



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.834.4

(07/2003)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

A CORBA interface specification for Broadband Passive Optical Networks based on UML interface requirements

ITU-T Recommendation Q.834.4

ITU-T Q-SERIES RECOMMENDATIONS
SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4, 5, 6, R1 AND R2	Q.120–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	Q.800–Q.849
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Q.834.4

A CORBA interface specification for Broadband Passive Optical Networks based on UML interface requirements

Summary

This Recommendation provides a CORBA IDL definition for the management interface between a Supplier Management System and an Operator Management System. This work defines part of the management aspects for network resources defined by the G.983.x series of ITU-T Recommendations for Broadband Passive Optical Network (BPON) equipment.

Generally speaking, the Supplier Management System is an Element Management System (EMS) and the Operator Management System (OMS) is a Network Management System (NMS). However, the Supplier Management System is required to present a "network view" of connection management to the Operator Management System. So it was deemed necessary for clarity's sake to use the terminology adopted in naming the systems involved.

In addition, it should be noted that ITU-T Rec. Q.834.1 contains a set of functionality requirements and a listing of managed entity definitions forming the basis for management information required for a "network element view" of BPON equipment. ITU-T Rec. Q.834.2 completes the management information definition for management of BPON equipment by providing the definitions of managed entities for the "network view". ITU-T Rec. Q.834.3 addresses management interface behaviour through use of UML diagrams and Use Case descriptions. The management information modelled in ITU-T Recs Q.834.1 and Q.834.2 is referenced throughout this Recommendation and ITU-T Rec. Q.834.3.

Source

ITU-T Recommendation Q.834.4 was approved on 7 July 2003 by ITU-T Study Group 4 (2001-2004) under the ITU-T Recommendation A.8 procedure. This edition includes the modifications introduced by Corrigendum 1 approved on 13 January 2004.

Keywords

APON, BPON, CORBA, IDL, PON, UML.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

© ITU 2004

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

Page

1	Scope	1
2	References.....	1
2.1	Normative references.....	1
2.2	Other references.....	2
3	Terms and definitions	2
3.1	Terms imported from ITU-T Rec. M.3010	2
3.2	Terms imported from UML.....	2
3.3	Terms imported from OMG Naming Service	3
3.4	Terms imported from ITU-T Rec. Q.834.1	3
3.5	Terms imported from ITU-T Rec. Q.834.3	3
3.6	New terms.....	3
4	Abbreviations.....	3
5	Conventions	5
5.1	Module description conventions.....	5
5.2	IDL file conventions.....	5
5.3	NULL values	6
6	Interface architecture overview	6
7	Names and naming constraint.....	7
7.1	Service objects and OMG naming service	8
7.2	Domain objects.....	9
8	Organization of IDL files.....	9
9	Modules	10
9.1	AccessControl module.....	10
9.2	Build module	16
9.3	Q834Common module	25
9.4	ControlArchive module	26
9.5	SoftwareDownload module	30
9.6	EventPublisher module.....	37
9.7	MIBTransfer module	40
9.8	PerformanceManager module	43
9.9	ProfileManager	52
9.10	Registrar module	54
9.11	ResourceAllocation module	59
9.12	SchedulerManagement module	62
9.13	ServiceProvisioning module.....	66
9.14	Synchroniser module	70
9.15	Test module	73
9.16	FileTransfer module	79

	Page
10 Compliance statement.....	82
Annex A – Data dictionary	82
Annex B – Exceptions	104
Annex C – IDL files.....	109
C.1 Q834AccessControl.idl	109
C.2 Q834Build.idl	114
C.3 Q834Common.idl	120
C.4 Q834ControlArchive.idl	134
C.5 Q834SoftwareDownload.idl	137
C.6 Q834EventPublisher.idl.....	142
C.7 Q834MIBTransfer.idl	146
C.8 Q834PerformanceManager.idl	149
C.9 Q834ProfileManager.idl	154
C.10 Q834Registrar.idl	165
C.11 Q834ResourceAllocation.idl	168
C.12 Q834SchedulerManagement.idl	171
C.13 Q834ServiceProvisioning.idl.....	173
C.14 Q834Synchroniser.idl.....	176
C.15 Q834Test.idl	178
C.16 Q834Filetransfer.idl.....	183
Annex D – Example endpoint templates	186

ITU-T Recommendation Q.834.4

A CORBA interface specification for Broadband Passive Optical Networks based on UML interface requirements

1 Scope

This Recommendation addresses the design of an interactive mechanized interface between the Supplier Management System managing BPON network resources and an Operator Management System (OMS). The design is based on the UML diagrams and Use Case descriptions of ITU-T Rec. Q.834.3. The general approach adopted in ITU-T Rec. Q.834.3, and implemented here, is that the Supplier Management System provides management services to the Operator Management System. These services provide the OMS with a high level abstraction of the following capabilities:

- Provisioning installed network resources;
- Provisioning uninstalled network resources including capacity reservation;
- Service provisioning;
- Archive management;
- NE software management;
- NE configuration data backup and restoral;
- Performance management;
- NE event publication;
- Profile management;
- Testing;
- Activity scheduling;
- Bulk transfer management;
- NE-EMS synchronisation;
- Access control.

This Recommendation describes the CORBA IDL interfaces supporting the services listed above.

2 References

2.1 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [1] ITU-T Recommendation M.3010 (2000), *Principles for a telecommunications management network*.
- [2] ITU-T Recommendation M.3200 (1997), *TMN management services and telecommunications managed areas: Overview*.
- [3] ITU-T Recommendation M.3400 (2000), *TMN Management Functions*.
- [4] OMG Document formal/99-06-01, *Unified Modelling Language, Section 1*.



- [5] ITU-T Recommendation G.983.1 (1998), *Broadband optical access systems based on Passive Optical Networks (PON)*, plus Amendment 1 (2001).
- [6] ITU-T Recommendation Q.834.1 (2001), *ATM-PON requirements and managed entities for the network element view*.
- [7] ITU-T Recommendation Q.834.2 (2001), *ATM-PON requirements and managed entities for the network view*.
- [8] ITU-T Recommendation Q.834.3 (2001), *A UML description for management interface requirements for Broadband Passive Optical Networks*.
- [9] ITU-T Recommendation M.3020 (2000), *TMN Interface Specification Methodology*.
- [10] ITU-T Recommendation G.983.2 (2002), *ONT management and control interface specification for B-PON*.
- [11] ITU-T Recommendation G.983.3 (2001), *A broadband optical access system with increased service capability by wavelength allocation*.
- [12] ITU-T Recommendation G.983.4 (2001), *A broadband optical access system with increased service capability using dynamic bandwidth assignment (DBA)*.
- [13] ITU-T Recommendation G.983.5 (2002), *A broadband optical access system with enhanced survivability*.
- [14] ITU-T Recommendation G.983.6 (2002), *ONT management and control interface specifications for B-PON system with protection features*.
- [15] ITU-T Recommendation G.983.7 (2001), *ONT Management and Control interface specification for Dynamic Bandwidth Assignment (DBA) B-PON system*.
- [16] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects*.
- [17] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services*.

2.2 Other references

- [18] OMG Document formal/02-09-02, *CORBA Services – Naming Service specification*.
- [19] OMG Document formal/02-08-04, *CORBA Services – Notification Service specification*.

3 Terms and definitions

3.1 Terms imported from ITU-T Rec. M.3010

The following term from ITU-T Rec. M.3010 is used in this Recommendation:

- User.

3.2 Terms imported from UML

The following terms from UML [4] are used in this Recommendation:

- Actor;
- Class;
- Class Diagram;
- Use Case.

3.3 Terms imported from OMG Naming Service

The following terms from OMG Naming Service [18] are used in this Recommendation:

- Naming graph;
- Name.

3.4 Terms imported from ITU-T Rec. Q.834.1

The following term from ITU-T Rec. Q.834.1 is used in this Recommendation:

- Managed Entity.

3.5 Terms imported from ITU-T Rec. Q.834.3

The following terms from ITU-T Rec. Q.834.3 are used in this Recommendation:

- Activate;
- Assign;
- Autodiscovery;
- BPON resource;
- Build;
- Filtering;
- Install;
- Range;
- Register;
- Reserve;
- Service Instance;
- User Label.

3.6 New terms

This Recommendation defines the following terms:

3.6.1 service object: The set of objects providing access to the management services implemented in the Supplier Management System. Interfaces to service objects constitute the specification for the portion of IF1 defined in this Recommendation.

3.6.2 domain object: The set of objects defining the management information for management of BPON network resources. Domain objects are primarily provided by the Managed Entity listings of ITU-T Recs Q.834.1 and Q.834.2.

3.6.3 internal object: The set of objects used to support Supplier Management System internal logic. They are completely obscure from the point of view of the OMS.

4 Abbreviations

This Recommendation uses the following abbreviations:

AAL	ATM Adaptation Layer
APON	ATM-PON
ATM	Asynchronous Transfer Mode
BICI	Broadband Inter-Carrier Interface
BISSI	Broadband Inter-Switching System Interface
BPON	Broadband Passive Optical Network

CAC	Call Admission Control
CCITT	Consultative Committee for International Telephone and Telegraph
CES	Circuit Emulation Service
CNM	Customer Network Management
CORBA	Common Object Request Broker Architecture
CTP	Connection Termination Point
CTT	Customer Trouble Ticket
DCN	Data Communications Network
DSx	Digital Signal x
EM	Element Management
EML	Element Management Layer
EMS	Element Management System
EOC	Embedded Operations Channel
Ex	European Digital Signal x
FSAN	Full Services Access Network
GUI	Graphical User Interface
IDL	Interface Definition Language
ITU	International Telecommunication Union
ME	Managed Entity
MIB	Management Information Base
NE	Network Element
NMS	Network Management System
NT	Network Terminal
ODN	Optical Distribution Network
OLT	Optical Line Terminal
OMG	Object Management Group
OMS	Operator Management System
ONT	Optical Network Terminal
ONU	Optical Network Unit
OS	Operations System
PON	Passive Optical Network
PVC	Permanent Virtual Circuit
QoS	Quality of Service
TCA	Threshold Crossing Alert
TMN	Telecommunications Management Network
TP	Termination Point
TTP	Trail Termination Point
UML	Unified Modelling Language
UNI	User-Network Interface
VC	Virtual Channel
VCC	Virtual Channel Connection

VCi	Virtual Channel Identifier
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier

5 Conventions

There are two clauses in this Recommendation where specific conventions have been followed. The first is the description of the IDL Modules in clause 8 and the second is the IDL files found in Annex C. Finally, this Recommendation uses certain conventions involving null values.

5.1 Module description conventions

Module descriptions have the following pattern:

- Module Name – services overview, including any relevant high-level modelling, key business data element descriptions, and scenarios as required.
 - Interface Name – overview of the specific services provided by this interface.
 - Operation Name₁ – detailed description of the behaviour of this operation including signature of the operation, description of each input parameter, and description of the return value.
 - ...
 - Operation Name_n – detailed description of the behaviour of this operation including signature of the operation, description of each input parameter, and description of the return value.
 - Exceptions – context-specific conditions that cause the named exceptions to be raised.

5.2 IDL file conventions

Each IDL file has the following format:

- **#ifndef __<MODULENAME>_DEFINED** (module name is the same as the filename);
- **#define __<MODULENAME>_DEFINED**
- **#include "<idlfilename>"¹**
- **#pragma prefix "itu.Int"**
- **module q834_4 {**
- **module <modulename> {**
- all the specific data type definitions and interface specifications needed and indicated by the interface design initiated from examination of ITU-T Rec. Q.834.3;
- **}; //module <modulename>**
- **}; //module q834_4**

Within each module definition, data type definitions precede and are segmented from the interface definitions. All data types defined in this Recommendation start with an initial capitalized alpha character. If data type names are joins of several proper nouns, pronouns, or adjectives, the join will expect a capitalized alpha character at the start of each component. Labels for parameters used in any interface operation signature start with a lower case alpha character. Each operation name begins with a lower case alpha character.

¹ Q834Common.idl file is included with every other IDL file in this Recommendation.

5.3 NULL values

In this Recommendation, when there is a mention of "null string", it is meant to be an empty string and not a null object supported by languages such as JAVA. Similarly, when a "null sequence" is mentioned, a sequence with zero elements is passed.

6 Interface architecture overview

Figure 1 illustrates the BPON network architecture and management system. This technology provides an integrated loop access delivery mechanism for every telecommunications service currently deployed by operators. A list of such services includes telephony and voice grade services, circuit-emulation, Ethernet, ATM, xDSL, and video. This standardization effort is ongoing in its consideration of layer 1 and 2 transport solutions on the ODN, although initial implementations exist in operator networks involving APON and ATM as defined in ITU-T Rec. G.983.1.

The figure also shows the IF1 (Q) interface between the Supplier Management System and Operator Management Systems. The IF1 interface covers all aspects of network and service provisioning management, network performance management, traffic management, maintenance, testing, and user security administration. Specification of IF1 would have been an overwhelming challenge if the approach adopted had not pursued a level of abstraction above the transport delivery technology details as well as the complexities inherent in managing so many service types. The approach included the definition and description of interface requirements using Unified Modelling Language, followed the methodology of ITU-T Rec. M.3020, and was documented in ITU-T Rec. Q.834.3.

