Project Number:	AC112
Project Title:	TRUMPET
	Inter Domain Management with Integrity
Deliverable Type:	I
CEC Deliverable Number:	AC112/GMD/WP3/DS/R/011/b1
Contractual Date of Delivery:	31st December 1997
Actual Date of Delivery:	14th January 1997
Date of This Version:	November, 1997
Title of Deliverable:	Implementation
Workpackage contributing:	WP3
Nature of the Deliverable:	T (PLEASE CHECK)
Document Location:	Trumpet/wp3/D11/d11.zip
Authors:	Editors: Philippe Emeriau, Marcus Wittig

Abstract:

This deliverable describes the implementation of the secure inter-domain service management systems which have been developed by TRUMPET. It serves as a baseline document for both, the software developers in TRUMPET, and the system's operators which need to install and run the software as part of the TRUMPET trials.

Keyword list:

TMN, inter-domain management, user access, assets, threats, vulnerabilities, security, integrity, accountability, availability, confidentiality, security measures, security profiles, security policies.

Executive Summary

This document describes the implementation of the secure inter-domain service management systems which have been developed by TRUMPET. It serves as a baseline document for both, the software developers in TRUMPET, and the system's operators which need to install and run the software as part of the TRUMPET trials. Accordingly this deliverable has been organised as a set of handbooks which describe the various aspects. While the developers handbooks in particular address the details of implementation design, the installation handbooks describe how to install and run software. Moreover, the security components have been described in a separate set of handbooks since the security software and documentation will be distributed to other projects such as the ACTS MISA project.

As part of the implementation design the developer handbooks describe the details of component engineering models, APIs and communication interfaces. It presents a refined view on the detailed component design presented in deliverables 8 and 9 [TRUMPET-D8, TRUMPET-D9] and describes how the components have been mapped onto the selected implementation technologies. While these handbooks are in particular addressed to the software developers, the installation handbook should help systems operators with the installation and execution of the software. The latter also includes a description of hardware and software prerequisites for each component so that all the information is given which is needed to set-up a site to run the TRUMPET system.

The structure of the document is as follows: Chapter 1 provides an introduction which gives an overall overview on the TRUMPET management architecture, the security architecture, and the technology environment. Chapters 2 and 3 present the handbooks describing the implementation design and the installation of the management system components, while the handbooks for the security components are presented in Chapter 4 and 5.

List of Contributors

Philippe Emeriau	**Ascom Monetel (P01)** CICA, 229 rte des Cretes F 06560 Sophia Antipolis France	tel.: +33.92.94.21.99 fax: +33.92.94.20.20 email: emeriau@ascom.eurecom.fr
Cyril Autant **Nicolas Ganivet**	**Alcatel ISR (P02)** 3, rue Ampere 91349 Massy Cedex France	tel.: +33 1 69 76 23 49 fax: +33 1 69 76 25 50 email: autant@isr.alcatel-alsthom.fr ganivet@isr.alcatel-alsthom.fr
Dominique Maillot **Christiane Pace**	**Telis / Sema (P03)** Departement Projects Europeens 3-9 rue Helene Boucher F-78280 Guyancourt France	tel.: +33.1.30.96.42.12 fax.: +33.1.30.96.44.72 e-mail: Maillot@sqy.sema.fr
Shahrzade Mazaher **Jonn Sketting**	**Norwegian Computing Center (P08)** P.O. Box 114 Blindern N-0314 Oslo Norway	tel.: +47.2285.2500 fax: +47.2269.7660 e-mail: Shahrzade.Mazaher, Jonn.Skretting@nr.no
Marcus Wittig **Oliver Schittko**	**GMD - Fokus (P09)** Kaiserin-Augusta-Allee 31 D-10589 Berlin Germany	tel.: +49.30.3463.7218 fax: +49.30. 3463.7218 e-mail: wittig@fokus.gmd.de schittko@fokus.gmd.de,
Lionel Sacks **Mathew Loryman** **Matthieu Verdier**	**University College London (P10)** Torrington Place London WC1E7JE England	tel.: +44.1.71.419.3198 fax: +44.1.71.387.4350 e-mail: l.sacks@eleceng.ucl.ac.uk
George Andrianopoulos	**ATC Bull (P12)** Aharnon 438 str, GR 11143 Athens Greece	tel.: +30.1.2182008-9 fax: +30.1.2182010 e-mail: andriano@01p.gr
Bernhard Bowler	**Salfort Software Systems (P13)** Technology House, Lissadel Street, Salford M6 6AP, England	tel.: +44 161 278 2555 fax: +44 161 278 2506 email: bb@e.sss.co.uk

Table of Contents

Table of Figures

Table of Tables

1. INTRODUCTION

The TRUMPET secure inter-domain service management systems have been developed as part of the activities in Work Package 3, which is responsible for the architectural design, the definition of test scenarios, and the implementation of the TRUMPET system. To validate the results of the design and implementation work the TRUMPET system will be executed and tested as part of trials in which real users work in a TMN domain environment established by National Hosts.

This document is intended to serve as a baseline document for both, the software developers in TRUMPET, and the system's operators which need to install and run the software as part of the TRUMPET trials. Accordingly this deliverable has been organised as a set of handbooks which describe the various aspects. While the developers handbooks in particular address the details of implementation design, the installation handbooks describe how to install and run software. Moreover, the security components have been described in a separate set of handbooks since the security software and documentation will be distributed to other projects such as the ACTS MISA project.

As part of the implementation design the developer handbooks describe the details of component engineering models, APIs and communication interfaces. It presents a refined view on the detailed component design presented in deliverables 8 and 9 [TRUMPET-D8, TRUMPET-D9] and describes how the components have been mapped onto the selected implementation technologies. While these handbooks are in particular addressed to the software developers, the installation handbook should help systems operators with the installation and execution of the software. The latter also includes a description of hardware and software prerequisites for each component so that all the information is given which is needed to set-up a site to run the TRUMPET system.

The structure of the document is as follows: The remainder of this chapter gives an overall overview on the TRUMPET management architecture, the security architecture, and the technology environment. Chapters 2 and 3 present the handbooks describing the implementation design and the installation of the management system components, while the handbooks for the security components are presented in Chapter 4 and 5.

1.1 Management System Architecture

TRUMPET focuses on the secure operation of inter-domain management systems within the Open Network Provisioning (ONP) framework. The TRUMPET scenario shown in Figure 1 involves the following players: two (or more) Public Network Operators (PNOs), a Value Added Service Provider (VASP), and a number of customers at various sites - Customer Premises Networks (CPNs) [TRUMPET-D6].

The customers see an end-to-end connection and are not necessarily aware of which PNOs are contributing to establish the connection. The VASP sees the connection as a set of *segments*, each supported by a different PNO, but does not know how each segment has been set up within the corresponding PNO (*i.e.*, what ATM switches are used).

The management systems of the players mentioned above form a service provisioning system for management and provision of broadband (ATM) network connections between two customers/end users. CPN is a dedicated service in the customer organisation, which already has a contract with the VASP. The VASP management system provides network connectivity to customers by utilising the resources of one or more PNOs. VASP allows customers to create, modify and delete connections, thus effectively providing the Virtual Private Network (VPN) service to the customers. PNOs provide the physical infrastructure, i.e. the network, and the adequate management interface to interact with it.

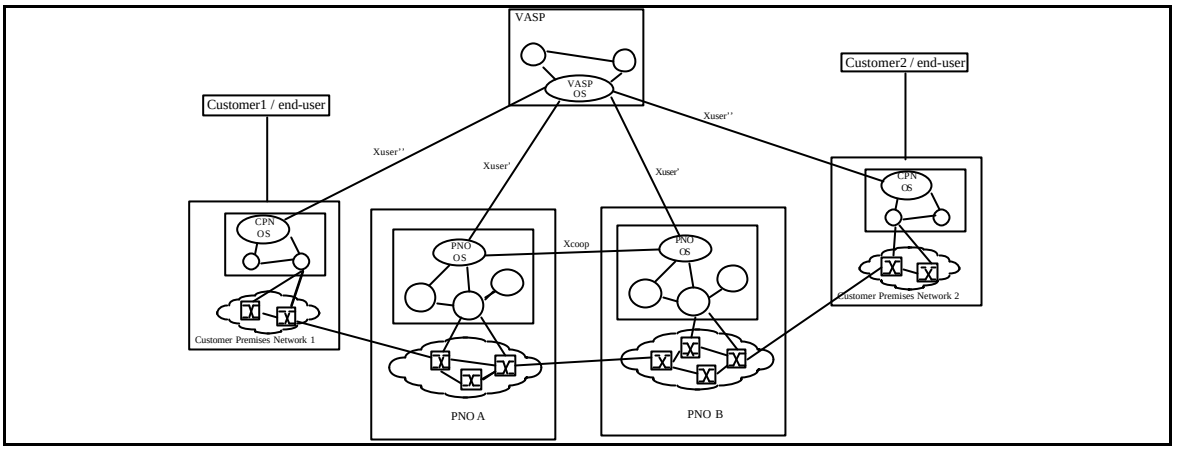


Figure 1: The TRUMPET Reference Architecture

This scenario thus presents a reasonable view of emerging service provisioning and market players. Within this context the needs for high integrity and secure management information exchange between the players can also be clearly seen. That each player must have publicly accessible data network interconnections there is a clear exposure of both the data in transit and the accessibility of operational interfaces. Further, the management platforms employed by each player cannot be restricted to that of any individual vendor. Thus there is also a demand for understanding the robustness impacts of the inter-working of different technologies. The exposure of data and interfaces points to the need for data encryption and integrity requirements for data in transit; the exposure of the interfaces requires access control. Both these security requirements imply that the organisations can exchange security information such as public keys, private keys and access rights. These concerns are the principle focus of the implementation of the security elements of TRUMPET. These security functions are, moreover, explored in the kinds of technology – both apparent and emerging – which can be used of open service provisioning. The functional requirements on the reference architecture are:

- VASP receives requests for services from a customer across a TMN like interface Xuser''. These requests concern the establishment of inter-domain connections between two customers. The requirement placed on the VASP consists in finding the best solutions to connect the two customers, taking into account the requirements concerning the connection (Quality of Service, bandwidth...), the physical resources available in the PNO domains, and optional criteria related to financial costs.
- PNO provides the physical infrastructure, i.e. the networks. It has a contract with the VASP, which knows the entry points of the PNO. During the establishment of the connection, a negotiation takes place between the PNO and the VASP to reach an agreement for an offer from the PNO which corresponds to the VASP requests.
- Customer is the end-point of the connection. Essentially, two kinds of customer will be considered, although here they have been merged in a single entity. The customer/end-user is the user of the application requiring the connection. The customer/network (CPN) is the organisation which will send the connection request to the VASP. It usually is a dedicated service in the customer organisation, who has already subscribed a contract with the VASP.

1.2 Security Architecture

The security architecture consists of a set of security components, which can be used by TMN platforms with open or closed protocol stacks. The distinction is described as:

- For a research management platform, the internals of the protocol stack can be accessed for additions and modifications. In this case, TRUMPET suggests use of the Generic Upper Layer Security (GULS) specifications and the Transport Layer Security Protocol (TLSP).
- For a commercial management platform, security features can only be added on top of the interface to the protocol stack, i.e. over CMISE or ACSE. Data integrity, confidentiality and non-repudiation are especially difficult to implement if the protocol stack is not accessible.

Since TRUMPET has selected a commercial TMN platform for implementation, only the commercial platform architecture has been further developed into component specifications, and is presented here. However, the

security architecture is designed to be as generic as possible by insulating the security-relevant code from the actual environment through an integration layer. Therefore, the applicability of the TRUMPET security architecture to other management environments (open protocol stack, WBEM) should be straightforward.

TRUMPET's security policies are based on the use of public key mechanisms for authentication and prescribes the use of Trusted Third Parties (TTP) in the role of Certification Authorities (CA). Symmetric encryption is used for confidentiality and data integrity protection.

The architecture is shown in Figure 2, with dashed lines indicating the components added by TRUMPET, and solid lines indicating existing components.

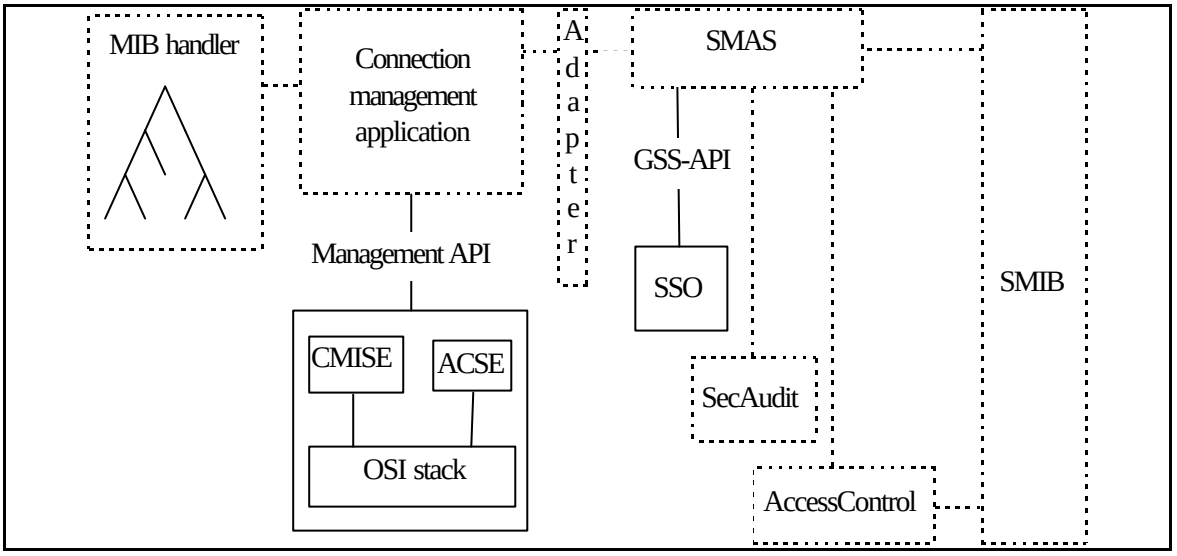


Figure 2: Architecture for a Commercial Management Platform.

This security architecture aims at securing communications between a Management Application Entity (MAE) belonging to a domain and another MAE belonging to another domain. Different MAEs within a domain may use different security profiles, and the choice of security profile is made during the initialisation of the security context. Selection of a security profile may be constrained by the mechanisms supported by an implementation, by internal policies, or by target OS policies. The architecture must support all security profiles because it is not a priori possible to determine which security policies that will be applied to a particular X interface. However, some policy decisions inherently affect the architecture, like the decision to use public key cryptography.

When a MAE belonging to the initiator OS performs inter-domain management operations, it may be working on behalf of another entity (a human user or another MAE) or on its own behalf. To preserve privacy of users, and to facilitate management of access privileges (authorisation), the MAE will always use its own identity and associated set of privileges to perform the management operations on the target OS. This implies that proper internal security measures must be enforced before human operators or MAEs are given access to the management capabilities of MAEs that perform inter-domain management.

To be as independent as possible from the management application and its environments, the security components are accessed through an *Adapter* component. Ideally, the interfaces provided by the Adapter to the application should be identical to the interfaces the application uses for access to the communication services. Alternatively, a defined interface (like the Generic Security Service API (GSS-API) [RFC 1508]) could be offered from the security components, in which case the applications must incorporate this interface, and perform necessary transformations on the data passed over the interface.

To achieve a high degree of flexibility to particular security mechanisms, the architecture is based on the GSS-API. The GSS-API interfaces to a Security Support Object (SSO) used to establish a security context between the communication parties and to perform security transformations on the application data.

With respect to a commercial management platform, there is some minimal support required for the transfer of security data between communicating parties. The specific requirements that must be satisfied are:

- The authentication field of ACSE must be supported to establish the security context and for authentication;
- The access control field of CMIS management operations must be supported to transfer security related information.

The security architecture also requires that agents have control over accesses to the MIB. This is necessary to enforce access control to MOs. Although TRUMPET is responsible for the implementation of agents, their design may be restricted by tools provided by the platform provider. For example, the code generated by a GDMO compiler may not be compatible with the introduction of access control mechanisms.

The architecture shown in Figure 2 is able to support most of the security services required by the TRUMPET policies. The security services that cannot be fully supported are integrity, confidentiality, non-repudiation and security negotiations.

When encryption for confidentiality is performed above CMISE, the encrypted data must be inserted into one of the fields of the particular CMIS operation being requested. However, most of the fields have specific pre-defined types that cannot accommodate an encrypted data type. In general, the only exception, and the only fields that can be encrypted, are the fields used to carry the attribute values to and from the target MIB.

1.3 High-Level Technology Viewpoint

This section presents the choices for implementation technologies used in the system as described in the previous sections, at a global or high level. These choices have been made so as to cover a number of requirements derived from the project goals and demands from the trials scenarios.

The figure below gives an overview of the technology viewpoint for this project.

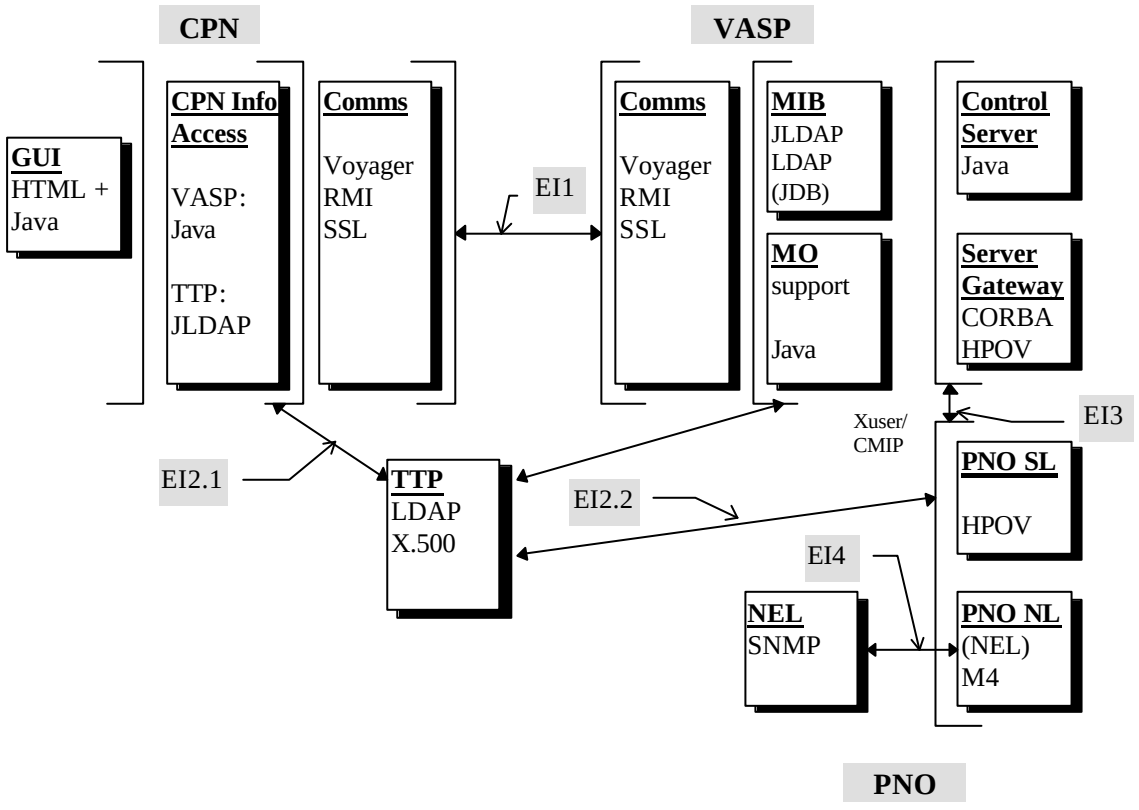


Figure 3: Overview of Technology Deployment

There are four major sites, each with specific communication needs the; CPN, VASP, PNOs, TTP. These are communication interfaces *external* to each domain and are labelled EI1-4. Although the components within each site also have inter process communications, these are not highly problematic and need not be considered in detail here. The communications channels considered are those which concern inter-domain security.

Key to Figure 3:

CORBA: Common Object Request Broker. An Object Oriented RPC like system suitable for building distributed systems over heterogeneous systems.

LDAP: Light weight Directory Access Protocol. A slimmed down implementation of X.500.

JLDAP: A Java API to LDAP

HTML: Hyper-Text Mark-up Language. A specialisation of SGML, adding functionality for linking material together in a networked environment (although loosing much of the type setting and document data base functionality of SGML).

Java: A trendy programming language plus a set of libraries suitable for building distributed systems and (light weight) user interfaces. Cross platform portability is provided by defining a Virtual Machine operating environment, rather than by forcing cross platform development in all environments and the utilisation of platform independent communications (*e.g.* CORBA or CMISE)

SNMP: Simple Network Management Protocol. For internet based network elements.

RMI: Remote Method Invocation. The low-level functionality for Java communications, in addition to pipes and sockets.

Voyager: A product which extends / completes Java communications to cover full functionality of such things as synchronous & asynchronous communications.

SSL: Secure Sockets Layer. A facility in Java for adding confidentiality, integrity and authentication to the classic socket and pipe communication model.

M4: A OSI management interface at the network and network element level defined by the ATMForum for ATM network management.

TTP: Is the trusted third party.

Summary of External Interfaces:

EI1: Between the Customer Premises and the VASP (customer) server. Using Java facilities. Low-level over IP (backbone or over ISDN) , High level protocol to support object reference model as supported by LDAP.

EI2: Between clients and TTP site. There are alternatives here supported by SecuDE; LDAP access over IP or (full) X.500 (TBD).

EI3: Between VASP Control server and PNO sites. Xuser over CMIP. Requires Java to CMIP gateway to be supported *via* CORBA.

EI4: From Network Elements Adapters to network elements. SNMP over IP.

2. SERVICE MANAGEMENT DEVELOPERS HANDBOOK

2.1 CPN User Application

2.1.1 Technology Object Model

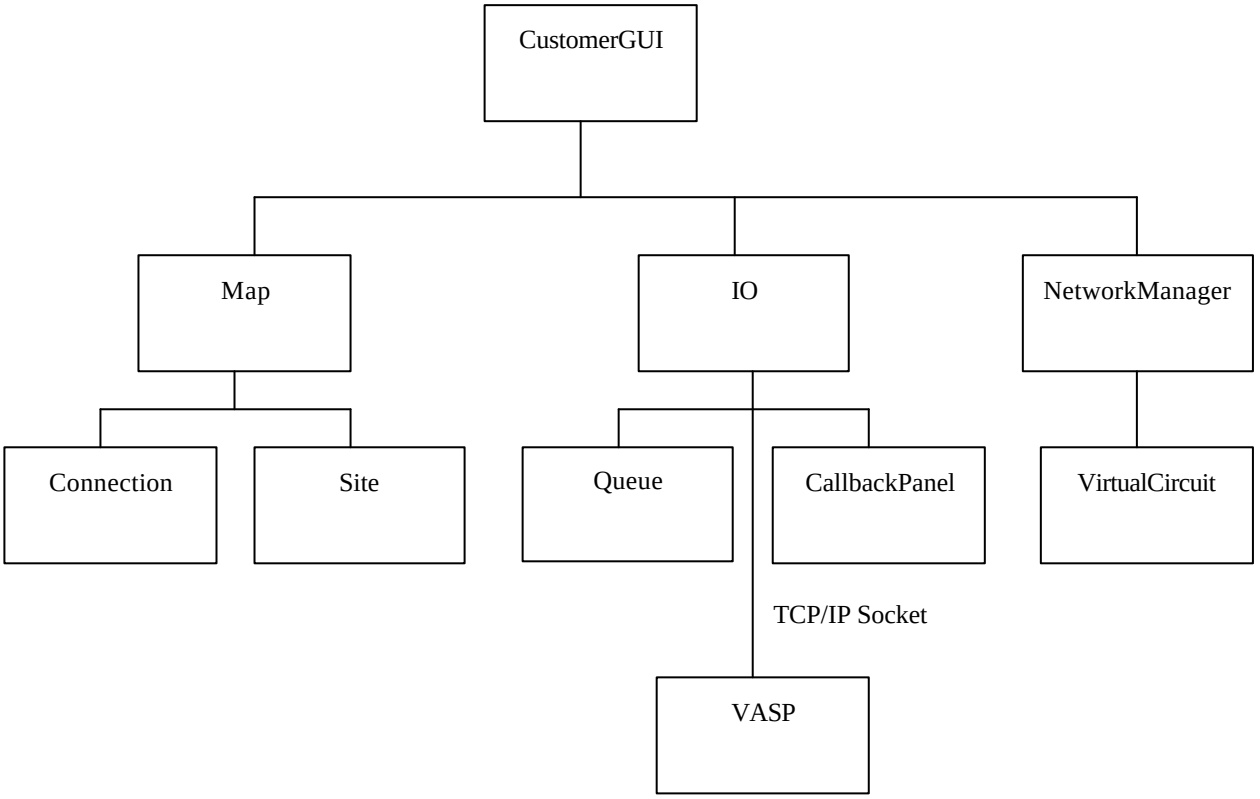


Figure 4: Engineering Object Model of the CPN User Application

The Map and CustomerGUI components extend class NameConstants in order to share global constants.

- **CustomerGUI**: is an applet designed to display a map on which sites and connections can be displayed and modified. The map on which the network is displayed is a background image of the applet. Buttons and checkboxes are placed on the screen in order to allow modification to the network. Events relating to the use of the buttons, mouse actions etc. are caught and methods of the Map class are called to make network modifications.
- **Map**: the Map component manipulates the map image displayed in the CustomerGUI applet to show the location of sites and the connections between them. Map also holds lists of the current sites and the connections between them.
- **Connection**: an instance of this type exists for each connection on the map. It holds the information required for each connection and has methods to modify or to obtain information on the connection.
- **Site**: an instance of this type exists for each site on the map. It holds the information required for each site and has methods to modify or to obtain information on the connection.
- **IO**: this component performs the communication between the CustomerGUI and the VASP. It uses a Queue class to store incoming and outgoing messages. The CallbackPanel passes messages from the CustomerGUI through to the IO component, prepending a unique instance Id.
- **Queue**: this class is used to buffer message to be sent to, or that have been received from the VASP.
- **CallbackPanel**: this class passes messages from CustomerGUI to the IO, prepending a unique instance Id.

- **NetworkManager**: this component handles messages to be passed to/from the vasp and calls appropriate methods in the Map class to manipulate the map as required.
- **VirtualCircuit**: this is a class to hold a representation of a virtual circuit.

2.1.2 Required and supported component interfaces

The CPN user application (the GUI) has only one interface which is a connection to the CPN Trumpet component via a socket.

- **Messages to the CPN**: messages are constructed as a string of tokens with the first token indicating the type of message this is. The messages sent are described below.
- **User Details**: the message header contains the string "UserDetails", the next two tokens in the string are the distinguished name and the password for the VASP LDAP server.
- **Get Connections**: the message header is "getConnections, no other string tokens are required.
- **Create Connections**: the message header is "createConnection". The following tokens describe the connection details such as site, location etc.
- **Messages from the CPN**: status messages are passed back from the CPN.

2.2 CPN Server

2.2.1 Engineering Object Model

The CPN structure closely follows that described in deliverable 8. The component hierarchy is shown below:

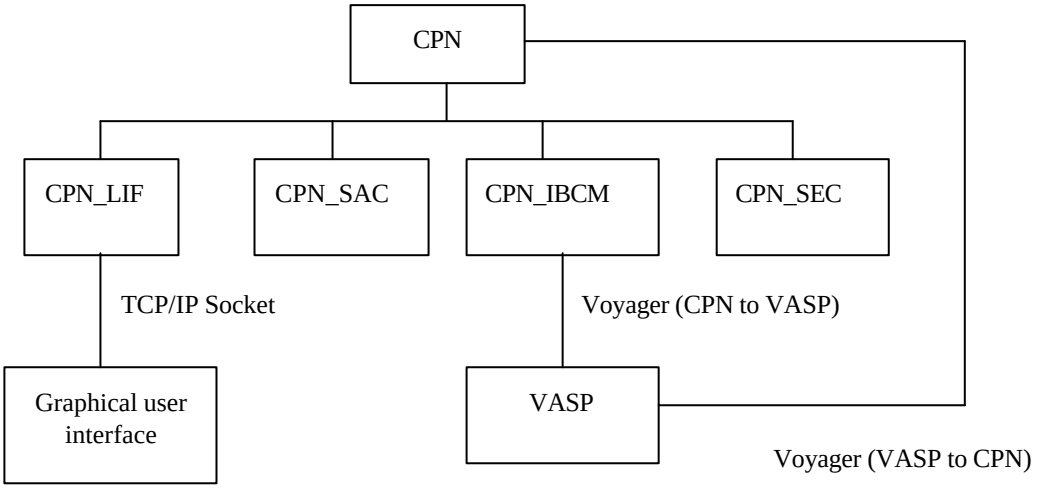


Figure 5: Engineering Object Model of the CPN User Application

All CPN components except for CPN_IBCM extend class CPN_CONSTANTS in order to share global constants between all components.

- **CPN:** invokes new instances of the other classes required constructing them with suitable parameters to enable communication with the GUI and VASP. Callbacks from the VASP via Voyager are also handled in this component as the use of Voyager precludes any other component performing this function. A new CPN_LIF class is invoked each time a connection is made to the local socket on which the CPN resides. This component also receives callbacks from the VASP using Voyager. This requires that the CPN is made Voyager aware in order that it can be accessed as a remote object from the VASP.
- **CPN_LIF:** is the CPN local interface for interfacing with the local system (currently via the GUI application). This component is threaded which allows more than one user to attach to the CPN. When constructed, the LIF registers itself with the CPN_SAC component. The CPN_SAC keeping a table of each connection with an associated id.
- **CPN_SAC:** the service access control point to provide a common interface between local facilities and the VASP. All calls from the LIF and callbacks received from the CPN_IBCM and the CPN itself are passed through the SAC.
- **CPN_IBCM:** provides the communications interface from the CPN to the VASP and uses Voyager as the interface medium. Voyager allows an association to be made to a remote instance of the VASP and call methods of the VASP object such as associate(), get() etc. directly.
- **CPN_SEC:** the security interface will handle security issues such as certification and key exchange.

2.2.2 Required and supported component interfaces

The CPN interfaces on one side to the graphical user interface using a TCP/IP socket and on the other side to the VASP using Voyager.

- **Interface to the Graphical User Interface:** the CPN_LIF interfaces to the GUI via a socket. More than one GUI can be connected to the CPN at any one time, a new instance of a CPN_LIF is created for each GUI requesting connection. The LIF then listens on the socket for any requests from the GUI. Currently, all requests from the GUI are passed to the method decodeDBQ which parses any messages from the GUI and calls methods in the CPN_SAC as appropriate. Messages recognised, and the action performed by the LIF are described below.

- **User Details**: the message header from the GUI contains the string "UserDetails", the next two tokens in the string are taken to be the distinguished name and the password for the VASP LDAP server. If tokens are received as expected then call the Make Association method in the CPN_SAC as below**:**

```
call CPN_SAC.MakeAssociation(distinguished name, password)
```

- **Get Connections**: the message header from the GUI is "getConnections", call the GetConnections method in the LIF as below**:**

```
CPN_SAC.GetConnections()
```

- **Create Connections**: the message header is "createConnection". Create an new instance of the virtual connection class and write the tokens from the string received into appropriate variables of the virtual connection class. Once completed successfully call the ReserveConnection method in the CPN_SAC as below.

```
CPN_SAC.ReserveConnection(VC);
```

  VC is the instance of the virtual connection class created.

- **Interface to the VASP**: the interface between the CPN and the VASP is made using the Voyager package. The CPN_IBCM component performs all interfacing TO the VASP. The IBCM must first register an instance of the VASP virtual object VAssociationServer. In order for this to be successful, the VASP must already be up and running. Once the association is made, the CPN can call methods in the VassociationServer class. Calling associate() on the VassociationServer passes back a reference to a VCustomerService class, which also resides in the VASP. The VcustomerService class has most of the methods which the CPN uses to communicate with the VASP.

