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.

CONTENTS

Page

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 Name$_1$ – 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

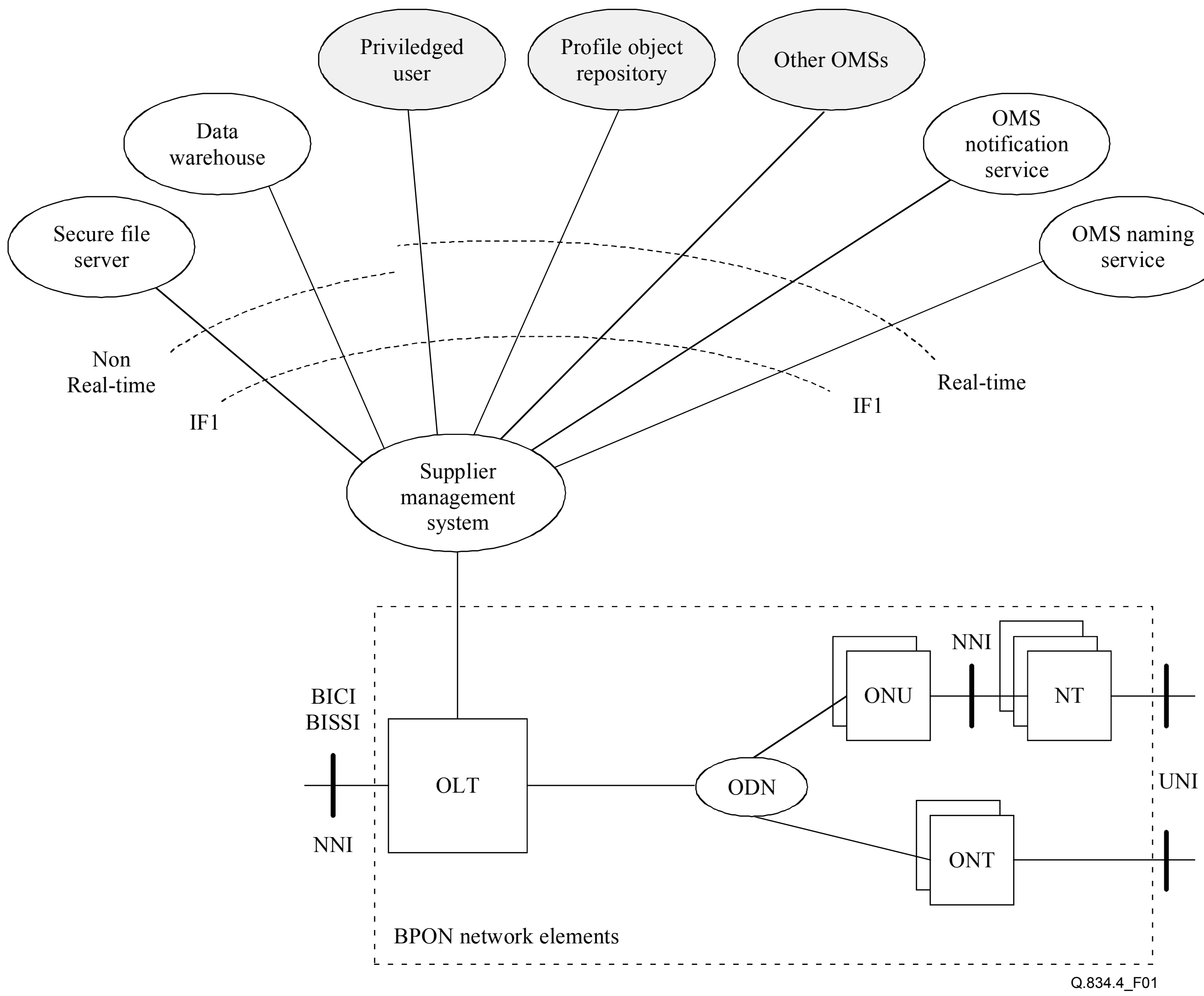
In this Recommendation, when there is a mention of "null string", it is meant to be an empty string and not a null object supported by languages such as JAVA. Similarly, when a "null sequence" is mentioned, a sequence with zero elements is passed.

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.



BICI	Broadband Inter-Carrier Interface	ODN	Optical Distribution Network
BISSI	Broadband Inter-Switching System Interface	OLT	Optical Line Terminal
BPON	Broadband Passive Optical Network	ONT	Optical Network Terminal
NNI	Network to Network Interface	ONU	Optical Network Unit
NT	Network Terminal	UNI	User-Network Interface

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.

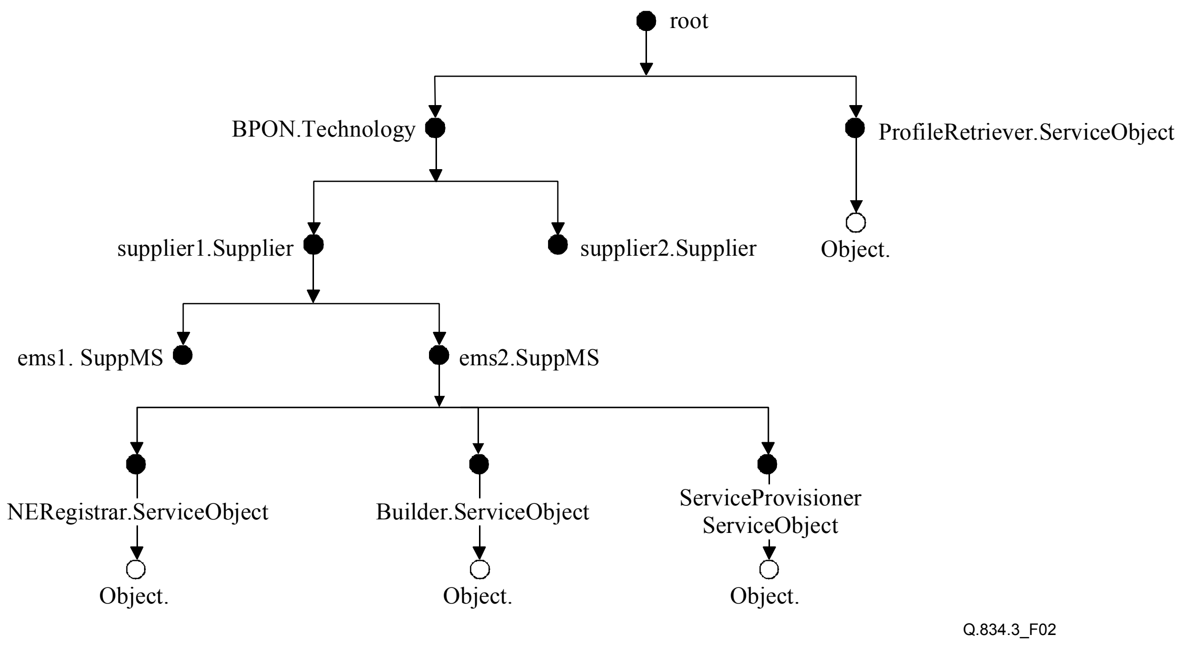


Figure 2/Q.834.4 – Naming graph

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 t**argetAdditions** 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 **physicalPathTPId**.

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

```
void     modifyPort    (
                in ManagedEntityIdType physicalPathTPId,
                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 **physicalPathTPId** 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 in 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> RDNType;
typedef sequence<RDNType> RDNSeqType;
typedef sequence<NameType> NameSeqType;

enum IdType {
     none,
     x780_fineGrained,
     x780_coarseGrained
};

typedef RDNType 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 **deliverDistSWGlobal** is provided below:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
                        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 deliverDistSWGlobal 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 **SoftwareDownloadTrackingObjectIdTypeId** 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 MEstruct. 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 MEstruct. 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 points[5] 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 type[6] 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 nEManagedEntityId 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 nEManagedEntityId 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 nEManagedEntityId,
            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.

The input parameter **nEManagedEntityId** identifies the network element. The input parameter **monitoredParameterList** identifies the parameters to be monitored. Values for the parameters are provided in q834_4::Q834Common::MonitoredParameter. The input parameter **totConsecutiveIntvls** identifies the value of total consecutive monitored intervals. The input parameter **persistenceMinimum** identifies the minimum number of occurrences of the TCA for the monitoredParameter. The input **sysScopeInd** indicates whether or not the sliding window settings should be applied to all network elements subtending from the initial one designated in the operation.

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

9.8.1.2 setSpecificSlidingWindowParameters

This operation is similar to the operation setSlidingWindowParameters except the scope of assignment of settings is limited to a specific monitoring point identified in the operation.

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