The operator systems identified in this diagram correspond directly to system actors defined in ITU-T Rec. Q.834.3. This Recommendation specifies the interactive real-time transactions involving the Supplier Management System and does not explicitly describe the file structures transferred between the Supplier Management System and either the Data Warehouse or Secure File Server. Since the interfaces to OMG Naming Service and OMG Notification Service are already specified, the focus of this Recommendation narrows down to messaging between the Supplier Management System and the shaded system actors.

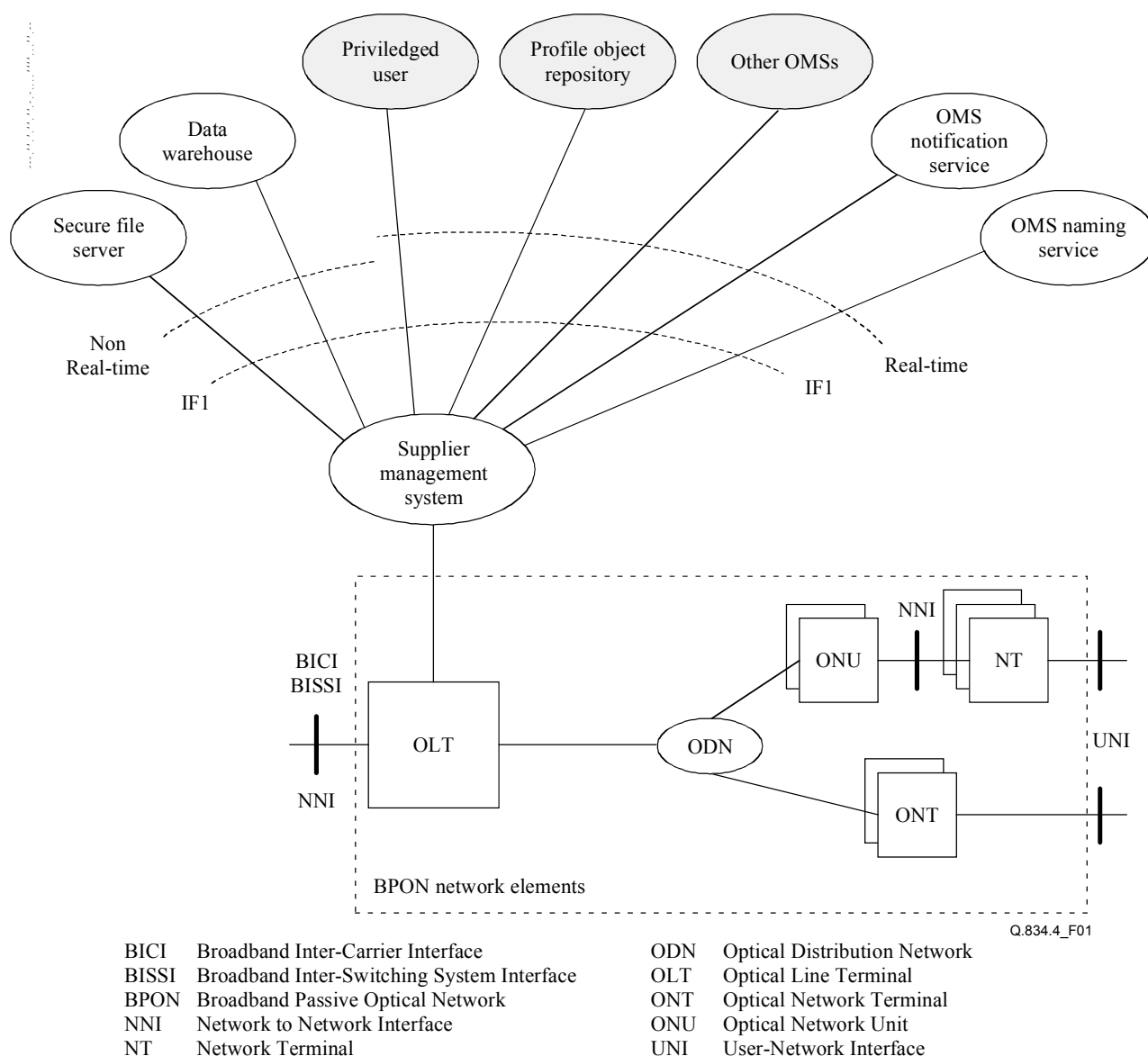


Figure 1/Q.834.4 – Interface architecture overview

7 Names and naming constraint

This clause provides guidelines for object names and managed entity identifiers referenced on IF1.

Review of all the class diagrams developed in ITU-T Rec. Q.834.3 shows that the Supplier Management System manages three distinct types of "objects". These can be classified as follows:

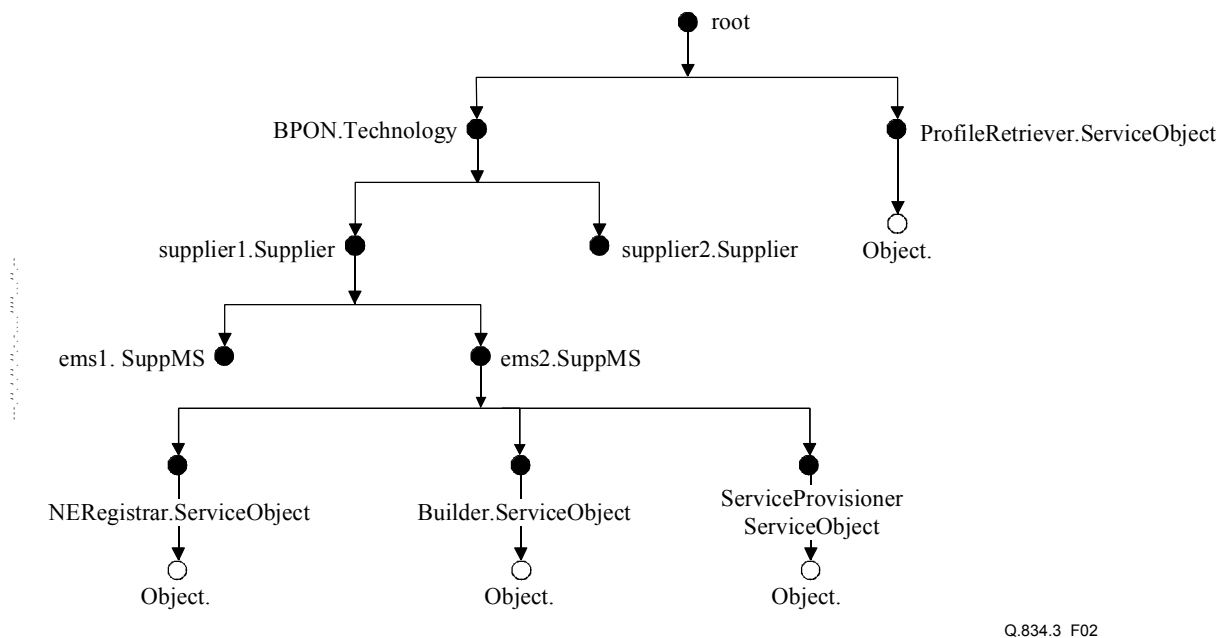
- **Service Objects:** provide access to the management services implemented in the Supplier Management System. The interfaces of these services are the main subject of this Recommendation. A few examples are NERegistrar, ServiceProvisioner, TestActionPerformer, and DownloadMgr. They are aligned with ITU-T Rec. X.780 as derivations of Managed Object Interface in this Recommendation.
- **Domain Objects:** define the management information for management of BPON network resources. Domain objects are primarily resources (managed entities) identified by ITU-T Recs Q.834.1 and Q.834.2. These domain objects, as well as others defined in ITU-T Rec. M.3120 and other Recommendations following the CORBA Framework of ITU-T, may be made available for use on IF1. A mechanism for harmonizing the use of Service Objects and Domain Objects is explained later on in this clause.

- Internal Objects: used to support Supplier Management System internal logic. They are completely obscure from the point of view of the OMS.

7.1 Service objects and OMG naming service

All Service objects (either instantiated by the Supplier Management System or the OMS) have their IDL interface defined in this Recommendation and their object reference registered with the OMG Naming Service pictured in Figure 1 using the bind() operation of Naming Service interface. Subsequently, the location of the object instance can be found using the resolve() operation of Naming Service, if required.

Object names can be either "well known" (meaning that the name has been documented and agreed to prior to runtime), or communicated at run-time through another interface. In the case of this Recommendation, all of the Service objects are "well-known". The name of a well-known object can be documented via a naming graph. Figure 2 provides an example of a naming graph for some of the Service Objects in this Recommendation, following the naming convention described in ITU-T Rec. Q.816. Shown in the diagram is the Id, followed by a dot, followed by the Kind value. Only four of the Service Objects are shown, for the sake of simplicity, in the figure. The other Service Objects would be named in an analogous fashion.



Q.834.3_F02

Figure 2/Q.834.4 – Naming graph

The complete path for the name is required for a "well known" object. At start-up of a newly installed Supplier Management System, all instances of Service objects are automatically registered to a single OMG Naming Service maintained by the Operator. The Name of any object registered with OMG Naming Service has the following syntax:

```

typedef      string      Istring;

struct NameComponent {
    Istring id;
    Istring kind;
};

typedef sequence<NameComponent> Name;
  
```

One interpretation of the naming graph of Figure 2 is that the "kind" field in each of the naming components of the service objects has the value of null string, and that each of the nodes shown

represents the values for the "id" field. The details of the naming graph, as well as its interpretation, should be determined via agreement between the supplier and operator and these details lie outside the scope of this Recommendation.

7.2 Domain objects

The Supplier Management System maintains a large number of Managed Entities (or Domain Objects)². These entities are managed indirectly through the management services offered by the Supplier Management System. However, references (identifiers) to specific resources instances are often needed for the OMS to make effective use of the management services of the Supplier Management System. When the identifier is returned as result of a management operation (or included in notifications) it is assumed that the identifier is "known" in subsequent management service invocations from the OMS. In this specification, the identifier of a Managed Entity has been provided a syntax that allows the Supplier Management System to inform the OMS of at least one of three ways to interact with the Managed Entity in subsequent invocations. If the Managed Entity is a Managed Object available on IF1, then the CORBA name of the object is provided as described in ITU-T Rec. X.780. If the Managed Entity has a supporting Façade object available on IF1, then the CORBA name of the façade object is provided as described in ITU-T Rec. X.780.1. If neither of these capabilities is available yet, then the identifier consists of a reference unique within the context of the management domain of the Supplier Management System. In general, details of this reference are based on a specific containment hierarchy determined via agreement between the supplier and operator and the details lie outside the scope of this Recommendation.

8 Organization of IDL files

The proposed interface specification found in this Recommendation consists of a parent module called "q834_4". The parent module is partitioned into IDL files (smaller modules) corresponding to smaller named modules. Each of these files contains one or more interface definitions representing a specific management service supported on IF1. Table 1 provides a listing correlating Annex C Reference number, IDL File Name, contained IDL Interfaces, and reference to the associated Use Case Description of ITU-T Rec. Q.834.3.

Table 1/Q.834.4 – q834_4 module organization

Annex reference	IDL files	IDL interface	Use case title
C.1	Q834AccessControl	AccessControlMgr	AccessControl
C.2	Q834Build	Builder	Build BPON Resources
C.3	Q834Common	ProbableCause MonitoringParameter RecordSetType PMCategory PhysicalLayerLoopback	
C.4	Q834ControlArchive	RecordSetMgr	Control Archiving

² If the Supplier Management System manages 100-200 BPON systems (desirable sizing requirement) and fully supports the shared management knowledge base contained in ITU-T Recs Q.843.1 and Q.834.2, then the number of managed entity instances supported by the Supplier Management System varies between 10s of millions to billions depending on the service offerings.

Table 1/Q.834.4 – q834_4 module organization

Annex reference	IDL files	IDL interface	Use case title
C.5	Q834SoftwareDownload	DownloadMgr	Distribute Software
		VersionRepository	NE Software Version Control
C.6	Q834EventPublisher	AlarmEventSupplier SecurityEventSupplier DiscoveryEventSupplier	Publish Events
C.7	Q834MIBTransfer	MIBMover	NE Restoral
C.8	Q834PerformanceManager	ImpairmentPersistence	RCAA & RCIA
		ReportController	Performance & Traffic Monitoring Reporting Control
C.9	Q834ProfileManager	ProfileConsumer ProfileUsageMgr ProfileRetriever	Profile Object Management
C.10	Q834Registrar	NERegistrar	Provision Installed BPON Resources
C.11	Q834ResourceAllocation	ResourceAllocator	Reserve Resources
C.12	Q834SchedulerManagement	SchedulerMgr	Scheduler
C.13	Q834ServiceProvisioning	ServiceProvisioner	Provision Service
C.14	Q834Synchroniser	Synchroniser	Provide Current Event Summary Listings
			NE Synchronisation
C.15	Q834Test	TestActionPerformer	Testing
C.16	Q834FileTransfer	TransferMgr	Bulk Transfer

9 Modules

9.1 AccessControl module

This module describes functionality for creating, deleting, assigning, and using access control information for operators using the GUI client application of the Supplier Management System. In this module, users can be assigned to user groups. Permission can be granted to individual users or to user groups. The user possesses the permission of any group to which he or she has been assigned. Any individually defined permission (see 9.1.1.14 and 9.1.1.15) for a specific user takes precedence over a group assignment. With group assignments the highest level of permission prevails for any user assigned to the groups. It is assumed that the privileged user does not request any ambiguous target activities when creating or modifying a user group, or when specifying the permission for an individual user.