VAssociationServer supports the following method**:**

METHOD associate

SYNOPSIS

```
VCustomerService associate(distinguished name, password, CPN host URL)
```

DESCRIPTION

The CPN_IBCM calls associate to make an association to the VASP LDAP server giving the required distinguished name and password. The path to the CPN is also given to allow the VASP to create a reference to the voyager aware CPN. A reference to a voyager aware VCustomerService object on the VASP is passed back.

VCustomerService supports the following methods**:**

METHOD get

SYNOPSIS

```
HashSet h = VCustomerService.get(
                                distinguished name,
                                scope,
```



```
                                    filter,
                                    Attribs);
```

DESCRIPTION

Returns a Voyager HashSet containing entries from the VASP LDAP server matching the arguments passed.

METHOD create

SYNOPSIS

```
vcs.create (Entryname, Entry);
```

DESCRIPTION

Create a new on the VASP with attributes as set in the Entry class passed. The Entry names recognized are “Customer” to create a new customer and “VASPVPConnection", to create a new connection.

In order to receive communications from the VASP the CPN object requires a Voyager aware version. This is constructed from the main() method in the CPN class. This object is then referenced remotely by the VASP. Currently the only method that the VASP calls on the CPN is

```
public void eventReport(String ConnectionID, boolean status)
```

which is used to pass back the current status of the given connection Id to the CPN.

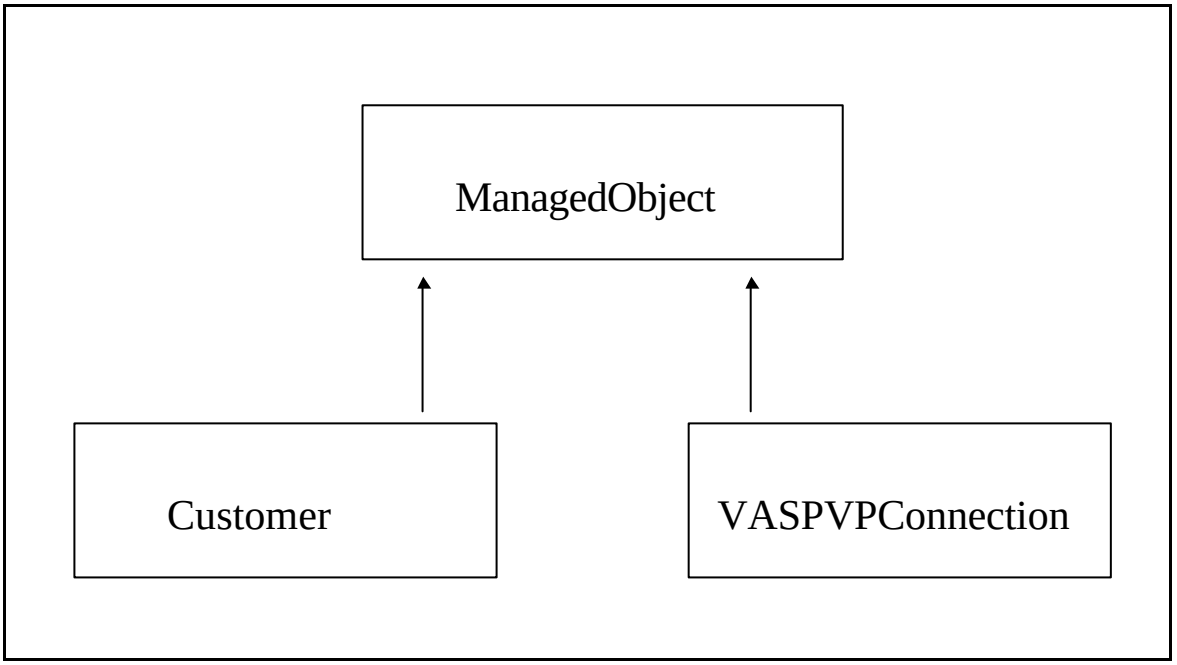
2.3 VASP Customer Server

2.3.1 Engineering Object Model

From section seven of D8, "VASP Information and Computational Models" several computational objects have to be realised as part of the Customer Server. The Computational objects incorporated into the Customer Server all form part of the MIB, or Management Information Base. The elements of the MIB are as follows:

- Customer MIB. This contains attributes such as Customer CPN ID, Termination point ID and Port ID.
- Connection MIB. This contains information about connection type, bandwidth and performance.

These are implemented using the classes Customer and VASPVPConnection. Both of these classes are managed objects are therefore extend the class ManagedObject. As part of the CustomerServer there is a facility for filtering and scoping for Managed Object selection. This is provided for by having Managed Objects reproduced along with certain of their Attributes in an LDAP Directory Server. Therefore a major function of Managed Objects is to make appropriate entries in the Directory Server. This is mostly accomplished from within the class Managed Object.



2.3.1.1 The Class Managed object

```
Class ManagedObject
{
String DistinguishedName;
AttributeList ALAttributes;
LDAPConnection LDAPmyConnection;

public ManagedObject(Entry, LDAPConnection)
{
    Constructs the managed object and assigns values to the above variables.
}

protected final void LDAPcreate(LDAPAttributeSet)
```



```
{
```

Takes the LDAPAttributeSet passed and constructs an Entry suitable for adding into the LDAP from this and the Distinguished name.

```
}

public void AlterAttributes(String, String)
{
```

Adds an attribute into the AttributeList held by the object.

```
}

protected final LDAPAttributeSet organizeAttributes(boolean)
{
```

Takes the AttributeList held in the ManagedObject class and converts it into an LDAPAttributeSet. If the value of the boolean is true, then the current Voyager Object Name is used as an LDAPAttribute to identify the Object. Otherwise a new Voyager Object Name is generated and used.

```
}

public Entry ReturnSelectAttrs(String[] )
{
```

The values of the Attributes held in the AttributeList are selected according to the contents of the String array passed as a parameter to this method. These are then built up into an Entry and returned.

```
}

private final CMISException  createAppropriateException(int iErr)
{
```

Generates and returns an appropriate error message.

```
}
}
```

2.3.1.2 The Class Customer

```
class Customer extends ManagedObject
{
public Customer(Entry, LDAPConnection)
{
```

calls the constructor of Managed Object with an Entry and a reference to the connection to the LDAP Directory Server. Then calls organizeAttributes( ) to return an LDAPAttributeSet, adds the attribute “objectClass” as being “Customer”. Then calls LDAPCreate( ) with this LDAPAttributeSet to create the entry in the LDAP.

```
}
}
```

2.3.1.3 The class VASPVPConnection.

```
class VASPVPConnection extends ManagedObject
{
public VASPVPConnection(Entry, LDAPConnection)
```



```
{
```

Calls the constructor of Managed Object with an Entry and a reference to the connection to the LDAP Directory Server. Obtains the Connection ID of this particular Connection and adds it as an attribute using AlterAttributes( ). A message is then passed to the ControlServer for it to set up the connection using reserveConnection( ).

Then calls organizeAttributes( ) to return an LDAPAttributeSet, adds the attribute "objectClass" as being "VASPVPConnection". Then calls LDAPCreate( ) with this LDAPAttributeSet to create the entry in the LDAP.

```
}

public void createVPSegment(String, String, String, String, String)
{
```

This is called from the ControlServer to allocate the individual PNO Segments.

```
}
}
```

2.3.2 Required and supported component interfaces

There are two sets of interfaces to the Customer Server. One between the CustomerServer and the code running on the Customer Premises Network, or CPN and one between the CustomerServer and the ControlServer.

2.3.2.1 Interface to the CustomerServer as seen by the CPN.

There are currently three methods that interface to the CustomerServer as seen by the CPN:

```
public synchronised VCustomerService associate (   String
DistinguishedName,
                            String password,
                            String address)

public EntrySet get(   String BaseDN,
           int scope,
           String filter,
           String[] SAattrs)

public void create(    String Stype,
           Entry Eentry)
```

2.3.2.2 Interface to the CPN as seen from the CustomerServer.

```
public void eventReport(    String ConnectionID, boolean status)
```

2.3.2.3 Interface to the ControlServer as seen by the CustomerServer.

```
public void reserveConnection(    String VaspVPId,
                String sourceCustId,
```



```
                String targetCustId,
                String sourceAddr,
                String targetAddr,
                ScheduleType schedule,
                QosSequenceTypeOpt qosParsOpt)
```

2.3.2.4 Interface to the CustomerServer as seen by the ControlServer.

```
public void AllocateSegment(     String ConnectionID,
                String pnoID,
                String pnoSegmentID,
                String accesspoint1,
                String accesspoint2)

public void allocateConnection(   String ConnectionID, boolean status)
```


2.4 VASP Control Server

2.4.1 Engineering Object Model

The controlServer's class hierarchy is rather flat. The only hierarchy that exists has to do with Managed Objects (MO) and is shown below:

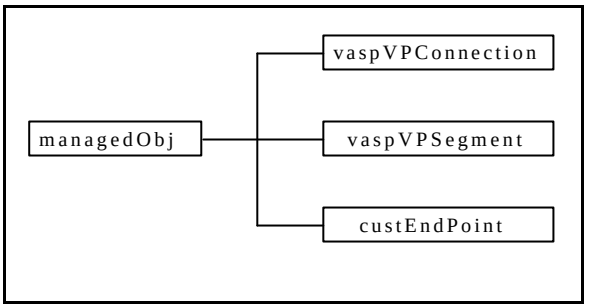


Figure 6: Engineering Object Model of the VASP Control Server

The controlServer keeps a local copy of the connection MIB for its own use. The objects that comprise the MIB are instances of the three leaf classes above. The class *managedObj* is an abstract class containing all the operation to have to do with the containement tree representing the MIB. For example, by inheriting from *managedObj*, all MIB objects are able to insert themselves into or remove themselves from the containement tree.

The class *VaspVpnManager* is the core of the controlServer. Only one instance of it is created. It is responsible for communicating with the customerServer on the one side, and the PnoConnectionMgr on the other. It receives customer requests from customerServer and further negociates with and forwards the requests to the involved PNOs, and vice versa, receives notifications from the PNOs and if necessary forwards them to the customerServer. As a result of the different received requests and notifications the MIB (containement tree) is updated.

The class *ControlServer* acts like a proxy of the *vaspVpnManager* with respect to the customerServer and likewise has only one instance. In other words, it is the *ControlServer* that really receives the requests from the customerServer. It then forwards them with no modification to the *vaspVpnManager*.

The rest of the classes are only support the functionality of the controlServer such as, reading the route table or keeping a list of PNOs with active connections to the controlServer.

2.4.2 Required and Supported Component Interfaces

The VASP controlServer module interfaces on one side with the VASP customerServer module and on the other side with the PnoConnectionManager object. This latter is an OrbixWEB proxy object representing the Xuser-Agent.

2.4.3 Interfaces with the customerServer

There are two interfaces defined between the customerServer and the controlServer modules. One defines the methods offered by the controlServer to the customerServer and is used to relay customer requests to the controlServer. The other one defines the methods offered by the customerServer to the ControlServer and is used to send notifications and the result of the requests to the customerServer. They are as follows:

2.4.3.1 ControlServer as seen by the CustomerServer

```
interface VPNService
{
public void reserveConnection( String VaspVPId,
```



```
                String sourceCustId,
                String targetCustId,
                String sourceAddr,
                String targetAddr,
                ScheduleType schedule,
                QoSSequenceTypeOpt qosParsOpt ) throws vaspException;

public void modifyConnection( String VaspVPId,
                 ScheduleTypeOpt  schedOpt,
                 QoSSequenceTypeOpt   qosParsOpt) throws vaspException;

public void releaseConnection( String VaspVPId ) throws vaspException;
}
```

2.4.3.2 customerServer as seen by the controlServer

```
interface customerServerFromControlServer
{
public void allocateConnection( String VaspVPId, boolean   status );
public void activateConnection( String VaspVPId, boolean   status );
public void releaseRequest( String VaspVPId );
public void modifyAccepted( String VaspVPId, boolean   status );
public void releaseNotify( String VaspVPId );
public void notify( String VaspId ReasonType   reason );
public void AllocateSegment(      String    VaspVPId, String    PnoId,
String    VaspSegId,  String    AccAddr1, String    AccAddr2 );
}
```

2.4.3.3 PnoConnectionManager as seen by the ControlServer

```
interface VPConnectionServiceOperations
{
public  XuserTypes.ReserveConnectionResultType
reserveConnection(XuserTypes.ReserveConnectionInfoType
connectionInformation )
throws
     PnoConnectionMgr._VPConnectionService.ConnectionRequestFailure,
IE.Iona.Orbixe.CORBA.SystemException;

public void
modifyConnection(XuserTypes.ModifyConnectionInfoType
connectionInformation )
throws
     PnoConnectionMgr._VPConnectionService.ConnectionRequestFailure,
IE.Iona.Orbixe.CORBA.SystemException;

public void
```



```
releaseConnection(XuserTypes.ReleaseConnectionInfoType
connectionInformation )
throws
     PnoConnectionMgr._VPConnectionService.ConnectionRequestFailure,
IE.Iona.Orbixe.CORBA.SystemException;
}
```

2.4.3.4 PnoConnectionManager as seen by the ControlServer

```
interface VPConnectionServiceEventHandlerOperations
{
public void
activateConnectionNotify(                          XuserTypes.NameType,
pnoId,XuserTypes.NameType, vpConnectionId, int status)
throws  IE.Iona.Orbixe.CORBA.SystemException;

public void
releaseConnectionNotify(    XuserTypes.NameType                    pnoId,
XuserTypes.NameType vpConnectionId,
XuserTypes.ReleaseReasonType   reason)
throws  IE.Iona.Orbixe.CORBA.SystemException;

public void
connectionNotify(  XuserTypes.NameType    pnoId,  XuserTypes.ReasonType
reason, String eventInformation )
throws  IE.Iona.Orbixe.CORBA.SystemException;
}
```

2.5 VASP CORBA/TMN Gateway

2.5.1 Engineering Object Model

The VASP CORBA/TMN Gateway has been introduced throughout the implementation design to provide the glue between the JAVA-based VASP management system and the TMN management solution provided for the PNO domain. Its primary purpose is to map between the TMN Xuser interface to an JAVA-based API which can be integrated with the VASP Control Server.

The gateway provides a set of adapter objects which exhibit a subset of the TMN Xuser interface. The interfaces of the adapter objects are defined using the interface definition language (IDL) which is part the CORBA specification [OMG CORBA]. At the programming level the IDL interfaces are mapped to suitable programming constructs (i.e. an JAVA API) according to language bindings. The implementation design of the gateway as shown in Figure 7 represents a refinement of the computational design for the PNO Service Layer Management which has been described in D8, Section 8.1.2 [TRUMPET-D8].

The object *CORBA/TMN Gateway* Server presents the initial object of the gateway which registers the gateway application with the communications infrastructure. Moreover, it creates two factory objects which are provided to the *VASP Control Server* to create and delete instances of the service objects. The gateway provides two kinds of service objects called *VPConnectionService* and *VPSubscriptionService* which represent the core interface to the PNO Service Management. These objects provide the required functionality to manage VP connections and to maintain customer access points. Additionally, the *VPConnectionServiceEventHandler* provides means to forward event reports of the PNO Service Management to the VASP Control Server. This object is part of the VASP Control Server and provides functions which may be invoked by the gateway to indicate event reports as described in D8, section 8.1.2.

The core implementation of the service objects *VPConnectionService* and *VPSubscriptionService* is provided by the *XuserMgrRequestHandler* which maps the operations of the service objects down to CMIP requests using the XMP/XOM API of HP OpenView DM. The request handler also serves as a co-ordinator for the CORBA and HP-OV communication channels. It realises an event loop which is waiting for indications of CORBA IIOP or CMIP protocol requests.

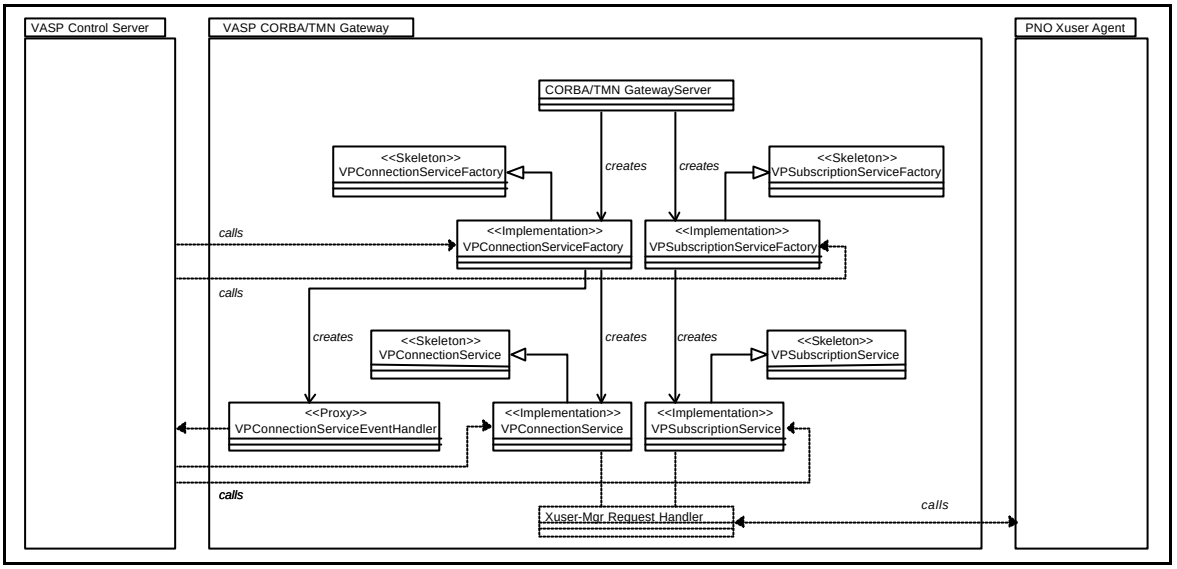


Figure 7: Engineering Object Model of the VASP CORBA/TMN Gateway

2.5.2 Required and supported component interfaces

Figure 8 presents an overview on the required and supported interfaces of the VASP CORBA/TMN gateway. All the supported interfaces are provided to the VASP Control Server which utilises the VP connectivity services of the PNO Service Layer Management. For the notification on event reports received from the PNO Service Layer Management a *VPConnServiceEventHandler* interface is required which is provided by the VASP Control Server. Moreover, the VASP CORBA/TMN gateway makes use of the Xuser interface which is provided by the PNO Xuser Agent.

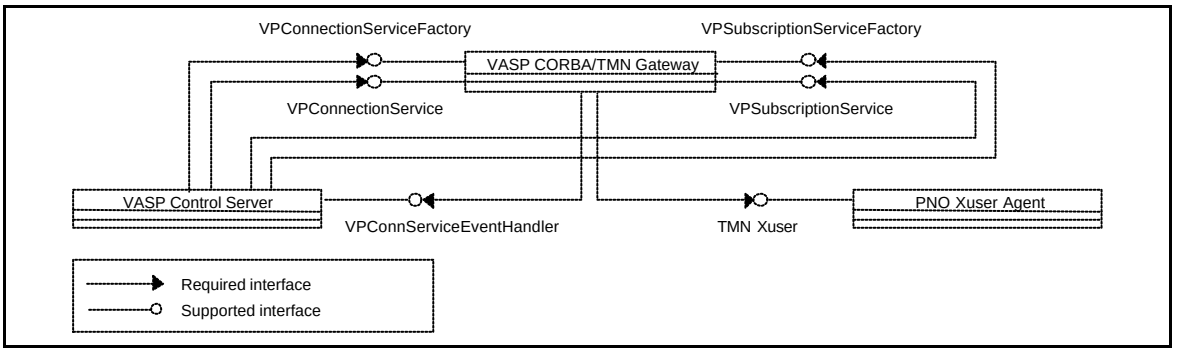


Figure 8: Required and supported interfaces of the VASP CORBA/TMN Gateway

The remainder of this section only describes the details of the supported interfaces as the required interfaces are described already in the corresponding sections of required and supported interfaces for the VASP Control Server (Section 2.4.2) and for the PNO Xuser Agent (Section 2.6.2).

2.5.2.1 VP Connection Service Factory

FUNCTION

```
PnoConnectionMgr::VPConnectionServiceFactory::create()
```

SYNOPSIS

```
VPConnectionService create(
      in XuserTypes::NameType pnoId,
      in VpnManager::VPConnectionServiceEventHandler eventHandler);
```

DESCRIPTION

Creates a new service object of type *VPConnectionService* for the interaction with the Service Layer Management System of the PNO identified by *pnoId*. The new service object is associated with an event handler which is referenced by the *eventHandler* parameter.

ARGUMENTS

- *pnoId*: Identifies the PNO Service Layer Management System. The identifier may be either a presentation string containing the global distinguished name of the PNO MAE or a number which can be mapped to the distinguished name according to an mapping table of the CORBA/TMN gateway.
- *eventHandler*: Contains an object reference to an event handler object *VPConnectionServiceEvent-Handler* which is associated with the new service object. The event handler is triggered by the service object if an event report from the PNO Service Layer Management System is indicated.

RESULTS

Returns an object reference to the new service object on success. If the service object cannot be created a nil object reference is returned.

BUGS

An exception type should be defined to indicate possible failures (i.e., PNO not found, communication failure, etc.)

FUNCTION

`PnoConnectionMgr::VPConnectionServiceFactory::delete()`

SYNOPSIS

```
void delete(
      in VPConnectionService vpConnectionServiceRef);
```

DESCRIPTION

Deletes the service object of type *VPConnectionService* identified by the object reference given in *vpConnectionServiceRef*. After successful deletion of the service object the event handler object which had been be associated with service object will not be triggered anymore to indicate event reports.

ARGUMENTS

- *vpConnectionServiceRef*: Contains an object reference to an service object of type *vpConnection-ServiceRef*.

RESULTS

∅.

BUGS

An exception type should be defined to indicate possible failures of this operation (i.e., object not found).

2.5.2.2 VP Connection Service

FUNCTION

```
PnoConnectionMgr::VPConnectionService::reserveConnection()
```

SYNOPSIS

```
XuserTypes::ReserveConnectionResultType reserveConnection(
      in XuserTypes::ReserveConnectionInfoType connectionInformation)
            raises (ConnectionRequestFailure);
```

DESCRIPTION

Reserves a new VP connection at the associated PNO according to the details given by the parameter *connectionInformation.*

ARGUMENTS

The parameter *connectionInfo*, contains the following components:

- *userId*: Identifies the user of the connectivity service offered by the PNO. According to the TRUMPET scenario this parameter identifies the VASP.
- *sourceE164AddressOpt*: May contain the E164 source address of the customer access point.
- *destinationE164Address*: Contains the E164 destination address of the customer access point.
- *connectionProtectionLevelOpt*: May contain the protection level for the VP connection. Possible values (if not omitted) are *protected*, *unprotected-lowpriority*, *unprotected-highpriority*.
- *routingCriteriaOpt*: May contain customised settings for the routing algorithm implemented as part of the PNO's connectivity service. Currently no options have been defined for the routing algorithm used by the TRUMPET PNO Service Layer Management System.
- *directionality*: Identifies the directionality of the requested VP connection. The value may be either *unidirectional* or *bidirectional*.
- *schedule*: Defines the schedule for the requested VP connection.
- *qosParametersOpt*: May contain a set of QOS parameters for the requested VP connection.

RESULTS

Returns an object type *ReserveConnectionResultType* on successful operation. This object contains the connection id for the schedule connection and may additional contain the select source customer access point. If the operation fails an exception of type *ConnectionRequestFailure* is raised which contains an object of type *ReasonType.* In the current implementation of the CORBA/TMN gateway this object will contain an error number.

BUGS

Instead of a general *ConnectionRequestFailure* exception which provides an error number, there should be rather several exception types defined for different types of errors which may occur.

SEE ALSO

```
PnoConnectionMgr::VPConnectionServiceFactory
```


FUNCTION
`PnoConnectionMgr::VPConnectionService::modifyConnection()`

SYNOPSIS
```
void modifyConnection(
      in XuserTypes::ModifyConnectionInfoType connectionInformation)
            raises (ConnectionRequestFailure);
```

DESCRIPTION

Modifies the characteristics of an pending or scheduled VP connection at the associated PNO as identified by the parameter *connectionInformation*. Modifications of both the schedule and the set of QoS parameters are possible.

ARGUMENTS

The parameter *connectionInfo*, contains the following components:

- *userId*: Identifies the user of the connectivity service offered by the PNO. According to the TRUMPET scenario this parameter identifies the VASP.
- *connectionId*: Identifies the VP Connection to be modified.
- *scheduleOpt*: may contain a new schedule for the given VP connection.
- *qosParametersOpt*: May contain a set of QOS parameters for the given VP connection.

RESULTS

Nothing is returned on successful operation. If the operation fails an exception of type *ConnectionRequest-Failure* is raised which contains an object of type *ReasonType*. In the current implementation of the CORBA/TMN gateway this object will contain an error number.

BUGS

Instead of a general *ConnectionRequestFailure* exception which provides an error number, there should be rather several exception types defined for different types of errors which may occur.

SEE ALSO
`PnoConnectionMgr::VPConnectionServiceFactory`

FUNCTION
`PnoConnectionMgr::VPConnectionService::releaseConnection()`

SYNOPSIS
```
void releaseConnection(
      in XuserTypes::ReleaseConnectionInfoType connectionInformation)
            raises (ConnectionRequestFailure);
```

DESCRIPTION

Releases an pending or scheduled VP connection at the associated PNO as identified by the parameter *connectionInformation*.

ARGUMENTS

The parameter *connectionInfo*, contains the following components:

- *userId*: Identifies the user of the connectivity service offered by the PNO. According to the TRUMPET scenario this parameter identifies the VASP.
- *connectionId*: Identifies the VP Connection to be released..

RESULTS

Nothing is returned on successful operation. If the operation fails an exception of type *ConnectionRequest-Failure* is raised which contains an object of type *ReasonType*. In the current implementation of the CORBA/TMN gateway this object will contain an error number.

BUGS

Instead of a general *ConnectionRequestFailure* exception which provides an error number, there should be rather several exception types defined for different types of errors which may occur.

SEE ALSO
`PnoConnectionMgr::VPConnectionServiceFactory`

2.5.2.3 VP Subscription Service Factory

FUNCTION
`PnoConnectionMgr::VPSubscriptionServiceFactory::create()`

SYNOPSIS
```
VPSubscriptionService create(in XuserTypes::NameType pnoId);
```

DESCRIPTION

Creates a new service object of type *VPSubscriptionService* for the interaction with the Service Layer Management System of the PNO identified by *pnoId*.

ARGUMENTS

- *pnoId*: Identifies the PNO Service Layer Management System. The identifier may be either a presentation string containing the global distinguished name of the PNO MAE or a number which can be mapped to the distinguished name according to an mapping table of the CORBA/TMN gateway.

RESULTS

Returns an object reference to the new service object of type *VPSubscriptionService* on success. If the service object cannot be created a nil object reference is returned.

BUGS

An exception type should be defined to indicate possible failures (i.e., PNO not found, communication failure, etc.)

FUNCTION
`PnoConnectionMgr::VPSubscriptionServiceFactory::delete()`

SYNOPSIS
```
void delete(in VPSubscriptionService vpSubscriptionServiceRef);
```

DESCRIPTION

Deletes the service object of type *VPSubscriptionService* identified by the object reference given in *vpSubscriptionServiceRef*. After successful deletion of the service object the event handler object which used to be associated with service object will not be triggered anymore to indicate event reports.

ARGUMENTS

- *vpSubscriptionServiceRef*: Contains an object reference to an service object of type *vpSubscription-ServiceRef*.

RESULTS

∅.

BUGS

An exception type should be defined to indicate possible failures of this operation (i.e., object not found).

2.5.2.4 VP Subscription Service

FUNCTION

```
PnoConnectionMgr::VPSubscriptionService::createAccessPoint
```

SYNOPSIS

```
void createAccessPoint(
      in XuserTypes::IdentifierType userId,
      in XuserTypes::NameType accessPointId,
      in XuserTypes::E164AddressType E164Address)
            raises (InvalidAccessPoint);
```

DESCRIPTION

Registers a customer access point with PNO Service Layer Management System for the user identified by the parameter *userId*. Note, that in the TRUMPET scenario the user role is always taken by the VASP management system which may register access points on behalf of its customers. The new access point is identified by *accessPointId* which serves as the value for the naming attribute of the created object instance within the PNO Service Layer Management System. The third parameter defines an globally unique number which is associated with the new access point.

ARGUMENTS

- *userId*: Identifies the user of the connectivity service offered by the PNO. According to the TRUMPET scenario this parameter identifies the VASP.
- *accessPointId*: Serves as the value for the naming value of the object instance to be created within the PNO Service Layer Management System. For the given user the identifier provided has to be unique.
- *E164Address*: Defines an globally unique number which is associated with the new access point.

RESULTS

Nothing is returned on successful operation. If the operation fails an exception of type *InvalidAccessPoint* is raised which indicates an invalid access point identifier or an invalid E164 address.

BUGS

Additional exception types should be defined to indicate the error conditions.

SEE ALSO

```
PnoConnectionMgr::VPSubscriptionServiceFactory
```


FUNCTION
`PnoConnectionMgr::VPSubscriptionService::deleteAccessPoint`

SYNOPSIS
```
void deleteAccessPoint(
      in XuserTypes::IdentifierType userId,
      in XuserTypes::NameType accessPointId)
            raises (NotFound);
```

DESCRIPTION

Removes a customer access point from the PNO Service Layer Management System for the user identified by the parameter *userId*. Note, that in the TRUMPET scenario the user role is always taken by the VASP management system which may register access points on behalf of its customers. The access point to be deleted is identified by *accessPointId*. Note, that access points can only be deleted by its creator.

ARGUMENTS

- *userId*: Identifies the user of the connectivity service offered by the PNO. According to the TRUMPET scenario this parameter identifies the VASP.
- *accessPointId*: Serves as the value for the naming value of the object instance to be deleted within the PNO Service Layer Management System. For the given user the identifier provided has to be unique.

RESULTS

Nothing is returned on successful operation. If the operation fails an exception of type *InvalidAccessPoint* is raised which indicates an invalid access point identifier.

BUGS

Additional exception types should be defined to indicate the error conditions.

SEE ALSO
`PnoConnectionMgr::VPSubscriptionServiceFactory`

2.6 PNO Xuser-Agent

2.6.1 Engineering Object Model

The PNO Xuser-Agent realises the PNO Service Layer Management System which provides the VP Connection Management Service. This service qualifies the VASP Management System to manage a segment of an end-to-end virtual path between to customer access points of the public network domain.

The implementation design of the Xuser-Agent (see Figure 9) is based on the computational of the PNO Service Layer Management which has been described in D8, Section 8.1.2 [TRUMPET-D8]. The TMN Xuser-interface provided the Xuser-Agent has been adopted from the MISA project according to the agreements between the TRUMPET and MISA projects. However, TRUMPET uses only a subset of the Xuser-interface, namely those functions for the connection management (MISA Path Provisioning Ensemble [MISA-D3-A1]) as well as some basic functions provided with the subscription management (MISA Subscription Ensemble [MISA-D3-A2]).

The object PNO Xuser Agent Co-ordinator presents the initial object of the Xuser-Agent which registers application with communication infrastructure, and creates the basic agent object, namely *the MIT Manager* and *Agent Request Handler*. The *MIT Manager* is responsible for data management of the Management Information Base and provides functions to invoke operations on the contained Managed Objects. The *Agent Request Handler* realises an event loop which awaits indications for CMIP operation requests. Moreover, the *Agent Request Handler* initially creates the Dispatcher Object which maps CMIP request to operations called on Managed Object maintained by the *MIT Manager*.