```
void setSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
            in short persistenceMinimum,
            in boolean allowGlobalOverwrite)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, UnknownManagedEntity, CommFailure, EquipmentFailure);
```

The input parameter **nEManagedEntityId** identifies the network element managed by the Supplier Management System. The input parameter **monitoringPoint** identifies the specific monitoring point for setting the sliding window parameters for a monitored parameter. The input parameter **monitoredParameterList** identifies the parameters to be monitored values for which are defined in Q834Common::MonitoredParameter. The input parameter **totConsecutiveIntvls** identifies the value of total consecutive monitored intervals. The input parameter **persistenceMinimum** identifies the minimum number of occurrences of the TCA for the monitoredParameter. The parameter **allowGlobalOverwrite** indicates if the specific sliding window parameters can be overwritten in a subsequent invocation of **setSlidingWindowParameters** operation.

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

9.8.1.3 getSpecificSlidingWindowParameters

This operation provides a listing of monitoring points and their sliding window settings for a specified parameter.

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

```
SWPValueSeqType getSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterType monitoredParameter)
            raises (UnknownNE, UnknownParameters, CommFailure);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoredParameter** identifies the parameter to be monitored and is defined in q834_4::Q834Common::MonitoredParameter.

The return value is of type **SWPValueSeqType** providing the sliding window assignments for a specific monitoring point.

9.8.1.4 setThreshold

This operation sets the threshold value identified by a profile to a monitoring point in a network element.

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

```
void setThreshold (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in NameType thresholdDataProfileName )
            raises (UnknownNE, AccessDenied,UnknownManagedEntity,
            UnknownProfiles, InvalidAssociation, CommFailure, ProfileSuspended);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoringPoint** identifies the specific monitoring point for setting the sliding window parameters for a monitored parameter. The input parameter **thresholdDataProfileName** provides listing of profile names for threshold values.

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

9.8.1.5 setThresholds

This operation allows the setting of a collection of threshold value, monitoringPointType pairs on a particular NE. The exception CommFailure is raised if the Supplier Management System is unable to communicate with the NE identified in the operation, and if threshold 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 nEManagedEntityId is the identifier for an OLT. This means that the threshold 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 nEManagedEntityId 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 threshold values.

If sysScopeInd has the value FALSE, then the threshold 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 **setThresholds** is shown below:

```
void setThresholds (
                   in ManagedEntityIdType nEManagedEntityId,
                   in boolean sysScopeInd,
                   in ThresholdsSeqType thresholdsList)
                   raises (UnknownNE, UnknownProfiles, AccessDenied,
                   UnknownMonitoringPointTypes, InvalidAssociation,
                   CommFailure, ProfileSuspended );
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **thresholdList** identifies a list of pairs consisting of Threshold Data name and monitoringPointType. The input **sysScopeInd** indicates whether or not the sliding window settings should be applied to all network elements subtending from the initial one designated in the operation.

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

9.8.1.6 getThresholdValues

This operation provides a listing of monitoring points and their threshold data setting names for a specified monitoring point type.

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

```
MonitoringPointThresholdsSeqType getThresholdValues (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoringKindType monitoringPointType)
          raises (UnknownNE, UnknownMonitoringPointTypes, CommFailure);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoringPointType** identifies the monitoring point on which the retrieval is based.

The return value is of type **MonitoringPointThresholdsSeqType** and provides the setting values for threshold crossings.

9.8.1.7 getSystemThresholdsSetting

This operation retrieves the system default values for Threshold Data.

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

```
ThresholdsSeqType getSystemThresholdsSetting (
                   in ManagedEntityIdType nEManagedEntityId)
                   raises (AccessDenied, UnknownManagedEntity);
```

The input parameter **nEManagedEntityId** uniquely identifies the system for which default settings are requested.

The return value is of type **ThresholdsSeqType** which provides the list of system defaults.

9.8.1.8 getSystemSWSettings

This operation retrieves the system default sliding window settings.

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

```
ParameterSettingSeqType getSystemSWSettings (
                   in ManagedEntityIdType nEManagedEntityId)
                   raises (AccessDenied, UnknownManagedEntity);
```

The input parameter **nEManagedEntityId** uniquely identifies the system for which default sliding window settings are requested.

The return value is of type **ParameterSettingSeqType** which provides the list of system defaults.

9.8.1.9 Exceptions

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

The exception **CommFailure** is raised when there was a DCN link failure between the NE and the Supplier Management System fails.

The exception **EquipmentFailure** is raised when the equipment is in failure condition and the settings cannot be applied.

The exception **InvalidAssociation** is raised when the given profile cannot be applied to a monitoring point.

The exception **IntervalCountTooLarge** is raised when the requested intervals exceed the maximum supported by the Supplier Management System. The exception indicates the maximum allowed monitoring intervals supported by the Supplier Management System.

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 **UnknownNE** is raised when the NE mentioned in the request is unknown to the Supplier Management System.

The exception **UnknownManagedEntity** is raised when the monitoring point is unknown to the Supplier Management System.

The exception **UnknownMonitoringPointTypes** is raised when the monitoring point is unknown to the Supplier Management System.

The exception **UnknownParameters** is raised when the given monitored parameter is unknown in 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.

9.8.2 ReportController interface

9.8.2.1 addCustomerMonitoringReporting

This operation adds a historydatatype to be monitored to a specific monitoring point in response to a customer complaint identified by the supplied serviceInstanceId or in order to support CNM services.

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

```
void addCustomerMonitoringReporting (
         in ManagedEntityIdType nEManagedEntityId,
         in ServiceInstanceIdType serviceInstanceId,
         in ManagedEntityIdType monitoringPoint,
         in GeneralizedTimeType stopTime,
         in HistoryDataType historyData,
         in short granularityPeriod)
         raises ( UnknownServiceInstance, AccessDenied, UnknownNE,
         UnknownManagedEntity, CollectionPeriodPast, CollectionLimitation,
         InvalidAssociation, UnknownHistoryDataType, CommFailure);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **serviceInstanceId** identifies the customer service instance associated with the monitoring point. The input parameter **monitoringPoint** identifies the specific monitoring point for collecting history data. The input parameter **stopTime** identifies the time at which the collection of history data will cease. If the reporting is to be ongoing, then stopTime is given as "0". Reporting also terminates automatically when the service is deleted. The inputParameter **historyData** identifies the type of history data to be collected, the values of which are defined in q834_4::Q834Common::RecordSetType. The input parameter **granularityPeriod** identifies the collection interval period in minutes. When the **stopTime** falls within the granularity period, the history data is collected for the whole granularity period.

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

9.8.2.2 removeCustomerMonitoringReporting

This operation removes all historydatatype collection on behalf of a supplied service instance.

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

```
void removeCustomerMonitoringReporting (
             in ServiceInstanceIdType serviceInstanceId )
             raises ( UnknownServiceInstance, AccessDenied,
             CollectionPeriodPast, CommFailure);
```

The input parameter **serviceInstanceId** identifies the customer service instance associated with the monitoring point.

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

9.8.2.3 selectByServiceInstance

This operation gets all the records available in the Supplier Management System associated with the given serviceInstanceId. The returned list has no redundant records. The Supplier Management System scans all relevant recordsets.

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

```
RecordsSeqType selectByServiceInstance (
             in ServiceInstanceIdType serviceInstanceId,
             in GeneralizedTimeType intervalStartTime,
             in GeneralizedTimeType intervalEndTime)
             raises ( UnknownServiceInstance, AccessDenied);
```

The input parameter **serviceInstanceId** identifies the customer service instance. The input parameter **intervalStartTime** filters out selection of history data records with periodEndTime before this value. The input parameter **intervalEndTime** filters out selection of history data records with periodEndTime after this value.

The return value is of type **RecordsSeqType** that provides a list of history data records associated with performance monitoring reported for this service instance.

9.8.2.4 displayActiveReporting

This operation gets all the monitoring points for which history data is currently being collected for a given serviceInstanceId.

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

```
MonitoringPointSeqType displayActiveReporting (
             in ServiceInstanceIdType serviceInstanceId)
             raises ( UnknownServiceInstance, AccessDenied);
```

The input parameter **serviceInstanceId** identifies the customer service instance.

The return value is of type **MonitoringPointSeqType** and provides the listing of all actively reporting monitoring points.