A target activity is defined as shown in Table 2, throughout the module:

Table 2/Q.834.4 – Target activity detail

Field name	Definition	Syntax	Comments
activityLevel	Specifies the access level of the activity	enum	monitorOnly allowedToExecute noAccess
activityType	Specifies the type of the activity	short	Defined as various constants in the interface AccessControlMgr
administrationDomainList	The identifier provided by the OMS or operator during registration to indicate the administration domain to which the NE belongs	UserLabel	

9.1.1 AccessControlMgr interface

9.1.1.1 setPasswordPolicy

This operation allows the privileged users to manage the password policy.

The operation signature for **setPasswordPolicy** is shown below:

```
void setPasswordPolicy (in PasswordPolicyType passwordPolicy)
    raises (AccessDenied);
```

The input parameter **passwordPolicy** identifies the password policy which is enforced by the Supplier Management System and it consists of UserLoginPolicy and SessionPolicy. UserLoginPolicy dictates the following parameters regarding userId and password: minimum size for user Id, minimum size for user password, length of time in days before a password can be reused, maximum number of login attempts allowed before the user access is blocked for a day, how long the password is valid (in days), whether the password must contain an alphanumeric mixture, whether or not at least one special character is required, whether or not the password can contain repeating characters, and whether or not the user Id can be part of the password.

SessionPolicy identifies the following parameters regarding the session: length of time that a session is inactive before the session is discontinued, length of time an inactive user Id is disabled, and the maximum number of active sessions per user Id is permitted.

Since only one passwordPolicy can exist in a Supplier Management System, there is no need to have a create method, either a default policy exists in the Supplier Management System or the first set method can act as set-by-create.

The return value is of type **void**.

9.1.1.2 passwordPolicyGet

This operation allows the privileged users to retrieve the policy concerning the syntax of data exchanged and timers used when logging into the Supplier Management System.

The operation signature for **passwordPolicyGet** is shown below:

```
PasswordPolicyType passwordPolicyGet ()
    raises (AccessDenied);
```

There is no input parameter required by this operation.

The return value is of type **PasswordPolicyType** providing the details governing user log on to the Supplier Management System.

9.1.1.3 userListGet

This operation allows the privileged users to retrieve the list of user ids having some form of access to the Supplier Management System as well as their target activities and group memberships.

The operation signature for **userListGet** is shown below:

```
UserSeqType userListGet ()  
    raises (AccessDenied);
```

There is no input parameter required by this operation.

The return value is of type **UserSeqType** providing the list of user ids having some form of access to the Supplier Management System as well as their target activities and group memberships.

9.1.1.4 userGroupListGet

This operation allows the privileged users to retrieve the user groups having access to the Supplier Management System with the members of the group and target activities identified.

The operation signature for **userGroupListGet** is shown below:

```
UserGroupSeqType userGroupListGet ()  
    raises (AccessDenied);
```

There is no input parameter required by this operation.

The return value is of type **UserGroupSeqType** providing the user groups having access to the Supplier Management System with the members of the group and target activities identified.

9.1.1.5 userGet

This operation allows the privileged users to retrieve the user's group membership and target activities.

The operation signature for **userGet** is shown below:

```
UserType userGet (  
    in UserIdType userId )  
    raises (AccessDenied, UnknownUserIds);
```

The input parameter **userId** identifies the user of interest.

The return value is of type **UserType** provides the user's group membership and target activities.

9.1.1.6 userGroupGet

This operation allows the privileged users to retrieve the member users of the group and their allowed target activities.

The operation signature for **userGroupGet** is shown below:

```
UserGroupType userGroupGet (  
    in UserLabelType userGroupId)  
    raises (AccessDenied, UnknownUserGroupId);
```

The input parameter **userGroupId** identifies the user group.

The return value is of type **UserGroupType** provides the identifier for the user group, the member users of the group, and the allowed target activities.

9.1.1.7 createUserGroup

This operation allows the privileged users to create a new user group. User group is used to give certain users certain accesses to certain network elements. The privileged user creates the new group providing the list of the activities allowed to the user group.

The operation signature for **createUserGroup** is shown below:

```
void createUserGroup (
    in UserLabelType userGroupId,
    in TargetActivitySeqType targetAdditions)
    raises (DuplicateUserGroupId, UnknownTargets, AccessDenied);
```

The input parameter **userGroupId** identifies the group to be created. UserGroupId is not allowed to have the empty string value. The input parameter **targetAdditions** identifies the TargetActivities allowed to the users belonging to this group.

The return value is of type **void**.

9.1.1.8 modifyUserGroup

This operation allows the privileged users to add/delete the target activities of a user group. The privileged user specifies the user group and the list of the target activities to be added or deleted. The Supplier Management System processes the deletions before the additions. This allows for example the raising of the permission level of a given activity.

The operation signature for **modifyUserGroup** is shown below:

```
TargetActivitySeqType modifyUserGroup (
    in UserLabelType userGroupId,
    in TargetActivitySeqType targetAdditions,
    in TargetActivitySeqType targetDeletions)
    raises (UnknownUserGroupId, UnknownTargets,
    AccessDenied);
```

The input parameter **userGroupId** identifies the user group to/from which an operator wants to add/delete target activities. The input parameter **targetAdditions** identifies the TargetActivities which need to be added to the group. The input parameter **targetDeletions** identifies the TargetActivities which need to be deleted from the group.

The return value is of type **TargetActivitySeqType** that provides the updated list of TargetActivities which are allowed to the users belonging to this user group.

9.1.1.9 deleteUserGroup

This operation allows the privileged users to delete an existing user group. The deletion can be completed only if the user group is empty.

The operation signature for **deleteUserGroup** is shown below:

```
void deleteUserGroup (
    in UserLabelType userGroupId)
    raises (AccessDenied, UserGroupNotEmpty, UnknownUserGroupId);
```

The input parameter **userGroupId** identifies the group to be deleted.

The return value is of type **void**.

9.1.1.10 addUsersToGroup

This operation allows the privileged user to add new users to an existing user group.

The operation signature for **addUsersToGroup** is shown below:

```
void addUsersToGroup (
    in UserLabelType userGroupId,
    in UserIdSeqType userIdList)
    raises (AccessDenied, UnknownUserGroupId);
```

The input parameter **userGroupId** identifies the group to which new users need to be added. The input parameter **userIdList** identifies a set of userIds which need to be added to the existing user group.

The return value is of type **void**.

9.1.1.11 deleteUsersFromGroup

This operation allows the privileged users to delete users from a user group.

The operation signature for **deleteUsersFromGroup** is shown below:

```
void deleteUsersFromGroup (  
    in UserLabelType userGroupId,  
    in UserIdSeqType userIdList)  
    raises (AccessDenied, UnknownUserGroupId, UnknownUserIds );
```

The input parameter **userGroupId** identifies the group from which users need to be deleted. The input parameter **userIdList** identifies a set of userIds to be deleted from the user group.

The return value is of type **void**.

9.1.1.12 getPermissionList

This operation allows the privileged users to obtain the list of the activities allowed to a specified user.

The operation signature for **getPermissionList** is shown below:

```
TargetActivitySeqType getPermissionList (  
    in UserIdType userId)  
    raises (UnknownUserIds, AccessDenied);
```

The input parameter **userId** identifies the user whose allowed TargetActivities the privileged user wants to obtain.

The return value is of type **TargetActivitySeqType** that provides the list of the TargetActivities which are allowed to a specified user.

9.1.1.13 modifyPermissionList

This operation allows the privileged users to modify the list of the activities allowed to a user. The privileged user specifies the userId and the list of the activities which need to be added or deleted. The Supplier Management System processes the deletions before the additions. This allows, for example, the raising of the permission level of a given activity.

The operation signature for **modifyPermissionList** is shown below:

```
TargetActivitySeqType modifyPermissionList (  
    in UserIdType userId,  
    in TargetActivitySeqType targetAdditions,  
    in TargetActivitySeqType targetDeletions)  
    raises (UnknownUserIds, UnknownTargets, AccessDenied);
```

The input parameter **userId** identifies the user whose allowed activities the privileged user wants to modify. The input parameter **targetAdditions** identifies the allowed activities which need to be added to a specified user. The input parameter **targetDeletions** identifies the allowed activities which need to be deleted from a specified user.

If a user belongs to multiple groups that have the same activity, then the user assumes the access level that is the highest. When both the user and the group the user belongs to are added to the same administration domain, then the users activityLevel takes precedence over the group's activityLevel.

The return value of type **TargetActivitySeqType** provides the list of the activities which are allowed to a specified user after the modification.

9.1.1.14 createUser

This operation allows the privileged users to create a new user. The privileged user provides a new **userId**, its password, and the list of the activities allowed to the user.

The operation signature for **createUser** is shown below:

```
void createUser (
    in UserIdType userId,
    in PasswordType password,
    in TargetActivitySeqType targetAdditions)
    raises (DuplicateUserId, UnknownTargets, AccessDenied,
    UserLoginPolicyViolation);
```

The input parameter **userId** identifies the identification information to be associated to the new user. The input parameter **password** identifies the password to be associated with the new user Id. The input parameter **targetAdditions** identifies the TargetActivities allowed to the new user.

The return value is of type **void**.

9.1.1.15 deleteUser

This operation allows the privileged users to delete an existing user.

The operation signature for **deleteUser** is shown below:

```
void deleteUser (
    in UserIdType userId)
    raises (UnknownUserIds, AccessDenied);
```

The input parameter **userId** identifies the user to be deleted.

The return value is of type **void**.

9.1.1.16 resetPassword

This operation allows the privileged users to reset the password for a user. This operation takes place when the old password is not available and the privileged user needs to set the new password for the user. On first use, the Supplier Management System will prompt the user to change their password.

The operation signature for **resetPassword** is shown below:

```
void resetPassword (
    in UserIdType userId,
    in PasswordType newPassword)
    raises (UnknownUserIds, UserLoginPolicyViolation, AccessDenied);
```

The input parameter **userId** identifies the user which needs to be deleted. The input parameter **newPassword** specifies the password which needs to be associated with the **userId**.

The return value is of type **void**.

9.1.1.17 Exceptions

The exception **AccessDenied** is raised when the system is not granted access to the interface object.

The exception **DuplicateUserId** is raised when the **userId** specified for the new creation of a user already exists in the Supplier Management System's database.

The exception **DuplicateUserGroupId** is raised when the **userGroupId** specified for the new creation of a user group already exists in the Supplier Management System. In other words, the

Supplier Management System polices the assignment of User Label for user group identifiers for all users recognized by it.

The exception **UnknownTargets** is raised when the specified list of the TargetActivities cannot be identified.

The exception **UnknownUserGroupId** is raised when the specified UserGroupId cannot be identified.

The exception **UnknownUserIds** is raised when any userId is unrecognized.

The exception **UserGroupNotEmpty** is raised when there is an attempt to delete a user group but the user group is not empty (i.e., there are registered users in the group).

The exception **UserLoginPolicyViolation** is raised when the password specified for a new user or for the password change is violating the UserLoginPolicy.

9.2 Build module

The Supplier Management System builds management model groupings for planned equipment on request of an OMS or Operator as part of preprovisioning activities. These resources include nodes (OLT, ONT, ONU, NT) and plug-in units. If equipment is installed, modify operations are used to complete the provisioning of the installed resources. Protection groupings are established for either preprovisioned or installed resources.

9.2.1 Builder interface

9.2.1.1 buildNode

This operation builds a NE in the Supplier Management System. A set of managed entities is automatically created in the management information model maintained by the Supplier Management System as a result of this operation. Depending on the equipment implementation of the supplier, instances of equipmentHolderF for shelves and slots, instances of NEFSAN, and instances of physicalPathTPF for integrated ports, are examples of what can be automatically created.

If the NE is an ONT or ONU, this activity automatically removes the bandwidth necessary to support the embedded operations channel between the OLT and ONT or ONU from the available system bandwidth.