Operations invoked on a Managed Objects will change the state of the Management Information Bases and may also result in operation invocations on a managed resource. In particular this is the case for the operations provided by *GBCServiceProvider* which are mapped internally to corresponding operation

invocations on the underlying PNO Network Layer Management System. The handling of the operations of the Network Layer Management System is provide by PNO Network Management.

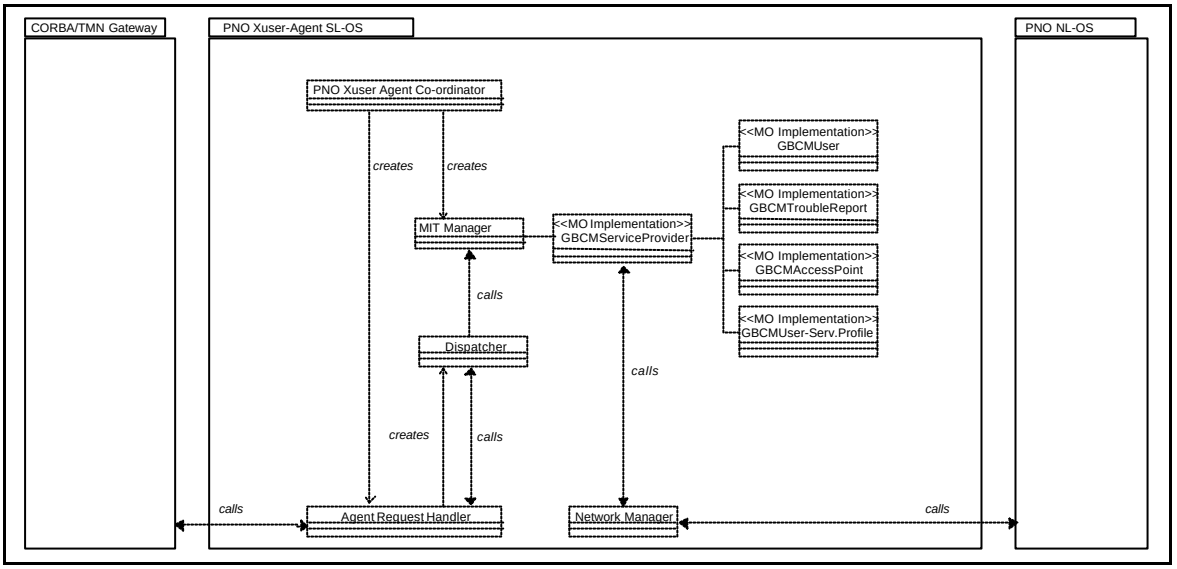


Figure 9: Engineering Object Model of the PNO Xuser-Agent

2.6.2 Required and supported component interfaces

The external interface provided by the Xuser-Agent corresponds to the Xuser-specification which has been developed by the MISA project [MISA-D3A1]. It is therefore not described here in detail. The latest release of the MISA Xuser specification which has been used as a basis for the implementation of the PNO Xuser-Agent can be found in Section XXX. In the remainder of this section only the basic operations provided for the connection management are described.

2.6.2.1 GBCServiceProvider

ACTION TYPE
`reserveGBCConnection`

BASE OBJECT CLASS
`GBCMServiceProvider`

BASE OBJECT INSTANCE
`GBCMServiceProvider instance`

DESCRIPTION

This action is performed by the GBCM User which requests a GBC connection reservation from the GBCM Service Provider. The result of this action is the acceptance or reject of the connection reservation request (regarding the start time, the stop time and eventually the periodicity requested). If the connection reservation is rejected, the reason is returned (not available resources, not possible in the interval time,...). If the connection reservation is accepted, a gBCConnection object instance is created.

ACTION INFORMATION

The information type *reserveGBCConnectionInformation*, contains the following components:

- *gBCMUserId*: Identifies the user of the connectivity service offered by the PNO.
- *sourceE164Address*: May contain the E164 source address of the customer access point.
- *destinationE164Address*: Contains the E164 destination address of the customer access point.
- *connectionProtectionLevel*: May contain the protection level for the VP connection. Possible values (if not omitted) are *protected*, *unprotected-lowpriority*, *unprotected-highpriority*.

- *routingCriteria*: May contain customised settings for the routing algorithm implemented as part of the PNO's connectivity service. Currently no options have been defined for the routing algorithm used by the TRUMPET PNO Service Layer Management System.
- *gBCType*: Identifies the type of connectivity service to be used. The current Xuser-specification defines the ATM and SDH Path Provisioning Service (APPS, SPPS).
- *gBCDirectionality*: Identifies the directionality of the requested VP connection. The value may be either *unidirectional* or *bi-directional*.
- *gBCSchedule*: Defines the schedule for the requested VP connection.
- *gBCPPSparameters*: May contain a set of QOS parameters for the requested VP connection.

ACTION RESULT

On successful operation the action results of type *modifyGBCConnectionResult* contains the connection id for the schedule connection and may additional contain the select source customer access point. If the operation fails the result contains an error number.

ACTION TYPE
`modifyGBCConnection`

BASE OBJECT CLASS
`GBCMServiceProvider`

BASE OBJECT INSTANCE
`GBCMServiceProvider instance`

DESCRIPTION

This action is performed by the GBCM User requesting the modification of the GBC connection. In case of SPPS (SDH), it is possible that modification is not supported. In this case the action request will be rejected."

ACTION INFORMATION

The information type *modifyGBCConnectionInformation*, contains the following components:

- *gBCMUserId*: Identifies the user of the connectivity service offered by the PNO.
- *gBCConnectionId*: Identifies the VP Connection to be modified.
- *gBCSchedule*: May contain a new schedule for the given VP connection.
- *gBCPPSparameters*: May contain a set of QOS parameters for the given VP connection.

ACTION RESULT

On successful operation the action results of type *modifyGBCConnectionResult* contains nothing. If the operation fails the result contains an error number.

ACTION TYPE
`releaseGBCConnection`

BASE OBJECT CLASS
`GBCMServiceProvider`

BASE OBJECT INSTANCE
`GBCMServiceProvider instance`

DESCRIPTION

This action is performed by the GBCM User requesting the clearing down of the GBC Connection. This will delete the *gBCConnection* object instance.

ACTION INFORMATION

The information type *releaseGBCConnectionInformation*, contains the following components:

- *gBCMUserId*: Identifies the user of the connectivity service offered by the PNO.
- *gBCConnectionId*: Identifies the VP Connection to be released.

ACTION RESULT

On successful operation the action results of type *releaseGBCConnectionResult* contains nothing. If the operation fails the result contains an error number.

3. SERVICE MANAGEMENT INSTALLATION GUIDE

3.1 CPN User Application

3.1.1 Hardware and software pre-requisites

The CPN User Application (the GUI) is a Java application and therefore it should be possible to run on any machine capable of running Java. **N.B. in order to ensure that the user application works correctly with the widest number of browsers, use JDK version 1.0.2 to compile it.** Unfortunately, not all browsers which claim to work with higher versions of Java do so in the correct manner, i.e. they have bugs. The file CustomerGUI.html is the file that is invoked by the browser.

Parameters within the CustomerGUI.html file may require modification. These set the name of the host on which the GUI will be run (default = localhost) and the socket with which communication is made with the CPN (default = 3001).

- Components needed for the GUI
- The GUI components themselves, all in a single directory.
- A web browser which supports Java 1.0.2 or the appletviewer from the JDK used to compile the GUI
- JDK for 1.0.2 for compilation if required.

3.1.2 Installation and configuration instructions

The software will be provided as a set of files, or the same files in .zip format. Copy/uncompress all files to a single directory.

The application is completely written in Java. If it is required to compile the Java code, make sure that the path variable contains the path to JDK 1.0.2 binary files and execute the following command

Javac *.java

This will create all the required classes.

Parameters within the CustomerGUI.html file may require modification. These set the name of the host on which the GUI will be run (default = localhost) and the socket with which communication is made with the CPN (default = 3001).

3.1.3 Runtime

To run the user application, select the CustomerGUI.html file from a web browser package. Or use the JDK appletviewer program to invoke the same file with the command

<path to JDK binary files>/appletviewer CustomerGUI.html

3.1.4 Version / release history

Version 2.0 to be released December 1997

3.1.5 1.2.5 Known bugs

None.

3.2 CPN Server

3.2.1 Hardware and software pre-requisites

The CPN is a Java application as are the associated packages. Thus it should be possible to run the CPN on any machine capable of running Java. Components needed for the CPN:

- The CPN components themselves, placed in a single directory.

- Copies of the VASP Voyager aware classes VAssociationServer and VCustomerService
- Objectspace Voyager 1.0
- Objectspace JGL Version 2.0.2
- Netscape LDAP Java SDK 1.0
- JDK for 1.1.4 for compilation if required.

3.2.2 Installation and configuration instructions

The software will be provided as a set of files, or the same files in .zip format. Copy/uncompress all files to a single directory.

The application is completely written in Java. If it is required to compile the Java code, make sure that the path variable contains the path to JDK 1.0.2 binary files and execute the following command

javac *.java

The classes **VAssociationServer** and **VCustomerService** provide the interface to the VASP and must be placed in the same directory as the CPN classes. Up to date versions of these classes must be obtained from the authors of the VASP component.

The environment variable CLASSPATH must include references to the following:

- voyager\lib\voyager1.0.0.jar
- jgl\jgl_2_0
- netscape\ldap\classes

The PATH environment variable should include references to the Java JDK (1.1.4) binary and Voyager executables directories. Ensure that the VASP Voyager objects are placed in the same directory as the CPN server classes.

3.2.3 Runtime

To initiate the CPN execute the following command line:

java CPNServer <Socket to GUI> <VASP host> <VASP server port>

If no arguments are given, default values are used as listed below:

- Socket to GUI = 3001
- VASP host = "localhost"
- VASP port = 8000

3.2.4 Version / release history

Version 2.0 to be released December 1997

3.2.5 Known bugs

None.

3.3 VASP Customer Server

3.3.1 Hardware and software pre-requisites.

The software for the CustomerServer is comprised of the following classes: AssociationServer, CustomerService, VASPVPConnection, Customer, VPSegment and the following associated classes for handling data in the form of entries and its components: Attribute, AttributeList, Entry and EntrySet.

The files ControlServer.java and CustomerServercomm.java which facilitate communication between the CustomerServer and the ControlServer.

Also included is the file Trumpet.new.ldif, which is used for building up the LDAP database and classes which are used purely for testing purposes and which simulate the operation of other components of Trumpet. These are: VASPgui.java, VCPN.java and CPN.java, which act as the CPN component, and SharTest.java and vaspVpnManager.java which represent the operation of the ControlServer. It is envisaged that these will eventually be discarded and the real CPN and ControlServer components will be substituted.

The following will be required in order to run the classes that comprise the CustomerServer properly:

- JDK 1.1.3 or later installed and available.
- An LDAP Directory server. Currently Netscape's Directory Server 1.02 is being used.
- ObjectSpace's Voyager 1.0.0 installed and available.
- Netscape's LDAPjava SDK1.0 available.
- Objectspace's JGL 2.0.2 set of Data Structure classes.

3.3.2 Installation and configuration instructions

The software will be provided as a set of files, or the same files in .zip format. Copy/uncompress all files to a single directory.

The application is completely written in Java. If it is required to compile the Java code, make sure that the path variable contains the path to JDK 1.1.4 binary files and execute the following command

javac *.java

The environment variable CLASSPATH must include references to the following:

- voyager\lib\voyager1.0.0.jar
- jgl\jgl_2_0
- netscape\ldap\classes

The PATH environment variable should include references to the Java JDK (1.1.4) binary and Voyager executables directories. Ensure that the VASP Voyager objects are placed in the same directory as the CPN server classes.

3.3.3 Runtime

Execute the following instructions:

1. java AssociationServer.
2. Build up a database in the LDAP Directory Server using the file Trumpet.new.ldif which is supplied with the release.
3. Run the CPN programs and when they call the Associate method supply a distinguished name, the Directory Server password and the URL of the machine the CPN is running on. Using the supplied CPN program the following would be entered:

java CPN "cn=Directory Manager, o=vasp" Directory-Server-Password URL-of-CPN-machine

3.3.4 Version / release history

Version 1: November 1997.

3.3.5 Known bugs

None.

3.4 VASP Control Server

3.4.1 Hardware and software pre-requisites.

The following will be required in order to run the classes that comprise the ControlServer properly:

- JDK 1.1.3 or later installed and available.
- IONA OrbixWeb 2.1 RunRime

3.4.2 Runtime

This module is started by running the *startContServer* java class. This class contains only one *main* method that creates one instance of the *vaspVpnManager* class. The main method expects one parameters which is the Id of the VASP used in communication with the PNOs. Moreover, the controlServer module uses a routing table containing static routing information. The routing information should be described in a file, the routing table, prior to starting up the VASP.

The full name of this table file, i.e., the full path-name plus the file name, is made known to the controlServer by means of Java's *property* mechanism. The property name chosen for the route table is **ROUTETABLE_PATH**. The value of this property is set by mean of the **-D** option of the **java** command. For example, assuming that the routing information is in the file *myRoutingTable* with a full path *myRoutingTablePath*, and that the Id of the VASP is *TRUMPETVasp*, one should start up the VASP in the following way:

```
java -DROUTETABLE_PATH=/myRoutingTablePath/myRoutingTable
                    startContServer  TRUMPETVasp
```

Note that if the ROUTETABLE_PATH is not set, no connection can be managed/set up.

The format of this file is given below in the form of an example that illustrates its content:

```
cpnId: NR
accesspoint: NRAtmSW1

pnoId: pnoNorway
accesspoint: nAccessAddress1
accesspoint: nAccessAddress2

pnoId: pnoSwiss
accesspoint: swAccessAddress1
accesspoint: swAccessAddress2

pnoId: pnoScotland
accesspoint: scAccessAddress1
accesspoint: scAccessAddress2

cpnId: EPFL
accesspoint: EPFLAtmSW5
```

The semantic of this table is that the access point of the first customer (NR) connects to first access point of the following PNO (pnoNorway), and the second access point of that PNO connects to the first access point of the next PNO (pnoSwiss) in the list, and so on. Finally, the second access point of the last PNO in the list (pnoScotland) connects to the access point of the other customer (EPFL).

3.4.3 Version / release history

Version 1: November 1997.

3.4.4 Known bugs

None.

3.5 VASP CORBA/TMN Gateway

3.5.1 Hardware and software prerequisites

- SUN Solaris 2.x with SUN SPARCworks C++ compiler 4.x ***or*** HP HP-UX 10.x with HP aC++.
- IONA Orbix 2.x / C++
- IONA OrbixWeb
- HP Open View 4.21 DM

3.5.2 Installation and configuration instructions

3.5.2.1 Directory $TRUMPET_TOP/src/corbaGateway

This directory is composed of 3 subdirectories. The idl and manager subdirectories provide includes and libraries, while the third one (adapter) contains the main program, acting as a CORBA server for the VASP Control Server (JAVA client).

Files:

- *Makefile*: describes the list of different subdirectories needed during the compiling phase of the corbaGateway.
- *idl.mk*: describes all the variables needed to access the includes and libraries implementing the IDL definition.It is used in the manager and adapter directories.
- *manager.mk*: describes all the variables needed to access the includes and library implementing the mapping between C++ objects and C structures and all the functions needed to allow protocol with HP Open View and the Xuser agent. It is used in the manager and adapter directories.
- *xuseragent.mk*: describes all the variables needed to acces the includes and libraries needed to allow protocol with the Xuser agent. It is used in the manager and adapter directories.
- *runGateway.sh*: shell script which sets different environmental variables and run the gateway executable.

Directories:

- *idl*: contains source code for the includes/libraries implementing the IDL definition. This directory creates 2 libraries. The server library is used by the CORBA part of the gateway,and will be linked during the build process of the gateway, in the adapter directory. The client library is only used by a client proces used for testing only.
- *manager*: contains source code for the includes/library implementing the mapping between C++ objects and C structures and all the functions needed to allow protocol with HP Open View and the Xuser agent.
- *adapter:*

Installation instructions:

1. in the file ../Config.mk, check the following makefile variables:
   - TRUMPET_TOP: name of the Trumpet top directory.
   - CCC: access path to the C++ compiler.

- ORBIX_DIR: access path to the Orbix tool (Orbix_2.2MT means it is a multithreaded version, see below)
- ORBIX_INC: add or suppress "-mt -D_REENTRANT" option whether the Orbix tool version is multithreaded or not.
- ORBIX_LIB: add or suppress "-mt" whether the Orbix tool version is multithreaded or not.
- HPOV: access path to the HPOV tool

2. in the file xuseragent.mk: if the location of the Xuser agent code is not at the same level than the corbaGateway directory, e.g. $(TRUMPET_TOP)/src/xuser, modify XUSERAGENT_DIR to the appropriate location.
3. in the file runGateway.sh, check ...
4. the environmental variable SYSTEM must be set to an appropriate value: 'sun5' for Solaris 2.x, 'hp' for HP-UX 10.x, thanks to a command like: setenv SYSTEM sun5
5. when everything is ready, first launch the command “make clean” to be sure that all object files and executables have been deleted.
6. then launch the command “make depend” to be sure that the file deps.mk is corresponding to the current operating system.
7. then launch the command “make”

3.5.2.2 Directory $TRUMPET_TOP/src/corbaGateway/idl

Files:

- Makefile: describes the list of different C++ files to be generated thanks to the IDL files and tools, and how to generate the 2 libraries.
- ASN1Types.idl: basic IDL types declarations.
- XuserTypes.idl: IDL view of Xuser agent GDMO MIB described in the Xuser agent directories.
- PnoConnectionMgr.idl and VpnManager.idl: Trumpet definitions.

Installation / configuration instructions:

1. Usually, this is not a good idea to just recompile something in this directory. The build process should be started from the $TRUMPET_TOP directory, in order to have all changes propagated everywhere !
2. Follow the instructions 1,2,3,4,5,6,7,8 and 9 of the corbaGateway directory to compile and generate the 2 libraries: libcidl and libsidl in both 2 different formats (static and shared).

3.5.2.3 Directory $TRUMPET_TOP/src/corbaGateway/manager

This directory creates a library used by the Xuser manager part of the CORBA/TMN gateway. It will be linked during the build process of the gateway, in the adapter directory.

Files:

- Makefile: describes how to generate the library.
- manager.cc: supplies the basic methods to be used to launch the Xuser manager.
- process.cc: methods dealing with mapping from C++ objects to C structures.
- display.cc: useful functions to display data contents.
- init.cc: whole stuff for HP-OV initialization.
- action_req.cc: methods dealing with the XOM/XMP requests to the Xuser agent.
- action_cnf.cc: methods dealing with retrieving the results from the Xuser agent after a request.
- end.cc: function to terminate the use of HP-OV.

- error.cc: all basic functions processing errors.
- mem.cc: all the memory management functions to be supplied inside the Xuser manager.
- handler_mgr.cc: methods to process asynchronous events.
- receive.cc: functions to process asynchronous events from the Xuser agent

Installation instructions:

1. Xuser agent (supplied by GMD): everything has to be compiled. To be sure of that, only a "./make" in the src/xuser directory is needed (after configuration of course !) ...
2. Xuser manager library: everything has to be compiled. To be sure of that, only a "make" command in the manager directory is needed (after configuration of course !) ...
3. Usually, this is not a good idea to just recompile something in this directory. The build process should be started from the $TRUMPET_TOP directory, in order to have all changes propagated everywhere !
4. Follow the instructions 1,2,3,4,5,6 and 7 of the corbaGateway directory to compile and generate the library: libmgr in both 2 different formats (static and shared).

3.5.2.4 Directory $TRUMPET_TOP/src/corbaGateway/adapter

This directory creates the main executable , used as a gateway between CORBA objects and HP-OV C structures. It will be created thanks to

- Xuser agent (supplied by GMD): everything has to be compiled. To be sure of that, only a "./make" in the src/xuser directory is needed (after configuration of course !) ...
- IDL library: the server library has to be compiled. To be sure of that, only a "make" command in the idl directory is needed (after configuration of course !) ...
- Xuser manager library: everything has to be compiled. To be sure of that, only a "make" command in the manager directory is needed (after configuration of course !) ...

Files:

- PnoConnectionMgrImpl.hh
- PnoConnectionMgrImpl.cc: implementation on the CORBA server side of the PnoConnectionManager class.
- gateway.cc: core source code to launch the gateway.
- objGateway.cc: gateway methods and main CORBA server code.
- clientTest.cc: CORBA client code for testing

Installation / configuration instructions:

Usually, this is not a good idea to just recompile something in this directory. The build process should be started from the $TRUMPET_TOP directory, in order to have all changes propagated everywhere !

3.5.3 Runtime

Different environment variables must be set before running the gateway. A shell script is available, setting and using default values. This runGateway.sh shell script is available in the current directory (corbaGateway). To run the CORBA/TMN gateway with default parameters, just type:./runGateway.sh

3.5.4 Version / release history

Version 1, November 1997.

3.5.5 Known bugs

None.

3.6 PNO Xuser-Agent

3.6.1 Hardware and software prerequisites

The current version has been tested on:

- HP-UX B10.10 and Solaris 2.x
- HP OpenView DM 4.21

To install Xuser Agent you may also need:

- GNU C, C++ compiler (v2.6+)
- GNU make utility (v3.7+).

3.6.2 Installation and configuration instructions

After uncompress the package and go into the directory where you see this README file, you should modify "XUSER_TOP" in the "./make" and may change definitions in the file "./v2.0/XuserDefs.mk" depending on your platform.

If everything is ready then under your shell type: “./make”, the makefile will detect your platform and finish everything necessary.

3.6.3 Runtime

If your installation (default) using explicit ACSE facility, before you run the Xuser Agent you should set "./etc/xuser.confi" correctly for your Agent AP-Title and P-Address etc.

If your installation using ACM supported by HP OpenView Platform, you should properly regist "./etc/misaXuAgent-2.0.lrf" in the ORS.

To run Xuser Agent, type: ./runAgent

NOTE: You should set "XUSER_TOP" properly in this shell script file and set your environment variables LD_LIBRARY_PATH including "./lib".

3.6.4 Version / release history

Version 2.0: September 29th 1997.

3.6.5 Known bugs

None.

4. SECURITY DEVELOPERS HANDBOOK

4.1 Security Profile Management

4.1.1 Engineering Object Model

The **SecPolicyInfo** interface as described in D9 Section 4.8 and supported by the **SecPolicy** object described in D9 Section 4.7.2 is realized by a C++ class definition with public interface:

```
class SecurityPolicy {
public:
  SecurityPolicy();
  ~SecurityPolicy();
  SecurityProfile* secProfileQuery(DistName&       initiatorTitle,
                                   ManagingRole    initiatorRole,
                                   DistName&       responderTitle,
                                   QoP);           //is not being used
};
```

Security profiles are read from file (for format of this file cf. Section 5.2) in the *SecurityPolicy* constructor.

4.1.2 Required and supported component interfaces

The *SecurityPolicy* class does not make use of any other TRUMPET component. It supports an interface for the SMASC.

Security rules for interacting with a remote peer are provided by calling the *secProfileQuery* method. The return type of *secProfileQuery* is *SecurityProfile* and the public part of this class definition is:

```
class SecurityProfile {
public:
  RWCString      aCDirectory();
  SecProfileType secProfile();
};
```

The *aCDirectory* method returns a value that is the name of a directory containing access rules to be used by the access control component (passed on to this module by the SMASC). Four different security profiles are recognized and may be returned by the *secProfile* method (although only two are actually supported by TRUMPET). The profile values are given in the definition of *SecProfileType*:

```
enum SecProfileType {
  NULLP,   //supported by TRUMPET
  MIN,     //not supported
  BASIC,   //supported
  ADV      //not supported
};
```


The parameters to *secProfileQuery* are of three different types; *ManagingRole*, *QoP* and *DistName*. *ManagingRole* can be either agent or manager and is defined as an enumeration type:

```
enum ManagingRole {
  AGENT_ROLE,
  MANAGER_ROLE
};
```

As *QoP* is not in use in TRUMPET, it is defined as an enumeration type with one possible value only:

```
enum QoP {
  NULLQ = 0
};
```

The class *DistName* is used by several components in the security package. The aspects of *DistName* relevant to the interface to an object of class *SecurityPolicy* can be described like this:

```
class DistName {
public:
  DistName();
  DistName(const char*);

  const char* dN() const;

  //equality operator overload
  int operator== (const DistName& dn) const;
  int operator!= (const DistName& dn) const;
};
```

The *secProfileQuery* method throws three exceptions that all have one public method each:

```
class UnknownInitiator          { public: char* msg(); };
class UnknownResponder          { public: char* msg(); };
class BadInitiatorResponderPair { public: char* msg(); };
```

It is up to the SMASC to dispose of the message returned with a *delete* operation on the returned value.

The types *SecurityProfile*, *SecProfileType*, *ManagingRole*, *QoP* and all three exceptions are defined along with *SecurityPolicy* in the *secpolicy.hh* header file. *DistName* is used more extensively in the security package and is defined elsewhere (but included by *secpolicy.hh*).

4.1.3 Example

The intended usage of the *SecurityPolicy* class from the SMASC can be illustrated like this:

```
#include <trumpet/secpolicy.hh>
```



```
SecurityPolicy   sPol;

DistName         initiatorTitle("cn=bm, OU=NR,  O=TRUMPET Project");
DistName         responderTitle("cn=bm, OU=UCL, O=TRUMPET Project");
ManagingRole     initiatorRole = MANAGER;
SecurityProfile* sP;

try {
  sP = sPol.secProfileQuery(initiatorTitle,
                            initiatorRole,
                            responderTitle,
                            NULLQ);
}
catch (UnknownInitiator ui) {
  msg = ui.msg(); cout << msg; delete msg;
}
catch (UnknownResponder ur) {
  msg = ur.msg(); cout << msg; delete msg;
}
catch (BadInitiatorResponderPair irp) {
  msg = irp.msg(); cout << msg; delete msg;
}

//some code to consider the attributes of *sP should
//be inserted here, before sP is being disposed of

delete sP;
```

4.2 Security Support Object

4.2.1 Engineering Object Model

SECUDE (formerly SecuDE - Security Development Environment) is a security toolkit which incorporates well known and established symmetric and public-key cryptography. It offers a library of security functions, security APIs and a number of utilities.

4.2.1.1 SECUDE APIs

SECUDE contains the following APIs:

- **AF - Authentication Framework and Certification**: this module adds X.509 certification functionality to SECUDE. Both local (i.e. PSE-located) certificates and directory-located certificates can be addressed. Therefore SECUDE offers an integrated X.500 Directory User Agent or alternatively an AF-Database, which is an emulation of an X.500 Directory running on a file system. Additionally ASN.1 encoding and decoding routines and a lot more auxiliary functions are available.
- **CRYPT - cryptographic algorithms**
- **GSS - Generic Security Services**
- **PKCS - Public Key Cryptography Standard**

- **PEM - Privacy Enhanced Mail Support**: this module converts functions, which realize the Internet Specifications RFC 1421 - 1424. The basic idea of PEM is to define document oriented message encipherment and authentication procedures for the protection of messages through the use of end-to-end cryptography between originator and recipient with no special processing requirements imposed on the message transfer system. This makes them transparent to the mail transfer systems and applicable for local security services, too.
- **SECURE - Personal Security Environment and Cryptography**: the SECURE API provides functions for the secure storage of data and basic cryptographic functions for the generation and verification of digital signatures, and encryption and decryption of data. All security relevant objects of a user are stored in a Personal Security Environment (PSE). The PSE typically contains the user's private key, user certificate, public root key. Two PSE realisations are available either a smartcard or a software PSE. No higher level functionality, like certificate processing, is provided by this API.
- **S/MIME - Secure MIME**

4.2.1.2 PSE Personal Security Environment

In order to achieve a system independant bit representation, all security objects which are stored on files or exchanged via communication protocols are defined as ASN.1 structures and encoded by applying the Basic Encoding Rules for ASN.1 (BER) with X.509 DER restrictions to these ASN.1 structures. This is simply called DER code in the following.

Within a G-program those objects have corresponding C-structure representations. The following table describes all objects of the PSE with their ASN.1 definitions, the corresponding C-structures and the Object Identifiers. The mapping between the program internal C-structures and the corresponding ASN.1 DER-code and vice versa is provided by the respective encoding and decoding routines:

PSE_object	**C-structure**	**Object Identifier**	**Content**
SignCert	Certificate	{ 1 3 36 2 1 1 }	certificate of public verification key (two-keypair PSE only)
EncCert	Certificate	{ 1 3 36 2 1 2 }	certificate of public encryption key (two-keypair PSE only)
SignCSet	SET_OF_Certificate	{ 1 3 36 2 2 2 }	cross-certificates of public encryption key (two-keypair PSE only)
Cset	SET_OF_Certificate	{ 1 3 36 2 2 3 }	cross-certificates of public key (one-keypair PSE only)
CrossSet	SET_OF_CertificatePair	{ 1 3 36 2 8 1 }	cross-certificate pairs from CA Directory entry
SignSK	KeyInfo	{ 1 3 36 2 3 1 }	secret signature key (two-keypair PSE only)
DecSKnew	KeyInfo	{ 1 3 36 2 3 2 }	secret decryption key (two-keypair PSE only)
Sknew	KeyInfo	{ 1 3 36 2 3 4 }	secret key (one-keypair PSE only)
FCPath	FCPath	{ 1 3 36 2 4 1 }	forward certification path
PKRoot	PKRoot	{ 1 3 36 2 5 1 }	top level public

			verification key
PKList	SET_OF_ToBeSigned	{ 1 3 36 2 6 1 }	list of trusted public verification keys
EKList	SET_OF_ToBeSigned	{ 1 3 36 2 6 2 }	list of trusted public encryption keys (two-keypair PSE only)
PCAList	SET_OF_ToBeSigned	{ 1 3 36 2 6 3 }	list of recognized PCAs
CrlSet	CrlSet	{ 1 3 36 2 9 1 }	list of revocation lists
SerialNo	OctetString	{ 1 3 36 2 10 1 }	current serial number (CAs_only)
EDBKey	KeyInfo	{ 1 3 36 2 11 1 }	DSA database encryption key
AliasLst	aliaslist	{ 1 3 36 2 12 1 }	user's alias list
QuipuPWD	char	{ 1 3 36 2 13 1 }	password for X.500 directory access
Dhparam	AlgId	{ 1 3 36 2 15 1 }	algorithm parameters for a Diffie-Hellman key agreement
DSAcomm	AlgId	{ 1 3 36 2 15 4 }	common algorithm parameters for the Digital Signature Algorithm (DSA)

4.2.2 Required and supported component interfaces

The trumpet interface offers 3 functions to the SMASC:

4.2.2.1 Function: server_acquire_creds

FUNCTION

server_acquire_creds

SYNOPSIS

```
int server_acquire_creds( char * service_name, gss_cred_id_t * server_creds)
```

DESCRIPTION

Imports service name and acquires credentials for it. The service name is imported with gss_import_name, and service credentials are acquired with gss_acquire_cred. If either opertion fails, an error message is displayed and -1 is returned; otherwise, 0 is returned.

ARGUMENTS

- service_name (r) the ASCII service name.
- server_creds (w) the GSS-API service credentials.

RESULTS

Returns: 0 on success, -1 on failure.

4.2.2.2 Function: client_establish_context

FUNCTION

client_establish_context

SYNOPSIS

```
int client_establish_context(     char * service_name,
                gss_ctx_id_t * gss_context
                gss_buffer_desc recv_tok,
                gss_buffer_desc send_tok)
```

DESCRIPTION

establishes a GSS-API context with a specified service and returns the context handle. service_name is imported as a GSS-API name and a GSS-API context is established with the corresponding service;The default GSS-API mechanism is used, and mutual authentication and replay detection are requested. If successful, the context handle is returned in context. If unsuccessful, the GSS-API error messages are displayed on stderr and -1 is returned. returns 1 on GSS_S_CONTINUE_NEDDED.