9.8.2.5 addNewMonitoringReporting

This operation adds a history data type to be monitored on a specific monitoring point.

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

```
void addNewMonitoringReporting  (
             in ManagedEntityIdType nEManagedEntityId,
             in ManagedEntityIdType monitoringPoint,
             in GeneralizedTimeType stopTime,
             in HistoryDataType historyData,
             in short granularityPeriod)
             raises ( AccessDenied, UnknownNE, UnknownManagedEntity,
             CollectionPeriodPast, CollectionLimitation, InvalidAssociation,
             UnknownHistoryDataType, CommFailure);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoringPoint** identifies the specific monitoring point for collecting history data. The input parameter **stopTime** identifies the time at which the collection of history data will cease. If the reporting is to be ongoing, then stopTime is given as "0". Reporting also terminates automatically when the managedEntity (monitoring point) deleted. The inputParameter **historyData** identifies the type of history data to be collected, the values of which are defined in q834_4::Q834Common::RecordSetType. The input parameter **granularityPeriod** identifies the collection interval period in minutes. When the **stopTime** falls within the granularity period, the history data is collected for the whole granularity period.

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

9.8.2.6 selectByMonitoringPoint

This operation gets all the records available in the Supplier Management System associated with the given monitoring point. The returned list has no redundant records.

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

```
RecordsSeqType selectByMonitoringPoint (
              in ManagedEntityIdType monitoringPoint,
              in GeneralizedTimeType intervalStartTime,
              in GeneralizedTimeType intervalEndTime)
              raises ( UnknownManagedEntity, AccessDenied);
```

The input parameter **monitoringPoint** identifies the specific monitoring for which records should be retrieved. The input parameter **intervalStartTime** filters out selection of history data records with periodEndTime before this value. The input parameter **intervalEndTime** filters out selection of history data records with periodEndTime after this value.

The return value is of type **RecordsSeqType** and provides all the history data records stored within the Supplier Management System reported by the monitoring point within the time period specified.

9.8.2.7 createReportingSchedule

This operation adds a scheduled collection of history data to be monitored on a specific monitoring point. The Supplier Management System will collect a historyData record for the monitoring point based on the schedule. The schedule identifies the periodEndTime(s) for the records to be collected.

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

```
void createReportingSchedule (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in HistoryDataType historyData,
              in ServiceInstanceIdType serviceInstance,
              in short granularityPeriod,
              in UserLabelType schedulerName)
              raises (AccessDenied, UnknownNE, UnknownManagedEntity,
              CollectionLimitation, UnknownScheduler, InvalidAssociation,
              UnknownHistoryDataType, InvalidScheduler);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoringPoint** identifies the specific monitoring point for collecting history data. The inputParameter **historyData** identifies the type of history data to be collected, the values of which are defined in q834_4::Q834Common::RecordSetType. The input parameter **serviceInstance** provides an optional reference to a service instance. Empty string value is used if no reference is desired. The input parameter **granularityPeriod** identifies the collection interval period in minutes. The input parameter **schedulerName** identifies the name for a schedule that was created earlier.

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

9.8.2.8 modifyReportingSchedule

This operation modifies the scheduled collection of history data to be monitored on a specific monitoring point. If successful, the change to scheduled collection of history data will occur with the next iteration.

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

```
void modifyReportingSchedule (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in HistoryDataType historyData,
              in UserLabelType newSchedulerName)
              raises (AccessDenied, UnknownNE, UnknownManagedEntity,
              CollectionLimitation, UnknownScheduler, InvalidAssociation,
              UnknownHistoryDataType, InvalidScheduler);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoringPoint** identifies the specific monitoring point for collecting history data. The inputParameter **historyData** identifies the type of history data to be collected, the values of which are defined in q834_4::Q834Common::RecordSetType. The input parameter **newSchedulerName** identifies the name of the new schedule to be applied.

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

9.8.2.9 cancelReportingSchedule

This operation cancels subsequent scheduled reporting of history data where the reporting was requested by the operator previously. It does not interrupt history data reporting in progress or reporting that is supported as default behaviour between the Supplier Management System and the network resources.

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

```
void cancelReportingSchedule (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in HistoryDataType historyData,
              in UserLabelType schedulerName)
              raises (AccessDenied, UnknownNE, UnknownManagedEntity,
              UnknownScheduler, InvalidAssociation, UnknownHistoryDataType);
```

The input parameter **nEManagedEntityId** uniquely identifies the network element managed by the Supplier Management System. The input parameter **monitoringPoint** identifies the specific monitoring point for collecting history data. The inputParameter **historyData** identifies the type of history data to be collected, the values of which are defined in q834_4::Q834Common::RecordSetType. The input parameter **schedulerName** identifies the schedule to be cancelled.

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

9.8.2.10 Exceptions

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

The exception **CommFailure** is returned when there was a DCN link failure between the NE and the Supplier Management System.

The exception **CollectionLimitation** is raised when the Supplier Management System cannot collect data for the given time duration and granularity period due to implementation restrictions.

The exception **CollectionPeriodPast** is raised when period end time is lesser than equal to the current time.

The exception **InvalidAssociation** is raised when the given profile cannot be applied to a monitoringPoint.

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

The exception **UnknownNE** is raised when the NE mentioned in the request is unknown to the Supplier Management System.

The exception **UnknownHistoryDataType** is raised when history data type is unknown in the Supplier Management System.

The exception **UnknownManagedEntity** is raised when the monitoring point is unknown to the Supplier Management System.

The exception **UnknownScheduler** is raised when the named scheduler is unknown to the Supplier Management System.

The exception **UnknownServiceInstance** is raised when the Service Instance is unknown to the Supplier Management System.

9.9 ProfileManager

This module consists of three interfaces: ProfileConsumer, ProfileUsageMgr, and ProfileRetriever. ProfileConsumer specifies the contents of event notification for profile creations within the Profile Object Repository. ProfileUsageMgr supports operations to manage profile settings cached within the Supplier Management System. ProfileRetriever allows Supplier Management System to obtain the values for profile settings.

9.9.1 ProfileConsumer Interface

The purpose of the interface is to announce the existence of new profile settings and to convey those settings to the Supplier Management System. 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 "ProfileEvent", and the **event_name** is set to a constant string that has the value "ProfileCreation" as defined in the interface.

The mapping in the filterable data follows a strategy using a constant string identifier for the filterable data component followed by the value for that data element. Also, the filterable data items are listed in a specific order.

The order of the filterable items is ProfileName, ProfileType, EventTime, ProfileAttributeValues, and NotificationIdentifier. ProfileName value has the syntax of NameType. ProfileType value has the syntax "unsigned short", and it allows the consumer (Supplier Management System) to identify the type of profile created and to unmarshall the attribute values found later in profileStruct. The syntax of ProfileAttributeValues is profileStruct. EventTime has the syntax of GeneralizedTimeType, and it is the time that the profile was created. NotificationIdentifier has the syntax of NotificationIdentifierType. This identifier uniquely labels the profile creation event and is incremented with each profile creation.

9.9.2 ProfileUsageMgr interface

9.9.2.1 reName

This operation provides the capability to rename a profile.

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

```
void reName (
            in NameType oldProfileName,
            in NameType newProfileName)
            raises (UnknownProfiles, AccessDenied, DuplicateProfileName);
```

The input parameter **oldProfileName** is the old profile name to be renamed. The input parameter **newProfileName** is the new name for the profile.

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

9.9.2.2 inUse

This operation returns a boolean value to tell if the profile is in use.

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

```
boolean inUse (
            in NameType profileName)
            raises (UnknownProfiles, AccessDenied);
```

The input parameter **profileName** provides the profile name which is to be checked if it is used by the other party.

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

9.9.2.3 suspendUse

This operation suspends the use of a profile.

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

```
void suspendUse (
              in NameType profileName)
              raises (UnknownProfiles, AccessDenied);
```

The input parameter **profileName** is a profile name which is to be suspended.

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

9.9.2.4 resumeUse

This operation resumes the use of a named profile.

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

```
void resumeUse (
          in NameType profileName)
          raises (UnknownProfiles, AccessDenied);
```

The input parameter **profileName** names the profile whose values are to be again made available for use by the Supplier Management System.

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

9.9.2.5 deleteProfile

This operation deleteProfile provides the capability to remove a profile which is not in use.

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