The operation signature for **buildNode** is shown below:

```
ManagedEntityIdType buildNode (  
    in NEKindType nEKind,  
    in string supplierName,  
    in string location,  
    in VersionType version,  
    in SerialNumType serialNum,  
    in NameSeqType alarmSeverityProfiles,  
    in NameSeqType thresholdDataProfiles,  
    in SlotAssignmentSeqType slotAssignmentList,  
    in ManagedEntityIdType port,  
    in string modelCode,  
    in string systemTitle,  
    in VersionSeqType softwareVersions,  
    in UserLabelType nEUserLabel,  
    in ExternalTimeType externalTime,  
    in SystemTimingType systemTiming,  
    in AdministrationDomainType administrationDomain)  
    raises (UnrecognisedVersion, InvalidSerialNumSyntax,  
    DuplicateSerialNumber, UnknownProfiles,  
    UnknownManagedEntity, DuplicateUserLabel, AccessDenied,
```

```
InvalidExternalTime, UnknownSystemTimingSource,
ProfileSuspended);
```

The input parameter **nEKind** identifies the type of NE to be built. Possible choices of NE type are: OLT, ONT, ONU, or NT. The inputs **supplierName** and **version** identify the NE vendor and the hardware version of the NE to be built. The input parameter **location** designates the physical location of the planned NE. The input **serialNum** provides a unique string corresponding to the NE. The input **alarmSeverityProfiles** identifies profiles to configure alarm severity for individual alarms to be reported by the NE. The input **thresholdDataProfiles** identifies profiles to be used when configuring threshold values to generate TCAs from the NE. The input **slotAssignmentList** identifies acceptable plug-in unit assignments for the slots of the NE. If a slot number is not mentioned in the list, then the assumption is made that no assignment has been made for it and, therefore, there are no restrictions on the type of card that can be placed in the slot. The same is true if the value of the plugIn is the empty string. The input **port** identifies the PON port of the OLT if the NE under construction is either an ONT or ONU. The input **port** identifies the port of the ONU serving the NT if an NT is under construction. In case of building an OLT, the value for **port** is null. The input **modelCode** provides a unique string identifying the type of network resource. This input parameter is primarily of interest when pre-provisioning an ONT or NT. The input **systemTitle** identifies an operator-defined label to be applied to the node. The input **softwareVersions** identifies the versions of software to be used by the node. The input **nEUserLabel** provides a unique operator designation for the constructed NE. The input **externalTime** sets current time expressed in generalized time for the NE. The input **systemTiming** identifies the input clock source of the NE that will be used for timing synchronisation. Whenever unspecified is passed for the "enum" data type **systemTiming** from the OMS, the NE shall use its default timing source. The input **administrationDomain** identifies the domain to which the NE belongs.

The return value of type **ManagedEntityIdType** provides an identifier for the new NE created by the operation.

9.2.1.2 assignUserLabelsToNE

This operation assigns operator administrative designations to NEs. This operation is required when the OMS first learns of the NE through autodiscovery.

The operation signature for **assignUserLabelsToNE** is shown below:

```
void assignUserLabelsToNE (
    in SerialNumType serialNum,
    in UserLabelType nEUserLabel,
    in AdministrationDomainType administrationDomain)
    raises (InvalidSerialNumSyntax, DuplicateSerialNumber,
    DuplicateUserLabel, AccessDenied);
```

The input **serialNum** identifies specific NE. The input **nEUserLabel** provides an operator assigned name for the NE. The input **administrationDomain** identifies the domain to which the NE is assigned.

The return value is of type **void**.

9.2.1.3 modifyNode

The operation initiates reconfiguration and update of specific parameters associated with a NE.

The operation signature for **modifyNode** is shown below:

```
void modifyNode (
    in ManagedEntityIdType managedEntityId,
    in SlotAssignmentSeqType newSlotAssignmentList,
    in NameSeqType newAlarmSeverityProfiles,
    in NameSeqType newThresholdDataProfiles,
    in ManagedEntityIdType port,
    in string newModelCode,
    in UserLabelType newNEUserLabel,
    in ExternalTimeType externalTime,
    in AdministrationDomainType administrationDomain)
    raises(UnknownManagedEntity, UnknownNE, InvalidSlotAssignmentList,
    UnknownProfiles, DuplicateUserLabel, AccessDenied,
    InvalidExternalTime, ProfileSuspended );
```

The input **managedEntityId** identifies the target NE to be modified. The **newSlotAssignmentList** identifies a new assignment of acceptable plug in units for the slots of the NE. If a slot number is not mentioned in the list, then the assumption is made that there is no change for the assignment of that slot. If the value of the plugIn is the empty string then there are no restrictions on the type of card that can be placed in the slot and if a plug-in unit is subsequently removed from the slot no removal alarms are raised. The input **alarmSeverityProfiles** identifies profiles to configure alarm severity for individual alarms to be reported by the NE. The input **thresholdDataProfiles** identifies profiles to be used when configuring threshold values to generate TCAs from the NE. The input **port** identifies PON port on the OLT if there is a change in the relationship between the planned ONT or ONU and the OLT port. The input **port** identifies the ONU port if there is a change in the relationship between the planned NT and ONU port. The input **newModelCode** provides a unique string identifying the type of network resource. This input parameter is primarily of interest when modified node is an ONT or NT. The input **newNEUserLabel** provides a new User Label to be applied to the NE. The input **externalTime** gives a new current reference time for the NE. The input **administrationDomain** identifies the domain to which the NE is reassigned

The return value is of type **void**.

9.2.1.4 deleteNode

This operation deletes the NE from the Supplier Management System and cancels a previous preprovisioning request. As a consequence of this operation, all managed entities automatically created as a result of the corresponding buildNode operation are deleted as well. It will raise **RemainingContainedManagedEntities** exception if there are any other contained managed entities still present such as service connections. Any bandwidth reservations associated with this node are also deleted.

The operation signature for **deleteNode** is shown below:

```
void deleteNode (
    in ManagedEntityIdType managedEntityId)
    raises (UnknownNE, RemainingContainedManagedEntities, AccessDenied,
    RemainingReservations, RemainingSubnetworkConnections);
```

The input **managedEntityId** identifies the NE to be deleted by this operation.

The return value is of type **void**.

9.2.1.5 modifyPort

The operation modifies parameters of the designated port identified by **physicalPathTPIId**.

The operation signature for **modifyPort** is shown below:

```
void      modifyPort      (
    in ManagedEntityIdType physicalPathTPIId,
    in NameSeqType newAlarmSeverityProfiles,
    in NameSeqType newThresholdDataProfiles,
    in NameSeqType newPortProfiles,
    in string newFrameFormat,
    in AdministrativeStateType administrativeState,
    in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
    in LoopbackLocationIdSeqType newLoopbackLocationIds,
    in unsigned long newInterfaceSpeed,
    in unsigned long aRCTimer)
    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
    InterfaceSpeedNotChangeable, ProfileSuspended );
```

The input **physicalPathTPIId** identifies the port to be modified. The input **alarmSeverityProfiles** identifies profiles to configure alarm severity for individual alarms to be reported by the NE. The input **thresholdDataProfiles** identifies profiles to be used when configuring threshold values to generate TCAs from the NE. The input parameter **newPortProfiles** identifies profiles used to finish the provisioning of the port. Examples of such profiles include but are not limited to the following: ATMNetworkAccessProfile, UNIInfo, EthernetProfile, CESServiceProfile, MACBridgeServiceProfile, LESServiceProfile, AAL1Profile, and AAL5Profile. The input **newFrameFormat** identifies a new frame format that will be terminated and generated over the physical port provided the frame format is configurable. The input **administrativeState** specifies the modified setting for this parameter. The input **newLoopbackLocationIds** identifies new location identifiers for loopback associated with the physical port. The input **newInterfaceSpeed** identifies a new interface speed of the physical port provided this is configurable. The input parameter **aRCTimer** provides the non-negative time (in seconds) for which the network resource initially detects a valid signal before raising any communication alarms on the port.

The return value is of type **void**.

9.2.1.6 buildPlugInUnit

This operation builds a plug-in unit in the Supplier Management System as part of the preprovisioning activities. A set of managed entities are automatically created in the management information model maintained by the Supplier Management System as a result of this operation. Depending on the type of plug-in unit, various termination points are automatically created. The slot assignment of the containing node is automatically updated to include the change requested by this operation.

The operation signature for **buildPlugInUnit** is shown below:

```
ManagedEntityIdType buildPlugInUnit (
    in ManagedEntityIdType nEId,
    in NameType alarmSeverityProfile,
    in UserLabelType plugInUnitUserLabel,
    in string modelCode,
    in AdministrativeStateType administrativeState,
    in ManagedEntityIdType equipmentHolder)
    raises (UnknownNE, DuplicateUserLabel,
    AccessDenied, UnknownManagedEntity,
    InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
    InvalidSlotAssignmentList, UnknownProfiles,
    ProfileSuspended );
```

The input **nEId** provides a unique name for the NE that contains the plug-in unit to be built. The input **plugInUnitUserLabel** gives an identifier for the plug-in unit to be built. The input **alarmSeverityProfile** identifies the profile to configure alarm severity assignments for equipment

failures relating to the plug-in unit. The input **modelCode** identifies the type of the plug-in unit. The input **administrativeState** specifies the initial setting for this parameter. The input **equipmentHolder** identifies the location of the slot where the built plug-in unit will occupy.

The return value of type **ManagedEntityIdType** provides a unique identifier for the constructed circuit pack.

9.2.1.7 modifyPlugInUnit

The operation modifies attributes of the plug-in unit in the Supplier Management System.

The operation signature for **modifyPlugInUnit** is shown below:

```
ManagedEntityIdType modifyPlugInUnit (  
    in ManagedEntityIdType plugInUnitId,  
    in NameType alarmSeverityProfile newAlarmSeverityProfile,  
    in string newModelCode,  
    in ManagedEntityIdType newEquipmentHolder,  
    in UserLabelType newPlugInUnitUserLabel,  
    in AdministrativeStateType newAdministrativeState)  
    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,  
    InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,  
    InvalidSlotAssignmentList, InvalidUserLabelSyntax,  
    ProfileSuspended);
```

The input **plugInUnitId** identifies the managed entity to be modified by this operation. The input **newAlarmSeverityProfile** changes the profile to configure alarm severity assignments for equipment failures relating to the plug-in unit. The input **newModelCode** changes the type of the plug-in unit. The input **newEquipmentHolder** changes the slot assignment for the plug-in unit. The input **newPlugInUserLabel** gives a new User Label for the plug-in unit. The input **newAdministrativeState** specifies the modified setting for this parameter.

The return value of type **ManagedEntityIdType** provides a unique identifier for the modified circuit pack.

9.2.1.8 deletePlugInUnit

This operation deletes a plug-in unit from the Supplier Management System. As a consequence of this operation, all managed entities automatically created as a result of the corresponding **buildPlugInUnit** operation are deleted as well. It will raise **RemainingSubnetworkConnections** exception if there are any connections still present. It is employed to remove preprovisioning information that is no longer desired by the operator.

The operation signature for **deletePlugInUnit** is shown below:

```
void deletePlugInUnit (  
    in ManagedEntityIdType plugInUnitId)  
    raises (UnknownManagedEntity, RemainingSubnetworkConnections,  
    AccessDenied, RemainingReservations);
```

The input **plugInUnitId** identifies the plug-in unit to be deleted.

The return value is of type **void**.

9.2.1.9 buildProtectionGrouping

This operation builds a protectionGrouping in the Supplier Management System. No port can belong to more than one protection grouping and must be provisioned prior to this operation. Outside the scope of IF1, the supplier and operator have come to a mutual understanding of the valid protection arrangements supported by the supplier equipment. If a protected port is listed in the protectionUnitList, there must also be at least one protecting port. All ports must have the same physical path characteristics.

The operation signature for **buildProtectionGrouping** is shown below:

```
ManagedEntityIdType buildProtectionGrouping (  
    in ProtectionParameterType protectionParameters,  
    in ProtectionUnitSeqType protectionUnitList)  
    raises (InvalidProtectionScheme, AccessDenied );
```

The input **protectionParameters** provides the characteristics for the protection scheme. The input **protectionUnitList** identifies the protecting and protected ports on the network resource.

The return value of type **ManagedEntityIdType** provides a unique identifier for the constructed relationship between protected and protecting ports.

9.2.1.10 modifyProtectionParameters

This operation modifies the protection schema for the identified protection grouping. If either m (= number of protecting ports) or n (= number of protected ports) is decreased, it will be necessary for the OMS to consider invoking modifyProtectionUnitList to avoid an invalid protection schema exception. If either m or n is increasing, it is not necessary to invoke modifyProtectionUnitList.

The operation signature for **modifyProtectionParameters** is shown below:

```
void modifyProtectionParameters (  
    in ManagedEntityIdType protectionGroupingId,  
    in ProtectionParameterType newProtectionParameters)  
    raises (UnknownManagedEntity, InvalidProtectionScheme,  
    AccessDenied);
```

The input **protectionGroupingId** identifies the managed entity to be modified by this operation. The input **newProtectionParameters** provides a replacement for the existing protection parameters.

The return value is of type **void**.

9.2.1.11 modifyProtectionUnitList

This operation either adds to or removes from the list of protected and protecting ports for the identified protection grouping. A removal of all protecting ports can only occur if all protected ports are also removed. An addition of protected or protecting ports cannot exceed the bounds of m or n specified within the data structure of protectionParameters associated with the protection grouping. No port can belong to more than one protection grouping and must be provisioned prior to this operation. All ports must have the same physical path characteristics.