ARGUMENTS

- service_name (r) the ASCII service name of the service.
- context (w) the established GSS-API context.
- recv_tok (r) the received token. Has to be set to = GSS_C_NO_BUFFER for the first call.
- send_tok (w) the token to send if 1 is returned.

RESULTS

Returns: 0 on success, 1 on GSS_S_CONTINUE_NEDDED, -1 on failure.

4.2.2.3 Function: server_establish_context

FUNCTION

server_establish_context

SYNOPSIS

```
int server_establish_context(     gss_cred_id_t server_creds,
                gss_ctx_id_t *context,
                gss_buffer_t client_name,
                gss_buffer_desc recv_tok,
                gss_buffer_desc send_tok)
```

DESCRIPTION

establishses a GSS-API context as a specified service with an incoming client, and returns the context handle and associated client name. Any valid client request is accepted. If a context is established, its handle is returned in context and the client name is returned in client_name and 0 is returned. If unsuccessful, an error message is displayed and -1 is returned. 1 is returned on GSS_S_CONTINUE_NEDDED.

ARGUMENTS

- service_creds (r) server credentials, from gss_acquire_cred
- context (w) the established GSS-API context
- client_name (w) the client's ASCII name
- recv_tok (r) the received token
- send_tok (w) the token to send if 1 is returned

RESULTS

Returns: 0 on success, 1 on GSS_S_CONTINUE_NEDDED, -1 on failure.

4.3 Access Control

4.3.1 Engineering Object Model

This is the object model for the access control component implementation:

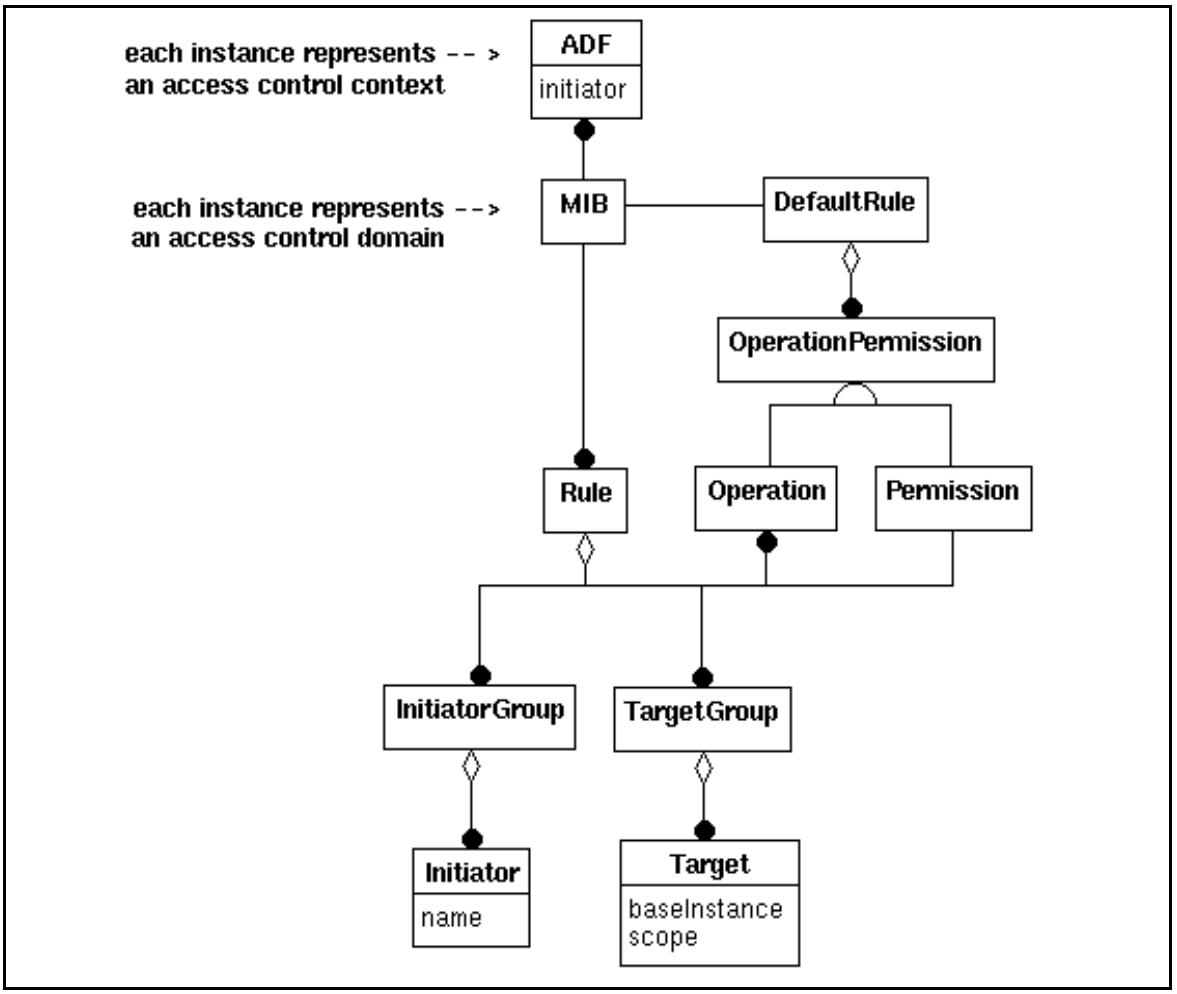


Figure 10: Engineering Object Model of the Access Contro Servicel

4.3.2 Required and supported component interfaces

The access control component interfaces with the SMASC. It does not depend on other TRUMPET system components.

The class ***ADF*** provides the external interface of the access control component:

```
class ADF {
public:
     ADF (const DistName& initiator, const RWCString& domain);
     Permission associationPermission (const DistName& target);
     Permission operationPermission (
          Operation operation,
          RWTValSlist<ac_target>& targets,
          RWTValSlist<ac_target>& denied);
};
```


METHOD

ADF: generates a new access decision context

SYNOPSIS

```
#include <trumpet/securityPackage/acControl/adf.hh>
ADF (const DistName& initiator, const RWCString& domain);
```

DESCRIPTION

This function is used to instantiate a new access decision context. It contains the name of the associated initiator and the set of access control rules to be used.

ARGUMENTS

- *initiator*: the application entity title of the requesting MAE (distinguished name form).
- *domain*: identifies the access control domain.

RESULTS

This function may return: **Allow, DenyWithResponse, DenyWithoutResponse, AbortAssociation**

METHOD

associationPermission: determine access control permission for association between given MAEs

SYNOPSIS

```
#include <trumpet/securityPackage/acControl/adf.hh>

AccessDecision associationPermission (const DistName& target)
```

DESCRIPTION

This function determines the access permission of the initiator MAE to establish an association to the target MAE.

ARGUMENTS

- *target*: the application entity title of the peer MAE to connect to. This must be a distinguished name.

RESULTS

This function may return: **Allow, DenyWithResponse, DenyWithoutResponse, AbortAssociation**

METHOD

operationPermission: determine access permissions to perform operation on given management information objects.

SYNOPSIS

```
#include <trumpet/securityPackage/acControl/adf.hh>

AccessDecision operationPermission (
          OM_sint primitive,
          List<ac_target>& targets,
          List<ac_target>& denied);
```

DESCRIPTION

This function is used to determine the access permissions of the initiator to perform the given operation on the set of managed objects.

ARGUMENTS

- *primitive* Identifies which type of operation has been requested. Possible values are: MP_GET_IND, MP_SET_IND, MP_ACTION_IND, MP_CREATE_IND, MP_DELETE_IND
- *targets* Identifies the set of managed objects selected for the operation. The type of list elements is *ac_target*. which is defined as:

```
struct ac_target {
OM_object object;
AccessDecision permission;
void* link};
```

  - *object* identifies a MO instance. It is an instance of OM class Base-Managed-Object-Id.
  - *permission* returns the permission to perform the operation determined by the access decision function.
  - *link* can be used by the caller to establish a link between the structure and the MO instance.

RESULTS

- *targets*: subset of objects access is granted. *permission* is set to Allow.
- *denied*: set of object access is denied. *permission* is set to one of the following values: **DenyWithResponse, DenyWithoutResponse, AbortAssociation**.

This function may return: **Allow, DenyWithResponse, DenyWithoutResponse, AbortAssociation**

4.4 Secure Management Association

4.4.1 Engineering Object Model

The Secure Management Association Support Component (SMASC) is that component of the management system which provides the management applications with the means to secure the management association with other management applications located in another management system. The main purpose of the *SMASC* is to isolate the security-related components from the application code. This approach has the following advantages:

- the security can easily be added to / removed from an existing application, without affecting its internal structure,
- the security-related code can be designed, programmed and verified independently by security-aware personnel,
- the addition of auditing capacities for security-related events is made easier and safer,
- the resulting code can easily be customised to accommodate new security policies.

On behalf of a management entity, the **SMASC** authenticates and control the access to peer management applications; it also initialises the security context for further security services to be used on the association, in particular it establishes a session secret key if the requested Quality of Protection requires integrity and / or confidentiality of communicated data.

Figure 11 shows the main components of the security architecture, the internal structure of the *SMASC* and the contract interfaces of the *SMASC* to other components. The security services of the *SMASC* can be accessed through *Adapter* Components. The purpose of the *Adapter* Components is to transform technology specific syntax (e.g. XOM objects [XOM]) to generic data structures (e.g. BER encoding). With this approach, platform specific code can be restricted to the Adapter Component and the SMASC can be reused without major modifications for other management platforms. The SMASC is also interfaced with:

- the *AccessControl* component to control the access to the management association,
- the Security Event Logging and Forwarding (SELF) component to keep track of all relevant security events occurring on the management associations.

The *SMASC* is decomposed into object classes as follows:

- a Secure Management Association Support (*SMAS*) object which co-ordinates the behaviour of the whole component,
- a *SSO* object, which provides generic security services such as peer authentication, integrity, encryption and digital signatures; the *SMAS* accesses the *SSO* through the Generic Security Service API (GSS-API). The *SSO* may be implemented using already existing commercial products which provide a standard [RFC 1508] interface.
- a *SecPolicy* object which contains the policy rules for inter-domain security to interact with a remote MAE,
- a *CertificateHandling* object, which performs key handling such as caching, fetching certificates from directories and checking certificates revocation lists. The *CertificateHandling* object may interact with extemal services for certificate and CRL distribution, for example offered in conjunction with a CA TTP. Use of the LDAP protocol (RFC 1777, 1995) is envisaged for fetching of certificates and CRLs.

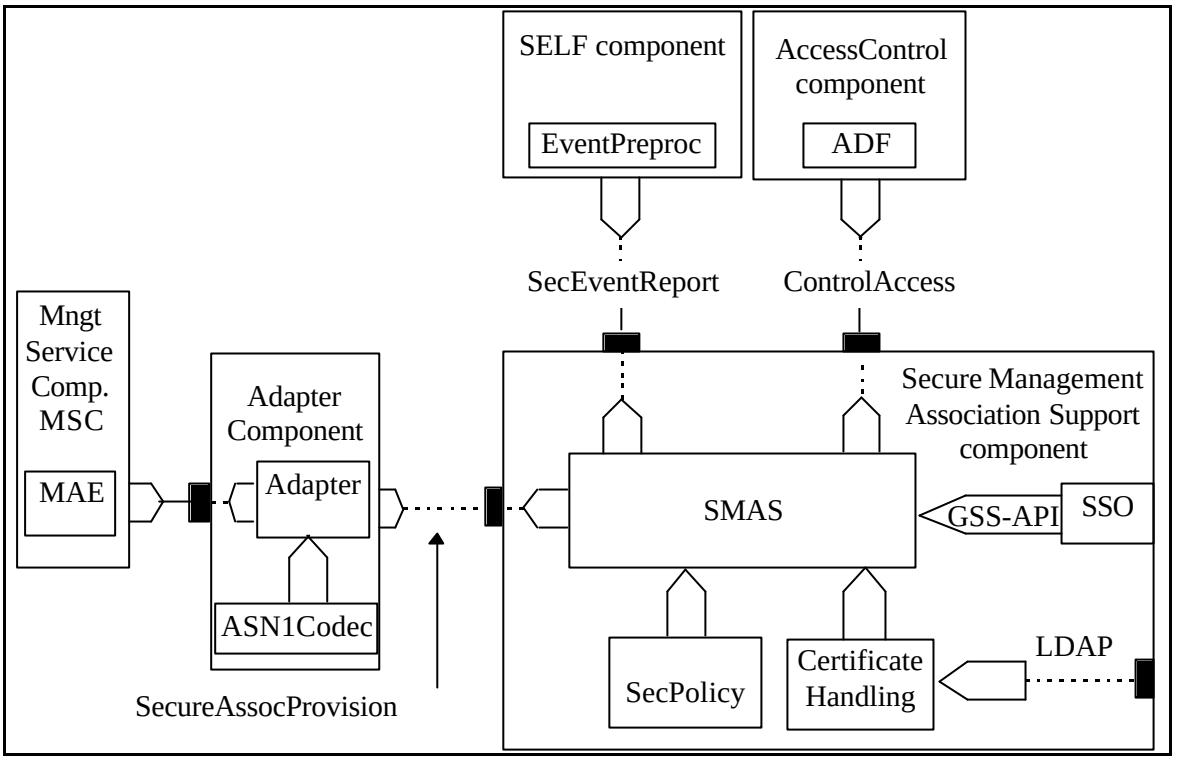


Figure 11: Graphical Representation of the Secure Management Association Component

Note that in order to establish a secure management association, a management entity must have disabled any type of automatic connection mechanism and take explicit control over the association.

4.4.2 Required and supported component interfaces

NAME

smasc_assoc_req - Initiates the authentication process with the peer entity

SYNOPSIS

```
#include <smasc.h>

int smasc_assoc_req(
     char *target,
     smasc_token *token);
```

DESCRIPTION

This function is used by the Adapter(s) to initiate the authentication process by computing the first token to be sent to the target MAE.

Parameters

target	The target MAE's LDAP Distinguished Name.
token	Returned upon successful completion of the function call. The authentication token to be sent to the target MAE.

RESULTS

This function may return:

0	**Function successfully completed**

NAME

smasc_assoc_rec - Forwards the authentication token in the incoming PDU to the SSO

SYNOPSIS

```
#include <smasc.h>

int smasc_assoc_rec(
     char *target,
     smasc_token token);
```

DESCRIPTION

This function is used by the Adapter(s) to forward the input authentication token received in an association request/response to the SSO.

Parameters

target	The target MAE's LDAP Distinguished Name.
token	The authentication token to be forwarded to the SSO.

RESULTS

This function may return:

0	**Function successfully completed**

NAME

smasc_assoc_rsp - Calculates the next token in the authentication process

SYNOPSIS

```
#include <smasc.h>

int smasc_assoc_rsp(
     char *target,
     smasc_token *token);
```

DESCRIPTION

This function is used by the Adapter(s) to initiate the get the next token needed in the authentication process with the target MAE.

Parameters

target	The target MAE's LDAP Distinguished Name.
token	Returned upon successful completion of the function call. The next authentication token to be sent to the target MAE.

RESULTS

This function may return:

-1	**The authentication process is unsuccessfull**
0	**The authentication process is completed and successfull**
1	**Another token is needed in the authentication process**

4.5 Adapter Object

4.5.1 Engineering Object Model

The X/Open Management Protocol (XMP) API provides a common access mecanism to both CMIP and SNMP services. XMP employs the X/Open OSI-Abstract-Data Manipulation (XOM) API to manipulate variables and parameters.

XMP consists of a library of C functions that reflect the services of CMIS and SNMP. It embodies the object-oriented OSI model of management. Manager applications use XMP to access managed objects, and agents applications use XMP to respond. XMP can be used synchronously or asynchronously.

The functions in XOM are used to create, examine, modify and destroy the arguments to XMP functions. It provides a generalized data handling mechanism, and manipulates data types that arise from Abstract Syntax Notation 1 (ASN.1) definitions.

As reflected in Figure 12, all network management applications can use XMP, with XOM, for standards-based process-to-process communications.

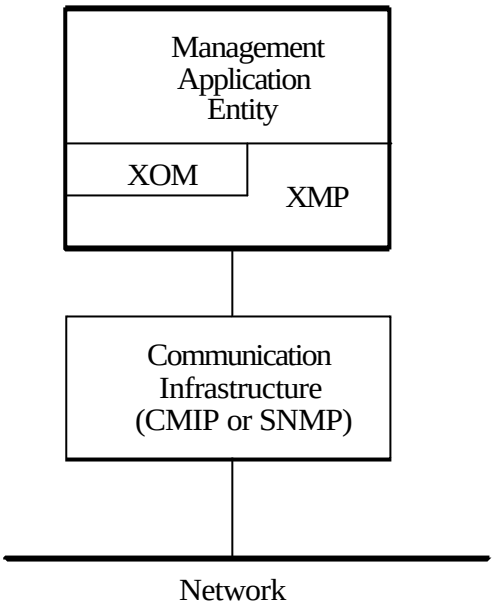


Figure 12: XOM, XMP and MAE

The XMP Adapter is responsible for securing the XMP function calls, which in turn map to CMIP requests and responses.

The approach chosen in Trumpet is to modify the MAE's source code to include security-related calls to the adapter where appropriate. The MAE is then made "security-aware".

As the adapter must protect all PDU exchanged between manager and agents applications, implementers have to insert security-related code before XMP "send request" function calls to protect the outgoing PDU ; and after XMP "get response" function calls to deprotect the incoming PDU.

The adapter interface is designed to simplify this process, as all the functions defined closely match that of XMP, having the same name and accepting similar arguments.

4.5.1.1 Secured Association Establishment

When using XMP, you can either rely on TMN Platform Infrastructure to control associations, providing automatic establishment and release, or you can retain application control over these functions. The Association Control Service Element (ACSE) extensions to the XMP API give you this capability.

Explicit association control is very useful in certain situations:

- an application may need to set certain ACSE parameters in order to interoperate with peer applications based on other TMN Platforms,
- with explicit association control, an application can establish multiple associations with a peer, each association having its own context. This can be useful if several kinds of transactions are occurring between pair of applications,
- a management association established using the XMP ACSE extensions is never terminated unless one side explicitly terminates it, or unless connectivity between the two sides is lost. Therefore, a management application can set up an association and use it as a test of connectivity with the remote peer: connectivity can be tested at any time by sending a message on that association.

Management associations established with Automatic Connection Management (ACM) are **shared associations**, that can be used by any Management Application Entity residing on the same host. By contrast, a management association established by two applications, both using the ASCE extensions, is a **private association** between those applications, and can only be used by them.

When using the Trumpet security package, secured associations estalibshed between applications must be private: each secured association rely on a distinct security context that must be negotiated during the association establishment, and terminated when releasing the association. Therefore, the ACSE extensions must be used by client Management Application Entities when they want to establish secured associations.

The XMP ACSE extensions add five functions to the XMP API. They are listed and descibed in Table 4. For more details, see the manpage for each function.

ASCE Function	Desciption
`mp_assoc_req()`	Called after mp_bind() to request the establishment of a connected session.
`mp_assoc_rsp()`	Used to reply to a previously invoked association request.
`mp_release_req()`	Used to request the release of a connected session.
`mp_release_rsp()`	Used to reply to a previously invoked release request.
`mp_abort_req()`	Used to abort a management association. This service is defined as non-confirmed.

Table 1: ACSE Functions

The XMP Adapter provides a set of functions that are responsible for negociating the security context. Calls to these functions must be inserted in the MAE's source code before any ACSE function is called. They act by inserting and/or transforming proper parameters in the XOM structures that are later used by the ACSE functions.

Table 5 lists and describes the ACSE-related functions that are provided by the XMP Adapter.

XMP Adapter Function	Desciption
`sp_assoc_req()`	Called before mp_assoc_req() to supply the necessary security parameters for the establishment of a connected session.
`sp_assoc_rsp()`	Called before mp_assoc_rsp() to supply the necessary security parameters for the reply to a previously invoked association request.
`sp_release_req()`	Called before mp_release_req() to supply the necessary security parameters to request the release of a connected session.
`sp_release_rsp()`	Called before mp_release_rsp() to supply the necessary security parameters to release a connected session.
`sp_abort_req()`	Called before mp_release_rsp() to supply the necessary security parameters to abort a connected session.

Table 2: ACSE-related Adapter functions

Figure 13 outlines the steps needed in order to establish and release a secured association. For the sake of simplicity, calls to XOM functions are not presented, but the main parameters needed to XMP functions are indicated where appropriate.

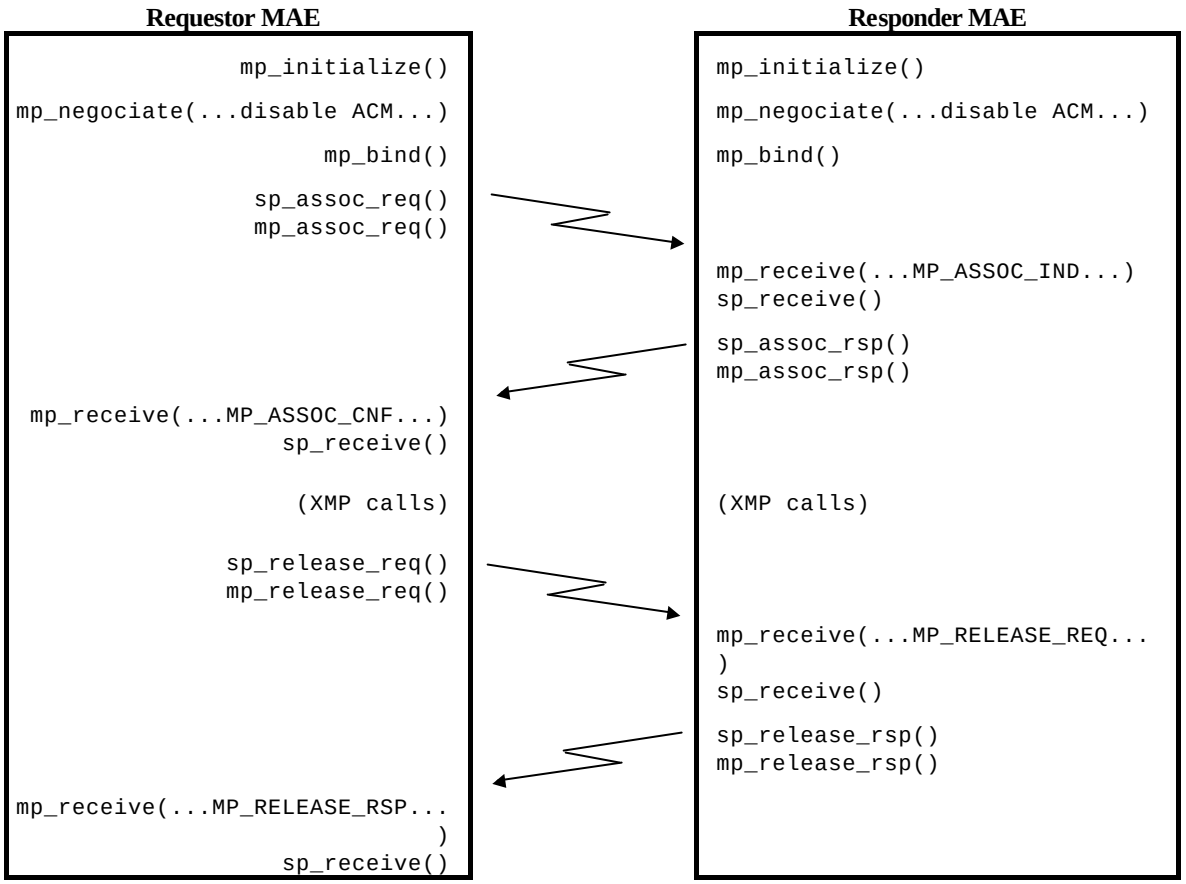


Figure 13: Secured association establishment & release

The sequence of operations is exactly the same as for a non-secured association ; calls to the Trumpet security package are just are inserted before all XMP functions that send data and after all XMP functions that receive data (the `mp_receive()` and `sp_receive()` functions are described later).

4.5.1.2 Secured XMP Requests

As previously mentionned, XMP supports the seven CMIS services through the CMIS OM package. The CMIS services are mapped to XMP function names, as shown in Table 6.

CMIS service	XMP functions	Description (of request only)
ACTION	`mp_action_req()` `mp_action_rsp()`	Requests that the responder perform one of the actions defined for an object.
CANCEL-GET	`mp_cancel_get_req()` `mp_cancel_get_rsp()`	Requests that the responder terminate servicing an earlier "get" request that has not yet completed.
CREATE	`mp_create_req()` `mp_create_rsp()`	Requests that the responder create an instance (object) of the specified object class.
DELETE	`mp_delete_req()` `mp_delete_rsp()`	Requests that the responder destroy a particular instance (object) of an object class.
EVENT-REPORT	`mp_event_report_req()` `mp_event_report_rsp()`	Issues one of the notifications (events) defined for a managed object.
GET	`mp_get_req()` `mp_get_rsp()`	Requests that the responder supply the value(s) of one or more object attributes.
SET	`mp_set_req()` `mp_set_rsp()`	Requests that the responder modify the value(s) of one or more object attributes.

Table 3: XMP functions to support CMIS services

As mentioned earlier, security-related calls to the adapter have to be made before XMP "send request" function calls and after XMP "get response" function call.

These calls are described in Table 7 below, they basically share the same input parameters as the corresponding XMP function calls. Their role is either to protect or deprotect an XOM object, provided as an input argument, and providing the result in an output OM object.

Protection of the OM input arguments, depending on the security context and policy requirements, can consist of confidentiality and/or integrity.

The returned protected XOM objects must not be modified or tampered in any way after they are produced ; they can only be send to the peer entity through the corresponding XMP call.

CMIS service	XMP functions	Corresponding Adapter functions
ACTION	`mp_action_req()` `mp_action_rsp()`	`sp_action_req()` `sp_action_rsp()`
CANCEL-GET	`mp_cancel_get_req()` `mp_cancel_get_rsp()`	`sp_cancel_get_req()` `sp_cancel_get_rsp()`
CREATE	`mp_create_req()` `mp_create_rsp()`	`sp_create_req()` `sp_create_rsp()`
DELETE	`mp_delete_req()` `mp_delete_rsp()`	`sp_delete_req()` `sp_delete_rsp()`
EVENT-REPORT	`mp_event_report_req()` `mp_event_report_rsp()`	`sp_event_report_req()` `sp_event_report_rsp()`
GET	`mp_get_req()` `mp_get_rsp()`	`sp_get_req()` `sp_get_rsp()`
SET	`mp_set_req()` `mp_set_rsp()`	`sp_set_req()` `sp_set_rsp()`

Table 4: CMIS-related Adapter functions

4.5.1.3 Secured Asynchronous Operations

When a synchronous function call is performed, the function does not return unless the effect of the call is complete. In opposition, asynchronous function calls do start some process and return. They are used by applications that need to do multiple independent function calls, an example being a network management application that interrogates multiple distinct network equipment.

The XMP API allows you to make any call (except `mp_cancel_get_req()`) synchronously, and to use any requester function asynchronously.

When you make synchronous requester calls, the parameters returned by the responder application are available through the `result_return` OM object, provided the request was successfully processed.

When you make an asynchronous function call, the XMP interface first determines if the call is valid. If so, the transaction with the responder is initiated. Your application is then allowed to continue processing while the request is serviced. If a response is expected, you must later call `mp_receive()` to determine the outcome of the request.

When such asynchronous function calls are secured with the Trumpet security package, the requester application must call the `sp_receive()` function in order to exploit the information provided by the `mp_receive()` function.

On the responder side, the application has no knowledge whether the call is performed synchronously or asynchronously, therefore the sequence of operations is the same as described in the previous chapter.

4.5.2 Required and supported component interfaces

NAME

sp_abort_req - Protects the parameters to an ACSE association abort request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_assoc_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to an ACSE association abort request just before the corresponding XMP function call is made.

Parameters

session	The OM object against which the operation will be performed. It must be a private OM object previously returned as part of an **Assoc-Argument** or **Assoc-Result** object. This object must belong to an ACM-disabled workspace.
context	Represents the management context to be used for this operation. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	The unprotected information supplied as the argument of an Abort operation. It is an instance of a subclass of the OM class **Abort-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_abort_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

MP_ACCESS_CONTROL_FAILURE

NAME

sp_action_req - Protects the parameters to a CMIS Action request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_action_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Action request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation/notification will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about the Action request and the data for that action. It is an instance of a subclass of the OM class **Action-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_action_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_action_rsp - Protects the parameters to a CMIS Action reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_action_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Action reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation/notification was requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the Action request. It is an instance of one of the following OM classes: **Action-Result**, **Linked-Reply-Argument**, **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_action_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_assoc_req - Protects the parameters to an ACSE association establishment request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_assoc_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to an ACSE association establishment request just before the corresponding XMP function call is made.

Parameters

session	The OM object against which the operation will be performed. It must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	Represents the management context to be used for this operation. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	The unprotected information that represents the argument of an Associate operation. It is an instance of a subclass of the OM class **Assoc-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_assoc_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

MP_ACCESS_CONTROL_FAILURE

NAME

sp_assoc_rsp - Protects the parameters to an ACSE association establishment reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_assoc_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to an ACSE association establishment reply just before the corresponding XMP function call is made.