```
void deleteProfile (in NameType profileName) raises (UnknownProfiles,
AccessDenied, ProfileInUse);
```

The input parameter **profileName** is the name of the profile to be deleted.

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

9.9.2.6 Exceptions

The exception **AccessDenied** is raised when the OMS is not granted access to the requested operation.

The exception **DuplicateProfileName** is raised when duplicate profile name is found.

The exception **ProfileInUse** is raised when deleting a profile, and found when the profile is in use by the other party.

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.

9.9.3 ProfileRetriever interface

9.9.3.1 retrieve

This operation provides the capability to retrieve a profile. This operation is used by the Supplier Management System to retrieve a profile from the Operator management system.

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

```
ProfileInfoType retrieve (
              in NameType profileName)
              raises (UnknownProfiles);
```

The input parameter **profileName** is the name of the profile to be retrieved.

The return value is of type **ProfileInfoType** and provides the attribute values for the named profile.

9.9.3.2 Exceptions

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.

9.10 Registrar module

This module supports the process of bringing a new NE into the management jurisdiction of the Supplier Management System. It supports both initial installations and equipment replacements motivated by maintenance or service upgrades. It explicitly supports the ranging function that is used to measure the round trip delay between the OLT and each ONU or ONT and establish security mechanisms (churning key algorithm) and communication timing including automatic setup of the embedded operations channel.

A set of managed entities is automatically created in the management information model maintained by the Supplier Management System as a result of registration. Depending on the equipment type, instances of OLT, ONT, ONU, APONLink, APONTTP, APONTrail, APONNetworkCTP, APONNetworkTTP, APONLinkConnection, tcAdapterF, vpLinkConnectionF, vpTopologicalLinkF, and physicalPathTPF are automatically created. Depending on supplier implementation as well as what is physically installed, other managed entities are also automatically created.

9.10.1 NERegistrar interface

9.10.1.1 registerNE

This operation registers an NE with a particular instance of Supplier Management System. The NE should be installed at this point. The Managed Entity Id of the NE is returned in order to indicate that the Supplier Management System can perform at least one rudimentary management function.

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

```
ManagedEntityIdType registerNE (
                    in DCNAddressType nEDCNAddress,
                    in UserLabelType nEUserLabel,
                    in AdministrationDomainType administrationDomain)
                    raises (AccessDenied, DCNTimeout, AddressLabelMismatch,
                    DuplicateUserLabel, TooManyNEs, InvalidDCNAddress,
                    DeniedAccess, InvalidUserLabelSyntax);
```

The input parameter **nEDCNAddress** identifies a DCN address to be used to access this NE. The NE should already have been configured with this address. The input parameter **nEUserLabel** provides an operator defined label for the NE. The NE should already have been provisioned with the User Label. The input parameter **administrationDomain** identifies the management domain to which this NE is assigned.

The return value of type **ManagedEntityIdType** identifies the newly registered NE if this operation completes successfully.

9.10.1.2 modifyNEDCNAddress

This operation changes the DCN address of a registered NE. Communications may be temporarily lost between Supplier Management System and NE during this operation. The new DCN address should take effect immediately.

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

```
void modifyNEDCNAddress (
        in ManagedEntityIdType nEManagedEntityId,
        in DCNAddressType newNEDCNAddress)
        raises(AccessDenied, DeniedAccess, AddressLabelMismatch,
        DCNTimeout, CommFailure, UnknownNE, InvalidDCNAddress, BackupInProgress);
```

The input parameter **nEManagedEntityId** identifies the NE to which this operation applies. The input parameter **newDCNAddress** identifies the new DCN address for the NE.

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

9.10.1.3 rangeONTorONU

This operation ranges an ONT or ONU using the serial number of the PON interface of the NE. If the NE being ranged has not been built, then buildNode will be called before ranging occurs. After the ranging has completed successfully a NE synch is initiated. This will update the Supplier Management System with the configuration data from the newly ranged NE.

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

```
ManagedEntityIdType rangeONTorONU(
          in ManagedEntityIdType oLTManagedEntityId,
          in UserLabelType nEUserLabel,
          in SerialNumType serialNum,
          in ManagedEntityIdType port)
          raises(AccessDenied, CommFailure, EquipmentFailure, UnknownNE,
          UnknownPort, MaxSubtendingNodesExceeded, InsufficientPONBW,
          InvalidSerialNumSyntax, APONLayerFailure, DuplicateUserLabel,
          InvalidUserLabelSyntax, BackupInProgress, SynchInProgress );
```

The input parameter **oLTManagedEntityId** identifies the OLT to which this operation applies. The input parameter **nEUserLabel** is a label given to the NE to be ranged. The input parameter **serialNum** identifies the NE serial number. The input parameter **port** identifies the specific PON port on the OLT where the NE is to be ranged.

The return value of type **ManagedEntityIdType** is the managed entity id of the ranged NE.

9.10.1.4 rangeReplacementNE

This operation ranges a replacement ONT or ONU using the serial number of the replacement equipment. All existing service connection information is automatically assigned to the replacement NE. Replacement scenarios occur due to maintenance activities for existing customers and services. The old and new NE User Label can be the same. Outside the scope of IF1 the supplier and the operator have come to an understanding of what type of hardware may replace any existing hardware.

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

```
ManagedEntityIdType rangeReplacementNE(
               in ManagedEntityIdType oldNEManagedEntityId,
               in UserLabelType newNEUserLabel,
               in SerialNumType replacementSerialNum)
               raises (AccessDenied, CommFailure, UnknownNE,
               InvalidSerialNumSyntax, APONLayerFailure, EquipmentFailure,
               InvalidUserLabelSyntax, HWServicesMismatch, DuplicateUserLabel,
               BackupInProgress, SynchInProgress );
```

The input parameter **oldNEManagedEntityId** identifies the NE to be replaced. The input parameter **newNEUserLabel** identifies a label for the new ONT or ONU to be ranged. The input parameter **replacementSerialNum** identifies the NE serial number.

The return value of type **ManagedEntityIdType** is the managed entity id of the newly ranged NE.

9.10.1.5 rangeUpgradeNE

This operation ranges a replacement NE for the purposes of upgrading the hardware. All existing service connection information is automatically assigned to the replacement NE. Additionally, it is assumed that the replacement NE has been preprovisioned (using buildNode operation) and any new service connections have been provisioned or bandwidth has been reserved. The new NE User Label can be the same as the old. Outside the scope of IF1 the supplier and the operator have come to an understanding of what type of hardware may replace any existing hardware.

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

```
ManagedEntityIdType rangeUpgradeNE(
               in ManagedEntityIdType oldNEManagedEntityId,
               in ManagedEntityIdType newNEManagedEntityId,
               in UserLabelType newNEUserLabel,
               in SerialNumType newNESerialNum )
               raises (AccessDenied, CommFailure, APONLayerFailure,
               EquipmentFailure, InvalidUserLabelSyntax, DuplicateUserLabel,
               UnknownNE, HWServicesMismatch, InsufficientPONBW,
               BackupInProgress, SynchInProgress );
```

The input parameter **oldNEManagedEntityId** identifies the NE being replaced. In this case, the type of NE being replaced is either an ONT or an ONU. The input parameter **newNEManagedEntityId** identifies the preprovisioned replacement. All new service connection or bandwidth reservation information is accessible to the Supplier Management System through this identifier. The input parameter **newNEUserLabel** provides a label for the new NE to be ranged. The old and new ONU or ONT User Label can be the same. The input parameter **newNESerialNum** provides the serial number to be used for ranging the replacement.

The return value of type **ManagedEntityIdType** is the managed entity id of the newly ranged NE.

9.10.1.6 moveONTorONU

This operation is used to move an ONT or ONU from one PON to another and also to move all of the associated services.

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

```
ManagedEntityIdType moveONTorONU(
                   in ManagedEntityIdType oldNEManagedEntityId,
                   in ManagedEntityIdType newPONPort)
                   raises(AccessDenied, CommFailure, UnknownNE, UnknownPort,
                   APONLayerFailure,EquipmentFailure, InsufficientPONBW,
                   BackupInProgress, SynchInProgress );
```

The input parameter **oldNEManagedEntityId** identifies the NE to be replaced. The input parameter **newPONPort** identifies the new PON to which these services are being moved.

The return value of type **ManagedEntityIdType** is the managed entity id of the newly ranged NE.

9.10.1.7 getSubtendingNEList

This operation returns all subtending NEs of a particular NE.

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