The operation signature for **modifyProtectionUnitList** is shown below:

```
void modifyProtectionUnitList (  
    in ManagedEntityIdType protectionGroupingId,  
    in ProtectionUnitSeqType deltaProtectionUnitList,  
    in boolean addDeleteInd)  
    raises (UnknownManagedEntity, InvalidProtectionScheme,  
    AccessDenied);
```

The input **protectionGroupingId** identifies the managed entity to be modified by this operation. The input **deltaProtectionUnitList** provides the changes to the protected and protecting ports. The input parameter **addDeleteInd** shows whether or not the change is an addition or removal.

The return value is of type **void**.

9.2.1.12 deleteProtectionGrouping

This operation deletes a protection grouping of ports from the Supplier Management System. As a consequence of this operation, all managed entities automatically created as a result of the corresponding buildProtectionGrouping operation are deleted as well.

The operation signature for **deleteProtectionGrouping** is shown below:

```
void deleteProtectionGrouping (  
    in ManagedEntityIdType protectionGroupingId)  
    raises (UnknownManagedEntity, AccessDenied);
```

The input **protectionGroupingId** identifies the protection grouping to be deleted.

The return value is of type **void**.

9.2.1.13 buildBridge

This operation builds a MAC Bridge in the Supplier Management System. All ports must be provisioned prior to this operation. All UNI ports must be LAN ports. This operation is not required if all UNI LAN ports on a network element automatically belong to the same bridge and the default settings for the bridge parameters is desired by the operator.

The operation signature for **buildBridge** is shown below:

```
ManagedEntityIdType buildBridge (  
    in NameType mACBridgeProfile,  
    in ManagedEntityIdType uplinkPort,  
    in ManagedEntityIdSeqType uNIPortList)  
    raises (UnknownProfiles, AccessDenied,  
    UnknownManagedEntity, ProfileSuspended);
```

The input **mACBridgeProfile** provides the characteristics for the bridge conforming to ANSI/IEEE 802.1D. The input **uplinkPort** identifies the OLT physical port interfacing with the IP layer backbone network. This parameter has the value empty string when there is no such interfacing on the OLT. The input **uNIPortList** identifies the UNI physical ports associated with the bridge.

The return value of type **ManagedEntityIdType** provides a unique identifier for the constructed bridge.

9.2.1.14 modifyBridgeProfile

This operation modifies the characteristics of the bridging function by changing the MAC Bridge Service Profile associated with the bridge.

The operation signature for **modifyBridgeProfile** is shown below:

```
void modifyBridgeProfile (  
    in ManagedEntityIdType bridgeId,  
    in NameType newMACBridgeProfile)  
    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,  
    ProfileSuspended);
```

The input **bridgeId** identifies the managed entity to be modified by this operation. The input **newMACBridgeProfile** provides a replacement for the existing bridge profile.

The return value is of type **void**.

9.2.1.15 modifyBridgePortList

This operation either adds to or removes from the list of UNI ports belonging to the bridge grouping. A removal of all UNI ports can occur. A UNI port cannot be removed if there exists an active service connection associated with it. If every UNI LAN port must belong to the same bridge, then this operation is not needed.

The operation signature for **modifyBridgePortList** is shown below:

```
void modifyBridgePortList (
    in ManagedEntityIdType bridgeId,
    in ManagedEntityIdSeqType deltaUNIPortList,
    in boolean addDeleteInd)
    raises (UnknownManagedEntity, RemainingSubnetworkConnections,
    AccessDenied);
```

The input **bridgeId** identifies the managed entity to be modified by this operation. The input **deltaUNIPortList** provides the changes to the UNI ports desired. The input parameter **addDeleteInd** shows whether or not the change is an addition or removal.

The return value is of type **void**.

9.2.1.16 deleteBridge

This operation deletes a bridge provisioning from the Supplier Management System. As a consequence of this operation, all managed entities automatically created as a result of the corresponding buildBridge operation are deleted as well. A bridge cannot be deleted if there are outstanding active subnetwork connections associated with it.

The operation signature for **deleteBridge** is shown below:

```
void deleteBridge (
    in ManagedEntityIdType bridgeId)
    raises (UnknownManagedEntity, AccessDenied,
    RemainingSubnetworkConnections);
```

The input **bridgeId** identifies the bridge to be deleted.

The return value is of type **void**.

9.2.1.17 buildVPNetworkCTP

This operation builds a VP Network CTP in the Supplier Management System. The port containing this VP Network CTP must be provisioned prior to this construction. This operation is used to construct VP Network CTPs in order to support installation testing. Both build and delete operations are needed.³

The operation signature for **buildVPNetworkCTP** is shown below:

```
ManagedEntityIdType buildVPNetworkCTP (
    in ManagedEntityIdType port,
    in short vPI,
    in NameType trafficDescriptorProfileName,
    in ATMOverbookingFactorType overbookingFactor,
    in UserLabelType userLabel,
    in SegmentEndpointIndType segmentEndpointInd)
    raises (UnknownProfiles, AccessDenied,
    UnknownManagedEntity, ParameterViolation,
    ProfileSuspended);
```

The input **port** identifies the ATM interface on the network resource. The input **vPI** gives the value of the index. The input **trafficDescriptorProfileName** identifies the ATM traffic descriptor profile associated with the CTP. The parameter **overbookingFactor** gives the percentage overbooking factor that can be used in the call admission control algorithms for any PVC have the same VPI value for the ATM interface port. The input parameter **userLabel** provides an operator-provided name for the VPNetworkCTP. The input parameter **segmentEndpoint** identifies whether or not the CTP is a segment endpoint.

³ Delete operation is provided in 9.2.1.18.

The return value of type **ManagedEntityIdType** provides unique identifier for the constructed CTP.

9.2.1.18 deleteVPNetworkCTP

This operation deletes a VP Network CTP provisioning from the Supplier Management System. As a consequence of this operation, all managed entities automatically created as a result of the corresponding buildVPNetworkCTP operation are deleted as well.

The operation signature for **deleteVPNetworkCTP** is shown below:

```
void deleteVPNetworkCTP (  
    in ManagedEntityIdType vPNetworkCTP)  
    raises (UnknownManagedEntity, AccessDenied);
```

The input **vPNetworkCTP** identifies the bridge to be deleted.

The return value is of type **void**.

9.2.1.19 createdNodesGet

This operation retrieves the list of network elements that have been constructed through invocation of this interface.

The operation signature of **createdNodesGet** is provided below:

```
ManagedEntityIdSeqType createdNodesGet () raises (AccessDenied);
```

There is no input parameter.

The return value is of type **ManagedEntityIdSeqType** which provides the list nodes that have been constructed through invocation of the Q834:Builder interface.

9.2.1.20 Exceptions

The exception **AccessDenied** is raised when the system is not granted access to the interface object.

The exception **DuplicateSerialNumber** is raised if there exists other equipment of the same type with this serial number.

The exception **DuplicateUserLabel** is raised if the User Label provided in the request has been used to label another NE or plug-in unit. In other words, the Supplier Management System is responsible for policing User Labels assigned for NEs and plug-in units within its management jurisdiction.

The exception **InterfaceSpeedNotChangeable** is raised if the physical port cannot support the new interface speed or if the speed is not configurable.

The exception **InvalidEquipmentCode** is raised if the equipment code does not conform to syntax agreed to by the operator and supplier.

The exception **InvalidExternalTime** is raised if the External Time specified is not valid.

The exception **InvalidProtectionScheme** is raised if the network resource does not support the protection parameters specified in context with the port listing or if the protection units are ports of dissimilar physical path characteristic.

The exception **InvalidSerialNumSyntax** is raised if the syntax of the serial number provided violates the supplier syntax.

The exception **InvalidSlotAssignmentList** will be raised if the designated equipmentHolder cannot accept the requested type of plug-in unit.

The exception **InvalidUserLabelSyntax** is raised if the User Label provided violates business rules of syntax defined by the operator and implemented in the Supplier Management System.

The exception **ParameterViolation** is raised when the VPI is out of range or a duplicate.

The exception **ProfileSuspended** is raised when profile(s) named in the invocation has been suspended for use within the Supplier Management System by the OMS or operator.

The exception **RemainingContainedManagedEntities** will be raised if there are any remaining managed entities contained by the managed entity to be deleted.

The exception **RemainingReservations** will be raised if there are remaining resource reservations associated with the managed entity to be deleted.

The exception **RemainingSubnetworkConnections** will be raised if the managed entity which is being deleted contains one or more valid Subnetwork Connections.

The exception **SlotAlreadyAssigned** is raised if the requested slot is already provisioned.

The exception **UnknownManagedEntity** is raised if the identified managed entity is unknown to the Supplier Management System.

The exception **UnknownNE** is raised if the identified NE is unknown to the Supplier Management System.

The exception **UnknownProfiles** is raised if the profile name provided is unknown to the Supplier Management System and cannot be retrieved from the Profile Object Repository.

The exception **UnknownSlot** is raised if the requested slot is unknown to the NE.

The exception **UnknownSystemTimingSource** is raised if the External time source is unknown to the Supplier Management System or NE.

The exception **UnrecognizedVersion** is raised if the Equipment version provided does not match known values.

9.3 Q834Common module

This module contains the definitions of data types used by more than one of the other IDL modules in this Recommendation. It also provides three groupings of constants. Many of the data types mentioned in Q834Common are imported from ITU-T Rec. X.780. The most important aspect of this IDL file is the definition of ManagedEntityIdType. The definition is extracted from the IDL file and presented below for discussion purposes.

```
struct NamingComponentType {
    string type; // managed entity type
    string id;
};

typedef sequence<NamingComponentType> RDNTType;
typedef sequence<RDNTType> RDNSeqType;
typedef sequence<NameType> NameSeqType;

enum IdType {
    none,
    x780_fineGrained,
    x780_coarseGrained
};

typedef RDNTType MEIdType;
```

```

struct ManagedEntityIdType {
    IdType id;
    MEIdType mEId;
};

```

ManagedEntityIdType is a return value for many of the operations throughout this Recommendation. Examining the definition of this data type (reading from bottom to top) shows that the Supplier Management System returns a value providing a reference either to a fine-grain Managed Object, a coarse-grain Façade, or a RDN identifier for a data structure within the Supplier Management System. In all three cases, the fundamental syntax for the mEId is a sequence of pairs of naming components which is the syntax employed by ITU-T Rec. X.780 for Managed Objects and the syntax employed by ITU-T Rec. X.780.1 for Façades.

9.3.1 ProbableCauseConst Module and ProbableCause interface

This module definition follows the pattern specified in ITU-T Rec. X.780 and uses the same syntax for the values of probable cause. It provides some technology-specific values in addition to the values defined in ITU-T Rec. X.780.

9.3.2 MonitoringParameter interface

This interface provides the names for monitored performance and traffic parameters to be used by interfaces AlarmEventSupplier, ImpairmentPersistence and ProfileConsumer. The MonitoringParameter values are expressed as a string of values and are part of filterable data in a structured event.

9.3.3 RecordSetType interface

This interface provides the data value(s) to be used for recordset data type value by interfaces ReportController and RecordSetMgr. The recordSet type values are expressed as unsigned short values. The Values 1-99 are reserved for HistoryDataType.

9.3.4 PhysicalLayerLoopback interface

This interface provides the data value(s) to be used for loopbackTestType data type value by the TestActionPerformer interface. The recordSet type values are expressed as unsigned short values.

9.4 ControlArchive module

The Supplier Management System provides the functionality to manage logs for specific groups of events including the clearing of the contents of the logs. The Privileged User can create, initialize, suspend, resume, and remove event logs. The Supplier Management System also provides the functionality to control the short-term archiving of performance monitoring and traffic monitoring reports, including the clearing of contents of these record sets. This function also includes the reporting of status for current logs or statistics record sets. The majority of short-term archives within the Supplier Management System are created automatically when the Supplier Management System is first instantiated. The names of these archives, identified by ManagedEntityIdType syntax, can be provided by the supplier to the operator or obtained by invocation of the operation recordSetListGet with creationModeType value "initialList" and iterative invocations of the operation getStatusAttributes for each record set identified in the first operation.

9.4.1 RecordSetMgr interface

9.4.1.1 createLog

CreateLog operation is used to create a RecordSet in the Supplier Management System for the purposes of archiving event information.