Parameters

session	The OM object against which the operation will be performed. It must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	Represents the management context to be used for this operation. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	The unprotected information supplied as a response of an Associate operation. It is an instance of a subclass of the OM class **Assoc-Result**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_assoc_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_ACCESS_CONTROL_FAILURE

NAME

sp_cancel_get_req - Protects the parameters to a CMIS Cancel-Get request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_cancel_get_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Cancel-Get request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about which Get operation is to be cancelled. It is an instance of a subclass of the OM class **Cancel-Get-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_cancel_get_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_cancel_get_rsp - Protects the parameters to a CMIS Cancel-Get reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_cancel_get_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Cancel-Get reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the result of the Cancel-Get operation. It is an instance of one of the following OM classes: **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_cancel_get_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_create_req - Protects the parameters to a CMIS Create request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_create_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Create request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about the managed object to create and any data values attributes of the managed object. It is an instance of a subclass of the OM class **Create-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_action_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_create_rsp - Protects the parameters to a CMIS Create reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_create_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Create reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the Create request. It is an instance of one of the following OM classes: **Create-Result**, **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_create_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_delete_req - Protects the parameters to a CMIS Delete request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_delete_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Delete request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about the managed object to delete. It is an instance of a subclass of the OM class **Delete-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_delete_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_delete_rsp - Protects the parameters to a CMIS Delete reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_delete_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Delete reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the Delete operation. It is an instance of one of the following OM classes: **Delete-Result**, **Linked-Reply-Argument**, **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_delete_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_event_report_req - Protects the parameters to a CMIS Event-Report request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_event_report_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Event-Report request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about the event to be generated. It is an instance of a subclass of the OM class **Event-Report-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_event_report_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_event_report_rsp - Protects the parameters to a CMIS Event-Report reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_event_report_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a CMIS Event-Report reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the Event-Report. It is an instance of one of the following OM classes: **Event-Report-Result**, **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_event_report_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_get_req - Protects the parameters to a Get request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_get_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a Get request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about which attributes are to be retrieved. It is an instance of a subclass of the OM class **Get-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_get_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_get_rsp - Protects the parameters to a Get reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_action_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a Get reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the Get operation. It is an instance of one of the following OM classes: **Get-Result**, **Linked-Reply-Argument**, **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_get_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_receive - Unprotects the result or notification to an asynchronous operation.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_receive(
     OM_private_object session,
     OM_private_object context,
[other parameters might be needed]
     OM_object protected,
     OM_object *result);
```

DESCRIPTION

This function is used to unprotect the partial or complete result of an invoked management operation, or its reported management notification.

Parameters

session	An OM object that identifies the management session in which the operation or notification was performed. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation was performed. This must be a private OM object ; the constant **Default-Context** { MP_DEFAULT_CONTEXT } is explicitly not permitted.
protected	An protected OM object obtained through a preceeding `mp_receive()`function call. The abstract class of this object is dependent on the value of the *primitive* and *completion_flag* parameters. It might be an instance of one of the following OM classes: **Action-Argument, Action-Result, Cancel-Get-Argument, Absent-Object, Create-Argument, Create-Result, Delete-Argument, Delete-Result, Event-Report-Argument, Event-Report-Result, Get-Argument, Get-Result, Set-Argument, Set-Result, Assoc-Argument, Assoc-Result, Release-Argument, Release-Result** or **Abort-Argument**.
protected	Returned upon successful completion of the function call. This object is of the same OM class as the *protected* argument.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_AUDIT_ALARM_FAILURE

NAME

sp_release_req - Protects the parameters to an ACSE association release request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_release_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to an ACSE association release request just before the corresponding XMP function call is made.

Parameters

session	The OM object against which the operation will be performed. It must be a private OM object previously returned as part of an **Assoc-Argument** or **Assoc-Result** object. This object must belong to an ACM-disabled workspace.
context	Represents the management context to be used for this operation. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	The unprotected information that represents the argument of a Release operation. It is an instance of a subclass of the OM class **Release-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_release_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_release_rsp - Protects the parameters to an ACSE association release reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_assoc_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to an ACSE association release reply just before the corresponding XMP function call is made.

Parameters

session	The OM object against which the operation will be performed. It must be a private OM object previously returned as part of an **Assoc_Argument** or **Assoc-Result** object. This object must belong to an ACM-disabled workspace.
context	Represents the management context to be used for this operation. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	The unprotected information supplied as a response to a Release operation. It is an instance of a subclass of the OM class **Release-Result**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_release_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_set_req - Protects the parameters to a Set request.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_get_req(
     OM_private_object session,
     OM_private_object context,
     OM_object argument,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a Set request just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
argument	An unprotected OM object which provides the information about which attributes are to be modified. It is an instance of a subclass of the OM class **Set-Argument**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to `mp_set_req()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

NAME

sp_set_rsp - Protects the parameters to a Set reply.

SYNOPSIS

```
#include <xom.h>
#include <tsp.h>

SP_status sp_action_rsp(
     OM_private_object session,
     OM_private_object context,
     OM_object response,
     OM_object *protected);
```

DESCRIPTION

This function is used to protect the parameters to a Set reply just before the corresponding XMP function call is made.

Parameters

session	An OM object that identifies the management session in which the operation will be requested. This must be a private OM object previously returned from `mp_bind()`. This object must belong to an ACM-disabled workspace.
context	The context in which the operation should be performed. This must be a private OM object or the constant **Default-Context** { MP_DEFAULT_CONTEXT }.
response	An unprotected OM object supplied as a response information about the Set operation. It is an instance of one of the following OM classes: **Set-Result**, **Linked-Reply-Argument**, **Absent-Object** or **Service-Error**.
protected	Returned upon successful completion of the function call. The protected information that is to be supplied to the `mp_set_rsp()`.

RESULTS

This function may return:

MP_SUCCESS

MP_NO_WORKSPACE

MP_INVALID_SESSION

MP_INSUFFICIENT_RESSOURCES

SP_NO_SECURITY_CONTEXT

SP_ACCESS_CONTROL_FAILURE

SP_AUDIT_ALARM_FAILURE

4.6 Audit and Alarm

4.6.1 Engineering Object Model

4.6.1.1 Implementation of the Security Audit Manager

4.6.1.1.1 Introduction

The implementation of the audit management is described in this section with emphasis on the Graphical User Interface, the Event Forwarding Discriminator management and the security related event collecting and displaying. In the Graphical User Interface, a collection of user friendly mask windows are defined and realised to help the user to access easily to the event forwarding discriminator construction and the alarm reporting function. The security related event management is implemented using the XMP API.

The implementation work is composed of:

- The basic audit management function library
- The top level GUI
- The GUI for alarm viewing
- The GUI for EFD management
- The GUI for log management
- The graphical editor for DiscriminatorConstruct

4.6.1.1.2 Implementation Architecture

The audit management is performed by the security alarm management application and the audit trail management application.

These management applications collect security-related events from a set of known agents according to the conditions defined in the Event Forwarding Discriminators. The alarm manager allows these events to be displayed in human readable format, as the audit trail manager saves them into a log for further analysis.

The audit management is implemented in one process.

The GUI provides:

- an agent viewer to show the location and the state of the agents
- an agent controller to manage the EFD in that agent (EFD creation, deletion, etc.)
- an alarm viewer to display collected security alarms
- an event log browser

The Security Alarm Manager implements the following management functions:

- event forwarding discriminator management
- security alarm collection (the alarm detection is performed by the agent side)
- alarm reporting

The Audit Trail Manager logs not only security alarms (`SecurityServiceAndMechanism-Violation`: a subset of security-related events) but also `ServiceReport`. These events are very valuable resources for further security audit analysis.

4.6.1.1.3 GUI

This section presents the design of the GUI for managing audit functions, including Event Forwarding Discriminators and security-related event collecting and displaying. The corresponding sponsor component, accepting messages from the audit management application and mapping them to real resources mechanisms is not discussed here.

In the GUI a collection of user-friendly mask windows are defined and realised to help the user access easily the event forwarding discriminator construction and the alarm reporting function. The security-related event management is implemented using the XMP API.

The main goal of the GUI is to allow the user not necessarily familiar with the security management concept and the OSI model to be able to access the management functions. This is the reason for which the GUI should be user-friendly not only as to how the information is displayed, but also in the manner in which the user participates in the management process.

Top-Level Window

This window allows to start the TRUMPET management session. The administrator can select the security services to select in the *Administration* menu: currently only the audit management is implemented; potentially, management of the security policy, management of the authentication and access control services and key management could be added. The *Security Status* menu displays the number of security alarms for each severity level. Two buttons allow to display more information about the alarms.

Agent Configuration Window

This window allows to see each agent configuration from which security alarms will be collected. In its menu bar, the administrator can select an audit manager configuration containing some agent bitmaps which he can move on the trials bit map and perform actions on them., see agent specifications ...

Agent-EFD Control Window

This window gives some useful information on the selected agent and controls the EFD on it. The agent control panel shows the name and the current co-ordinates of the agent. The user can give the current operational state of the agent (present, running, locked, etc.). Using the EFD panel, the user can create a new EFD instance in the selected agent. A scroll list shows the EFDs currently associated with the agent; by choosing one of the EFD identifier from the list, the user may delete, edit or get help from it. When the user clicks on the Modify or Create button, the EFD construction mask window is displayed.

EFD Construction Mask Window

This window contains all the attributes of the EFD class. The security administrator can specify the appropriate value to build a new EFD or edit an existing EFD. By clicking on the corresponding button, the administrator is displayed a specific window to fill all the EFD attributes fields: Discriminator Construct, Start Time, Stop Time, Intervals of Days, Week Mask, Destination, Backup Destination List, Active Destination, Administrative State, Operational State, Availability Status, Confirmed Mode.

Filters Editor Window

This window allows to construct a discriminator graphically. On the drawing area, the administrator has to click on a rectangle bitmap and to control it by using the mouse menu button.

Control Window for Managing Logs

This windows is used for log management. The existing logs are displayed as icons with log names below. The security auditor can select a log with the mouse to display the contents of the log or delete the log. He can create a new log (Log Creation Window). By double-clicking on the log icon, the administrator can display the log attributes.

Log Creation Window

This window allows the creation of a new log. A default log can be created or the administrator can create a customised log by specifying – using the same windows as for the EFDs – the discriminator construct, the start time, the stop time, the intervals of days, the week mask. He can modify the operational and administrative state of the log.

Log Attributes Display Window

The administrator can display and modify the attributes of the selected log.

Alarm Viewer Window

This window shows the collected security alarms. Only 10 alarms can be displayed simultaneously. The alarm buffer size is limited. The older alarms will be discarded when the buffer is full.

4.6.1.1.4 Basic audit management function library

This library provides some basic functions to:

- initialise the XMP library
- create OpenView kernel EFD
- create, delete, enable and disable EFD in the specified agent
- wait an event (with time out)

This library is written upon the XMP interface.

4.6.1.1.5 Interprocess Communication Interface Library

The Interprocess Communication Interface Library defines a specific protocol and a set of service functions to allow one or more management application processes to communicate with the GUI process. The protocol is based on message passing mechanisms.

4.6.1.1.5.1 Principle

Management applications can not be implemented using event driven scheme. They have to perform a loop to wait for the CMIP messages and GUI commands. The X11 main loop does the same thing. It is possible to add into X11 main loop the CMIP messages .

The GUI implemented in X11 and Motif incorporates the messages from management processes (`XtAppAddInput`). The XMP based applications will include the GUI command handler into their main loop.

The GUI is responsible to load and shutdown a XMP process. The GUI can also control the XMP process execution, for example pause and resume the alarm collection.

4.6.2 Required and supported component interfaces

- The manager audit and alarms communicates with the agent ovead of the HP OpenView 4.21 plateform through XMP.The GDMO model used for XOM objects is the GDMO X721 (ems.mib) of the plateform.The manager manages object of class HPEventForwardingDiscriminator.
- The manager exchanges with SELF agent with XMP: Event Report which field Event Info is of class SecurityAlarmInfo (in ems.mib) or ServiceReport (GDMO X740).

Contents of the Event Report

Name of the attribute (GDMO)	**Contents**
Managed Object Class	Service which raise the event (integer converted in oid)
Managed Object Instance (in the attribute value of the first ava)	Instance of the service (integer)
Event Time	actual time (generalized time)
Event Type	oid of the notification
Event Info	Security Alarm Info or Security Audit Info

Contents of the Security Alarm Info

Name of the attribute (GDMO)	Contents
Security Alarm Cause	oid of the security alarm cause
Security Alarm Severity	integer
Security Alarm Detector	empty
Service User (attribute Detail)	oid of the entity which caused the raising of the alarm
Service Provider (attribute Detail)	oid of the entity which has been «attacked »

Contents of the Security Audit Info (Service Report)

Name of the attribute (GDMO)	Contents
Service Report Cause	oid of the service report cause
Additional Text	oid of the entity which caused the raising of the report
Additional Information	oid of the entity which has been «attacked »

4.7 SELF

4.7.1 Engineering Object Model

There is a number of XOM objects used to implement the audit agent. These are as follows:

- feature_list (MP_feature): negotiate the features of the platform environment where the agent runs.
- attributeId , ava , ds_rdn , ds_dn (OM_descriptor): they are used to identify an instance of a security service.
- bmoi: (OM_descriptor): is used to identify a managed object class which is a class of a service. An integer is used converted to an OID.
- bmoc: (OM_descriptor): is used to identify an instance of a managed object class i.e. an instance of a service using an integer.
- eType: (OM_descriptor): it contains the OID of the notification.
- sUser: (OM_descriptor): DN of the initiator MAE.
- sProvider: (OM_descriptor): DN of the target MAE.
- addInfo: (OM_descriptor): OID of the target MAE in case of emmision of a service report notification.
- addText: (OM_descriptor): OID of the initiator MAE in case of emmision of a service report notification.
- sACause: (OM_descriptor): OID of the security alarm cause.
- sASeverity: (OM_descriptor): Severity of the event (MAJOR / MINOR / WARNING / CRITICAL / INTERMEDIATE)

- srCause: (OM_descriptor): OID of the service report cause.
- saInfo: (OM_descriptor): Security Alarm Info in case of a service report notification.
- setEventReport: (OM_descriptor) contains a CMIP event report.
- eInfo - Security Alarm Info.

4.7.2 Required and supported component interfaces

4.7.2.1 Interface between the Self and the SMASC

The SMASC and the Self communicate through a UNIX socket created at the current directory where the agent runs. The definition is as follows:

```
#define SER_ADDR "./server.soc
```

The above and all the definitions that follow reside in the AA.h file which represents the library used by the SMASC in order to communicate with the agent. There is a number of security events that the SMASC sends to the agent and these are listed below.

```
/* Security events as defined in D9 + "NULL string event" in order to
use the serviceReport Notification */
#define AUTH_F "authenticationFailure"
#define KEY_EXP "keyExpired"
#define IMD "informationModificationDetected"
#define DI "duplicateInformation"
#define BOC "breachOfConfidentiality"
#define UAA "unauthorizedAccessAttempt"
#define OOS "outOfService"
#define HAA "outOfHoursActivity"
#define UR "unspecifiedReason"
#define NO_EVENT "NULL"
```

In what follows the structure of the notification information received by the agent process is described:

FUNCTION open_AA_connection

SYNOPSIS

```
void open_AA_connection ()
```

DESCRIPTION

This function is used to open a connection with the agent process.

ARGUMENTS

None.

FUNCTION send_info

SYNOPSIS

```
void send_info( enum Service_Type service, char *EType, char *Suser,
char *Sprovider)
```

DESCRIPTION

This is the function used to send the notification. For the moment only the first four in the set of services are considered since there is no connection of the TRUMPET management system with SMASC. Later versions may include the rest of the services.

ARGUMENTS

- *service* defines a service chosen from the available set of services defined by the Service_Type type (Auth,KeyMgmnt,Int_Conf,AC,Assoc,Mgmnt,Notif,Other)
- *EType* defines the event from the list of available events described above.
- *Suser* represents the DN of the initiator MAE
- *Sprovider* represents the DN of the target MAE.

FUNCTION send_AA_notif

SYNOPSIS

```
int send_AA_notif (char *str)
```

DESCRIPTION

This function sends actually the notification by filling in the notification string with other information, which is the library version and the instance of the service.

FUNCTION close_AA_connection

SYNOPSIS

```
void close_AA_connection ()
```

DESCRIPTION

This function closes the connection to the agent process and should always be called after sending the notification.

4.7.2.2 Interface between the Self and the audit manager

This interface is accomplished though a number of XOM objects (all of type OM_descriptor) since the communication between the agent and the manager is done using the CMIP protocol:

- setEventReport: this is the XOM object representing the CMIP event report sent to the manager. It follows the X736, X740 standards and the type of information set includes the Managed Object Class, the Managed Object Instance the Event Type the Event Time and the Event Info. The Event Info is realised by the following two XOM objects.
- eInfo: this object is used for a security related event.
- saInfo: this object is used when a service report notification is sent.

5. SECURITY INSTALLATION GUIDE

5.1 Security Profile Management

5.1.1 Hardware and software prerequisites

- RogueWave version 6.0.4 or later.

5.1.2 Installation / configuration instructions

The Security Profile Management module is present in the distribution in the location: src/securityPackage/secProfile

It consists of the files:

- README
- Makefile -- the Makefile
- secpolicy.cc -- policy object c++ file
- secpolicyP.cc -- policy object c++ file
- trumpet/secpolicy.hh -- the header file to include for users
- trumpet/secpolicyP.hh -- private header file
- sp-tst.cc -- an example/test program
- secprofs.txt -- a sample policy file

Installation is done by doing a *make* and a *make install*. The example/test program is not installed as it is considered to be of little general interest.

Security Policy rules are set up as a table of entries:

(initiatorTitle, initiatorRole, responderTitle, accessControlDirectory, securityProfile).

The table that defines the security profiles to be used is expected to be found on a file which name is retrieved from the environment variable SECURITY_PROFILES. In case this variable is not set, a file **secprofs.txt** is looked for, and in case this file neither exists, an empty table is set up.

A file with security profiles must have entries like this:

iT: cn=BMA,OU=NR,O=TRUMPET Project

iR: [AGENT | MANAGER]

rT: cn=ABMA,OU=NR,O=TRUMPET Project

aC: /path/to/access/control/directory

sP: [NULL | BASIC]

where each group of five lines are separated with one or more lines starting with a space (or just newline). If multiple (iT,iR,rT) tuples are present in the file, the last one is used. In case of a bad entry specification, the entry is skipped.

The meaning of the two sP values are:

- NULL: No security
- BASIC: Access Control & Authentication & Alarm, Audit, Recovery & Integrity

A sample secprofs.txt file:


```
iT: cn=BMA, OU=NR,O=TRUMPET Project
iR: AGENT
rT: cn=ABMA,OU=NR,O=TRUMPET Project
aC: /path/to/null-rules
sP: NULL

iT: cn=BMA,OU=NR,O=TRUMPET Project
iR: AGENT
rT: cn=mv,OU=UCL,O=TRUMPET Project
aC: /path/to/basic-rules
sP: BASIC
```

5.1.3 Runtime

If the security profile file is named secprofs.txt and exists in the current directory, no environment variable needs to be set to point out the file. If the file is called something else, or is placed in another directory, the environment variable SECURITY_PROFILES must be set to file's full or relative path. If no file is indicated, one way or the other, no security profiles will be known to the system.

5.1.4 Version / release history

The implementation was updated after the integration meeting held beginning of November and a new version released on the 12th. This version is by the end of November still the last one, and no new one is planned.

5.1.5 Known bugs

None.

5.2 Security Support Object

5.2.1 Hardware and software prerequisites

The Security Support Objects are implemented with the SECUDE security software package:

- SECUDE-5.1 Preview Release III

5.2.2 Installation / configuration instructions

You have downloaded a SECUDE package: secude-5.0c-<platform>.<packer>

where <platform> is the selected platform, <packer> is one of

"zip" - ZIP, unpack with 'unzip <package>.zip'

"tgz" - GNU TAR + Zip, unpack with 'tar zxf <package>.tgz'

"tz" - Unix TAR + COMPRESS, unpack with 'cat <package>.tz | uncompress | tar xv'

To develop secured programs, SECUDE delivers all necessary header files. Unpack the following file:

../<your-secude-directory>/lib/include.<packer>

Unpack it to a directory of your choice. Then copy the file

../<your-secude-directory>/lib/configur.h

to that directory.

SECUDE ships the following files:

Unix platforms:

../RELEASE	known bugs within SECUDE 5.0c
../bin/secude	SECUDE Single Utility
../lib/libsecude.[o or so]	SECUDE shared library
../lib/configur.h	system dependend configuration file; it belongs to SECUDEs include files.
../lib/include.<packer>	SECUDE packed header files
../etc/af-db	<empty>

After unpacking the downloaded SECUDE package, run the platform-specific setup procedures.

Unix platforms:

On some UNIX-Systems, you have to add the path to SECUDEs shared library to

LD_LIBRARY_PATH (e.g. SunOS, DEC) and to add the path to SECUDEs binary to your path.On others you only have to modify your path settings.

The following environment settings are mandatory:

HOME	HOME <directory>, default path for a PSE - Personal Security Environment.
SECUDE_ETC	SECUDE_ETC <directory>, path for some secude relevant configuration files; the directory is also used as repository for testkeys. The default is set to /usr/local/secude/etc. If you have no access to /usr/local/secude/etc, set it to a directory of your choice.

5.2.3 Known bugs

None.

5.3 Access Control

5.3.1 Hardware and software prerequisites

To compile and run the access control component you need to have the RogueWave Tools.h++ installed (v6.0 or above).

5.3.2 Installation / configuration instructions

The access control domains used in a program are configured by ASCII files contained in UNIX directories with the same name. These directories must be located in the directory the program is started from or the path can be specified by defining the environment variable *MIB_PATH*. The directory must contain the following files (filenames are fixed):

- *initiator.MIB* contains a set of initiator groups which can be referred by rules.

  An initiator group definition consists of a name followed by at least one initiator name (string DN). All of this had to be on separate lines. A definition will be closed by a line starting with the character *'#'*.

  EXAMPLE:

```
; initiators
initiators1
   DN1
```



```
   DN2
#
initiators2
   DN1
   DN3
#
```

- *target.MIB* contains a set of target groups which can be referred by rules.

  An target group definition consists of a name followed by at least one target. All of this had to be on separate lines.

  A target is defined by the name of a base object instance (string DN) otional followed by scope in brackets. A scope is specified according to this:

  [*] whole subtree

  [-n] base to nth Level

  [n] individual Level n

  A definition will be closed by a line starting with the character '#'.

  EXAMPLE:

```
; targets
targets1
   DN1
   DN2[2]
#
targets2
   DN1[*]
   DN2[-3]
#
```

- *rule.MIB* describes the rules of the access control domain

  A rule definition consists of a name followed by at least one initiator group reference, a set of target group references and an access right specification. The different parts are separated by a line starting with the character '-' (minus).

  The access rigths are defined starting with the permission followed by an optional list of operations. If no operation is specified the rule is valid for all operations.

  A definition will be closed by a line starting with the character '#'.

  EXAMPLE:

```
; access control rules
globalRule1
   initiators1
   initiators2
   -
   -
   allow create
# end rule
itemRule1
   initiators2
   -
   targets2
   -
   deny
#
```

- *ruleDefault.MIB* contains the default rule of the access control domain

A default rule is specified by a line with one permission for each operation separated by ':'. The sequence of operations is:

get, set, create, delete, action

EXAMPLE:

```
; Default permission for
; get:set:create:delete:action
allow:deny:allow:deny:deny
```

5.3.3 Runtime

5.3.4 Version / release history

- v1.0 Initial Release Nov 97

5.3.5 Known bugs

None.

5.4 Secure Management Association

5.4.1 Hardware and software prerequisites

The SMASC compiles with any C++ compiler on Solaris or HPUX workstations. It requires all the other security components (Security Profile Manager, Security Support Object, Access Control, SELF, Audit and Alarm).

5.4.2 Installation / configuration instructions

At a unix shell prompt, type:

```
cd Trumpet.cur/src/secPackage/smasc
make all
make install
```

5.4.3 Version / release history

0.1 Basic integration with the SSO and SELF components.

5.4.4 Known bugs

Still much work to be done.

5.5 Adapter Object

5.5.1 Hardware and software prerequisites

The Adapter Object compiles with any C++ compiler on Solaris or HPUX workstations. It requires:

- HP Open View 4.21 or later (DM, OSI and XMP packages installed).
- TRUMPET SMP (SMASC and all the other security components).

5.5.2 Installation / configuration instructions

At a unix shell prompt, type:

```
cd Trumpet.cur/src/secPackage/xmpV7_adapter
make all
make install
```

A simple testing program is provided with the XMP Adapter. It can be installed using:

```
cd Trumpet.cur/src/secPackage/xmpV7_assoc
make all
```


To run this testing tool, open two shell windows (xterms). Type **ag** in the first one and then **ma** in the second. The output messages are self-explicit.

5.5.3 Version / release history

0.1 First level of capability: authentication.

5.5.4 Known bugs

The authentication-Method and authentication-Value ACSE Session fields are not transmitted to the target MAE. This bug might be platform-specific and may not appear under HPUX.

5.6 Audit and Alarm

5.6.1 Hardware and software prerequisites

- HP OpenView version 4.21
- Motif 1.2
- C compiler

5.6.2 Installation / configuration instructions

In the main directory of the application, you have a file Makefile which is used by the command make to generate the executable « auditman » and the needed library «smli_audit.a ».

In the file Makefile you need to set the following variables with the appropriate paths specific to the current host:

- INCLUDE must contain all the paths of the include files needed to compile (standard C, Motif, X11, XMP, XOM).
- LDFLAGS contains the path of the HP OpenView and standard C libraries.
- LDFLAGS_X contains the path of the Motif and X11 libraries.

When all these variables are filled, you run the compilation by executing the make command.

To install the audit and alarm manager, you need:

- the executable auditman, to execute you need to type «auditman » with no parameter, in the directory /auditman,
- a configuration file audit.cfg which contains the IP address of the manager and agent in the same directory as the executable, this file must have 2 lines:one for the agent and one for the manager with the following information:

[AGENT]

Section1: EntityName;coordinates X;coordinates Y;Graphical object class;

Section2: XMP requestor/responder title and addresses. The AE-Title should be declared in form2. The structure is: entityName;AP-title-form2;AE-qualifier-form2;psel;ssel;tsel;nadd;

[MANAGER]

Section1:EntityName;Graphical object class;

Section2: XMP requestor/responder title and addresses. The AE-Title should be declared in form2. The structure is: entityName;AP-title-form2;AE-qualifier-from2;psel;ssel;tsel;nadd;

Here is an example:


```
[MANAGER]:
     auditman;1.9.2.134.145.116;1;
#0302;030100;0101;08005ac7442d;
[END]
[AGENTS]:
     Selfagent;220;460;1.9.2.134.145.116;1;
#0202;020200;0202;08005ac7442d;
[END]
```

- a directory « log » to create the log files in the same directory as the executable.

5.6.3 Runtime

To run the audit and alarm manager, you need:

- 2 generated packages (with ovpacgen): HP_DMI_PKG (from /opt/gdmo_mibs/ems.mib) and AUD_PKG on the plateform HP Openview in the directory pacgen. In order to use these packages, it is necessary to set the environmental variables:

setenv MP_PACKAGE_DIR $HOME/pacgen

setenv MP_PACKAGE_DIR_NAME package.dir

- the postmaster of the platform and the ovead agent need to be started.

5.6.4 Version / release history

Actually, the agent of the HP OpenView platform does not implement the behaviour of all the packages of X721 GDMO. The Duration package is not implemented in the ovead agent, the GUI implements the Duration package (week mask, intervals of day, ...), but since it is not implemented in the agent, setting these attributes in the manager will have no effect on the behaviour.

5.6.5 Known bugs

None.

5.7 SELF

5.7.1 Hardware and software prerequisites

In order to run the Self component HP OpenView libraries should be in place. Also the directory /pacgen should be present. In there the MIB resides.

5.7.2 Software installation

All the following sources are available:Selfagent1.c , Selfagent.h (MIB file) , tools.h, macrotools.h, AA.h, tools.c, mactorools.c (additional libraries). Using the make command, the executable Self will be generated.

5.7.3 Runtime

The steps in order to run the agent are as follows:

1. The following environmental variables have to be set, thanks to the commands:

   setenv MP_PACKAGE_DIR $HOME/pacgen

   setenv MP_PACKAGE_DIR_NAME package.dir

2. The Self component is the audit and alarm agent. You can run at the /audagent directory by typing at the prompt Self &. This command runs the agent at the background. The agent behaves as a server process.

3. The input comes from the SMASC. There are 5 different client programs that contain instructions for opening and closing the socket and sending notifications. The source code for these clients exists in the /sockets directory under the prefix test_client. These lines of code should appear somewhere in the code of SMASC.

4. So each time you run one of these clients the Self agent sends an event report to the manager. So while the agent runs in the background if you type e.g. Client1 then the client sends the notification and the agent preprocess it and sends the CMIP event report to the manager.

5. The client programs should be run from the same directory so they exist (the executables) in the audagent directory. This is so, because the socket the server opens exists in the current directory where the Self runs so the client program should read from it (the socket is a file). In the trials this should be set to a fixed directory on the filing system of the machine where the SMASC and the agent run.

6. The Self code should always be compiled with the AA.h library. The SMASC code should be compiled using the AA.c and AA.h files. These two exist in the /sockets directory and there is a Makefile of how to use AA.c and AA.h. In this Makefile the test_client.c resembles the SMASC and the server.c resembles the Self agent.

5.7.4 Version / release history

version 1.0 , November 1997.

5.7.5 Known bugs

None.

6. REFERENCES

[Gos95] The Java Language Environment, A White Paper, James Gosling, Henry McGilton, Sun Microsystems, October 1995.

[Kern83] The ANSI C Programming Language, Brian Kernighan and Dennis Ritchie, Prentice Hall, 1983.

[MISA-D3-A1] Initial MISA High Level Design, Annex A1: Xuser Interface Definition, ACTS AC080 MISA Deliverable 3, Annex A1, September 1996

[MISA-D3-A2] Initial MISA High Level Design, Annex A2: Path Provisioning Ensemble, ACTS AC080 MISA Deliverable 3, Annex A2, September 1996

[Orbix96a] Orbix 2 Programming Guide, IONA Technologies, October 1996.

[Orbix96b] The Orbix Architecture, IONA Technologies, November 1996.

[Orfali97] Client/Server Programming with JAVA and CORBA, Robert Orfali and Dan Harkey, John Wiley & Sons, Inc., ISBN 0-471-16351-1, USA, 1997.

[OMG CORBA] http://www.omg.org

[OMG-970301] IDL/Java Language Mapping, Joint Revised Submission, OMG TC Document orbos/97-03-01, 19/3/1997

[RFC 1508] RFC 1508, “Generic Security Service Application Program Interface”, September 1993

[Str86] The C++ Programming Language, Bjarne Stroustrup, Addison Wesley, 1986

[TRUMPET-D6] ACTS AC112 Trumpet Deliverable 6, NIL-Security Prototype Report, February 1997.

[TRUMPET-D8] ACTS AC112 TRUMPET Deliverable 8, "Detailed Component and Scenario Design ", June. 1997

[TRUMPET-D9] ACTS AC112 TRUMPET Deliverable 9, " System Component Specification ", October. 1997

[UML] Unified Modelling Language, RATIONAL Software Corporation, http://www.rational.com/

[XOM] X/Open Company Limited & X.400 API Association, XOM, OSI-Abstract-Data Manipulation API, CAE Specification, 1991

7. ACRONYMS

API	Application Programmer's Interface
ATM	Asynchronous Transfer Mode
BSP	Binary Space Partition
CA	Certification Authority
CIM	Common Information Model
CIS	Communications Interface System
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	CMIS Element
CORBA	Common Object Request Broker Architecture
CPN	Customer Premises Network
CRL	Certificate Revocation List
DII	Dynamic Invocation Interface
DN	Distinguished Name
DSI	Dynamic Skeleton Interface
EML	Element Management Layer
EMS	Event Management System
GBC	Global Broadband Connection
GBCM	Global Broadband Connectivity Management
GDMO	Guidelines for the Definition of Managed Objects
GUI	Graphical User Interface
HMM	Hyper-Media Management
HPOV	Hewlert-Packard Open View
HTML	Hyper-Text Mark-up Language
HTTP	Hyper-Text Transfer Protocol
IBC	Integrated Broadband Connection
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JLDAP	Java LDAP
JMAPI	Java Management API
LAN	Local Area Network
LDAP	Lite Weight Directory Access Protocol
LIF	Local Interface
MIB	Management Information Base
MO	Managed Object
MOS	Managed Object Server
NEL	Network Element Layer
NEV	Network Element View
NL	Network Layer
NML	Network Management Layer
NMS	Network Management System
NV	Network View
ODP	Open Distributed Processing
OMG	Object Management Group
ONP	Open Network Provision
ORB	Object Request Broker
OS	Operation System
OSF	Operation System Function
OSI	Open Systems Interconnection
PC	Personal Computer
PNO	Public Network Operator
PTT	Postal, Telegraph and Telephone
QAF	Q Adapter Function
QATM	ATM QAF
QoS	Quality of Service
RDN	Relative Distinguished Name
RMI	Remote Method Invocation
SAC	Service Access Control
SDH	Synchronous Digital Hierarchy
SII	Static Invocation Interface
SL	Service Layer
SME	Small and Medium Enterprise (Market)
SNC	Subnetwork Connection (as in releaseSNC, etc.)
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office (Market)
SSI	Static Skeleton Interface
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TMN	Telecommunications Management Network
TTP	Trusted Third Party
UML	Unified Modelling Language
URL	Universal Resource Locator
VASP	Value Added Service Provider
VP	Virtual Path
VPI	Virtual Path Identifier
WAN	Wide Area Network
WPx	Work Package Number x

8. APPENDIX A - CORBA/TMN GATEWAY INTERFACE

This section contains the complete set interface definitions for the CORBA/TMN gateway. Parts of this interface definitions have been derived from the MISA Xuser GDMO specification (see Section 10) according to the mapping rules have been developed within the Network Management Forum to map between GDMO and IDL definitions. The specification is organised as follows:

- *PNO Connection Manager*: contains all the supported interfaces of the CORBA/TMN gateway excluding the parameter types which are defined separately as Xuser types.
- *VASP VPN Manager*: contains the definition of the interfaces which is required from the VASP Control Server.
- *Xuser Types*: Defines all the parameter types and its constituents.
- *ASN.1 Types*: Defines types definitions according to ASN.1 basic types.

8.1 PNO Connection Manager