```
ManagedEntityIdSeqType getSubtendingNEList(
                   in ManagedEntityIdType nEManagedEntityId)
                   raises (UnknownNE, AccessDenied);
```

The input parameter **nEManagedEntityId** identifies the specific NE for which the list should be returned.

The return value of type **ManagedEntityIdSeqType** provides a list of subtending network elements.

9.10.1.8 nEListGet

This operation retrieves the network elements under the management jurisdiction of the Supplier Management System.

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

```
ManagedEntityIdSeqType nEListGet () raises (AccessDenied);
```

There are no input parameters.

The return value is of type **ManagedEntityIdSeqType** which provides the list of network elements managed by the Supplier Management System.

9.10.1.9 deRegisterNE

This operation removes the network element from the management jurisdiction of the Supplier Management System.

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

```
void deRegisterNE (
            in ManagedEntityIdType nE)
            raises (AccessDenied);
```

The input parameter **nE** identifies the network element to remove from the management jurisdiction of the Supplier Management System.

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

9.10.1.10 associateNE

This operation associates preprovisioning information with a network element that has been installed and autodiscovered. The activities of preprovisioning and installation both produce a managed entity within the Supplier Management System that is made available to the OMS. This operation unifies the two managed entities into one.

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

```
ManagedEntityIdType associateNE (
                    in ManagedEntityIdType preProvisionedNE,
                    in ManagedEntityIdType discoveredNE)
                    raises (AccessDenied, UnknownManagedEntity);
```

The input parameter **preProvisionedNE** identifies the information associated with the preprovisioned network element. The input parameter **discoveredNE** identifies the information associated with the installed network element.

The return value of type **ManagedEntityIdType** provides the identification of the merged information.

9.10.1.11 Exceptions

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

The exception **AddressLabelMismatch** is raised when the identified NE does not have the current DCN Address provided in the request.

The exception **APONLayerFailure** is raised when there was an APON protocol ranging failure between the OLT and the designed subtending node.

The exception **BackupInProgress** is raised when the cancellation request is issued while the backup is in progress.

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 **DCNTimeout** is raised when the DCN communications link between at least one of the NEs and the Supplier Management System is so congested that current state or status information cannot be transferred within a system defined synch time.

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

The exception **DuplicateUserLabel** is raised if the User Label provided in the request has been used to label another NE or plugInUnit. 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 **EquipmentFailure** is raised when the equipment where the data is backed up from is in failure condition.

The exception **HWServicesMismatch** is raised when the replacement NE cannot perform the provisioned services.

The exception **InsufficientPONBW** is raised when the ONT or ONU cannot be ranged due to insufficient bandwidth on the APONLink.

The exception **InvalidDCNAddress** is raised when the specified DCN address is not valid.

The exception **InvalidSerialNumSyntax** is raised when the Syntax of the serial number provided does not match definition rules.

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

The exception **MaxSubtendingNodesExceeded** is raised when the maximum engineered number of subtending nodes for the identified PON interface has been exceeded with this request for service provisioning.

The exception **SynchInProgress** will be raised when any operation is requested while the Supplier Management System is in the process of synchronising with the NE.

The exception **TooManyNEs** is raised when the Supplier Management System cannot manage one more OLT.

The exception **UnknownManagedEntity** is raised when the equipment is unknown to the Supplier Management System.

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

The exception **UnknownPort** is raised when the identified port is unknown to the Supplier Management System.

9.11 ResourceAllocation module

This service allows the OMS to reserve bandwidth on a system resources for an anticipated service connection. The reserved bandwidth can be deleted or retrieved. The ResourceAllocator interface provides the means for creating, deleting, or displaying a resource reservation. This operation is used prior to dispatch of personnel for the installation of a network element. Once resource capacity is reserved, the reserved resource can be used only for the service specified in the reservation.

9.11.1 ResourceAllocator Interface

9.11.1.1 reserveForService

This operation reserves bandwidth for a network resource such as ONT, ONU, or NT whose installation is pending. This operation is used when the ONT, ONU, or NT that serves the bandwidth associated with the designated OLT is first provisioned. When the operation is completed, the return value, **ReservationBandwidthType**, provides an accounting of the reserved bandwidth to the OMS.

The operation signature for the **reserveForService** is shown below:

```
ReservationBandwidthType reserveForService(
                        in EndPointType endPointA,
                        in EndPointType endPointZ,
                        in NameSeqType networkCharacteristicsProfiles,
                        in ServiceInstanceIdType serviceInstanceId)
                        raises(UnknownNE, UnknownPort, UnknownProfiles,
                        InsufficientBW, MaxSubtendingNodesExceeded,
                        ConnectionCountExceeded, CommFailure, AccessDenied,
                        ProfileSuspended );
```

The input parameters **endPointA** and **endPointZ** identify the two endpoints of the service connection that will determine the resources necessary to support an anticipated service connection. The input parameter **networkCharacteristicsProfiles** is a list consisting of the following; abTrafficDescripterProfile, and baTrafficDescripterProfile (in this order). The input parameter **serviceInstanceId** identifies the associated service instance for this reserved bandwidth.

The return value is of type **ReservationBandwidthType** and includes a Reservation Id and the amount of network resource bandwidth that has been reserved by the Supplier Management System. The Reservation Id can be used by the OMS during service provisioning or to cancel this reservation.

9.11.1.2 cancelReservation

This operation is used to delete the reservation and release the resources from the reserved system capacity.

The operation signature for the **cancelReservation** is shown below:

```
AvailableSysBandwidthSeqType cancelReservation (
                   in ReservationIdType reservationId )
                   raises (UnknownReservationId, CommFailure, AccessDenied);
```

The input parameter **reservationId** identifies the existing reservation associated with the bandwidth that was allocated.

The return value of type **AvailableSysBandwidthSeqType** indicates the current available bandwidth of the OLT after the deletion of the reserved bandwidth.

9.11.1.3 getReservationId

This operation is used to display the reservation Id associated with the service instance id that is assigned for the reserved bandwidth.

The operation signature for the **getReservationId** is shown below:

```
ReservationIdType getReservationId (
                 in ServiceInstanceIdType serviceInstanceId)
                 raises(UnknownServiceInstance, AccessDenied);
```

The input parameter **serviceInstanceId** is used to identify the existing service instances assigned during the time the resource is reserved.

The return value of type **ReservationIdType** is associated with the given service instance id.

9.11.1.4 reportReservedResources

This operation is used by the OMS to display the current reserved bandwidth in the specific headend NE (in this case the OLT).

The operation signature for the **reportReservedResources** is shown below:

```
ReservedBandwidthSeqType reportReservedResources (
                   in ManagedEntityIdType nEManagedEntityId)
                   raises (UnknownNE, AccessDenied);
```

The input parameter **nEManagedEntityId** is used to identify the network element for which current bandwidth reservations are requested.

The return value of type **ReservedBandwidthSeqType** has the all current reserved bandwidth information for the network element.

9.11.1.5 getReservations

This operation is used by OMS to retrieve all the reservations associated with the given NE.

The operation signature for the **getReservations** is shown below:

```
ReservationIdSeqType getReservations(
                        in ManagedEntityIdType nEManagedEntityId)
                        raises (UnknownNE, AccessDenied);
```

The input parameter **nEManagedEntityId** is used to identify the NE for retrieval of the current reserved bandwidth.

The return value of type **ReservationIdSeqType** provides all the reservation id(s) currently allocated to the OLT.

9.11.1.6 cancelAllRemainingReservations

This operation is used to delete all the remaining reservations against capacity associated with a given network element. The network element can have bandwidth that is either assigned, reserved, or available for service connections. This operation only changes reserved resources to available resources.

The operation signature for the **cancelAllRemainingReservations** is shown below:

```
AvailableSysBandwidthSeqType cancelAllRemainingReservations(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, CommFailure, AccessDenied);
```

The input parameter **nEManagedEntityId** is used to identify the network element.

The return value of type **AvailableSysBandwidthSeqType** indicates the current available bandwidth of the network element after the deletion of the reserved bandwidth.

9.11.1.7 getReservation

This operation is used to investigate the origins of a reservation of capacity within an NE.

The operation signature for the **getReservation** is shown below:

```
ReservationInfoType getReservation (
               in ReservationIdType reservationId)
               raises (UnknownReservationId, AccessDenied);
```

The input parameter **reservationId** is used to specify the reservation.

The return value of type **ReservationIdInfoType** indicates the service connection information provided as part of the reservation request with the provided **reservationId**.

9.11.1.8 getAvailableSysBandwidth

This operation is used to determine the amount of capacity remaining for service connection reservation or assignment for a network element.

The operation signature for the **getAvailableSysBandwidth** is shown below:

```
AvailableSysBandwidthSeqType getAvailableSysBandwidth (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (UnknownNE, CommFailure, AccessDenied);
```

The input parameter **nEManagedEntityId** is used to identify the network element.

The return value of type **AvailableSysBandwidthSeqType** indicates the current available bandwidth of the network element. It provides the available bandwidth for each provisioned port on the network element as characterized by the supplier implementation.

9.11.1.9 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 ONT or ONU is down.

The exception **ConnectionCountExceeded** is raised when the maximum number of connections for the OLT or PON port has been exceeded with this request for service provisioning.

The exception **InsufficientBW** is raised when the bandwidth is insufficient to the requested service.

The exception **MaxSubtendingNodesExceeded** is raised when the maximum engineered number of subtending nodes for the identified PON interface has been exceeded with this request for service provisioning.

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

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

The exception **UnknownPort** is raised when the identified port 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 **UnknownReservationId** is raised when the Supplier Management System does not recognize this Reservation Id.

The exception **UnknownServiceInstance** is raised when the service instance is unknown to the Supplier Management System.

9.12 SchedulerManagement module

This service is used to provide OMS interfaces for managing schedulers to be used for invoking various activities. Once the scheduler is created, the OMS can initiate requests to schedule an activity such as NE MIB uploading, bulk transfer, testing, or software downloading by referencing the scheduler. Schedulers are determined by needs of the operations environment. It is assumed that there will be no schedulers named or established automatically upon instantiation of the Supplier Management System. References to a specific scheduler will always be made via User Label. A trigger matrix is used to describe the schedule. The values in the matrix are interpreted based on the value of HourlyDailyWeeklyMonthlyInd.

9.12.1 SchedulerMgr interface

9.12.1.1 makeScheduler

This operation creates a new scheduler object. The OMS can then associate activities with the scheduler object by referencing the User Label name of the scheduler.

The operation signature for the **makeScheduler** is shown below:

```
void makeScheduler (
            in UserLabelType schedulerName,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime,
            in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
            in TriggerTimeMatrixSeqType matrix)
            raises (InvalidStartTime, InvalidStopTime, DuplicateUserLabel,
            MatrixSchedulerTypeMismatch, AccessDenied, InvalidTrigger);
```

The input parameter **schedulerName** identifies the scheduler which can be referenced by different activities. The input parameters **startTime** and **stopTime** identify the time range in which the scheduled activiti[es] is[are] applicable. The input parameter **hourlyDailyWeeklyMonthlyInd** indicates the frequency (hourly, daily, weekly or monthly) with which the schedule should be triggered. The input parameter **matrix** provides specific schedule invocation information and is affected by the **hourlyDailyWeeklyMonthlyInd** parameter value.

Table 4 below provides a table showing the dependency of **hourlyDailyWeeklyMonthlyInd** and **matrix**:

Table 4/Q.834.4 – Matrix details

hourlyDailyWeeklyMonthlyInd value	matrix value
Hourly	time – Any value between 0 and 3600[7] dayOfWeek – must be 'unspecified' dayOfMonth – must be 0
Daily	time – Any value between 0 and 86400[8] dayOfWeek – must be 'unspecified' dayOfMonth – must be 0
Weekly	Time – Any value between 0 and 86400 dayOfWeek – different from 'unspecified' dayOfMonth – must be 0
Monthly	Time – Any value between 0 and 86400 dayOfWeek – must be 'unspecified' dayOfMonth – different from 0

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

9.12.1.2 suspendScheduler

This operation is used to suspend a schedule. It essentially sets the schedule's administrativeState from 'unlocked' to 'locked'. This will be applicable starting from the next schedule iteration. This operation causes any associated scheduled activity to be suspended.

The operation signature for the **suspendScheduler** is shown below:

```
void suspendScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied);
```

The input parameter **schedulerName** is used to identify the schedule to be suspended.

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

9.12.1.3 resumeScheduler

This operation is used to resume a suspended schedule. It essentially sets the schedule's administrativeState from 'locked' to 'unlocked'. This will be applicable starting with the next schedule iteration. This operation causes any associated scheduled activity to be resumed.

The operation signature for the **resumeScheduler** is shown below:

```
void resumeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied );
```

The input parameter **schedulerName** is used to identify the schedule to be resumed.

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

[7] Represents the number of seconds from the beginning of the hour for the trigger time.

[8] Represents the number of seconds from the beginning of the day for the trigger time where the day begins after midnight.

9.12.1.4 modifyTime

This operation is used in order to change the **startTime** and **stopTime** of a schedule. It is used by the OMS to extend or to shorten the time range in which the schedule is applicable.

The operation signature for the **modifyTime** is shown below:

```
void modifyTime (
            in UserLabelType schedulerName,
            in GeneralizedTimeType newStartTime,
            in GeneralizedTimeType newStopTime)
            raises (InvalidStartTime, InvalidStopTime, UnknownScheduler,
            AccessDenied);
```

The input parameter **schedulerName** is used to identify the schedule. The input parameter **newStartTime** provides the time at which this scheduler starts and the input parameter **newStopTime** provides the time at which the scheduler stops. If the value of newStartType is earlier than the current system time, the exception InvalidStartTime will be raised. A value of 0 for either **newStartTime** or **newStopTime** indicates that the previous value should not be changed.

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

9.12.1.5 changeSchedulerName

This operation is used to change the name of the scheduler object. Upon success of this operation, the new scheduler name will immediately be applicable. Any reference to this schedule will have to be made using the newly assigned name.

Many activities can be associated with the same scheduler name. Changing the scheduler name will not affect the activities associated with the scheduler. All activities associated with the changed scheduler will maintain their association.

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

```
void changeSchedulerName (
            in UserLabelType oldSchedulerName,
            in UserLabelType newSchedulerName)
            raises (UnknownScheduler, DuplicateUserLabel, AccessDenied);
```

The input parameter **oldSchedulerName** is used to identify the existing schedule. The input parameter **newSchedulerName** is used to indicate the new name to assign for this scheduler.

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

9.12.1.6 modifyTriggerTimes

This operation is used by the OMS to specify new trigger times and iteration for a scheduler[9]. If this operation is successful, then any associated scheduled activity will occur at the new schedule times starting from the next schedule iteration.

The operation signature for the **modifyTriggerTimes** is shown below:

```
void modifyTriggerTimes (
         in UserLabelType schedulerName,
         in HourlyDailyWeeklyMonthlyIndType newHourlyDailyWeeklyMonthlyInd,
         in TriggerTimeMatrixSeqType newMatrix)
         raises (UnknownScheduler, MatrixSchedulerTypeMismatch,
         InvalidTrigger, AccessDenied );
```

The input parameter **schedulerName** is used to identify the schedule to be modified. The input parameter **newHourlyDailyWeeklyMonthlyInd** indicates the new frequency with which the

[9] Please refer to makeScheduler operation for details on the dependency of used variables.

schedule should be triggered. The input parameter **newMatrix** is used to provide the new specific schedule invocation information. Please refer to Table 4 for dependency of the variables. Both **newDailyWeeklyMonthlyInd** and **newMatrix** must be explicitly specified.

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

9.12.1.7 removeScheduler

This operation is used to delete a scheduler. This operation will be allowed only if there are no activities associated with this scheduler.

The operation signature for the **removeScheduler** is shown below:

```
void removeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied, ScheduleInUse );
```

The input parameter **schedulerName** is used to identify the schedule to be deleted.

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

9.12.1.8 retrieveScheduler

This operation is used to retrieve information on the schedule. The schedule object will be returned upon successful invocation of this operation.

The operation signature for the **retrieveScheduler** is shown below:

```
SchedulerType retrieveScheduler (
              in UserLabelType schedulerName)
              raises (UnknownScheduler, AccessDenied);
```

The input parameter **schedulerName** is used to identify the schedule to be displayed.

The return value of type **Scheduler** provides the following information: schedulerName; startTime; stopTime; hourlyDailyWeeklyMonthlyInd; matrix; operationalState and administrativeState.

9.12.1.9 schedulerListGet

This operation is used to retrieve the names of all existing schedulers defined for the Supplier Management System.

The operation signature for the **schedulerListGet** is shown below:

```
SchedulerSeqType schedulerListGet ()
                  raises (AccessDenied);
```

There are no input parameters.

The return value is of type **SchedulerSeqType** and provides the listing desired.

9.12.1.10 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 Scheduler. In other words, the Supplier Management System is responsible for policing User Labels assigned for schedulers within its management jurisdiction.

The exception **InvalidStartTime** is raised when the specified start time is inconsistent with the current trigger time matrix or the stop time.

The exception **InvalidStopTime** is raised when the specified stop time is inconsistent with the current trigger time matrix or the start time.

The exception **InvalidTrigger** is raised when the specified Trigger has values that cannot be interpreted by the Scheduler.

The exception **MatrixSchedulerTypeMismatch** is raised when the syntax for the Trigger Time Matrix is mismatched with the type of scheduler named.

The exception **ScheduleInUse** will be returned from removeScheduler operation in case there are operations associated with the schedule.

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

9.13 ServiceProvisioning module

In this module, the Supplier Management System selects ports, facilities and bandwidth in order to be able to complete the design, selection and assignment process associated with a service.

9.13.1 ServiceProvisioner interface

9.13.1.1 provisionConnection

This operation provisions a connection between any two endpoints of a BPON fibre access system. A connection may be established between an NNI and a UNI, between two UNIs, or between two NNIs. Figure 1 illustrates these endpoints.

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

```
ManagedEntityIdType provisionConnection(
               in EndpointType endPointA,
               in EndpointType endPointZ,
               in NameSeqType networkCharacteristicsProfiles,
               in ServiceInstanceIdType serviceInstanceId,
               in AdministrativeStateType administrativeState)
               raises (UnknownNE, UnknownProfiles, UnknownPort, InsufficientBW,
               ConnectionCountExceeded, CommFailure, EquipmentFailure,
               ParameterViolation,AccessDenied, InsufficientPONBW,
               ProfileSuspended, ConnectionAlreadyExists);
```

The input parameters **endPointA** and **endPointZ** identify the two endpoints of the connection request. An endpoint is defined by the data structure provided in Table 5.

Table 5/Q.834.4 – Endpoint details

Field name	Definition	Syntax	Comments
portId	Specifies the physical port containing one end point of the connection.	ManagedEntityIdType	The assumption is that this is the physicalPathTP Managed Entity for the port.
endPointParameters	Identifies service instance specific parameters assisting in the joining together of network connection components.	any	Structures to be provided as part of implementation. An ATM connection, for example, would have the VPI, VCI parameters.
serviceCharacteristicsProfile	Lists references to profiles characterizing the service at the endpoint.	NameSeqType	Examples are AAL1Profile, AAL5Profile, ATMNetworkAccess Profile, UNIProfile, CESServiceProfile, DS1Profile, DS3Profile, EthernetProfile, AAL2Profile, LESProfile, SSCSParameter Profile1, SSCSParameter Profile2, VoiceService ProfileAAL2, BridgedLAN ServiceProfile, MACBridgeService Profile. (Note)
NOTE – Specific choices depend on service and equipment characteristics. Profiles are listed by the names provided in the module Q834ProfileManager. See Annex D for example relationships.			

The input parameter **networkCharacteristicsProfiles** provides the list of transport related profiles to be used for service provisioning including azTrafficDescriptorProfile and zaTrafficDescriptorProfile. The az Traffic Descriptor Profiles are listed before the za in the listing. The parameter **serviceInstanceId** indicates the service instance identifier to be used as the key when referring to network resources associated with the service. The input parameter **administrativeState** specifies whether or not the subnetwork connection is able to carry subscriber traffic once this operation is executed.

The return value of type **ManagedEntityIdType** is a managed entity identifier to identify the subnetworkConnection created as the result of this request.

9.13.1.2 provisionReservation

This operation provisions service between the NNI of an OLT and the UNI of an ONT or between two UNIs based on an outstanding reservation. Once the service connection is provisioned, the reserved bandwidth and associated reservation Id is removed from the Supplier Management System because the reserved resources are assigned.

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

```
ManagedEntityIdType provisionReservation(
                   in ReservationIdType reservationId,
                   in AdministrativeStateType administrativeState)
                   raises ( UnknownReservationId, AccessDenied);
```

The input parameter **reservationId** specifies the reservation Id. It points to all the other information (such as service instance Id and endpoints) to associate with the provisioned connection. The input parameter **administrativeState** specifies whether or not the subnetwork connection is able to carry subscriber traffic once this operation is executed.

The return value of type **ManagedEntityIdType** is a managed entity identifier to identify the subnetworkConnection created as the result of this request.

9.13.1.3 deleteConnection

The operation tears down the existing service and connections.

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

```
void deleteConnection(
            in ManagedEntityIdType subnetworkConnectionId)
            raises (UnknownConnection, CommFailure, EquipmentFailure,
            AccessDenied);
```

The input parameter **subnetworkConnectionId** is the subnetwork that was created previously via service provisioning request.

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

9.13.1.4 modifyConnection

This operation provides the capability to modify the existing service.

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

```
ManagedEntityIdType modifyConnection (
                    in ManagedEntityIdType subnetworkConnectionId,
                    in ManagedEntityIdType portB,
                    in NameSeqType newNetworkCharacteristicsProfiles,
                    in NameSeqType newServiceCharacteristicsProfiles)
                    raises (UnknownConnection, UnknownProfiles, InsufficientBW,
                    UnknownPort, AccessDenied, ProfileSuspended );
```

The input parameter **subnetworkConnectionId** is the subnetwork that was created previously via service provisioning request. The input parameter **portB** helps to clarify the directionality of the traffic descriptor profiles named in the next input parameter as well as to specify where the service profiles are meant to be applied. This port is either the A or Z port of the original connection request. The input parameter **newNetworkCharacteristicsProfiles** provides the modified list of network related profiles to be used for service provisioning where "b-to-opposite endpoint" traffic descriptor profiles are listed before "the opposite endpoint-to-b" traffic descriptor profiles. The input parameter **newServiceCharacteristicsProfiles** provides the new list of service related profiles.

The return value of type **ManagedEntityIdType** is a managed entity identifier to identify the subnetworkConnection created as the result of this request.

9.13.1.5 suspendService

This operation disables the flow of user traffic through the service subnetwork connection. In effect, suspension of service is equivalent to locking the administrative state of the subnetwork connection.

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

```
void suspendService (
            in ServiceInstanceIdType serviceInstanceId,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime)
            raises (UnknownServiceInstance, AccessDenied, InvalidStartTime,
            InvalidStopTime);
```

The input parameter **serviceInstanceId** identifies the service connection to be suspended. The input parameter **startTime** provides the time point at which the service is to be suspended. The input parameter **stopTime** provides the time point at which the service is to be resumed.

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

9.13.1.6 resumeService

This operation enables the flow of user traffic through the service subnetwork connection. This operation can be invoked either after service has been suspended (see operation above) or if the original service connection is configured in an inactive state. The service is to be resumed immediately.

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

```
void resumeService (
              in ServiceInstanceIdType serviceInstanceId)
              raises (UnknownServiceInstance, AccessDenied);
```

The input parameter **serviceInstanceId** identifies the service connection to be resume.

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

9.13.1.7 Exceptions

The exception **AccessDenied** is raised when the NE access is denied due to a security reason.

The exception **CommFailure** is raised when communication between the Supplier Management System and NEs failed.

The exception **ConnectionAlreadyExists** is raised when there already exists a subnetwork connection with the same endpoints.

The exception **ConnectionCountExceeded** is raised when the connection count exceeds the allowable connection count.

The exception **EquipmentFailure** is raised when the connection requested cannot be applied to an installed network resource because of the failure condition of the NE.

The exception **InsufficientBW** is raised when the bandwidth is insufficient to the requested service.

The exception **InsufficientPONBW** is raised when the available PON bandwidth is not sufficient to support the requested service provisioning.

The exception **InvalidStartTime** is raised when the specified start time is inconsistent with the current time or the stop time.

The exception **InvalidStopTime** is raised when the specified stop time is inconsistent with the current time or the start time.

The exception **ParameterViolation** is raised when the endpoint parameters do not match the protocol characteristics of the port, or if the value(s) are out of range or invalid duplicates.

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

The exception **UnknownConnection** is raised when the connection to be deleted is not found.

The exception **UnknownNE** is raised when the OLT name is not known to the Supplier Management System.

The exception **UnknownPort** is raised when the input port id is not known 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 **UnknownReservationId** is raised when the reservation identifier is not known to the Supplier Management System.

The exception **UnknownServiceInstance** is raised when the service identifier is not known to the Supplier Management System.

9.14 Synchroniser module

This module manages the synchronisation process between the Supplier Management System and a specific NE. The requested synchronisation process can be narrowed to a specific set of current event listings. The details on how inconsistencies are detected and what data is retrieved is up to the suppliers implementation. However, this process results in the removal of any inconsistencies between the management information found with the NE and the information model maintained in the Supplier Management System.

The operations in this module can only be invoked by a privileged user. The synchronisation process is "best effort" in the sense that any inconsistencies detected and corrected prior to a system defined timeout period[10] are preserved. The synchronisation process may impact the performance of the Supplier Management System in a negative way prior to its completion.

When the network element chosen is an OLT, then the synchronisation process is system-wide.

9.14.1 NESynchroniser interface

9.14.1.1 synchNE

This operation initiates a synchronisation process between the Supplier Management System and a specific NE. This operation is blocking only as long as it takes for the Supplier Management System to validate that the process can be initiated, and performs the synchronisation operation in the background.

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