The signature of **createLog** is as follows:

```
ManagedEntityIdType createLog (  
    in UserLabelType recordSetUserLabel,  
    in AdministrativeStateType administrativeState,  
    in NameType filterName,  
    in FullActionType fullAction,  
    in MaxSizeType maxSize,  
    in SizeThresholdType sizeThreshold)  
    raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

The input parameter **recordSetUserLabel** uniquely identifies the recordSet within the Supplier Management System. The input parameter **administrativeState** specifies if the new log is initialized for recording of event information. The input parameter **filterName** explains the entrance criteria determining what event notifications are recorded in the log to be created. If the **filterName** is unrecognized by the Supplier Management System, it looks up the IOR for the filterObject using **filterName** by consulting the directory of Naming Service. Using the IOR, the system is then able to retrieve the entrance criteria details from CosNotifyFilter interface available on Notification Service. The input parameter **fullAction** specifies the behaviour of the recordset when it reaches its maximum size. The input parameter **maxSize** specifies the maximum size of the recordSet. The input parameter **sizeThreshold** specifies the threshold on size of the RecordSet which triggers the Supplier Management System to generate an alarm/event.

The return value is of type **ManagedEntityIdType** and provides the identifier for the log archive created with this operation.

9.4.1.2 createArchive

CreateArchive operation is used to create a recordSet for storing history data. The archiving will halt automatically its recording when it hits the maximum size.

The signature of **createArchive** is as follows:

```
ManagedEntityIdType createArchive (  
    in UserLabelType recordSetUserLabel,  
    in AdministrativeStateType administrativeState,  
    in RecordKindType recordKind,  
    in MaxSizeType maxSize)  
    raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

The input parameter **recordSetUserLabel** uniquely identifies the record Set within the Supplier Management System. The input parameter **administrativeState** specifies if the new archive is initialized for recording of appropriate records. The input parameter **recordKind** identifies the type of record to be stored in the record set. The input parameter **maxSize** specifies the maximum size of the recordSet.

The return value is of type **ManagedEntityIdType** and provides the identifier for the archive created with this operation.

9.4.1.3 getStatusAttributes

At any time the Operator or OMS can view the current status of a short-term archive.

The signature of **getStatusAttributes** is as follows:

```
RecordSetStatusType getStatusAttributes (  
    in ManagedEntityIdType recordSetId)  
    raises (AccessDenied, UnknownRecordSet);
```

The input parameter **recordSetId** uniquely identifies the recordSet within the Supplier Management System.

The return value is of type **RecordSetStatusType** which contains the current status of the recordset.

9.4.1.4 suspendArchive

Once the short-term archive has been created and initialized for use, the OMS can suspend its use.

The signature of **suspendArchive** is as follows:

```
void suspendArchive (  
    in ManagedEntityIdType recordSetId)  
    raises (AccessDenied, UnknownRecordSet);
```

The input parameter **recordSetId** uniquely identifies the recordSet within the Supplier Management System.

The return value is of type **void**.

9.4.1.5 resumeArchive

This operation resumes recording in an archive or initializes the recording within a recordSet that has been constructed in a locked state.

The signature of **resumeArchive** is as follows:

```
void resumeArchive (  
    in ManagedEntityIdType recordSetId)  
    raises (UnknownRecordSet, AccessDenied);
```

The input parameter **recordSetId** uniquely identifies the recordSet within the Supplier Management System.

The return value is of type **void**.

9.4.1.6 deleteArchive

This operation removes an archive from the Supplier Management System.

The signature of **deleteArchive** is as follows:

```
void deleteArchive (  
    in ManagedEntityIdType recordSetId )  
    raises (UnknownRecordSet, AccessDenied );
```

The input parameter **recordSetId** identifies the name that the Privileged User wishes to use to identify the log to be created in subsequent interactions.

The return value is of type **void**.

9.4.1.7 purgeArchive

The operation removes the information contained within a specified archive. However, the archive continues its recording.

The signature of **purgeArchive** is as follows:

```
void purgeArchive (  
    in ManagedEntityIdType recordSetId )  
    raises (UnknownRecordSet, AccessDenied );
```

The input parameter **recordSetId** identifies the name that the Privileged User wishes to use to identify the log to be created in subsequent interactions.

The return value is of type **void**.

9.4.1.8 selectRecords

After the creation of an archive, the OMS may select some of the records from it.

The signature of **selectRecords** is:

```
RecordSeqType selectRecords (  
    in FilterType selectionFilter,  
    in ManagedEntityIdType recordSetId)  
    raises (UnknownRecordSet, Timeout, NoSuchRecords,  
    AccessDenied, TooManyRecords);
```

The input parameter **recordSetId** identifies the name that the Privileged User wishes to use to identify the log to be created in subsequent interactions. The input parameter **selectionFilter** is defined by specific record set as defined in the data structure.

The return value is of type **RecordSeqType** supplying the requested information.

9.4.1.9 recordSetListGet

This operation allows an OMS to get a complete listing of record sets managed by the Supplier Management System.

The signature of **recordSetListGet** is:

```
ManagedEntityIdSeqType recordSetListGet (  
    in CreationModeType creationMode)  
    raises (AccessDenied);
```

The input parameter **creationMode** identifies the way the archive was creation, i.e., whether by an operator or automatically as part of Supplier Management System initialization.

The return value is of type **ManagedEntityIdSeqType** listing the names for the short-term archives of the Supplier Management System.

9.4.1.10 changeUserLabel

After the creation of a short-term archive, the OMS can change the User Label assigned to the archive.

The signature of **changeUserLabel** is:

```
void changeUserLabel (  
    in ManagedEntityIdType recordSetId,  
    in UserLabelType newUserLabel)  
    raises (UnknownRecordSet, AccessDenied, DuplicateUserLabel);
```

The input parameter **recordSetId** identifies the archive. The input parameter **newUserLabel** identifies the new name for a specific archive.

The return value is of type **void**.

9.4.1.11 Exceptions

The exception **AccessDenied** is raised when the system is not granted access to the interface object.

The exception **DuplicateUserLabel** is raised if the User Label provided in the request has been used to label another archive. In other words, the Supplier Management System is responsible for policing User Labels assigned for record sets within its management jurisdiction.

The exception **LockedAlready** is raised if the set value of the administrativeState is "locked".

The exception **NoSuchRecords** is raised if no records among the designated record sets matches the selection criteria.

The exception **RecordSetExists** is raised if the archive defined by the parameters of creation request already exists in the Supplier Management System.

The exception **Timeout** is raised if the process duration reached a system-defined timeout before the process could complete.

The exception **TooManyRecords** is raised if the number of records selected for retrieval produces a response to the request that exceeds a predetermined size.⁴

The exception **UnknownRecordSet** is raised if the designated record set is unknown to the Supplier Management System.

9.5 SoftwareDownload module

The software download process consists of four phases: delivery, distribution, installation (commit), and activation. The Supplier Management System supports these stages for the download of software generic programs, software upgrades, and software maintenance changes (patches) to NEs in this module. The Supplier Management System can accept requests involving one or multiple NEs at once. All software management activities can be individually scheduled. Any request for downloading software generics is accompanied by security credentials whereby the Supplier Management System is allowed to communicate with the source server. Which security credentials are being used depends on implementation: If the software load is resident on the EMS and the EMS will push it (through file transfer) to the NE, then the credentials are for the NE. If the software load is resident on the EMS and the NE will pull, then the credentials are for the EMS. If the software load is resident on a separate repository, then the security credentials are for the separate repository and the NE pulls the load from the repository.

Requests for a software management activity may generate the creation of a software management tracking object. This lifecycle of this object must extend until all associated activities are completed successfully or unsuccessfully. The object is only deleted after a predefined retention period and the length of time of the retention period is agreed to outside the scope of this interface. The retention period for the tracking object should be generous enough to include any revert activity desired. If the tracking object has been automatically deleted by the Supplier Management System, then the revert activity is viewed as download activity of a previous software load. The OMS can also delete this tracking object before its automatic termination if desired.

The Supplier Management System creates a log record for every software distribution activity including delivery, commitment, and activation, regardless of the result of the activity. The record contains detailed results for every affected target including start/stop times and success or failure indication.

The Supplier Management System maintains a current listing of all software activities in progress.

This module also includes support for retrieval of hardware and software version information from installed NEs. It is also possible to verify that a software set may be downloaded to a NE or to a specific plug-in unit.

9.5.1 DownloadMgr interface

9.5.1.1 deliverDistSWGlobal

This operation requests the Supplier Management System to download software generics from a software source machine for the purpose of software upgrades and software maintenance changes (patches) to NEs. The Supplier Management System can accept requests for one or multiple NEs at once. The operation is best effort, meaning the Supplier Management System attempts to execute the delivery and distribution to all specified targets. As appropriate, it continues to attempt to

⁴ This size to be agreed upon by supplier and operator in advance.

deliver and distribute software to all named targets, even when it encounters a failure in either process for a specific target. The activity completion log is used to record successes and failures of both the delivery and distribution attempts. The `SoftwareDownloadTrackingObject` continues to be available for commit, activate attempts and those activities are only attempted for targets where the delivery and distribution was successful. The `SoftwareDownloadTrackingObject` is retained beyond the successful activation stage (in part to allow for revert activities if required).

Subsequently, if a new ONT is ranged, or if a broken fibre connection is repaired, or if a new plug-in unit is installed, and in each of these cases the active software load differs from that specified in this operation, then the Supplier Management System (or network resource) will be responsible for upgrading the software automatically.

The signature of the operation **deliverDistSWGloab** is provided below:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWGloab (
    in FilenameSeqType softwareSet,
    in DCNAddressType softwareSourceAddr,
    in UserIdType userId,
    in PasswordType password,
    in ManagedEntityIdSeqType deliverDistTargets)
raises (CommFailure, UnrecognisedTarget,
    InsufficientMemory, SoftwareLoadHWMismatch,
    SourceUnreachable, UnknownSoftwareLoad, Timeout,
    AccessDenied, DeniedAccess);
```

The input parameter **softwareSet** identifies the software load and where it is located (file names and full path) to be downloaded. The input parameter **softwareSourceAddr** provides the DCN address of the server where the set of software resides. The input parameters **userId** and **password** provide the login mechanism to the softwareSource (assuming such security credentials are needed). The input parameter **deliverDistTargets** indicates the list of OLTs where the software is to be delivered.

The return value of type **SoftwareDownloadTrackingObjectIdType** provides a reference to be used when attempting to commit, activate or check the status of the delivery and distribution process. The result of software download is logged by the Supplier Management System.

9.5.1.2 deliverDistSWSpecific

This operation is the same as `deliverDistSWGloab` except the scope is within a single NE/PlugInUnit/slot as described in the input parameter `distributionTarget`.

The operation signature of **deliverDistSWSpecific** is provided below:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
    in FilenameSeqType softwareSet,
    in DCNAddressType softwareSourceAddr,
    in UserIdType userId,
    in PasswordType password,
    in TargetType deliverDistTarget)
raises (CommFailure, UnrecognisedTarget,
    InsufficientMemory, SoftwareLoadHWMismatch,
    SourceUnreachable, UnknownSoftwareLoad, Timeout,
    AccessDenied, DeniedAccess);
```

The input parameter **softwareSet** identifies the software load and where it is located (file names and full path) to be downloaded. The input parameter **softwareSourceAddr** provides the DCN address of the server where the set of software resides. The input parameters **userId** and **password** provide the login mechanism to the softwareSource (assuming such security credentials are needed). The input parameter **deliverDistTarget** indicates the specific target at the NE/PlugInUnit/slot level. The components for the value of this parameter are described in Table 3.

Table 3/Q.834.4 – deliverDistTarget details

Field name	Description
containingSystem	The system is identified by the Managed Entity Id of the headend OLT.
containingNE	Identifies the destination network resource. When the value is the empty sequence, the operation distributes the software to all NEs in the system.
plugInUnitType	Identifies the plugInUnitType. When this value is the empty string, the distribution target is all suitable plugInUnitTypes within the containingNE.
slot	Identifies the slot. When this value is the empty sequence, then any suitable slot is considered.

The return value of type **SoftwareDownloadTrackingObjectIdType** provides a correlation key reference to be used when attempting to commit, activate or check the status of the delivery and distribution process. The result of software download is logged by the Supplier Management System.

9.5.1.3 deleteSoftwareDownloadTrackingObject

This operation allows the OMS to notify the Supplier Management System that the software tracking object specified is no longer needed.

The operation signature of **deleteSoftwareDownloadTrackingObject** is provided below:

```
void deleteSoftwareDownloadTrackingObject (
    in SoftwareDownloadTrackingObjectIdType id)
    raises (UnknownSoftwareDownloadTrackingObject, AccessDenied,
    SoftwareTrackingObjectInUse);
```

The input parameters **id** identifies the software tracking object.

The return value is of type **void**.

9.5.1.4 commit

This operation requests the Supplier Management System to install (commit) the downloaded software to target locations.

The operation signature of **commit** is provided below:

```
void commit (
    in SoftwareDownloadTrackingObjectIdType id,
    in TargetType commitTarget)
    raises (InstallationFailure, UnknownSoftwareDownloadTrackingObject,
    AccessDenied, UnrecognisedTarget);
```

The input parameters **id** provides a reference to software download tracking object. The input parameter **commitTarget** specifies the specific target at the NE/PlugInUnit/slot level. The **commitTarget** can be a subset of the original target.

The return value is of type **void**.

9.5.1.5 activate

This operation activates installed software at target locations.

The operation signature of **activate** is provided below:

```
void activate (
    in SoftwareDownloadTrackingObjectIdType id,
    in TargetType activateTarget)
```

```
raises (UnknownSoftwareDownloadTrackingObject,
SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
UnrecognisedTarget);
```

The input parameters **id** provides a reference to software download tracking object. The input parameter **activateTarget** specifies the specific target at the NE/PlugInUnit/slot level. The activateTarget can be a subset of the original target.