```
/* $Id:$ */

/*
** Copyright (C) 1997
** by GMD - German National Research Center for Information Technology
**
** Permission to use, copy, modify, distribute, and sell this software
** and its documentation for any purpose is hereby granted without fee,
** provided that the above copyright notice appear in all copies and
** that both that copyright notice and this permission notice appear
** in supporting documentation, and that the name of GMD not be used in
** advertising or publicity pertaining to distribution of the software
** without specific, written prior permission. GMD makes no
** representations about the suitability of this software for any
** purpose. It is provided "as is" without express or implied warranty.
**
** GMD DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING
** ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT
** SHALL GMD BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES
** OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
** WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
** ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
** SOFTWARE.
*/

#ifndef _PnoConnectionMgr_IDL_
#define _PnoConnectionMgr_IDL_

#include "XuserTypes.idl"
#include "VpnManager.idl"

module PnoConnectionMgr {
/*
This IDL module defines the supported external interfaces of the component
PNO Connection Manager. The PNO Connection Manager requires the
external interface VpnManager::VPConnectionServiceEventHandler which
is needed to propagate Xuser event reports to the VASP. This interface
is defined by the IDL module VpnManager.
*/

//forward declaration of interfaces
interface VPConnectionServiceFactory;
interface VPConnectionService;
interface VPSubscriptionServiceFactory;
interface VPSubscriptionService;
interface VPConnectionServiceEventHandler;
```



```
//_____________________ Interface: VPConnectionServiceFactory ____________________

interface VPConnectionServiceFactory {
      exception ServiceProfileNotFound {
      };

      VPConnectionService
      create(in XuserTypes::NameType pnoId,
            in VpnManager::VPConnectionServiceEventHandler eventHandler);

      void
      destroy(in VPConnectionService vpConnectionServiceRef);
};

//_____________________ Interface: VPConnectionService ____________________

interface VPConnectionService {

      exception ConnectionRequestFailure {
            XuserTypes::ReasonType      reason;
      };

      XuserTypes::ReserveConnectionResultType
      reserveConnection(
            in XuserTypes::ReserveConnectionInfoType connectionInformation)
      raises (ConnectionRequestFailure);

      void
      modifyConnection(
            in XuserTypes::ModifyConnectionInfoType connectionInformation)
      raises (ConnectionRequestFailure);

      void
      releaseConnection(
            in XuserTypes::ReleaseConnectionInfoType connectionInformation)
      raises (ConnectionRequestFailure);

};

//_____________________            Interface:            VPSubscriptionServiceFactory
____________________

interface VPSubscriptionServiceFactory {
      exception ServiceProfileNotFound {
      };

      VPSubscriptionService
      create(in XuserTypes::NameType pnoId);

      void
      destroy(in VPSubscriptionService subscriptionServiceRef);
};

//_____________________ Interface: VPSubscriptionService ____________________

interface VPSubscriptionService {

      exception InvalidAccessPoint {
      };

      exception NotFound {
      };

      void
      createAccessPoint(in XuserTypes::IdentifierType userId,
            in XuserTypes::NameType accessPointId,
```



```
            in XuserTypes::E164AddressType E164Address)
      raises (InvalidAccessPoint);

      void
      deleteAccessPoint(in XuserTypes::IdentifierType userId,
            in XuserTypes::NameType accessPointId)
      raises (NotFound);
};

}; // End of Module

#endif
```

8.2 VP Connection Manager

```
/* $Id:$ */

/*
** Copyright (C) 1997
** by GMD - German National Research Center for Information Technology
**
** Permission to use, copy, modify, distribute, and sell this software
** and its documentation for any purpose is hereby granted without fee,
** provided that the above copyright notice appear in all copies and
** that both that copyright notice and this permission notice appear
** in supporting documentation, and that the name of GMD not be used in
** advertising or publicity pertaining to distribution of the software
** without specific, written prior permission. GMD makes no
** representations about the suitability of this software for any
** purpose. It is provided "as is" without express or implied warranty.
**
** GMD DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING
** ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT
** SHALL GMD BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES
** OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
** WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
** ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
** SOFTWARE.
*/

#include "XuserTypes.idl"

module VpnManager {
/*
*/

//_____________________          Interface:          VPConnectionServiceEventHandler
_________________________

interface VPConnectionServiceEventHandler {

      oneway void
      activateConnectionNotify(in XuserTypes::NameType pnoId,
            in XuserTypes::NameType vpConnectionId,
            in XuserTypes::ActivationNotifInfoType status);

      oneway void
      releaseConnectionNotify(in XuserTypes::NameType pnoId,
            in XuserTypes::NameType vpConnectionId,
            in XuserTypes::ReleaseReasonType reason);

      oneway void
      connectionNotify(in XuserTypes::NameType pnoId,
            in XuserTypes::ReasonType reason,
            in ASN1_PrintableString eventInformation);
};
```



```
}; // End of Module
```

8.3 ASN.1 Basic Types

```
//
// ASN1Types.idl
//

#ifndef _ASN1TYPES_IDL_
#define _ASN1TYPES_IDL_

// ASN.1 base types

typedef octet              ASN1_Null;
//const ASN1_Null ASN1_NullValue = 0;
typedef boolean            ASN1_Boolean;
typedef long               ASN1_Integer;
typedef long               ASN1_Integer64[2];
typedef float              ASN1_Real;
typedef sequence<octet>    ASN1_BitString; // PIDL defined
typedef sequence<octet>    ASN1_OctetString;
typedef string             ASN1_ObjectIdentifier;
typedef any                ASN1_Any;
typedef any                ASN1_DefinedAny;

struct                     ASN1_External {
      ASN1_ObjectIdentifier  syntax;
      ASN1_DefinedAny        data_value;  // by syntax
};

// ASN.1 strings which may not contain binary zeros

typedef string             ASN1_IA5String;
typedef string             ASN1_NumericString;
typedef string             ASN1_PrintableString;
typedef string             ASN1_TeletexString;
typedef string             ASN1_T61String;
typedef string             ASN1_VideotexString;
typedef string             ASN1_VisibleString;

typedef ASN1_VisibleString ASN1_GeneralizedTime; // PIDL defined
typedef ASN1_VisibleString ASN1_UTCTime;

// ASN.1 strings which may contain binary zeros

typedef sequence<octet>    ASN1_BMPString;
typedef sequence<octet>    ASN1_GeneralString;
typedef sequence<octet>    ASN1_GraphicString;
typedef sequence<octet>    ASN1_ISO646String;
typedef sequence<octet>    ASN1_UniversalString;

typedef ASN1_GraphicString ASN1_ObjectDescriptor;

// define constants for ASN.1 Real infinity

//#include "ASN1Limits.idl"

#endif /*  _ASN1TYPES_IDL_ */
```

8.4 Xuser Types

```
/* $Id:$ */

/*
** Copyright (C) 1997
** by GMD - German National Research Center for Information Technology
**
```



```
** Permission to use, copy, modify, distribute, and sell this software
** and its documentation for any purpose is hereby granted without fee,
** provided that the above copyright notice appear in all copies and
** that both that copyright notice and this permission notice appear
** in supporting documentation, and that the name of GMD not be used in
** advertising or publicity pertaining to distribution of the software
** without specific, written prior permission. GMD makes no
** representations about the suitability of this software for any
** purpose. It is provided "as is" without express or implied warranty.
**
** GMD DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING
** ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT
** SHALL GMD BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES
** OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
** WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
** ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
** SOFTWARE.
*/

#ifndef _XuserTypes_IDL_
#define _XuserTypes_IDL_

#include "ASN1Types.idl"

module XuserTypes {
/*
This IDL module defines the basic types of the CORBA adapter object to
the Xuser interface. The type definitions has been translated from
corresponding definitions of the MISA Xuser ASN.1 Module using
the translation algorithm resulted from JIDM (X/Open-NMF joint activity).
However, in some cases, the translated definitrions have been slightly
modified to simply their usage for the application programmer.
*/

//_____________________________ Type: ActivationNotifInformation _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition

ActivationNotifInformation::= ENUMERATED { ok(0), ko(1)}
*/

enum  ActivationNotifInfoType {ok, ko
};

//_____________________ Type: NameType _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from M.3100)

NameType::=    CHOICE {
     numericName     INTEGER,
     pString GraphicString
}

NOTE: TRUMPET uses PrintableString instead of GraphicString to
simplify the handling of pString components. GraphicString is
mapped to sequence<octect> and PrintableString is mapped
to string. The latter implies the limitation that the sequence
of chars shall not include the null char (binary zeros for 8-bit chars).
*/

enum NameTypeChoice { numericName, pString };

union NameType switch(NameTypeChoice) {
     case numericName:   ASN1_Integer numericName;
     case pString:              ASN1_PrintableString pString;
};
```



```
//_____________________ Type: Time24Type _____________________

/* Corresponding MISA-Xuser ASN.1 type definition (from SMI):

Time24::= SEQUENCE {
      hour            INTEGER (0..23),
      minute  INTEGER (0..59)
}
*/

struct  Time24Type{
      unsigned short hour;
      unsigned short minute;
};

//_____________________ Type: DaySlot _____________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ETSI ETS300653):

GBCDaySlot::= SEQUENCE {
      slotBegin Time24,
      slotEnd Time24
}
*/

struct  DaySlotType  {
      Time24Type    slotBegin;
      Time24Type    slotEnd;
};

//_____________________ Type: DailySchedule _____________________

/* Corresponding MISA-Xuser ASN.1 type definition
   (originally taken from ETSI ETS300653):

GBCDailySchedule::= SEQUENCE OF GBCDaySlot
*/

typedef sequence <  DaySlotType > DailyScheduleType;

//_____________________ Type: WeekDay _____________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ETSI ETS300653):

WeekDay::= ENUMERATED {
      sunday (0),
      monday (1),
      tuesday (2),
      wednesday (3),
      thursday (4),
      friday (5),
      saturday (6)
}
*/

enum  WeekDayType{sunday, monday, tuesday, wednesday, thursday, friday, saturday};

//_____________________ Type: StopTime _____________________

/* Corresponding MISA-Xuser ASN.1 type definition (from SMI):

StopTime::= CHOICE {
           specific GeneralizedTime,
           continual       NULL
}
```



```
*/

enum StopTimeTypeChoice{
     specificChoice,
     continualChoice
};

union StopTimeType  switch(StopTimeTypeChoice){
     case specificChoice: ASN1_GeneralizedTime specific;
     case continualChoice:      ASN1_Null continual;
};

//_____________________________ Type: NA4StartTime _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ETSI ETS300653):

NA4StartTime::= StopTime
*/

typedef StopTimeType NA4StartTimeType;

//_____________________________ Type: TimeWeek _____________________________

/*
TimeWeek::= SEQUENCE {
     weekDay WeekDay,
     time Time24
}
*/

struct  TimeWeekType {
     WeekDayType   weekDay;
     Time24Type    time;
};

//_____________________________ Type: WeekSlot _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition
   (originally taken from ETSI ETS300653):

GBCWeekSlot::= SEQUENCE {
     slotBegin TimeWeek,
     slotEnd TimeWeek
}
*/

struct  WeekSlotType {
     TimeWeekType  slotBegin;
     TimeWeekType  slotEnd;
};

//_____________________________ Type: WeeklySchedule _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition
   (originally taken from ETSI ETS300653):

GBCWeeklySchedule::= SEQUENCE OF GBCWeekSlot
*/

typedef sequence <  WeekSlotType > WeeklyScheduleType;

//_____________________________ Type: OccasionalSlot _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition
   (originally taken from ETSI ETS300653):
```



```
GBCOccasionalSlot::= SEQUENCE {
      slotBegin NA4StartTime,
      slotEnd StopTime
}
*/

struct  OccasionalSlotType {
      NA4StartTimeType           slotBegin;
      StopTimeType        slotEnd;
};

//_____________________________ Type: OccasionalSchedule ____________________________

/*  Corresponding MISA-Xuser ASN.1 type definition
    (originally taken from ETSI ETS300653):

GBCOccasionalSchedule::= SEQUENCE OF GBCOccasionalSlot
*/

typedef sequence <  OccasionalSlotType > OccasionalScheduleType;

//_____________________ Type: E164Address ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

E164Address::= PrintableString
*/

typedef ASN1_PrintableString    E164AddressType;

//_____________________ Type: Identifier ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

Identifier::= INTEGER
*/

typedef ASN1_Integer      IdentifierType;

//_____________________________ Type: ProtectionLevel ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ProtectionLevel::=  ENUMERATED {
      protected (0),
      unprotected-lowpriority (1),
      unprotected-hightpriority (2)}
*/

enum ProtectionLevelType {
      protected, unprotected_lowpriority, unprotected_hightpriority
};

//_____________________ Type: Reason ____________________________

typedef NameType ReasonType;

/* Corresponding MISA-Xuser ASN.1 type definition:

Reason::= NameType
*/

//_____________________ Type: Schedule ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:
```



```
GBCSchedule::= SEQUENCE {
      startTime       NA4StartTime,
      stopTime        StopTime,
      CHOICE {        durationSchedule [0]    NULL,
             dailySchedule [1]       GBCDailySchedule,
             weeklySchedule  [2]     GBCWeeklySchedule,
             monthlySchedule [3]     GBCMonthlySchedule,
             occasionalSchedule [4]  GBCOccasionalSchedule
}
*/

enum ScheduleItemTypeChoice {
      durationSchedule,
      dailySchedule,
      weeklySchedule,
      occasionalSchedule
};

union ScheduleItemType  switch(ScheduleItemTypeChoice) {
      case durationSchedule: ASN1_Null durationSchedule;
      case dailySchedule: DailyScheduleType dailySchedule;
      case weeklySchedule: WeeklyScheduleType weeklySchedule;
      case occasionalSchedule: OccasionalScheduleType occasionalSchedule;
};

struct ScheduleType {
      NA4StartTimeType  startTime;
      StopTimeType  stopTime;
      ScheduleItemType scheduleChoice;
};

union ScheduleTypeOpt switch(boolean) {
      case TRUE: ScheduleType schedule;
      default: ASN1_Null undefined;
};

//_____________________________ Type: QosClass _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition
   (adopted from ATMforum M4):

QoSClass::= INTEGER {
      cbr(0),
      rt-vbr(1),
      nrt-vbr(2),
      ubr(3),
      abr(4),
      sdh-cbr(5)
}

NOTE: sdh-cbr does not apply for TRUMPET

*/

typedef ASN1_Integer     QoSClassType;

const QoSClassType cbr=0;
const QoSClassType rt_vbr=1;
const QoSClassType nrt_vbr=2;
const QoSClassType ubr=3;
const QoSClassType abr=4;
const QoSClassType sdh_cbr=5;

//_____________________________ Type: MaxNumVp _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:
```



```
MaxNumVp::= INTEGER
*/

typedef ASN1_Integer MaxNumVpType;

//_____________________________ Type: PeakBitRate _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

PeakBitRate::= INTEGER
*/

typedef ASN1_Integer PeakBitRateType;

//_____________________________ Type: MaxCellTransferDelay _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

MaxCellTransferDelay::= SEQUENCE {
      acceptableMaxCTD [0] INTEGER OPTIONAL,
      cumulativeMaxCTD [1] INTEGER OPTIONAL
      }
*/

union AcceptableMaxCTDTypeOpt switch(boolean) {
      case TRUE: ASN1_Integer  acceptableMaxCTD;
      default: ASN1_Null undefined;
};

union CumulativeMaxCTDTypeOpt switch(boolean) {
      case TRUE: ASN1_Integer  cumulativeMaxCTD;
      default: ASN1_Null undefined;
};

struct  MaxCellTransferDelayType {
      AcceptableMaxCTDTypeOpt acceptableMaxCTD;
      CumulativeMaxCTDTypeOpt cumulativeMaxCTD;
};

//_____________________________            Type:            PeakToPeakCellDelayVariation
_____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

PeakToPeakCellDelayVariation::= SEQUENCE {
      acceptablePeakToPeakCDV [0] INTEGER OPTIONAL,
      cumulativePeakToPeakCDV [1] INTEGER OPTIONAL }
*/

union CDVTypeOpt switch(boolean) {
      case TRUE: ASN1_Integer  acceptablePeakToPeakCDV;
      default: ASN1_Null undefined;
};

struct  PeakToPeakCellDelayVariationType{
      CDVTypeOpt acceptablePeakToPeakCDV;
      CDVTypeOpt cumulativePeakToPeakCDV;
};

//_____________________ Type: CellLossRatio _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

CellLossRatio::= INTEGER (1..15)
*/
```



```
typedef unsigned short CellLossRatioType;

//_____________________ Type: MaxDelay ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

MaxDelay::= INTEGER (1..15)
*/

typedef ASN1_Integer MaxDelayType;

//_____________________ Type: PeakCellRate ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):

PeakCellRate::= SEQUENCE {
     peakCellRateCLP0plus1 [0] INTEGER OPTIONAL,
     peakCellRateCLP0 [1] INTEGER OPTIONAL
}
*/

union PeakCellRateCLPTypeOpt switch(boolean){
     case TRUE: ASN1_Integer  peakCellRate;
     default: ASN1_Null undefined;
};

struct  PeakCellRateType {
     PeakCellRateCLPTypeOpt peakCellRateCLP0plus1;
     PeakCellRateCLPTypeOpt peakCellRateCLP0;
};

//_____________________ Type: MaxBurstSize ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):

/*
MaxBurstSize::= SEQUENCE {
     maxBurstSizeCLP0plus1 [0] INTEGER OPTIONAL,
     maxBurstSizeCLP0 [1] INTEGER OPTIONAL
}
*/

union MaxBurstSizeCLPTypeOpt switch(boolean) {
     case TRUE: ASN1_Integer  maxBurstSize;
     default: ASN1_Null undefined;
};

struct  MaxBurstSizeType{
    MaxBurstSizeCLPTypeOpt maxBurstSizeCLP0plus1;
    MaxBurstSizeCLPTypeOpt maxBurstSizeCLP0;
};

//_____________________ Type: MinimumCellRate ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):

MinimumCellRate:= PeakCellRate
*/

typedef ASN1_Integer MinimumCellRateType;

//_____________________ Type: SustainableCellRate ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):
```



```
/*
SustainableCellRate::= SEQUENCE {
      sustainableCellRateCLP0plus1 [0] INTEGER OPTIONAL,
      sustainableCellRateCLP0 [1] INTEGER OPTIONAL
}
*/

union SustainableCellRateCLPTypeOpt switch(boolean) {
      case TRUE: ASN1_Integer  sustainableCellRate;
      default: ASN1_Null undefined;
};

struct  SustainableCellRateType {
    SustainableCellRateCLPTypeOpt sustainableCellRateCLP0plus1;
    SustainableCellRateCLPTypeOpt sustainableCellRateCLP0;
};

//_____________________ Type: FloatingPointCoding ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):

/*
FloatingPointCoding::= SEQUENCE {
      e INTEGER (0..31),
      w INTEGER (0..31)
}
*/

struct  FloatingPointCodingType{
      unsigned short e;
      unsigned short w;
};

//_____________________ Type: CDVToleranceCoding ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):

/*
CDVToleranceCoding::= CHOICE {
      integerCoding INTEGER,
      floatingPointCoding  FloatingPointCoding
}
*/

enum CDVToleranceCodingTypeChoice{
      integerCodingChoice,
      floatingPointCodingChoice};

union  CDVToleranceCodingType switch(CDVToleranceCodingTypeChoice) {
      case integerCodingChoice: ASN1_Integer integerCoding;
      case floatingPointCodingChoice: FloatingPointCodingType floatingPointCoding;
};

//_____________________ Type: CDVTolerance ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T I.751):

/*
CDVTolerance::= SEQUENCE {
      sustainableCellRateCLP0plus1 [0] INTEGER OPTIONAL,
      sustainableCellRateCLP0 [1] INTEGER OPTIONAL
}
*/

union CDVToleranceCLPTypeOpt switch(boolean) {
      case TRUE: CDVToleranceCodingType  sustainableCellRate;
```



```
      default: ASN1_Null undefined;
};

struct  CDVToleranceType{
    CDVToleranceCLPTypeOpt cellDelayVariationToleranceCLP0plus1;
    CDVToleranceCLPTypeOpt cellDelayVariationToleranceCLP0;
};

//_____________________________ Type: Directionality _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition (from ITU-T M.3100):

Directionality::= ENUMERATED {
      unidirectional(0),
      bidirectional(1)
}
*/

enum  DirectionalityType {unidirectional, bidirectional};

//_____________________ Type: BlockErrorRate _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

BlockErrorRate::= INTEGER
*/

typedef ASN1_Integer BlockErrorRateType;

//_____________________ Type: QoSSequenceType _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ATMSpecificParameters::= SEQUENCE {
      qosClass                [0] QoSClass OPTIONAL,
      maxCTD                  [1] MaxCellTransferDelay OPTIONAL,
      peakToPeakCDV           [2] PeakToPeakCellDelayVariation OPTIONAL,
      cellLossRatio           [3] CellLossRatio OPTIONAL,
      peakCellRate            [4] PeakCellRate OPTIONAL,
      sustainableCellRate     [5] SustainableCellRate OPTIONAL,
      pcrCDVTolerance         [6] CDVTolerance OPTIONAL,
      scrCDVTolerance         [7] CDVTolerance OPTIONAL,
      maxBurstSize            [8] MaxBurstSize OPTIONAL,
      minCellRate             [9] MinimumCellRate OPTIONAL }

APPSQoSSequence::= SEQUENCE {
      forward [0] ATMSpecificParameters OPTIONAL,
      backward [1] ATMSpecificParameters OPTIONAL
}

APPSQoSSequence::= SEQUENCE {
      forward         [0] ATMSpecificParameters OPTIONAL,
      backward        [1] ATMSpecificParameters OPTIONAL
}

GBCQoSSequence::= SEQUENCE {
      aPPSQoSSequence [0] APPSQoSSequence OPTIONAL,
      sPPSQoSSequence [1] SPPSQoSSequence OPTIONAL
}

Note: APPSQoSSequence is mapped directly onto GBCQoSSequenceType since
SDH QoS parameters are not required for TRUMPET (SPPSQoSSequence)
*/

union QoSClassTypeOpt switch(boolean) {
      case TRUE: QoSClassType qosClass;
```



```
      default: ASN1_Null undefined;
};

union MaxCTDTypeOpt switch(boolean) {
      case TRUE: MaxCellTransferDelayType  maxCTD;
      default: ASN1_Null undefined;
};

union PeakToPeakCDVTypeOpt switch(boolean) {
      case TRUE: PeakToPeakCellDelayVariationType  peakToPeakCDV;
      default: ASN1_Null undefined;
};

union CellLossRatioTypeOpt switch(boolean) {
      case TRUE: CellLossRatioType  cellLossRatio;
      default: ASN1_Null undefined;
};

union PeakCellRateTypeOpt switch(boolean) {
      case TRUE: PeakCellRateType  peakCellRate;
      default: ASN1_Null undefined;
};

union SustainableCellRateTypeOpt switch(boolean){
      case TRUE: SustainableCellRateType  sustainableCellRate;
      default: ASN1_Null undefined;
};

union PcrCDVToleranceTypeOpt switch(boolean) {
      case TRUE: CDVToleranceType  pcrCDVTolerance;
      default: ASN1_Null undefined;
};

union ScrCDVToleranceTypeOpt switch(boolean) {
      case TRUE: CDVToleranceType  scrCDVTolerance;
      default: ASN1_Null undefined;
};

union MaxBurstSizeTypeOpt switch(boolean) {
      case TRUE: MaxBurstSizeType  maxBurstSize;
      default: ASN1_Null undefined;
};

union MinCellRateTypeOpt switch(boolean) {
      case TRUE: MinimumCellRateType  minCellRate;
      default: ASN1_Null undefined;
};

struct  ATMSpecificParametersType {
      QoSClassTypeOpt qosClass;
      MaxCTDTypeOpt maxCTD;
      PeakToPeakCDVTypeOpt peakToPeakCDV;
      CellLossRatioTypeOpt cellLossRatio;
      PeakCellRateTypeOpt peakCellRate;
      SustainableCellRateTypeOpt sustainableCellRate;
      PcrCDVToleranceTypeOpt pcrCDVTolerance;
      ScrCDVToleranceTypeOpt scrCDVTolerance;
      MaxBurstSizeTypeOpt maxBurstSize;
      MinCellRateTypeOpt minCellRate;
};

union ATMQosTypeOpt switch(boolean) {
      case TRUE: ATMSpecificParametersType  parameters;
      default: ASN1_Null undefined;
};
```



```
struct  QoSSequenceType {
      ATMQosTypeOpt forward;
      ATMQosTypeOpt backward;
};

union QoSSequenceTypeOpt switch(boolean) {
      case TRUE: QoSSequenceType qosSequence;
      default: ASN1_Null undefined;
};

//_____________________________ Type: RoutingCriteria _____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

RoutingCriteria::= NameType
*/

typedef NameType RoutingCriteriaType;

//_____________________ Type: ReserveConnectionInformation ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ReserveGBCConnectionInformation::= SEQUENCE {
      gBCMUserId Identifier,
      sourceE164Address [0] E164Address OPTIONAL,
      destinationE164Address E164Address,
      connectionProtectionLevel [1] ProtectionLevel OPTIONAL,
      routingCriteria RoutingCriteria OPTIONAL,
      gBCType GBCType,
      gBCDirectionality Directionality
      gBCSchedule GBCSchedule,
      gBCPPSparameters GBCQoSSequence OPTIONAL }

NOTE: The GBCType component is not mapped since TRUMPET only
uses/supports ATM Path Provisioning Service (APPS)

*/

union E164AddressTypeOpt switch(boolean) {
      case TRUE: E164AddressType  sourceE164Address;
      default: ASN1_Null undefined;
};

union ProtectionLevelTypeOpt switch(boolean) {
      case TRUE: ProtectionLevelType  connectionProtectionLevel;
      default: ASN1_Null undefined;
};

union RoutingCriteriaTypeOpt switch(boolean) {
      case TRUE: RoutingCriteriaType  routingCriteria;
      default: ASN1_Null undefined;
};

struct  ReserveConnectionInfoType {
      IdentifierType       userId;
      E164AddressTypeOpt   sourceE164AddressOpt;
      E164AddressType      destinationE164Address;
      ProtectionLevelTypeOpt     connectionProtectionLevelOpt;
      RoutingCriteriaTypeOpt     routingCriteriaOpt;
      DirectionalityType   directionality;
      ScheduleType         schedule;
      QoSSequenceTypeOpt   qosParametersOpt;
};

//_____________________ Type: ReserveConnectionResult ____________________________
```



```
/* Corresponding MISA-Xuser ASN.1 type definition:

ReserveGBCConnectionResult::= CHOICE {
      successful [0] SEQUENCE {
      gBCConnectionId   GBCConnectionId,
      gBCAccessPointId  GBCAccessPointId OPTIONAL},
      unsuccessfull [1] Reason
}
*/

union AccessPointIdTypeOpt switch(boolean) {
      case TRUE: NameType        accessPointId;
      default: ASN1_Null undefined;
};

struct ReserveConnectionResultType {
      AccessPointIdTypeOpt accessPointIdOpt;
      NameType             connectionId;
};

/*
NOTE: The type mapping only reflects the results returned on
successful operation of the reserveGBCConnection ACTION operation.
Failures will be indicated through an exception of type
ConnectionRequestFailure which is associated with the
reserveConnection operation.
*/

//_____________________ Type: ModifyConnectionInformation ____________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ModifyGBCConnectionInformation::= SEQUENCE {
      gBCMUserId Identifier,
      gBCConnectionId GBCConnectionId,
      gBCSchedule [0] GBCSchedule OPTIONAL,
      gBCPPSparameters [1] GBCQoSSequence OPTIONAL}
*/

struct  ModifyConnectionInfoType {
      IdentifierType       userId;
      NameType             connectionId;
      ScheduleTypeOpt      durationOpt;
      QoSSequenceTypeOpt   qosParametersOpt;
};

//_____________________ Type: ModifyConnectionResult _____________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ModifyGBCConnectionResult::= CHOICE {
      successful NULL,
      unsuccessfull Reason }
*/

/*
NOTE: No type definition is needed since the modifyGBCConnection ACTION operation
returns NULL on successful completion. Failures will be indicated through an
exception of type ConnectionRequestFailure which is associated with the
modifyConnection operation.
*/

//_____________________ Type: ReleaseConnectionInformation ____________________

/* Corresponding MISA-Xuser ASN.1 type definition:
```



```
ReleaseGBCConnectionInformation::= SEQUENCE {
    gBCMUserId Identifier,
    gBCConnectionId GBCConnectionId,
    reason  Reason }
*/

struct  ReleaseConnectionInfoType{
    IdentifierType        userId;
    NameType              connectionId;
    ReasonType     reason;
};

//_____________________ Type: ReleaseConnectionResult ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ReleaseGBCConnectionResult::= CHOICE {
    successful NULL,
    unsuccessfull Reason }
*/

/*
NOTE: No type definition is needed since the modifyGBCConnection ACTION operation
returns NULL on successful completion. Failures will be indicated through an
exception of type ConnectionRequestFailure which is associated with the
modifyConnection operation.
*/

//_____________________________ Type: ReleaseReason ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ReleaseReason::= CHOICE {
     fromPNO ENUMERATED {enduserRelease (0), timeout (1),
          pnoRelease (2), other (3) },
     fromOriginCPN PrintableString }
*/

enum ReleaseReasonfromPNOChoiceType {
     enduserRelease, timeout, pnoRelease, other
};

enum ReleaseReasonTypeChoice {
     fromPNOChoice, fromOriginCPNChoice
};

union ReleaseReasonType  switch(ReleaseReasonTypeChoice) {
     case fromPNOChoice: ReleaseReasonfromPNOChoiceType fromPNO;
     case fromOriginCPNChoice: ASN1_PrintableString fromOriginCPN;
};

//_____________________________ Type: ReleaseNotifInformation ____________________________

/* Corresponding MISA-Xuser ASN.1 type definition:

ReleaseNotifInformation::= ReleaseReason
*/

typedef ReleaseReasonType ReleaseNotifInfoType;

}; // End of Module

 #endif
```


9. APPENDIX B - CORBA/TMN GATEWAY LIBRARY

Different functions are offered by this library, allowing the gateway to handle all the requests sent by the VASP Control Server.

Some of these functions are available to allow a correct initialization with the HP-OV postmaster daemon:

FUNCTION initManager

SYNOPSIS

```
int initManager(void)
```

DESCRIPTION

initialization of all required HP OpenView data.

ARGUMENTS

∅

RESULTS

returns 0 if success, -1 if not.

FUNCTION exitManager

SYNOPSIS

```
void exitManager()
```

DESCRIPTION

release and free all data allocated for HP OpenView use**.**

ARGUMENTS

∅

RESULTS

∅

FUNCTION getManagerFileDescriptor

SYNOPSIS

```
int getManagerFileDescriptor()
```

DESCRIPTION

retrieves the file descriptor on HP OpenView postmaster daemon, in order to process asynchronous events coming from the Xuser agent.

ARGUMENTS

∅

RESULTS

On success, returns a filedescriptor value to the HPOV postmaster daemon..

In case of error, returns -1.

FUNCTION processEventFromAgent

SYNOPSIS

```
void processEventFromAgent()
```


DESCRIPTION

processing of an incoming event sent by the Xuser agent to the manager.

ARGUMENTS

∅

RESULTS

∅

The next functions are called in the implementation side of the CORBA part of the gateway. Precisely, each function is used appropriately in the methods of the class PnoConnectionMgImpl, when the VASP Control Server issues a request:

FUNCTION processReserveConnection

SYNOPSIS

```
XuserTypes::ReserveConnectionResultType *
processReserveConnection (const XuserTypes::ReserveConnectionInfoType
connectionInfo)
```

DESCRIPTION

processes a mapping from the C++ infornation given by the parameter to a C structure, and sends to the Xuser agent the ReserveConnection request with XOM/XMP facilities. After receiving a successful result, processes a mapping from the C Xuser agent result to a C++ gateway result and returns it. Then, the CORBA server side is able now to translate this in a required form, understandable by the CORBA client (the VASP Control Server written in Java).

ARGUMENTS

connectionInfo, composed of the following attributes:

- userId: User Id.
- sourceE164AddressOpt: optional source address.
- destinationE164Address: destination address.
- connectionProtectionLevelOpt: optional protection level.
- RoutingCriteriaOpt: optional routing criteria.
- Directionality: directionality.
- Schedule: schedule.
- qosParametersOpt: optional QOS parameters.

RESULTS

In case of success, returns a memory block containing all relevant information sent by the Xuser agent, after a ReserveConnection CMIP request, with the following attributes:

- accessPointIdOpt: optional access point Id.
- connectionId: connection Id.

In case of error, an exception is raised, of type ConnectionRequestFailure, and containing the reason of the failure.

FUNCTION processModifyConnection

SYNOPSIS

```
void processModifyConnection (const XuserTypes::ModifyConnectionInfoType
connectionInfo)
```

DESCRIPTION

process a mapping from the C++ information given by the parameter to a C structure, and sends to the Xuser agent the ModifyConnection request with XOM/XMP facilities. After receiving a successful result, does nothing else.

ARGUMENTS

connectionInfo, composed of the following attributes:

- userId: User Id.
- connectionId: connection Id.
- ScheduleOpt: optional schedule.
- qosParametersOpt: optional QOS parameters.

RESULTS

In case of success, nothing is returned. In case of error, an exception is raised, of type ConnectionRequestFailure, and containing the reason of the failure.

FUNCTION processReleaseConnection

SYNOPSIS

```
void processReleaseConnection (const
XuserTypes::ReleaseConnectionInfoType connectionInfo);
```

DESCRIPTION

process a mapping from the C++ information given by the parameter to a C structure, and sends to the Xuser agent the ReleaseConnection request with XOM/XMP facilities. After receiving a successful result, does nothing else.

ARGUMENTS

connectionInfo, composed of the following attributes:

- userId: User Id.
- connectionId: connection Id.

RESULTS

In case of success, nothing is returned. In case of error, an exception is raised, of type ConnectionRequestFailure, and containing the reason of the failure.

10. APPENDIX C - XUSER INTERFACE DEFINITION

This section contains the complete specification of the external interface to the TRUMPET PNO Xuser Agent. According to the collaboration between the TRUMPET and MISA project this specification has been adopted from MISA [MISA-D3-A1]. The GDMO definitions included below represent the latest revision of the MISA Xuser specification which has been used as a basis for the implementation of the PNO Xuser Agent and the VASP CORBA/TMN gateway in TRUMPET.

