [NameService]
   # Host name of the naming service(default value is local host name)
   hostname=10.70.71.61
   ```

   ```
   [NotifyService]
   # Host name of the notify service(default value is local host name)
   hostname=10.70.71.61
   ```

   Ensure that the two **hostname** parameters are set to the IP addresses of the naming service and the notification service respectively.

   NOTE
   Do not set hostname to localhost or 127.0.0.1.

3. Configuration items in **server/etc/oss_cfg/nbi/corba/conf/ii_corbaagent_bundle/bundle.cfg**

   ```
   #| Host name of the naming service located.
   NamingService_Host      = 10.70.71.61
   ```

```
#| Host name of the notify service located.
NotifyService_Host        = 10.70.71.61
```

```
#| Host name of the CORBAAgent located.
CORBAAgent_Host           = 10.70.71.61
```

Ensure that **NamingService_Host**, **NotifyService_Host**, and **CORBAAgent_Host** are set to the IP addresses of the naming service, the notification service, and the CORBA process service respectively.

NOTE

Do not set these three host names to **localhost** or **127.0.0.1**

4.3.2 Checking the Communication Mode of the CORBA NBI

1. Perform the following operation to check the communication mode of the naming service:

   Open the **server/etc/oss_cfg/nbi/corba/cbb/nbi/nbicbb_3p/share/bin/ns.cfg** file.

   ```
   # Is using SSL mode(if using SSL mode, sslconf must be set to enable)
   ssl=enable
   ```

   If **ssl** is set to **disable**, the SSL mode is disabled. If **ssl** is set to **enable**, the SSL mode is enabled.

2. Perform the following steps to check the communication mode of the notification service:

   Open the **server/etc/oss_cfg/nbi/corba/cbb/nbi/nbicbb_3p/share/bin/ntf.cfg** file.

   ```
   # Is using SSL mode(if using SSL mode, sslconf and naming service must be set to enable)
   ssl=enable
   ```

   If **ssl** is set to **disable**, the SSL mode is disabled. If **ssl** is set to **enable**, the SSL mode is enabled.

3. Perform the following operation to check the communication mode of the CORBA agent:

   Open the **server/etc/oss_cfg/nbi/corba/conf/ii_corbaagent_bundle/bundle.cfg** file.

   ```
   #| SSL protocol mode(0:IIOP; 1: SSLIOP)
   SSLMode=1
   ```

   If **SSLMode** is set to **1**, the SSL mode is enabled. If **SSLMode** is set to **0**, the SSL mode is disabled.

NOTE

Ensure that the communication modes of the naming service, the notification service, and the CORBA agent service are the same.

4.3.3 Checking the Configuration of the SSL Mode

If the SSL mode is enabled for CORBA communication, take the following precautions:

1. The certificate must take effect.
2. Path to the certificate must be correct in the configuration file.

   Check that the path to the certificate is correct in the following configuration files:

   The **nbi_ssl.conf** file in the **server\etc\oss_cfg\nbi\corba\conf\ii_corbaagent_bundle** directory


```
static Resource_Factory "-ORBProtocolFactory SSLIOP_Factory"
dynamic SSLIOP_Factory Service_Object * TAO_SSLIOP:_make_TAO_SSLIOP_Protocol_Factory()
"-SSLAuthenticate SERVER
-SSLPrivateKey PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/server_key.pem
-SSLCertificate PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/server_ca.cer
-SSLCAfile PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/client_ca.cer
-SSLCApath nbi/corba/conf/ii_corbaagent_bundle/certificate/"
```

The **notify_ssl.conf** file in the **server\etc\oss_cfg\nbi\corba\conf\ii_corbaagent_bundle** directory

```
static Server_Strategy_Factory "-ORBconcurrency thread-per-connection"
static Notify_Default_Event_Manager_Objects_Factory "-MTDispatching -DispatchingThreads 3 -MTSourceEval -SourceThreads 3"
static Resource_Factory "-ORBProtocolFactory SSLIOP_Factory"
dynamic SSLIOP_Factory Service_Object * TAO_SSLIOP:_make_TAO_SSLIOP_Protocol_Factory()
"-SSLAuthenticate SERVER
-SSLPrivateKey PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/server_key.pem
-SSLCertificate PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/server_ca.cer
-SSLCAfile PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/client_ca.cer
-SSLCApath nbi/corba/conf/ii_corbaagent_bundle/certificate/"
```

The **naming_ssl.conf** file in the **server\etc\oss_cfg\nbi\corba\conf\ii_corbaagent_bundle** directory

```
# This service must use a thread-per-connection model to enable the
# server to block in a dedicated thread, i.e., one for each client.
# The concurrency service will not run in the reactive model.
static Server_Strategy_Factory "-ORBconcurrency thread-per-connection"
static Client_Strategy_Factory "-ORBTransportMuxStrategy exclusive"
static Resource_Factory "-ORBProtocolFactory SSLIOP_Factory"
dynamic SSLIOP_Factory Service_Object * TAO_SSLIOP:_make_TAO_SSLIOP_Protocol_Factory()
"-SSLAuthenticate SERVER
-SSLPrivateKey PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/server_key.pem
-SSLCertificate PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/server_ca.cer
-SSLCAfile PEM:nbi/corba/conf/ii_corbaagent_bundle/certificate/client_ca.cer
-SSLCApath nbi/corba/conf/ii_corbaagent_bundle/certificate/"
```