The return value is of type **void**.

9.5.1.6 revert

This operation activates older installed software at target locations. The revert operation is only meaningfully invoked after activation of a new software load. The SoftwareDownloadTrackingObjectId refers to the most recent software download affecting the named Target. If revert completes successfully, then the active version becomes the standby version.

The operation signature of **revert** is provided below:

```
void revert (
    in SoftwareDownloadTrackingObjectIdType id,
    in TargetType revertTarget)
raises (UnknownSoftwareDownloadTrackingObject,
SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
UnrecognisedTarget, InvalidSoftwareTrackingObject);
```

The input parameters **id** provides a reference to software download tracking object. The input parameter **revertTarget** specifies the specific target at the NE/PlugInUnit/slot level. The revertTarget can be a subset of the original target.

The return value is of type **void**.

9.5.1.7 getStatus

This operation requests the status of software activities.

The operation signature of **getStatus** is provided below:

```
DownloadStatusSeqType getStatus (
    in SoftwareDownloadTrackingObjectIdType id)
raises (UnknownSoftwareDownloadTrackingObject, AccessDenied);
```

The input parameters **id** provides a reference to the software download tracking object.

The return value of type **DownloadStatusSeqType** provides the progress status for all the download activities tracked by the Software Download Tracking Object.

9.5.1.8 scheduleDeliverDist

This operation schedules the delivery and distribution of software to specified targets. The distribution of the software within the specified targets is up to the supplier's implementation.

The operation signature of **scheduleDeliverDist** is provided below:

```
SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
    in FilenameSeqType softwareSet,
    in DCNAddressType softwareSourceAddr,
    in UserIdType userId,
    in PasswordType password,
    in ManagedEntityIdSeqType deliverDistTargets,
    in GeneralizedTimeType deliverDistStartTime)
raises (SoftwareLoadHWMismatch, AccessDenied, InvalidStartTime );
```

The input parameter **softwareSet** identifies the software load and where it is located (file names and full path) to be downloaded. The input parameter **softwareSourceAddr** provides the DCN address of the server where the set of software resides. The input parameters **userId** and **password** provide the login mechanism to the softwareSource (assuming such security credentials are needed). The input parameter **deliverDistTargets** indicates the list of OLTs where the software is to be delivered. The input parameter **deliverDistStartTime** provide the scheduled time to start.

The return value of type **SoftwareDownloadTrackingObjectIdType** provides a reference to the scheduled process that can be used to track the progress of the process or to cancel it.

9.5.1.9 scheduleCommit

This operation schedules the installation (commit) of software to predetermined targets.

The operation signature of **scheduleCommit** is provided below:

```
void scheduleCommit (
    in SoftwareDownloadTrackingObjectIdType
        deliverDistSoftwareDownloadTrackingObjectId,
    in GeneralizedTimeType commitStartTime)
    raises (UnknownSoftwareDownloadTrackingObject,
        SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

The input parameters **deliverDistSoftwareDownloadTrackingObjectId** provides a reference to software delivery tracking object. The input parameter **commitStartTime** provides the time to start this scheduled activity.

The return value of type **void**.

9.5.1.10 scheduleActivate

This operation schedules the activation of software to predetermined targets.

The operation signature of **scheduleActivate** is provided below:

```
void scheduleActivate (
    in SoftwareDownloadTrackingObjectIdType id,
    in GeneralizedTimeType activateStartTime)
    raises (UnknownSoftwareDownloadTrackingObject,
        SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

The input parameters **id** provides a reference to the software delivery tracking object. The input parameter **activateStartTime** provides the time to start this activity.

The return value of type **void**.

9.5.1.11 cancelScheduledSoftwareActivity

This operation cancels all subsequently scheduled software download activities associated with this tracking object.

The operation signature of **cancelScheduledSoftwareActivity** is provided below:

```
void cancelScheduledSoftwareActivity (
    in SoftwareDownloadTrackingObjectIdType id)
    raises (UnknownSoftwareDownloadTrackingObject,
        ActivityCompleted, ActivityInProgress, AccessDenied);
```

The input parameters **id** provides a reference to software activity tracking object.

The return value is of type **void**.

9.5.1.12 scheduledSoftwareDownloadTrackingObjectListGet

This operation retrieves the list of outstanding scheduled activities for software download.

The operation signature of **scheduledSoftwareDownloadTrackingObjectListGet** is provided below:

```
SoftwareDownloadTrackingObjectIdSeqType  
scheduledSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

There is no input parameter.

The return value is of type **SoftwareDownloadTrackingObjectIdSeqType** which provides the list of outstanding scheduled activities.

9.5.1.13 onDemandSoftwareDownloadTrackingObjectListGet

This operation retrieves the list of outstanding non-scheduled activities for software download.

The operation signature of **onDemandSoftwareDownloadTrackingObjectListGet** is provided below:

```
SoftwareDownloadTrackingObjectIdSeqType  
onDemandSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

There is no input parameter.

The return value is of type **SoftwareDownloadTrackingObjectIdSeqType** which provides the list of outstanding non-scheduled activities.

9.5.1.14 Exceptions

The exception **AccessDenied** is raised when the system is not granted access to the interface object.

The exception **ActivityCompleted** is raised when the software activity has been executed and cannot be cancelled.

The exception **ActivationFailure** is raised if the software activation process failed even though the software has been installed (i.e., commit was successful).

The exception **ActivityInProgress** is raised when the software activity has been initiated and cannot be cancelled.

The exception **CommFailure** is raised when the DCN between Supplier Management System and OLT or the communication between OLT and source ONT is down.

The exception **DeniedAccess** is raised if the access to NE is denied as a result of access control restrictions.

The exception **InstallationFailure** is raised if the Software installation process failed.

The exception **InsufficientMemory** is raised if there is insufficient memory on the NE to load the software.

The exception **InvalidSoftwareTrackingObject** is raised if the referenced software tracking object is not the most recent associated with the installation of a software load on the NE.

The exception **InvalidStartTime** is raised if the start time is before the current system time.

The exception **SoftwareLoadHWMismatch** is raised if the designated software may not be loaded onto the equipment hardware since the version of the hardware cannot accept the software load.

The exception **SoftwareNotYetInstalled** is raised when activation is requested and the software has not been installed yet.

The exception **SoftwareTrackingObjectInUse** is raised when there are outstanding software activities tracked by this object and it may not be deleted.

The exception **SourceUnreachable** is raised if the server holding the software load to be downloaded could not be reached by the OLT.

The exception **Timeout** is raised if the process duration reached a system-defined timeout before the process could complete.

The exception **UnknownSoftwareLoad** is raised when the specified software set cannot be found.

The exception **UnknownSoftwareDownloadTrackingObject** is raised when the software download process is unknown to the Supplier Management System.

The exception **UnrecognizedTarget** is raised when the designated software in the Secure File Server is unknown to the Supplier Management System.

9.5.2 VersionRepository interface

9.5.2.1 retrieveVersions

This operation retrieves all version information (both software and hardware) for a network resource.

The operation signature of **retrieveVersions** is provided below:

```
VersionsSeqType retrieveVersions (  
    in ManagedEntityIdType containingManagedEntityId)  
    raises (CommFailure, UnknownManagedEntity, AccessDenied);
```

The input parameters **containingManagedEntityId** identifies the network resource.

The return value is of type **VersionsSeqType** and provides a listing of the hardware and software versions associated with this network resource.

9.5.2.2 validateNEVersion

This operation requests the Supplier Management System to validate if the proposed software is compatible with the NE.

The operation signature of **validateNEVersion** is provided below:

```
boolean validateNEVersion (  
    in ManagedEntityIdType managedEntityId,  
    in VersionType proposedSoftware)  
    raises (UnknownNE, AccessDenied);
```

The input parameters **managedEntityId** identifies the NE.

The input parameter **proposedSoftware** identifies the proposed software to be validated.

The return value is of type **boolean**.

9.5.2.3 validatePlugInVersion

This operation requests the Supplier Management System to validate if the proposed software is compatible with the plugInUnit.

The operation signature of **validatePlugInVersion** is provided below:

```
boolean validatePlugInVersion (  
    in ManagedEntityIdType plugInUnitId,  
    in VersionType proposedSoftware)  
    raises (UnknownManagedEntity, AccessDenied);
```

The input parameter **plugInUnitId** identifies the plugInUnit.

The input parameter **proposedSoftware** identifies the proposed software to be validated.

The return value is of type **boolean**.

9.5.2.4 Exceptions

The exception **AccessDenied** is raised when the system is not granted access to the interface object.

The exception **CommFailure** is raised when the DCN between Supplier Management System and OLT or the communication between OLT and source ONT is down.

The exception **UnknownManagedEntity** is raised if the Identifier for plugInUnit or port is unknown to the Supplier Management System.

9.6 EventPublisher module

On receipt of processed configuration, performance, or fault event information provided by other use cases within the Supplier Management System and based on rules concerning publication, the Supplier Management System queues and channels event information to all interested consumers, including Operators and OMS(s).

9.6.1 AlarmEventSupplier interface

The purpose of the interface is to announce alarm events to the Operator Management System via the OMG Notification Service. This interface has no operations. However, it does provide the fixed header mapping as well as the filterable data mappings for the structured event object used to push event information through the event channel of the OMG Notification Service. Both sets of mappings shall follow the guidelines specified in ITU-T Rec. X.780 for the Notifications interface.

In the fixed header, the **domain_type** is set to "telecommunications", the **type_name** is set to "Alarm", and the **event_name** is set to a constant string that has one of the following values: "Communications Alarm", "Environmental Alarm", "Equipment Alarm", "Processing Error Alarm", or "Quality of Service Alarm".

The mapping in the filterable data consists of pairs of items. The first component in the pair is a string identifier for a data name and the second is the value for that data element. The string identifiers are constants that are defined in this interface. Furthermore, the filterable data pairs must occur in a specific order.

The order of the filterable items is as follows:

- AlarmEmittingMEId
- EventTime
- ProbableCause
- SpecificProblems
- PerceivedSeverity
- ServiceAffectingInd
- BackUpStatus
- BackUpManagedEntityId
- ThresholdInfo
- NotificationIdentifier
- CorrelatedNotifications
- StateChangeDefinition
- AdditionalText

The value for ProbableCause has syntax ProbableCauseType and assumes one of the specific values defined in ITU-T Rec. X.780 or q834_4::Q834Common::ProbableCauseConst.

The value for ThresholdInfo has syntax MonitoredParameterType and assumes one of the specific values defined in q834_4::Q834Common::MonitoredParameter. The null string is supplied for this value for all event_names except "QualityOfServiceAlarm".

The syntax and interpretation for all other data types and interpretation of their use is found in ITU-T Rec. X.780.

9.6.2 SecurityEventSupplier

The purpose of the interface is to announce security events to the Operator Management System via the OMG Notification Service. This interface has no operations. However, it does provide the fixed header mapping as well as the filterable data mappings for the structured event object used to push event information through the event channel of the OMG Notification Service. Both sets of mappings shall follow the guidelines specified in ITU-T Rec. X.780 for the Notifications interface.

In the fixed header, the **domain_type** is set to "telecommunications", the **type_name** is set to "SecurityEvent", and the **event_name** is set to a constant string that has one of the following values: "IntegrityViolation", "OperationalViolation", "PhysicalViolation", "SecurityEventViolation", or "TimeDomainViolation".

The mapping in the filterable data consists of pairs of items. The first component in the pair is a string identifier for a data name and the second is the value for that data element. The string identifiers are constants that are defined in this interface. Furthermore, the filterable data pairs must occur in a specific order.

The order of the filterable items is as follows:

- EventEmittingMEId
- EventTime
- SecurityAlarmCause
- SecurityAlarmDetector
- ServiceUser
- ServiceProvider
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

The syntax for data types and interpretation of their use is found in ITU-T Rec. X.780.

9.6.3 DiscoveryEventSupplier

The purpose of the interface is to announce changes to installed equipment to the Operator Management System via the OMG Notification Service. This interface has no operations. However, it does provide the fixed header mapping as well as the filterable data mappings for the structured event object used to push event information through the event channel of the OMG Notification Service.

In the fixed header, the **domain_type** is set to "telecommunications", the **type_name** is set to "DiscoveryEvent", and the **event_name** is set to a constant string that has one of the following values: "ManagedEntityCreation" or "ManagedEntityDeletion".

The mapping in the filterable data consists of pairs of items. The first component in the pair is a string identifier for a data name, and the second is the value for that data element. The string identifiers are constants that are defined in this interface. Furthermore, the filterable data pairs must occur in a specific order.

The order of the filterable items for an event_name of "ManagedEntityCreation" is as follows:

- ManagedEntityType
- EventTime
- ManagedEntityAttributeValues
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

The value for ManagedEntityType has syntax of EquipmentType that indicates the type of inventory data discovered. The interface defines constants for various NE types as well as plugInUnits, equipmentHolders and software.

The value for EventTime has syntax of GeneralizedTimeType and refers to the moment when the discovery condition was detected by the network.