```
-- Filename/type = xuser.mib
-- Created automatically by MMF generator
--
-- Xuser Interface GDMO Specification
--
-- Editor: G. Liu, T. Zhang (GMD FOKUS)
--
-- DEPENDENCIES:
--      "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992"
--      "ITU-T Rec. M.3100 (1995)"
--      "I-ETS 300 653: 1996"
--      "ITU-T Rec. I.751 (1996)"
--      "ITU-T Rec. X.790 (1995)"
--
-- REVISION INFORMATION
--
-- Revision 2.3  1997/11/08 00:00:00  liu
--      Use the registration tree reserved for MISA Xuser
--      Change the syntax of gBCMUserId from INTEGER to M.3100 NameType
--      Enhance some NOTIFICATION definitions
--
-- Revision 2.2  1997/10/12 00:00:00  liu
--      Conclude the concrete class identifiers (integer) for QoSClass
--      Enhance some BEHAVIOUR definitions
--      Add an explicit directionality field for GBC reservation
--      Replace some text for more appropriate expressions
--
-- Revision 2.1  1997/09/26 00:00:00  liu
--      Modify definitions for GBC scheduling, GBC QoS parameters
--      and GBC Traffic parameters
--
-- Revision 2.0  1997/09/04 00:00:00  liu
--      Insert high level Probable Cause according to AC1 definitions
--      Add ReserveNotifReply for destination GBCMuser checking
--
-- Revision 1.6  1997/07/30 00:00:00  liu
--      Modify some definitions for Fault Management Ensemble
--      Modify some definitions for QoS issues
--
-- Revision 1.5  1997/02/17 00:00:00  zhang
--      Modify some BEHAVIOUR descriptions responding to Dieter's
--
-- Revision 1.4  1996/12/17 00:00:00  liu
--      Replace some identifiers, labels and references for
--      the globally unique identities in the MISA specifications.
--
-- Revision 1.3  1996/12/16 00:00:00  liu/zhang
--      Extent the functionalities to support the destination GUI etc.
--
-- Revision 1.2  1996/11/14 00:00:00  liu
--      Support the specification of RP1 interface which is out of
--      the scope of the Xuser Protocol itself.
--      This part should be struck off in the formal version of the
--      Xuser GDMO specification.
--
```



```
-- Revision 1.1  1996/10/14 00:00:00  liu
--
-- Revision 1.0  1996/09/30 00:00:00  liu
--
-- Revision 0.0  1996/09/27 00:00:00  liu
--      Construct this file from the MISA resource provided by zhang
--
--
-- OTHER CHANGES
-- Deviations from original:
-- 16 DEC 96: aro (ZRL)
-- - define statements added for OID resolution.
-- - changed all references to X.721, M.3100 to use the format shown above
--   under Dependancies.
-- - Added reference to "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992"
--   for attribute logRecordId, logId and class log, alarmRecord
-- - globally replaced gBCServiceProvider with gBCMServiceProvider

-- Index
--
        -- ASN.1 Module
        -- MANAGED OBJECT CLASS
        -- PACKAGE
        -- ATTRIBUTE
        -- ACTION
        -- NOTIFICATION
        -- BEHAVIOUR
        -- NAME BINDING

-- ************************************************************************
        -- MANAGED OBJECT CLASS
-- ************************************************************************

gBCServiceProvider MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":system;
        CHARACTERIZED BY
                gBCServiceProviderPackage,
                gBCMServicePerformerPackage;
REGISTERED AS {misaXuserObjectClass 1};

gBCMUser MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992": top;
        CHARACTERIZED BY
                gBCMUserPackage,
                gBCMReconfigurationPackage;
        CONDITIONAL PACKAGES
                gBCMTroubleReportFormatObjectPtrPkg
                        PRESENT IF "an instance supports it.";
REGISTERED AS {misaXuserObjectClass 2};

gBCMUserServiceProfile MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992": top;
        CHARACTERIZED BY
                gBCMUserServiceProfilePackage;
REGISTERED AS {misaXuserObjectClass 3};

gBCAccessPoint MANAGED OBJECT CLASS
        DERIVED FROM "I-ETS 300 653: 1996": networkTP;
        CHARACTERIZED BY
                gBCAccessPointPackage,
                gBCAccessPointAlarmReportPackage;
        CONDITIONAL PACKAGES
                gBCAccessPointNotifPackage
                        PRESENT IF "It is supported.";
REGISTERED AS {misaXuserObjectClass 4};
```



```
gBCConnection MANAGED OBJECT CLASS
        DERIVED FROM "I-ETS 300 653: 1996": connectivity;
        CHARACTERIZED BY
                gBCConnectionPackage,
                gBCAlarmReportPackage,
                "ITU-T Rec. M.3100 (1995)": createDeleteNotificationsPackage;
        CONDITIONAL PACKAGES
                gBCPPSParameterPackage
                        PRESENT IF "The GBCConnection has SDH/ATM specific
parameters";

REGISTERED AS {misaXuserObjectClass 5};

gBCTroubleReport MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.790 (1995)": troubleReport;
        CHARACTERIZED BY
                gBCTroubleReportPkg,
                "ITU-T Rec. X.790 (1995)": trObjectCreationDeletionPkg,
                "ITU-T Rec. X.790 (1995)": trAttributeValueChangePkg;
        CONDITIONAL PACKAGES
                "ITU-T Rec. X.790 (1995)": trTroubleClearancePersonAttributePkg
                        PRESENT IF "It is necessary to record the information",
                "ITU-T Rec. X.790 (1995)": trRelatedTroubleReportListPkg
                        PRESENT IF "One trouble has certain close relationship
with others";
REGISTERED AS {misaXuserObjectClass 6};

reconfigurationRecord MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":
eventLogRecord;
        CONDITIONAL PACKAGES
                "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":
eventTimePackage
                        PRESENT IF "The event time parameter was presenting in the
received RequiredReconfigurationNotification event report";
REGISTERED AS {misaXuserObjectClass  7};

troubleCreationRecord MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":
objectCreationRecord;
REGISTERED AS {misaXuserObjectClass 8};

troubleDeletionRecord MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":
objectDeletionRecord;
REGISTERED AS {misaXuserObjectClass  9};

gBCMTroubleNotificationRecord MANAGED OBJECT CLASS
        DERIVED FROM "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":
eventLogRecord;
        CHARACTERIZED BY gBCMTroubleNotificationRecordPkg PACKAGE
        ATTRIBUTES
                "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992": eventTime GET,
                "ITU-T Rec. X.790 (1995)": managedObjectInstance GET,
                "ITU-T Rec. X.790 (1995)": receivedTime        GET,
                "ITU-T Rec. X.790 (1995)": troubleFound        GET;;;
        CONDITIONAL PACKAGES gBCMTroubleTypePkg
                        PRESENT IF "an instance supports it.";
REGISTERED AS {misaXuserObjectClass 10};


gBCService MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.790 (1995)": service;
        CHARACTERIZED BY
                gBCServicePkg PACKAGE
                BEHAVIOUR
```



```
                        gBCServiceBehaviour BEHAVIOUR
                        DEFINED AS "gBCService is a class of managed objects
                        that represents the offerings from a GBC service
                        provider which supplies APPS(ATM Path Provisioning
                        Service) and/or SPPS(SDH Path Provisioning Service)
                        to one or more customers.";;
                ATTRIBUTES
                        gBCServiceDescription GET;;;
        CONDITIONAL PACKAGES
                gBCMTroubleReportFormatObjectPtrPkg
                        PRESENT IF "an instance supports it.";
REGISTERED AS {misaXuserObjectClass 11};

gBCTroubleReportFormatDefn MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.790 (1995)": troubleReportFormatDefn;
        CONDITIONAL PACKAGES
                gBCTroubleFormatPackage
                        PRESENT IF "an instance supports it.";
REGISTERED AS {misaXuserObjectClass 12};

gBCAlarmRecord MANAGED OBJECT CLASS
        DERIVED  FROM  "CCITT  Rec.  X.721  (1992)  |  ISO/IEC  10165-2:  1992":
alarmRecord;
        CONDITIONAL PACKAGES
                gBCAlarmTypePackage
                        PRESENT IF "an instance supports it.";
REGISTERED AS {misaXuserObjectClass 13};

-- ************************************************************************
        -- PACKAGE
-- ************************************************************************

gBCServiceProviderPackage PACKAGE
        BEHAVIOUR gBCServiceProviderBehaviour;
        ATTRIBUTES
                administrativeAddress GET-REPLACE;
REGISTERED AS {misaXuserPackage 1};

gBCMServicePerformerPackage PACKAGE
        BEHAVIOUR gBCMServicePerformerBehaviour;
        ACTIONS
                reserveGBCConnection,
                modifyGBCConnection,
                releaseGBCConnection,
                removeSubscription;
REGISTERED AS {misaXuserPackage 2};

gBCMUserPackage PACKAGE
        BEHAVIOUR gBCMUserBehaviour;
        ATTRIBUTES
                gBCMUserId GET,
                gBCMUserCategory GET-REPLACE,
                gBCMUserAdminAddress GET-REPLACE;
REGISTERED AS {misaXuserPackage 3};

gBCMUserServiceProfilePackage PACKAGE
        BEHAVIOUR gBCMUserServiceProfileBehaviour;
        ATTRIBUTES
                serviceProfileId GET,
                gBCMServiceType GET-REPLACE;
REGISTERED AS {misaXuserPackage 4};

gBCAccessPointPackage PACKAGE
        BEHAVIOUR gBCAccessPointBehaviour;
        ATTRIBUTES
                gBCAccessPointId GET,
```



```
                e164Address GET,
                gBCConnectionPtr GET,
                serviceProfilePtr GET-REPLACE,
                qoSLimitsSequence GET-REPLACE;
REGISTERED AS {misaXuserPackage 5};

gBCConnectionPackage PACKAGE
        BEHAVIOUR gBCConnectionBehaviour;
        ATTRIBUTES
                gBCSchedule GET-REPLACE,
                connectionProtectionLevel GET-REPLACE,
                routingCriteria GET-REPLACE,
                gBCConnectionId GET,
                gBCAccessPointPtr GET,
                listOfDestAddr GET-REPLACE;
        NOTIFICATIONS
                activationNotif,
                deactivationNotif,
                releaseNotif,
                modifyGBCConnectionNotif;
REGISTERED AS {misaXuserPackage 6};

gBCPPSParameterPackage PACKAGE
        BEHAVIOUR gBCPPSParameterBehaviour;
        ATTRIBUTES
                gBCQoSSequence GET-REPLACE;
REGISTERED AS {misaXuserPackage 7};

gBCMReconfigurationPackage PACKAGE
        BEHAVIOUR
                gBCMReconfigurationBehaviour BEHAVIOUR
                         DEFINED AS "In case of requiring reconfiguration on the
GBCM user's side due to the fault affecting the managed resources, a
requiredReconfigurationNotif is emitted to indicate the essential reconfiguration.
";;
        NOTIFICATIONS
                requiredReconfigurationNotif;
REGISTERED AS {misaXuserPackage 8};

gBCTroubleFormatPackage PACKAGE
        BEHAVIOUR
                gBCMTroubleFormatBehaviour BEHAVIOUR
                        DEFINED AS "In this package, there is only one attribute
troubleReportFormat which contains the templates for trouble reporting. Its value
can be updated by the manager.";;
        ATTRIBUTES
                troubleReportFormat     GET-REPLACE;
REGISTERED AS {misaXuserPackage 9};

gBCAccessPointAlarmReportPackage PACKAGE
        BEHAVIOUR
                gBCAccessPointAlarmReportBehaviour BEHAVIOUR
                         DEFINED AS "This package only contains one notification.
The emitting of this notification is as a consequence of faults reported by the
MISA network level OS regarding the GBC access point. ";;
        NOTIFICATIONS
                gBCAccessPointAlarmReportNotif;
REGISTERED AS {misaXuserPackage 10};

gBCAlarmReportPackage PACKAGE
        BEHAVIOUR
                gBCAlarmReportBehaviour BEHAVIOUR
                DEFINED AS "This package only contains one notification. The
emitting of this notification is as a sonsequence of faults reported by the MISA
netwrk level OS regarding the GBC";;
        NOTIFICATIONS
```



```
                gBCAlarmReportNotif;
REGISTERED AS {misaXuserPackage 11};

gBCTroubleReportPkg PACKAGE
        BEHAVIOUR
                gBCTroubleReportBehaviour BEHAVIOUR
                DEFINED AS "The gBCTroubleReport managed object represents the
problem detected by the GBCM user and reported to the GBCM provider. Instances of
this class describe the nature of the problem as well as ongoing status.  ";;
        NOTIFICATIONS
                troubleNotif;
REGISTERED AS {misaXuserPackage 12};

gBCAccessPointNotifPackage PACKAGE
        BEHAVIOUR
                gBCAccessPointNotifBehaviour BEHAVIOUR
                DEFINED AS "In case a GBC connection is reserved at the
destination access point, the related GBCMUser should informed.";;
        NOTIFICATIONS
                reserveGBCConnectionNotif;
REGISTERED AS {misaXuserPackage 13};

gBCMTroubleReportFormatObjectPtrPkg PACKAGE
        ATTRIBUTES
                "ITU-T Rec. X.790 (1995)": troubleReportFormatObjectPtr GET;
REGISTERED AS {misaXuserPackage 14};

gBCMTroubleTypePkg PACKAGE
        ATTRIBUTES
                "ITU-T Rec. X.790 (1995)": troubleType GET;
REGISTERED AS {misaXuserPackage 15};

gBCAlarmTypePackage PACKAGE
        BEHAVIOUR gBCAlarmTypeBehaviour
                BEHAVIOUR DEFINED AS "The gBCAlarmType indicate the type
                of GBC alarm.";;
        ATTRIBUTES
                gBCAlarmType GET;
REGISTERED AS {misaXuserPackage 16};

-- ************************************************************************
        -- ATTRIBUTE
-- ************************************************************************

administrativeAddress ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.AdministrativeAddress;
        MATCHES FOR EQUALITY;
        BEHAVIOUR administrativeAddressBehaviour;
REGISTERED AS {misaXuserAttribute 1};

connectionProtectionLevel ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.ProtectionLevel;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR connectionProtectionLevelBehaviour;
REGISTERED AS {misaXuserAttribute 2};

e164Address ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.E164Address;
        MATCHES FOR EQUALITY;
        BEHAVIOUR e164AddressBehaviour;
REGISTERED AS {misaXuserAttribute 3};

gBCAccessPointId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCAccessPointId;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCAccessPointIdBehaviour;
```



```
REGISTERED AS {misaXuserAttribute 4};

gBCAccessPointPtr ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCAccessPointPtr;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCAccessPointPtrBehaviour;
REGISTERED AS {misaXuserAttribute 5};

gBCConnectionId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCConnectionId;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCConnectionIdBehaviour;
REGISTERED AS {misaXuserAttribute 6};

gBCConnectionPtr ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCConnectionPtr;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCConnectionPtrBehaviour;
REGISTERED AS {misaXuserAttribute 7};

gBCMServiceType ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCMServiceType;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCMServiceTypeBehaviour;
REGISTERED AS {misaXuserAttribute 8};

gBCMUserAdminAddress ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.AdministrativeAddress;
        MATCHES FOR EQUALITY;
        BEHAVIOUR administrativeAddressBehaviour;
REGISTERED AS {misaXuserAttribute 9};

gBCMUserCategory ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCMUserCategory;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCMUserCategoryBehaviour;
REGISTERED AS {misaXuserAttribute 10};

gBCMUserId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCMUserId;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCMUserIdBehaviour;
REGISTERED AS {misaXuserAttribute 11};

gBCQoSSequence ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCQoSSequence;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCQoSSequenceBehaviour;
REGISTERED AS {misaXuserAttribute 12};

listOfDestAddr ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.ListOfDestAddr;
        MATCHES FOR EQUALITY;
        BEHAVIOUR listOfDestAddrBehaviour;
REGISTERED AS {misaXuserAttribute 13};

qoSLimitsSequence ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.QoSLimitsSequence;
        MATCHES FOR EQUALITY, ORDERING;
        BEHAVIOUR qoSLimitsSequenceBehaviour;
REGISTERED AS {misaXuserAttribute 14};

gBCSchedule ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCSchedule;
        MATCHES FOR EQUALITY;
        BEHAVIOUR gBCScheduleBehaviour;
```



```
REGISTERED AS {misaXuserAttribute 15};

routingCriteria ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.RoutingCriteria;
        MATCHES FOR EQUALITY;
        BEHAVIOUR routingCriteriaBehaviour;
REGISTERED AS {misaXuserAttribute 16};

serviceProfileId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.ServiceProfileId;
        MATCHES FOR EQUALITY;
        BEHAVIOUR serviceProfileIdBehaviour;
REGISTERED AS {misaXuserAttribute 17};

serviceProfilePtr ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.ServiceProfilePtr;
        MATCHES FOR EQUALITY;
        BEHAVIOUR serviceProfilePtrBehaviour;
REGISTERED AS {misaXuserAttribute 18};

troubleReportFormat ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.TroubleReportFormat;
        MATCHES FOR EQUALITY;
REGISTERED AS {misaXuserAttribute 19};

gBCServiceDescription ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCServiceDescription;
        BEHAVIOUR gBCServiceDescriptionBehaviour
                BEHAVIOUR DEFINED AS "The gBCServiceDescription describes
                the distinguishing characteristics of a specific GBC service
                provided by the GBC service provider. They are what a service
                provider can serve its customers";;
REGISTERED AS {misaXuserAttribute 26};

gBCAlarmType ATTRIBUTE
        WITH ATTRIBUTE SYNTAX MisaXuserASN1Module.GBCAlarmType;
REGISTERED AS {misaXuserAttribute 27};

-- ************************************************************************
        -- ACTION
-- ************************************************************************

reserveGBCConnection ACTION
        BEHAVIOUR reserveGBCConnectionBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX
        MisaXuserASN1Module.ReserveGBCConnectionInformation;
        WITH REPLY SYNTAX MisaXuserASN1Module.ReserveGBCConnectionResult;
REGISTERED AS {misaXuserAction 1};

modifyGBCConnection ACTION
        BEHAVIOUR modifyGBCConnectionBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX
        MisaXuserASN1Module.ModifyGBCConnectionInformation;
        WITH REPLY SYNTAX MisaXuserASN1Module.ModifyGBCConnectionResult;
REGISTERED AS {misaXuserAction 2};

releaseGBCConnection ACTION
        BEHAVIOUR releaseGBCConnectionBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX
        MisaXuserASN1Module.ReleaseGBCConnectionInformation;
        WITH REPLY SYNTAX MisaXuserASN1Module.ReleaseGBCConnectionResult;
REGISTERED AS {misaXuserAction 3};
```



```
removeSubscription ACTION
        BEHAVIOUR removeSubscriptionBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX MisaXuserASN1Module.RemoveSubscriptionInformation;
        WITH REPLY SYNTAX MisaXuserASN1Module.RemoveSubscriptionResult;
REGISTERED AS {misaXuserAction 4};

-- ************************************************************************
        -- NOTIFICATION
-- ************************************************************************

activationNotif NOTIFICATION
        BEHAVIOUR activationNotifBehaviour;
        WITH INFORMATION SYNTAX MisaXuserASN1Module.ActivationNotifInformation;
REGISTERED AS {misaXuserNotification 1};

releaseNotif NOTIFICATION
        BEHAVIOUR releaseNotifBehaviour;
        WITH INFORMATION SYNTAX MisaXuserASN1Module.ReleaseNotifInformation;
REGISTERED AS {misaXuserNotification 2};

gBCAlarmReportNotif NOTIFICATION
        BEHAVIOUR
                gBCAlarmReportNotifBehaviour BEHAVIOUR
                 DEFINED AS "Any failure and warnings which affect GBC Connection
are triggered conditions of this notification. The alarm over and protection
switching is also a triggered conditions of this notification.
                 At the Xuser interface the following high level ProbableCause(s)
are defined:
                - interDomainLinkInavailable is used when the primary alarm
affects an interdomain access point.
                - foreignConnectionInavailable is used when primary alarm affects
a GBCSubnetworkConnection in a different domain to the one to which the user
access point is located.
                - localConnectionInavailable is used when primary alarm affects a
GBCSubnetworkConnection in the domain where the user access point is located.
                - networkFault is used when PNO wants to be as much generic as
possible, e.g. because the real primary alarm has not been detected (just AIS
and/or RDI defects have been received).";;
        WITH INFORMATION SYNTAX
                MisaXuserASN1Module.GBCAlarmReportNotifInfo
        AND ATTRIBUTE IDS
                typeOfAlarm             gBCAlarmType,
                 probableCause           "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": probableCause,
                 perceivedSeverity       "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": perceivedSeverity,
                 notificationIdentifier  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": notificationIdentifier,
                 specificProblems        "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": specificProblems,
                 correlatedNotifications "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": correlatedNotifications,
                monitoredAttributes     "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": monitoredAttributes,
                 proposedRepairActions   "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": proposedRepairActions,
                 additionalText          "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": additionalText;
REGISTERED AS {misaXuserNotification 3};

gBCAccessPointAlarmReportNotif NOTIFICATION
        BEHAVIOUR
                gBCAccessPointAlarmReportNotifBehaviour BEHAVIOUR
```



```
                DEFINED AS "Any failure and warnings which affect GBC Access Point
are triggered conditions of this notification. The alarm over is also a triggered
conditions of this notification.";;
        WITH INFORMATION SYNTAX
                MisaXuserASN1Module.GBCAlarmReportNotifInfo
        AND ATTRIBUTE IDS
                typeOfAlarm             gBCAlarmType,
                 probableCause           "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": probableCause,
                 perceivedSeverity       "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": perceivedSeverity,
                 notificationIdentifier  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": notificationIdentifier,
                 specificProblems        "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": specificProblems,
                 correlatedNotifications "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": correlatedNotifications,
                 monitoredAttributes     "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": monitoredAttributes,
                 proposedRepairActions   "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": proposedRepairActions,
                 additionalText          "CCITT Rec. X.721 (1992) | ISO/IEC 10165-
2: 1992": additionalText;
REGISTERED AS {misaXuserNotification 4};

requiredReconfigurationNotif NOTIFICATION
        BEHAVIOUR requiredReconfigurationNotifBehaviour;
        WITH INFORMATION SYNTAX
                MisaXuserASN1Module.RequiredReconfigurationNotifInfo;
REGISTERED AS {misaXuserNotification 5};

troubleNotif NOTIFICATION
        BEHAVIOUR troubleNotifBehaviour;
        WITH INFORMATION SYNTAX
                MisaXuserASN1Module.TroubleNotifInfo
        AND ATTRIBUTE IDS
                managedObjectInstance           "ITU-T    Rec.    X.790    (1995)":
managedObjectInstance,
                receivedTime            "ITU-T Rec. X.790 (1995)": receivedTime,
                troubleFound            "ITU-T Rec. X.790 (1995)": troubleFound,
                troubleType             "ITU-T Rec. X.790 (1995)": troubleType;
REGISTERED AS {misaXuserNotification 6};

reserveGBCConnectionNotif NOTIFICATION
        BEHAVIOUR
                reserveGBCNotifBehaviour BEHAVIOUR
                DEFINED   AS   "To   notify   the   destination   GBCMUser   of   the
reservation.";;
        WITH                            INFORMATION                             SYNTAX
MisaXuserASN1Module.ReserveGBCConnectionNotifInformation;
        WITH REPLY SYNTAX MisaXuserASN1Module.ReserveNotifReply;
REGISTERED AS {misaXuserNotification 7};

modifyGBCConnectionNotif NOTIFICATION
        BEHAVIOUR
                modifyGBCNotifBehaviour BEHAVIOUR
                DEFINED AS "To notify the modification of GBC connection.";;
        WITH                            INFORMATION                             SYNTAX
MisaXuserASN1Module.ModifyGBCConnectionNotifInformation;
REGISTERED AS {misaXuserNotification 8};

deactivationNotif NOTIFICATION
        BEHAVIOUR
                deactivationNotifBehaviour BEHAVIOUR
                DEFINED   AS   "To   notify   the   deactivation   of   the   connections
according to the schedule.";;
```



```
        WITH INFORMATION SYNTAX MisaXuserASN1Module.DeactivationNotifInformation;
REGISTERED AS {misaXuserNotification 9};

-- ************************************************************************
        -- NAME BINDING
-- ************************************************************************

gBCMUser-gBCServiceProvider NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCMUser;
        NAMED BY
           SUPERIOR OBJECT CLASS "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":system AND SUBCLASSES;
        WITH ATTRIBUTE gBCMUserId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 1};

gBCAccessPoint-gBCMUser NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCAccessPoint;
        NAMED BY
           SUPERIOR OBJECT CLASS gBCMUser;
        WITH ATTRIBUTE gBCAccessPointId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 2};

gBCMUserServiceProfile-gBCMUser NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCMUserServiceProfile;
        NAMED BY
           SUPERIOR OBJECT CLASS gBCMUser;
        WITH ATTRIBUTE serviceProfileId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 3};

gBCConnection-gBCMUser NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCConnection;
        NAMED BY
           SUPERIOR OBJECT CLASS gBCMUser;
        WITH ATTRIBUTE gBCConnectionId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 4};

gBCTroubleReport-gBCMUser NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCTroubleReport;
        NAMED BY
           SUPERIOR OBJECT CLASS gBCMUser;
        WITH ATTRIBUTE "ITU-T Rec. X.790 (1995)": troubleReportID;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 5};

log-gBCServiceProvider NAME BINDING
          SUBORDINATE OBJECT CLASS    "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":log AND SUBCLASSES;
```



```
        NAMED BY
           SUPERIOR OBJECT CLASS  gBCServiceProvider AND SUBCLASSES;
        WITH ATTRIBUTE "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2: 1992":logId;
REGISTERED AS {misaXuserNameBinding 6};

gBCAlarmRecord-log NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCAlarmRecord;
        NAMED BY
              SUPERIOR OBJECT CLASS  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":log AND SUBCLASSES;
        WITH   ATTRIBUTE   "CCITT   Rec.   X.721   (1992)   |   ISO/IEC   10165-2:
1992":logRecordId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 7};

reconfigurationRecord-log NAME BINDING
        SUBORDINATE OBJECT CLASS   reconfigurationRecord;
        NAMED BY
              SUPERIOR OBJECT CLASS  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":log AND SUBCLASSES;
        WITH   ATTRIBUTE   "CCITT   Rec.   X.721   (1992)   |   ISO/IEC   10165-2:
1992":logRecordId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 8};

troubleCreationRecord-log NAME BINDING
        SUBORDINATE OBJECT CLASS   troubleCreationRecord;
        NAMED BY
              SUPERIOR OBJECT CLASS  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":log AND SUBCLASSES;
        WITH   ATTRIBUTE   "CCITT   Rec.   X.721   (1992)   |   ISO/IEC   10165-2:
1992":logRecordId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 9};

troubleDeletionRecord-log NAME BINDING
        SUBORDINATE OBJECT CLASS   troubleDeletionRecord;
        NAMED BY
           SUPERIOR  OBJECT  CLASS  "CCITT  Rec.  X.721  (1992)  |  ISO/IEC  10165-2:
1992":log AND SUBCLASSES;
        WITH   ATTRIBUTE   "CCITT   Rec.   X.721   (1992)   |   ISO/IEC   10165-2:
1992":logRecordId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 10};

gBCMTroubleNotificationRecord-log NAME BINDING
        SUBORDINATE OBJECT CLASS   gBCMTroubleNotificationRecord;
        NAMED BY
           SUPERIOR  OBJECT  CLASS  "CCITT  Rec.  X.721  (1992)  |  ISO/IEC  10165-2:
1992":log  AND SUBCLASSES;
        WITH   ATTRIBUTE   "CCITT   Rec.   X.721   (1992)   |   ISO/IEC   10165-2:
1992":logRecordId;
        CREATE
           WITH-AUTOMATIC-INSTANCE-NAMING;
```



```
        DELETE
           ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {misaXuserNameBinding 11};

gBCService-gBCServiceProvider NAME BINDING
        SUBORDINATE OBJECT CLASS gBCService;
        NAMED BY
           SUPERIOR OBJECT CLASS "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":system AND SUBCLASSES;
        WITH ATTRIBUTE "ITU-T Rec. X.790 (1995)":serviceID;
REGISTERED AS {misaXuserNameBinding 12};

gBCTroubleReportFormatDefn-gBCServiceProvider NAME BINDING
        SUBORDINATE OBJECT CLASS gBCTroubleReportFormatDefn AND SUBCLASSES;
        NAMED BY
           SUPERIOR OBJECT CLASS "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:
1992":system AND SUBCLASSES;
        WITH ATTRIBUTE "ITU-T Rec. X.790 (1995)":tRFormatID;
REGISTERED AS {misaXuserNameBinding 13};

-- ************************************************************************
        -- BEHAVIOUR
-- ************************************************************************

activationNotifBehaviour BEHAVIOUR
        DEFINED AS "This notification is issued by the gBCConnection MO, to
indicate
         all the involved GBCM Users that the activation of the VP Connection took
place.";

administrativeAddressBehaviour BEHAVIOUR
        DEFINED AS "The administrative address of an organisation.";

connectionProtectionLevelBehaviour BEHAVIOUR
        DEFINED AS "This attribute specifies the protection level of the GBC
connection. This
        information will be used in fault management/recovery.";

e164AddressBehaviour BEHAVIOUR
          DEFINED AS " It represents the address assigned to a particular access
point";

gBCAccessPointBehaviour BEHAVIOUR
         DEFINED AS "represents the network access point at which a GBC Connection
enters in the public
          network from the GBCM User's domain (e.g. from a CTN) or enters in the
GBCM User's domain
        from the public network. To each gBCAccessPoint is associated a particular
E.164 address.
        The GBC connections passing through a GBCAccessPoint may reach the
terminals of many end-
        users within e.g. a CTN.
        Let's consider for example a company having in its private network an ATM
MUX, through which
        it puts and receives ATM traffic to/from the public network.
        Let's suppose that the ATM MUX has an SDH physical interface and generates
ATM traffic  over
         SDH after multiplexing different traffic sources (IP over AAL5, pure ATM
etc.)  generated in the local
        private network.
         Many end-users inside the company will need GBC connections for their
applications (e.g. data
        transfer at high bit rate etc..). They express their needs to the manager
of the private
         network who (playing the role of GBCM User) associates to its (previously
created) APPS
```



```
          subscription the E.164 address of the ATM MUX physical interface. This
will be performed in
        practice by requesting the creation of a gBCAccessPoint instance and
specifying the E.164
        address and all the other parameters relevant for that interface (e.g. Max
number of Vps, max
        bandwidth...etc).
          In the case in which the GBCM User is a VASP, third party connection
reservation can be
        supported in the following way.
         The VASP subscribes itself as GBCM User of e.g. the APPS service (this is
realised by creating a
          GBCM User object instance and a GBCMUser-ServiceProfile object instance
that has ServiceType value =
         APPS). Then the VASP declares, in the list of gBCAccessPoints associated
to that APPS,
        and the E.164 addresses.
         When the VASP will request the reservation of a VP connection having as
source E.164 address the
          above address and having the time schedule, bandwidth parameters ..etc
originally requested to the VASP
          by the VASP user, all will happen (from the point of view of the GBCM
Service Provider) as if
        no third party reservation apply.";

gBCAccessPointPtrBehaviour BEHAVIOUR
          DEFINED AS "It points to the registered gBCAccessPoint(s) to which the
connection is
        associated.";

gBCAccessPointIdBehaviour BEHAVIOUR
        DEFINED AS "It identifies a GBC access point.";

gBCConnectionBehaviour BEHAVIOUR
        DEFINED AS "The gBCMConnection object class represents the GBC global
broadband
        connection that is managed through the GBCM-Xuser interface.";

gBCConnectionIdBehaviour BEHAVIOUR
          DEFINED AS "It identifies the GBC connection. In our context this ID is
envisaged
        global significant. E.g. it can contain several subnetwork IDs, the access
point ID,
        VPI/VCI and the GBCM user ID to identify the connection and to provide
enough
        information for processing the configuration.";

gBCConnectionPtrBehaviour BEHAVIOUR
          DEFINED AS "It points to the GBC Connections terminated at this access
point. His value is      set by
the GBCM Service Provider.";
gBCQoSSequenceBehaviour BEHAVIOUR
        DEFINED AS "The QoS tuple for GBCM connection/service.";

gBCMServicePerformerBehaviour BEHAVIOUR
          DEFINED AS "It supports the GBCM actions that can be applied to a GBC
Connection
        through the GBCM Provider domain.";

gBCServiceProviderBehaviour BEHAVIOUR
         DEFINED AS "It contains organizational information about the GBCM Service
Provider
        and its service characteristics.";

gBCMServiceTypeBehaviour BEHAVIOUR
        DEFINED AS "It identifies the GBCM service supported, either SDH or ATM.";
```



```
gBCMUserBehaviour BEHAVIOUR
          DEFINED AS "represents the entity which, as representative of a set of
end-users, is responsible for all
the direct interactions (through the Xuser) with the PNO Service Provider's MISA
OS. Thus it is envisaged that
an instance of such resource has to be created for each of the management systems
interacting with the MISA OS
through the Xuser interface. As a consequence, this could imply, in the case of a
Customer Telecommunication
Network (CTN) composed by different sites each with its own Service Level OS, that
as many GBCM Users
should be instantiated as the number of CTN sites. Naturally one more resource is
needed to contain all the
information that require to be stored in a centralized location (e.g. overall
billing for all the CTN). Being
accounting management out of the scope of MISA, it is left open the task to
identify, define and insert in our
Xuser information model all the additional resources (e.g. customer) that would
permit to extend our Xuser
specification to cover also the accounting issues.
The GBCMUser represents also the entity which subscribes to the GBCM Service and
thus plays both the roles
defined in the Eurescom Xuser: the manager role and the subscriber role.";

gBCMUserCategoryBehaviour BEHAVIOUR
         DEFINED AS "It identifies the category (normal or privileged) of the GBCM
User.";

gBCMUserIdBehaviour BEHAVIOUR
         DEFINED AS "It represents the unique identifier assigned by the GBCM
ServiceProvider to the
subscription of a GBCM User. It is returned to the GBCM User as positive result of
his request to create a
GBCMUser instance.";

gBCPPSParameterBehaviour BEHAVIOUR
         DEFINED AS "It contains the QoS parameters that specify the GBC
connection. As a MISA
          GBC connection will run through different domains with different sets of
QoS parameters,
         we envisage a sequence/tuple of such sets in the GBC connection
description.
         The major reason for using a tuple of QoS descriptions is that many
parameters, e.g.
         ATM parameters, must  transferred  transparent through a SDH domain.
Moreover, mapping
          of QoS parameters between layers can not be done in all the cases without
lost of       information.
         Therefore, it is not enough to have only one set of QoS parameters.";

listOfDestAddrBehaviour BEHAVIOUR
         DEFINED AS "indicates the destination addresses of the GBC connection.";

modifyGBCConnectionBehaviour BEHAVIOUR
           DEFINED AS "This action is performed by the GBCM User requesting the
modification of    the
GBC connection. In case of SPPS (SDH), it is possible that modification is not
supported.       In this case the
action request will be rejected.";

qoSLimitsSequenceBehaviour BEHAVIOUR
          DEFINED AS "contains two set of parameters (related to the GBC Access
Point)    representing the
limits of the QoS in the case of APPS or SPPS respectively.
```



```
        In the case of a gBCAccessPoint associated to a subscription to APPS, the
following     parameters may
be          specified:          MaxNumVPs,          ForwardUpperLimitPeackCellRate,
BackwardUpperLimitPeackCellRate.
        In the case of a gBCAccessPoint associated to a subscription to SPPS, the
following     attributes may
be specified: ForwardUpperLimitPeackBitRate,    BackwardUpperLimitPeackBitRate..";

releaseGBCConnectionBehaviour BEHAVIOUR
          DEFINED AS "This action is performed by the GBCM User requesting the
clearing down of   the
GBC Connection. This will delete the gBCConnection object instance.";

releaseNotifBehaviour BEHAVIOUR
        DEFINED AS "This notification is issued by the gBCConnection MO, to
indicate to the     involved
GBCM Users, that the GBC Connection has been released.";

removeSubscriptionBehaviour BEHAVIOUR
         DEFINED AS "This action is issued by the GBCM Service Provider when he
wants to stop    the
subscription to the GBCM Service. This action implies the deletion of all the
managed   object instances
(gBCMUserServiceProfile, gBCAccessPoint, gBCConnection, gBCTroubleReport) having
that
GBCMUserId (identifier of a specific subscription) in their Distinguished Name." ;

gBCScheduleBehaviour BEHAVIOUR
        DEFINED AS "It specifies at which time the GBC connection should be
activated and
optionally at what time the reserved GBC connection should be deactivated or
released.";

reserveGBCConnectionBehaviour BEHAVIOUR
        DEFINED AS "This action is performed by the GBCM User which requests a GBC
connection
reservation
        from the GBCM Service Provider. The result of this action is the
acceptance or reject of the
         connection reservation request (regarding the start time, the stop time
and eventually the
        periodicity requested). If the connection reservation is rejected, the
reason is returned
        (not available resources, not possible  in the interval time,...). If the
connection reservation
        is accepted, a gBCConnection object instance is created.";

routingCriteriaBehaviour BEHAVIOUR
        DEFINED AS "It specifies the routing criteria for GBC connection
establishment.";

gBCMUserServiceProfileBehaviour BEHAVIOUR
        DEFINED AS "It represents the service profile associated to the GBCMUser's
subscription. Besides the indication of the particular GBCM service instance (APPS
or SPPS) the GBCM User has subscribed to, it may contain other service related
information (e.g. billing rates, and what a GBC service customer get from the
service provider). A GBCM User having the interest to subscribe to both APPS and
SPPS services, will have two GBCMUser-ServiceProfiles associated to the same
subscription (i.e. to the same GBCMUser instance).";

serviceProfileIdBehaviour BEHAVIOUR
         DEFINED AS "It identifies a GBCM User's service profile. Its value is
assigned by the   GBCM Service
Provider when the GBCM User has requested the creation of his service   profile ";

serviceProfilePtrBehaviour BEHAVIOUR
```



```
         DEFINED AS " This attribute points to the GBCM User's service profile to
which the GBC  access
point is associated upon subscription.";

requiredReconfigurationNotifBehaviour BEHAVIOUR
        DEFINED AS "behaviour to be defined";

troubleNotifBehaviour BEHAVIOUR
        DEFINED AS "behaviour to be defined";

-- Deviations from original:
-- 16 DEC 96: aro (ZRL)
-- - replaced AtmMIBMod with ATMForumASN1Module
-- - replaced ASN1TypeModule with PR-ETS300469

-- ************************************************************************
        -- ASN.1 Module
-- ************************************************************************

MisaXuserASN1Module  {joint-iso-ccitt(2)  country(16)  ch(756)  apps(5)  misa(7)
xuser(3) informationModel(0) asn1Module(0)} DEFINITIONS IMPLICIT TAGS::=

BEGIN

-- exports everything

IMPORTS

        CharacteristicInformation,
        UserLabel,
        Directionality,
        NameType
                FROM  ASN1DefinedTypesModule  {ccitt(0)  recommendation(0)  m(13)
gnm(3100) informationModel(0) asn1Modules(2) asn1DefinedTypesModule(0)}

        Attribute,
        ObjectClass,
        ObjectInstance
                FROM   CMIP-1   {joint-iso-ccitt(2)   ms(9)   cmip(1)   modules(0)
protocol(3)}

        NotificationIdentifier,
        SpecificProblems,
        CorrelatedNotifications,
        ProposedRepairActions,
        SystemId,
        AdministrativeState,
        AvailabilityStatus,
        OperationalState,
        ProbableCause,
        MonitoredAttributes,
        PerceivedSeverity,
        AdditionalText,
        Time24,
        StopTime
                FROM   Attribute-ASN1Module   {joint-iso-ccitt(2)   ms(9)   smi(3)
part2(2) asn1Module(2) 1}

        NamingString,
        PremisesName,
        ReceivedTime,
        TroubleFound,
        TroubleLocation,
        TroubleType
                FROM  X790ASN1Module  {itu-t(0)  recommendation(0)  x(24)  x790(790)
informationModel(0) asn1module(2)}
```



```
        AssignmentState,
        LifecycleState,
        Mode,
        TimeWeek,
        TimeMonth,
        StartTime
                FROM  I-ETS300653  {ccitt(0)  identified-organization(4)  etsi(0)
ets(653) informationModel(0) asn1Module(2) i-ets300653(0)}

        PeakCellRate,
        CDVTolerance,
        MaxBurstSize,
        SustainableCellRate
                FROM  AtmMIBMod  {itu-t(0)  recommendation(0)  i(9)  atmm(751)
informationModel(0) asn1Module(2) atm(0)};

misaXuserInfoModel OBJECT IDENTIFIER::= {joint-iso-ccitt(2) country(16) ch(756)
apps(5) misa(7) xuser(3) informationModel(0)}

misaXuserSpecificExtention      OBJECT      IDENTIFIER::=      {misaXuserInfoModel
specificExtention(1) }

misaCharacteristicInfo OBJECT IDENTIFIER::= {misaXuserSpecificExtention 0}

atmoverE1pdh CharacteristicInformation::= {misaCharacteristicInfo 1}
atmoverE2pdh CharacteristicInformation::= {misaCharacteristicInfo 2}
atmoverE3pdh CharacteristicInformation::= {misaCharacteristicInfo 3}
atmoverVC4sdh CharacteristicInformation::= {misaCharacteristicInfo 4}
atmoverVC3sdh CharacteristicInformation::= {misaCharacteristicInfo 5}
atmoverVC2sdh CharacteristicInformation::= {misaCharacteristicInfo 6}
atmoverVC12sdh CharacteristicInformation::= {misaCharacteristicInfo 7}
opticalSTM1SPICI CharacteristicInformation::= {misaCharacteristicInfo 8}
opticalSTM4SPICI CharacteristicInformation::= {misaCharacteristicInfo 9}
opticalSTM16SPICI CharacteristicInformation::= {misaCharacteristicInfo 10}

subnetworkConnectionProtectionSwitchingFailure  ProbableCause::= localValue: 1
interDomainLinkProtectionSwitchingFailure       ProbableCause::= localValue: 2
aPPSinavailable ProbableCause::= localValue: 3
sPPSinavailable ProbableCause::= localValue: 4
degrationOfQos  ProbableCause::= localValue: 5
configurationError      ProbableCause::= localValue: 6
thresholdCrossed        ProbableCause::= localValue: 7
subnetworkConnectionProtectionSwitching ProbableCause::= localValue: 8
interDomainLinkProtectionSwitching      ProbableCause::= localValue: 9
faultCleared    ProbableCause::= localValue: 10
interDomainLinkInavailable      ProbableCause::= localValue: 11
foreignConnectionInavailable    ProbableCause::= localValue: 12
localConnectionInavailable      ProbableCause::= localValue: 13
networkFault    ProbableCause::= localValue: 14

misaXuserObjectClass        OBJECT        IDENTIFIER::=        {misaXuserInfoModel
managedObjectClass(3)}

misaXuserPackage OBJECT IDENTIFIER::= {misaXuserInfoModel package(4)}

misaXuserParameter OBJECT IDENTIFIER::= {misaXuserInfoModel paramter(5)}

misaXuserNameBinding OBJECT IDENTIFIER::= {misaXuserInfoModel nameBinding(6)}

misaXuserAttribute OBJECT IDENTIFIER::= {misaXuserInfoModel attribute(7)}

misaXuserAttributeGroup       OBJECT       IDENTIFIER::=       {misaXuserInfoModel
attributeGroup(8)}

misaXuserAction OBJECT IDENTIFIER::= {misaXuserInfoModel action(9)}
```



```
misaXuserNotification OBJECT IDENTIFIER::= {misaXuserInfoModel notification(10)}

AdministrativeAddress::= SEQUENCE {
        contactname [0] PrintableString,
        address [1] PrintableString,
        e-mail [2] PrintableString OPTIONAL,
        phone [3] PrintableString OPTIONAL,
        fax [4] PrintableString OPTIONAL }

E164Address::= PrintableString

GBCAccessPointId::= NameType

GBCAccessPointPtr::= CHOICE {
        singleTermination ObjectInstance,
        multipleTermination SET OF ObjectInstance}

GBCConnectionId::= NameType

GBCConnectionPtr::= SET OF ObjectInstance

GBCMServiceType::= ENUMERATED {apps (0), spps (1)}

GBCMUserCategory::= ENUMERATED {normal (0), privileged (1)}

GBCMUserId::= NameType

ListOfDestAddr::= SEQUENCE OF E164Address

ProtectionLevel::=    ENUMERATED  {protected  (0),  unprotected-lowpriority  (1),
unprotected-highpriority (2)}

Reason::= NameType

RoutingCriteria::= NameType

ServiceProfileId::= NameType

ServiceProfilePtr::= ObjectInstance

MaxCellTransferDelay::= SEQUENCE {
        acceptableMaxCTD [0] INTEGER OPTIONAL,
        cumulativeMaxCTD [1] INTEGER OPTIONAL }

PeakToPeakCellDelayVariation::= SEQUENCE {
        acceptablePeakToPeakCDV [0] INTEGER OPTIONAL,
        cumulativePeakToPeakCDV [1] INTEGER OPTIONAL }

CellLossRatio::= INTEGER (1..15)

MaxDelay::= INTEGER

MinimumCellRate::= PeakCellRate

BlockErrorRate::= INTEGER

QoSClass::= INTEGER {cbr(0), rt-vbr(1), nrt-vbr(2), ubr(3), abr(4), sdh-cbr(5)}

MaxNumVp::= INTEGER

PeakBitRate::= INTEGER

GBCSchedule::= SEQUENCE {
        startTime       StartTime,
        stopTime        StopTime,
```



```
        CHOICE {        durationSchedule [0]    NULL,
                        dailySchedule [1]       GBCDailySchedule,
                        weeklySchedule  [2]     GBCWeeklySchedule,
                        monthlySchedule [3]     GBCMonthlySchedule,
                        occasionalSchedule [4]  GBCOccasionalSchedule
                }
        }

GBCDailySchedule::= SEQUENCE OF GBCDaySlot

GBCWeeklySchedule::= SEQUENCE OF GBCWeekSlot

GBCOccasionalSchedule::= SEQUENCE OF GBCOccasionalSlot

GBCMonthlySchedule::= SEQUENCE OF GBCMonthlySlot

GBCDaySlot::= SEQUENCE {
        slotBegin       Time24,
        slotEnd         Time24
        }

GBCWeekSlot ::= SEQUENCE {
        slotBegin       TimeWeek,
        slotEnd         TimeWeek
        }

GBCOccasionalSlot::= SEQUENCE {
        slotBegin       StartTime,
        slotEnd         StopTime
        }

GBCMonthlySlot::= SEQUENCE {
        slotBegin       TimeMonth,
        slotEnd         TimeMonth
        }

ATMSpecificParameters::= SEQUENCE {
        qosClass                [0] QoSClass OPTIONAL,
        maxCTD                  [1] MaxCellTransferDelay OPTIONAL,
        peakToPeakCDV           [2] PeakToPeakCellDelayVariation OPTIONAL,
        cellLossRatio           [3] CellLossRatio OPTIONAL,
        peakCellRate            [4] PeakCellRate OPTIONAL,
        sustainableCellRate     [5] SustainableCellRate OPTIONAL,
        pcrCDVTolerance         [6] CDVTolerance OPTIONAL,
        scrCDVTolerance         [7] CDVTolerance OPTIONAL,
        maxBurstSize            [8] MaxBurstSize OPTIONAL,
        minCellRate             [9] MinimumCellRate OPTIONAL }

SDHSpecificParameters::= SEQUENCE {
        qosClass                [0] QoSClass OPTIONAL,
        throughput              [1] PeakBitRate OPTIONAL,
        delay                   [2] MaxDelay OPTIONAL,
        blockErrorRate          [3] BlockErrorRate OPTIONAL }

GBCQoSSequence::= SEQUENCE {
        aPPSQoSSequence [0] APPSQoSSequence OPTIONAL,
        sPPSQoSSequence [1] SPPSQoSSequence OPTIONAL }

APPSQoSSequence::= SEQUENCE {
        forward         [0] ATMSpecificParameters OPTIONAL,
        backward        [1] ATMSpecificParameters OPTIONAL }

SPPSQoSSequence::= SEQUENCE {
        forward         [0] SDHSpecificParameters OPTIONAL,
        backward        [1] SDHSpecificParameters OPTIONAL }
```



```
QoSLimitsSequence::= SEQUENCE {
        atmQoSLimitsSequence [0] ATMQoSLimitsSequence OPTIONAL,
        sdhQoSLimitsSequence [1] SDHQoSLimitsSequence OPTIONAL }

ATMQoSLimitsSequence::= SEQUENCE {
        maxNumVp                        MaxNumVp,
        forwardNegotiableParameter      ATMSpecificParameters,
        backwardNegotiableParameter     ATMSpecificParameters }

SDHQoSLimitsSequence::= SEQUENCE {
        forwardNegotiableParameter      SDHSpecificParameters,
        backwardNegotiableParameter     SDHSpecificParameters }

ReserveGBCConnectionInformation::= SEQUENCE {
        gBCMUserId GBCMUserId,
        sourceE164Address [0] E164Address OPTIONAL,
        destinationE164Address E164Address,
        connectionProtectionLevel [1] ProtectionLevel OPTIONAL,
        routingCriteria RoutingCriteria OPTIONAL,
        gBCType GBCType,
        gBCDirectionality Directionality,
        gBCSchedule     GBCSchedule,
        gBCPPSparameters GBCQoSSequence OPTIONAL }

GBCType::= GBCMServiceType

ReserveGBCConnectionResult::= CHOICE {
        successful [0] SEQUENCE {
                gBCConnectionId   GBCConnectionId,
                gBCAccessPointId  GBCAccessPointId OPTIONAL},
        unsuccessfull [1] Reason }

ReserveGBCConnectionNotifInformation::= SEQUENCE {
        gBCMUserId GBCMUserId,
        sourceE164Address E164Address,
        destinationE164Address E164Address,
        connectionProtectionLevel [1] ProtectionLevel OPTIONAL,
        routingCriteria RoutingCriteria OPTIONAL,
        gBCType GBCType,
        gBCDirectionality Directionality,
        gBCSchedule     GBCSchedule,
        gBCPPSparameters GBCQoSSequence OPTIONAL}

ReserveNotifReply::= CHOICE {
        acception  [0] AdditionalText,
        rejection [1] Reason }

ModifyGBCConnectionInformation::= SEQUENCE {
        gBCMUserId GBCMUserId,
        gBCConnectionId GBCConnectionId,
        gBCSchedule     [0] GBCSchedule OPTIONAL,
        gBCPPSparameters [1] GBCQoSSequence OPTIONAL}

ModifyGBCConnectionResult::= CHOICE {
        successful NULL,
        unsuccessfull Reason }

ModifyGBCConnectionNotifInformation::= SEQUENCE {
        gBCSchedule     [0] GBCSchedule OPTIONAL,
        gBCPPSparameters [1] GBCQoSSequence OPTIONAL }

ReleaseGBCConnectionInformation::= SEQUENCE {
        gBCMUserId GBCMUserId,
        gBCConnectionId GBCConnectionId,
        reason  Reason }
```



```
ReleaseGBCConnectionResult::= CHOICE {
        successful NULL,
        unsuccessfull Reason }

RemoveSubscriptionInformation::= SEQUENCE{
        gBCMUserId      GBCMUserId
                                }

RemoveSubscriptionResult::= CHOICE {
        successful NULL,
        unsuccessful Reason }

ActivationNotifInformation::= ENUMERATED { ok(0), ko(1)}

DeactivationNotifInformation::= ENUMERATED {ok(0), ko(1)}

ReleaseNotifInformation::= ReleaseReason

ReleaseReason::= CHOICE {
        fromPNO ENUMERATED {enduserRelease (0), timeout (1),
                                pnoRelease (2), other (3) },
        fromOriginCPN PrintableString }

GBCAlarmReportNotifInfo::= SEQUENCE {
                typeOfAlarm             GBCAlarmType,
                probableCause           ProbableCause,
                perceivedSeverity       PerceivedSeverity,
                notificationIdentifier  [1] NotificationIdentifier OPTIONAL,
                specificProblems        [2] SpecificProblems OPTIONAL,
                correlatedNotifications [3] CorrelatedNotifications OPTIONAL,
                monitoredAttributes     [4] MonitoredAttributes OPTIONAL,
                proposedRepairActions   [5] ProposedRepairActions OPTIONAL,
                additionalText          AdditionalText OPTIONAL }

RequiredReconfigurationNotifInfo::= SEQUENCE {
                relevantAlarmNotificationId     NotificationIdentifier,
                relevantMOC             ObjectClass,
                relevantMOI             ObjectInstance,
                reconfigurationDescription      ReconfigurationDescription,
                notificationIdentifier  [1] NotificationIdentifier OPTIONAL
 }

TroubleNotifInfo::= SEQUENCE {
        managedObjectInstance   [0] ObjectInstance,
        receivedTime    [1] GeneralizedTime,
        troubleFound    [2] TroubleFound,
        troubleType     [15] TroubleType OPTIONAL
}

GBCAlarmType::= CHOICE {
        number INTEGER {
                recoverableFailure      (1),
                unrecoverableFailure    (2),
                cleared                 (3)
        },
        identifier OBJECT IDENTIFIER
}

ReconfigurationDescription::= PrintableString

TroubleReportFormat::= SEQUENCE OF SingleFormat

SingleFormat::= SEQUENCE {
        managedObjectInstance   GraphicString,
        receivedTime    GraphicString,
        troubleType     GraphicString,
```



```
        troubleReportStatus     GraphicString
}

GBCServiceDescription::= GraphicString(SIZE(0..256))

END
```


11. APPENDIX D - LIST OF REQUIRED PLATFORMS & PACKAGES

Vendor/Package	Required for	Availability and Dependencies	Cost	Description/Comments
Sun Microsystems, Inc. JRE/JDK 1.0.2	CPN GUI & Trace System	Sun Microsystems, Inc. JRE/JDK 1.1.4 (or higher)	free	JDK, Java Development toolkit, needed to develop Java 1.0 applications/applets. JRE, Java runtime environment is a subset of JDK including the JAVA libraries and the virtual machine to run Java Applications. *NOTE: For a runtime-only configuration (without the need to modify the JAVA source code) the JRE/JDK package is optional since the JAVA runtime is available with Java-enabled web browsers. For the TRUMPET trials, however, JDK 1.0.2 should be in place to allow code changes during the integration phase.*
Sun Microsystems, Inc. JRE/JDK 1.1.4 (or higher)	CPN & VASP	Solaris 2.x Windows 95/NT HP/UX 10.x	free	Java Development (JDK) and Runtime (JRE) toolkit, JRE is a subset of JDK including the JAVA libraries and the virtual machine to run Java Applications. *NOTE: JDK for HP/UX is provided by Hewlett-Packard.*
Netscape Navigator 3 or Communicator 4 or Microsoft Internet Explorer 4.0	CPN GUI (and other GUIs, documenation)	Solaris 2.x Windows 95/NT HP/UX 10.x (Internet Explorer is not available for HP/UX 10.x, for Solaris 2.5 only a beta version is available to date)	free	A web browser which provides the execution environment to download and run Java Applets.
Objectspace Voyager 1.0.0	CPN & VASP	100% JAVA based (reqs JRE/JDK 1.1.x)	free	Voyager is a Java-centric distributed computing platform supporting transparent access to remote objects and facilitates object mobility.
Objectspace JGL 2.0.2	CPN(?) & VASP	100% JAVA based (reqs JRE/JDK 1.0.2 or higher)	free	JGL includes 11 optimized data structures including sequential containers, sets, maps, and queues. Both ordered and hashing

				versions of sets and maps are available. NOTE: Support for object persistence is only provided using JRE/JDK 1.1.
Netscape LDAP Java SDK 1.0beta 2	VASP	100% JAVA based (reqs JRE/JDK 1.1.x)	free	Java toolkit to build applications that access networked directory data through the Internet standard Lightweight Directory Access Protocol (LDAP, RFC 1777). Constitutes a subset of the Netscape Directory SDK.
IONA Orbix 2.x /C++ (Single Threaded version)	VASP (CORBA Adapter to Xuser Manager)	Solaris 2.x (reqs SunWSpro Compiler 4.x) Hewlett Packard HP/UX 10.x (reqs HP aC++ Compiler)	Developers: US$ 5000,- Runtime: US$ 100,- UNIX Support: US$ 750,- Windows Support: US$ 400,-	OMG CORBA 2 compliant C++ ORB, Required for the CORBA/TMN gateway, Runtime & Developers Licenses are available.
IONA OrbixWeb for Java 2.x	VASP (Client Proxies & Event Handler of the control server)	Solaris 2.x Windows 95/NT HP/UX 10.x (reqs. JRE/JDK 1.0.2 or higher)	Developers: US$ 799,- Runtime: free Ann. support: US$ 400,-	OMG CORBA 2 compliant Java ORB, Required for the CORBA/TMN gateway, Runtime & Developers Licenses are available.
Netscape Directory Services 3.0	VASP (& CA)	Solaris 2.x Windows 95/NT HP/UX 10.x	free	Directory Service implementation (X.500 based) supporting LDAP version 2 and 3. *NOTE: This package includes the Netscape Directory SDK which in turn includes LDAP Java SDK 1.0beta 2.* *NOTE: Currently the CA is based on the Michigan LDAP service. Support of Netscape Directory Services is considered as an optional work item.*
University of Michigan Michigan LDAP	CA	Solaris 2.x Windows 95/NT HP/UX 10.x	free	Various LDAP tools which have been developed at UMich. Needs to be in place to operate an CA. Pointers to LDAP-related sources can also be found at reference .com.
Rogue Wave tools.h++ 7.x	potentially all components developed with C++	Solaris 2.x Windows 95/NT HP/UX 10.x	Usually no extra cost as it is bundled with C++	C++ foundation class library contains over 120 classes, including dates, times and strings, sets,

			compilers, Software: US$ 594,- Support: US$ 234,-	bags, B-Trees, sorted collections, linked lists, queues, stacks, and more This library is bundled with SunWSpro C++ and the HP aC++ compilers.
Sun Microsystems, Inc. Workshop pro C/C++ compiler 4.2	all components developed with C++ (e.g., VASP CORBA Adapter, PNO, security package)	Solaris 2.x	variable R&D: ask for academic price list Named customers: approx. UK pounds 700,- Printed Docu.: UK pounds 180,-	SUN ANSI C/C++ Compiler, Linker and Libraries (includes Rogue Wave tools.h++ 7.0), Runtime libs could be provided by TRUMPET partner, Contact and price info. can be obtained through SunExpress
Hewlett Packard HP aC++ for HP/UX-10	all components developed with C++ (e.g., VASP CORBA Adapter, PNO, security package)	HP/UX 10.x	variable list price: US$ 1495,-	HP ANSI C/C++ Compiler, Linker, and Libraries (includes C++ standard lib and Rogue Wave tools.h++ 7.0), NOTE: HP offers another Compiler (Cfront) called C++ 3.0 or CSET which is NOT suitable since there is no proper support for Orbix, and limitations for library support such as tools.h++!!!
Hewlett Packard Cumulative Consolidated Patch PSOV_01730 for HPOV-DM 4.21 on Solaris 2.x	PNO & NMS	Solaris 2.x	free	Cumulative Consolidated Patch for HPOV-DM 4.21 on HP/UX 10.x. The patch is required to run the VASP CORBA/TMN gateway and the PNO Xuser Agent. The patch can be obtained from HP Support web site.
Hewlett Packard Cumulative Consolidated Patch PSOV_12211 for HPOV-DM 4.21 on HP/UX 10.x	PNO & NMS	HP/UX 10.x	free	Cumulative Consolidated Patch for HPOV-DM 4.21 on HP/UX 10.x. The patch is required to run the VASP CORBA/TMN gateway and the PNO Xuser Agent. The patch can be obtained from HP Support web site.
Hewlett Packard MOT 1.1	NMS	Solaris 2.x HP/UX 10.x (reqs HPOV-DM 4.2)	variable Runtime provided by GMD	HP OpenView Telecom Managed Object Toolkit, provides high-level C++ APIs which hides the complexity of XOM/XMP object manipulations and provides a generator to create OSI agent implementations from GDMO/ASN.1
Fore Systems	NMS	FORE ASX200	not available	ATM Networking

ForeThought 4.02		(embedded Sun Solaris system)		Software which also provides the SNMP management interfaces (MIBs) required for the NMS

Table 5: List of required platforms and packages