The value for ManagedEntityAttributeValues has syntax MStruct. However, the data type supplied for this value is one of a set of data structures defined in the module. The data item ManagedEntityType identifies the type of struct that is passed in this value in the structured event object.

The value for NotificationIdentifier has syntax NotificationIdentifierType and it provides a reference sequence number for the event. The value for CorrelatedNotifications has syntax of CorrelatedNotificationType and supplies a list of reference numbers for other event notifications provided by the Supplier Management System for associated inventory conditions. If there are no related notifications, the value of the empty set is supplied.

Finally, the value for AdditionalText has syntax string. This data item provides a location to pass any textual miscellaneous information from the Supplier Management System concerning the inventory change condition. If there is no additional information, the null string will be passed.

The order of the filterable items for an event_name of "ManagedEntityDeletion" is as follows:

- ManagedEntityType
- EventTime
- ManagedEntityAttributeValues
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

The value for ManagedEntityType has syntax of EquipmentType that indicates the type of inventory data removed. The interface defines constants for various NE types as well as plugInUnits and equipmentHolders.

The value for EventTime has syntax of GeneralizedTimeType and refers to the moment when the removal condition was detected by the network.

The value for ManagedEntityAttributeValues has syntax MStruct. However, the data type supplied for this value is one of a set of data structures defined in the module. The data item ManagedEntityType identifies the type of struct that is passed in this value in the structured event object.

The value for NotificationIdentifier has syntax NotificationIdentifierType and it provides a reference sequence number for the event. The value for CorrelatedNotifications has syntax of CorrelatedNotificationType and supplies a list of reference numbers for other event notifications

provided by the Supplier Management System for associated inventory change conditions. If there are no related notifications the value of the empty set is supplied.

Finally, the value for **AdditionalText** has syntax string. This data item provides a location to pass any textual miscellaneous information from the Supplier Management System concerning the inventory change condition. If there is no additional information, the null string will be passed.

Insertion or removal of a plug-in unit always creates a notification no matter what provisioning state surrounds the plug-in unit.

9.7 MIBTransfer module

This module manages the process for collection of system configuration data to be used to restore a system in the event that a catastrophic failure condition has occurred. The collection of the configuration data can be done either on a real-time or on a scheduled basis. It also provides status information for backup and restoral processes in progress. The results of backup and restore processes are logged by the Supplier Management System. Any request to backup or restore configuration data is accompanied by security credentials whereby the Supplier Management System or OLT is allowed to communicate with the external server. While the backup is progress, all provisioning requests involving the designated OLT system are rejected/blocked. Transfer Tracking Objects are automatically deleted by the Supplier Management System once the associated file transfer(s) have finished and results (whether successful or unsuccessful) are recorded in the completion log.

9.7.1 MIBMover interface

9.7.1.1 startBackup

This operation initiates an immediate backup of system configuration data from a system and/or Supplier Management System to a backup server destination.

The operation signature for **startBackup** is shown below:

```
TransferTrackingObjectIdType startBackup (  
    in ManagedEntityIdType nEManagedEntityId,  
    in DCNAddressType destinationServerAddr,  
    in UserIdType userId,  
    in PasswordType password,  
    in FilenameType destinationFile,  
    in boolean overwriteExistingFile)  
    raises (UnknownNE, UnknownDestinationServer, CommFailure,  
    EquipmentFailure, DeniedAccess, AccessDenied);
```

The input parameter **nEManagedEntityId** identifies the system to be backed up. The input parameter **destinationServerAddr** identifies the data communication networking address for the server that is the destination of the backup. The input parameter **userId** identifies the user login to the destination server. The input parameter **password** is the password to access the destination server. The input parameter **destinationFile** provides a full directory location for the backup file. Finally, the parameter **overwriteExistingFile** indicates whether or not the backup should allow the overwriting of a pre-existing file with the same destination directory location.

The return value is of type **TransferTrackingObjectIdType** providing a correlation key to track the progress status of the backup process.

9.7.1.2 getBackupStatus

This operation provides the capability to retrieve the status of a backup process.

The operation signature for **getBackupStatus** is shown below:

```
StatusAttributeSeqType getBackupStatus (  
    in TransferTrackingObjectIdType id)  
    raises (UnknownBackupProcess, AccessDenied);
```

The input parameter **id** identifies the backup process.

The return value is of type **StatusAttributeSeqType** and gives the status of the previously requested backup process.

9.7.1.3 scheduleBackup

This operation schedules backup processes.

The operation signature for **scheduleBackup** is shown below:

```
TransferTrackingObjectIdType scheduleBackup(  
    in ManagedEntityIdType nEManagedEntityId,  
    in UserIdType userId,  
    in PasswordType password,  
    in UserLabelType schedulerName,  
    in DCNAddressType destinationServerAddr,  
    in FilenameType destinationFile,  
    in boolean overwriteExistingFile)  
    raises (UnknownNE, UnknownScheduler, UnknownDestinationServer,  
    InvalidScheduler, AccessDenied);
```

The input parameter **nEManagedEntityId** identifies the OLT to be backed up. The input parameter **schedulerName** is the scheduler name to be used for backup. The input parameter **userId** identifies the user login to the destination server. The input parameter **password** is the password to access the destination server. The parameter **destinationServerAddr** indicates the DCN address for the destination server where the data is to be backed up. The input parameter **destinationFile** provides a full directory location for the backup file. Finally, the parameter **overwriteExistingFile** indicates whether or not the backup should allow the overwriting of a pre-existing file with the same destination directory location.

The return value is of type **TransferTrackingObjectIdType** and provides a correlation key to be used when attempting to track the status of the non-real time at some later point.

9.7.1.4 modifyBackupSchedule

This operation cancels all subsequent backup processes for a system based on a scheduler. This operation will not interrupt a backup process in progress.

The operation signature for **modifyBackupSchedule** is shown below:

```
void modifyBackupSchedule (  
    in TransferTrackingObjectIdType id,  
    in UserLabelType newSchedulerName)  
    raises (UnknownBackupProcess, AccessDenied, UnknownScheduler,  
    InvalidScheduler);
```

The input parameter **id** identifies the scheduled process to be modified. The input parameter **newSchedulerName** identifies the new time triggers.

The return value is of type **void**.

9.7.1.5 cancelScheduledBackup

This operation cancels all subsequent backup processes for a system based on a scheduler. This operation will not interrupt a backup process in progress.

The operation signature for **cancelScheduledBackup** is shown below:

```
void cancelScheduledBackup (
    in TransferTrackingObjectIdType id)
    raises (UnknownBackupProcess, AccessDenied);
```

The input parameter **id** identifies the scheduled process to be cancelled.

The return value is of type **void**.

9.7.1.6 abortBackup

This operation aborts a backup process in progress whether scheduled or not. Subsequent scheduled backup process are not affected by this operation.

The operation signature for **abortBackup** is shown below:

```
void abortBackup (
    in TransferTrackingObjectIdType id)
    raises (UnknownBackupProcess, CommFailure, EquipmentFailure,
    AccessDenied);
```

The input parameter **id** identifies the backup process to be aborted.

The return value is of type **void**.

9.7.1.7 startRestore

This **startRestore** operation provides functionality to restore a system based on a backed up copy of configuration data.

The operation signature for **startRestore** is shown below. The configuration data is resident on an external secure server.

```
TransferTrackingObjectIdType startRestore (
    in ManagedEntityIdType nEManagedEntityId,
    in DCNAddressType sourceServerAddr,
    in UserIdType userId,
    in PasswordType password,
    in FilenameType sourceFile)
    raises ( UnknownNE, CommFailure, EquipmentFailure,
    UnknownSourceServer, DeniedAccess,
    SoftwareLoadHardwareMismatch, AccessDenied );
```

The input parameter **nEManagedEntityId** identifies the OLT to be restored. The input parameter **sourceServerAddr** identifies the data communication networking address for the server that is the destination of the backup. The input parameter **userId** identifies the user login to the destination server. The input parameter **password** is the password to access the destination server. The parameter **sourceFile** indicates the file where the data has been backed up, from which the restoral will take place.

The return value is of type **TransferTrackingObjectIdType** and provides a correlation key to be used when attempting to track the status of the requested restoration.

9.7.1.8 getRestoreStatus

This operation provides the status of a restoral process.

The operation signature for **getRestoreStatus** is shown below:

```
StatusAttributeSeqType getRestoreStatus (
    in TransferTrackingObjectIdType id)
    raises (UnknownRestoreProcess, AccessDenied);
```

The input parameter **id** identifies the restoration process.

The return value is of type **StatusAttributeSeqType** and provides the progress state of the restoration process.

9.7.1.9 transferTrackingObjectIdListGet

This operation retrieves the list of outstanding system MIB transfers, both backup and restore.

The operation signature of **transferTrackingObjectIdListGet** is provided below:

```
TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
    raises (AccessDenied);
```

There is no input parameter.

The return value is of type **TransferTrackingObjectIdSeqType** which provides the list of outstanding system MIB transfers, both backup and restore.

9.7.1.10 Exceptions

The exception **AccessDenied** is raised when the system is not granted access to the interface object.

The exception **CommFailure** is raised when the DCN between Supplier Management System and OLT or the communication between OLT and source ONT is down.

The exception **DeniedAccess** is raised if the access to NE is denied.

The exception **EquipmentFailure** is raised when the equipment where the data is backed up from is in failure condition.

The exception **InvalidScheduler** is raised when the given scheduler is inappropriate for use in this operation or out-of-date.

The exception **SoftwareLoadHardwareMismatch** is raised when the source software does not match with the destination hardware.

The exception **UnknownBackupProcess** is raised when the given TransferTrackingObjectId related backup process is not found.

The exception **UnknownDestinationServer** is raised when the IP address of destination server is not found.

The exception **UnknownNE** is raised when the OLT is unknown to the Supplier Management System.

The exception **UnknownRestoreProcess** is raised when the given TransferTrackingObjectId related restoral process is unknown.

The exception **UnknownScheduler** is raised when the given scheduler name is not found.

The exception **UnknownSourceServer** is raised when the source server is unknown to the Supplier Management System.

9.8 PerformanceManager module

The Supplier Management System shall provide for the activation and deactivation of performance data or traffic measurements reporting on individual termination points contained in the NEs as required by the operator or OMS for OLTs that have been installed. Requirements for the Supplier Management System found in this Recommendation also include provision for the setting of threshold values and describe automatic reporting of performance measurements to the Supplier

Management System when thresholds have been crossed. When there is an occurrence of a set of Threshold Crossing Alerts associated with a single performance degradation condition, the Supplier Management System shall analyse and correlate the alert events within its domain to the best of its ability, determine the underlying root cause of the problem, and store this information in a log. If several occurrences of the same root cause impairment are detected within a period of time, the Supplier Management System shall prepare a QoS alarm record for publication any interested consumer (Operator or OMS). If the Supplier Management System can collect all the history data records for all the monitoring points⁵ for all the network elements in its management domain, there is no need for reporting control. In this case, it is assumed that reporting is constantly available.

9.8.1 ImpairmentPersistence interface

Throughout this Recommendation, it is assumed and required that profile objects that are of threshold data type⁶ are groupings of threshold values can be associated with one, and only one, type of monitoring point.

9.8.1.1 setSlidingWindowParameters

This operation sets the sliding window parameters for one, some or all monitored parameters in an NE. This setting is applied for all points monitoring the parameters. Subsequently, the Supplier Management System generates a QoS alarm if a monitoring point detects a TCA **persistenceMinimum** times within **totConsecutiveIntvls** consecutive monitoring intervals. This operation is best effort.

The exception **CommFailure** is raised if the Supplier Management System is unable to communicate with the NE identified in the operation, and if the sliding window parameter values are set directly on network resources.

This operation is used to establish system-wide default settings if the indicator **sysScopeInd** has the value **TRUE** and the **nManagedEntityId** is the identifier for an OLT. This means that the sliding window parameter values are to be applied to all monitoring points of any equipment components currently or potentially belonging to the system with the named OLT headend. If the OLT has lost communications with any subtending network resource prior to or during invocation of this operation, it is up to supplier implementation to guarantee that the settings are applied once communication is established. If the indicator **sysScopeInd** has the value **TRUE** and the **nManagedEntityId** is the identifier for an ONU, then the settings are viewed to apply to the ONU and all NTs subtending from it. As equipment components are added to the system with headend of OLT or ONU, the Supplier Management System automatically sets these parameter values.

If **sysScopeInd** has the value **FALSE**, then the sliding window parameter settings are to be applied only to the indicated resource. The settings are still applied to both existing and potential equipment components of the node.

The operation signature for **setSlidingWindowParameters** is shown below:

```
void setSlidingWindowParameters (
    in ManagedEntityIdType nManagedEntityId,
    in MonitoredParameterSeqType monitoredParameterList,
    in short totConsecutiveIntvls,
    in short persistenceMinimum,
    in boolean sysScopeInd)
    raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
    AccessDenied, CommFailure);
```

⁵ History data records and monitoring points are defined in ITU-T Rec. Q.834.1.

⁶ Of type 21 as specified in the ProfileManager module.