```
void synchNE(in ManagedEntityIdType nEManagedEntityId)
            raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
            BackupInProgress, SynchInProgress);
```

The input **nEManagedEntityId** identifies NE to synchronise with the Supplier Management System.

[10] This timeout period is a matter of agreement between the Operator and the Supplier.

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

9.14.1.2 abortSynchNE

This operation aborts a synchronisation process in progress between the Supplier Management System and a specific NE. Any inconsistencies between the Supplier Management System and the NE that have been resolved before the abort are retained.

The operation can only be invoked by a privileged user.

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

```
void   abortSynchNE(
             in ManagedEntityIdType nEManagedEntityId)
             raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
             NoSynchInProgress);
```

The input **nEManagedEntityId** identifies the NE whose synchronisation with the Supplier Management System is to be aborted.

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

9.14.1.3 scheduleSynchNE

This operation schedules a synchronisation process between the Supplier Management System and a specific NE. The scheduled synchronisation process is triggered by the scheduler which is predefined by the operator. At most, one scheduler can be associated with this activity for a specific NE.

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

```
void scheduleSynchNE(
           in ManagedEntityIdType nEManagedEntityId,
           in UserLabelType schedulerName)
           raises (AccessDenied, UnknownNE, UnknownScheduler, InvalidScheduler);
```

The input **nEManagedEntityId** identifies the NE to synchronise with the Supplier Management System. The input **schedulerName** identifies scheduler to be used to invoke scheduled synchronisation operation of Supplier Management System.

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

9.14.1.4 modifyNESynchSchedule

This operation modifies the schedule for NE synchronisation. This operation will not interrupt a synchronisation process in progress. If successful, the new schedule is applied with the next iteration.

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

```
void modifyNESynchSchedule(
            in ManagedEntityIdType nEManagedEntityId,
            in UserLabelType newSchedulerName)
            raises (AccessDenied, UnknownNE, UnknownScheduler,
            InvalidScheduler);
```

The input **nEManagedEntityId** identifies the NE where the scheduled NE synchronisation is activated. The input **newSchedulerName** identifies the schedule to be applied.

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

9.14.1.5 cancelScheduledSynchNE

This operation cancels all subsequent scheduled synchronisation processes for this NE. This operation will not interrupt a synchronisation process in progress.

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

```
void cancelScheduledSynchNE(
            in ManagedEntityIdType nEManagedEntityId )
            raises (AccessDenied, UnknownNE);
```

The input **nEManagedEntityId** identifies the NE where the scheduled NE synchronisation is activated.

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

9.14.1.6 synchCurrentEventListings

The operation initiates synchronisation between the Supplier Management System and specified NE for items in particular current event listings. The Supplier Management System normally retrieves current values of state, status, or management attributes and tracks these via current summary event listings for the system. If any automatic system retrieval process shows that the listing is not up-to-date with the current conditions of the system, then the listing is modified (through deletion of an entry or insertion of a new entry) in order to correct the listing. This operation is a manual version driven by OMS request. While this operation is in effect, the exception SynchInProgress for any other operation may not be invoked.

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

```
void synchCurrentEventListings(
              in ManagedEntityIdType nEManagedEntityId,
              in CurrentListingSeqType currentListingTypeList)
              raises (AccessDenied, CommFailure, DCNTimeout, UnknownNE,
              EquipmentFailure, Timeout);
```

The input **nEManagedEntityId** identifies the NE for synchronisation. The input **currentListingTypeList** identifies the list of current event that are to be synchronised between the Supplier Management System and the specified NE. The Supplier Management System retrieves the specified current event listing values from NE.

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

9.14.1.7 scheduledSynchNEListGet

This operation is used to retrieve the names of all NEs with synchronisation schedules.

The operation signature for the **scheduledSynchNEListGet** is shown below:

```
ScheduledSynchNESeqType scheduledSynchNEListGet ()
                  raises (AccessDenied);
```

There are no input parameters.

The return value is of type **ScheduledSynchNESeqType** and provides the listing of NEs desired.

9.14.1.8 Exceptions

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

The exception **BackupInProgress** is raised when the synchronisation request is issued while the backup is in progress.

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

The exception **DCNTimeout** is raised when the DCN communications link between at least one of the NEs and the Supplier Management System is so congested that current state or status information cannot be transferred within a system defined time period.

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 **SynchInProgress** will be raised when a new synchNe is requested while either synchNe or scheduledSynchNe is in progress.

The exception **SynchNotScheduled** is raised if no planned synchronisation exists for the NE at the specified time.

The exception **UnknownNE** is raised when the NE mentioned in the request is unknown to the Supplier Management System.

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

The exception **Timeout** is raised when the specified process has exceeded a system-defined default timeout period.

The exception **NoSynchInProgress** is raised if there is no synchronisation process in progress.

9.15 Test module

This service is used to provide interfaces for the operator or OMS to perform directed or scheduled testing procedures. Tests such as ATM OAM cell loopback testing, interface loopback set-up on subscriber cards or OLT network interface cards, and ATM continuity checks are specified using this service. Tests can be either scheduled or manually invoked following an identified fault or subscriber service complaint. The TestActionPerformer interface provides the operations to execute tests applicable on network resources.

9.15.1 TestActionPerformer interface

9.15.1.1 aTMLoopback

This operation is used to invoke an ATM OAM Loopback test. ATM Loopback test is unidirectional.

The operation signature for the **aTMLoopback** is shown below:

```
AggregateATMLoopbackResultSeqType aTMLoopback (
                  in UserIdType testRequestorId,
                  in ManagedEntityIdType ctp,
                  in ATMLoopbackInfoType aTMLoopbackInfo,
                  in TestIterationNumType testIterationNum,
                  in ServiceInstanceIdType serviceInstanceId)
                  raises (AccessDenied, CommFailure, UnknownManagedEntity,
                  NotAvailableForTest, InvalidLocationId, InvalidDirection);
```

The input parameter **testRequestorId** is used to identify the initiator of the ATM Loopback test. The input parameter **ctp** is used to uniquely identify the CTP for the loopback cell injection point. The input parameter **aTMLoopbackInfo** is used to provide specific ATM test information; the LoopbackLocationId is part of aTMLoopbackInfo. The input parameter **testIterationNum** identifies the iteration number for the ATM loopback test. The input parameter **serviceInstanceId** identifies the possible service instance associated with the loopback test request.

When specifying aTMLoopbackInfo, the only applicable values for directionality are 'egress' or 'ingress' only. The directionality is specified as part of the **aTMLoopbackInfo**.

The LoopbackLocationId identifies the point(s) along a virtual connection where the loopback is to occur. The default value of all ones is used by the transmitter to indicate the end point. When segmentCellInd is set to 'false', the LoopbackLocationId must be set to the default. If no

LoopbackLocationId is provided, it is assumed that there is a designated segment endpoint for the flow associated with the injection ctp.

For ctp, all zeros indicate a loopback request directed at all connection points having a LoopbackLocationId. All ones indicate a loopback request directed at the endpoint (segment or connection endpoint). 'x6A'H indicates no designated CP for loopback, and therefore no loopback should be performed. All other values for LoopbackLocationId indicate a loopback request directed at a specific LoopbackLocationId location.

The return value is of type **AggregateATMLoopbackResultSeqType** which provides information on the test, specifically the loopbackingLLID, responseTime in microseconds, and success/failure for each iteration.

9.15.1.2 initializeContinuityCheck

This operation is used to prepare for an ATM continuity check. The creation of a continuity test environment does not necessarily initiate the test, but rather schedules a test to be initiated. Upon successful creation of a continuity check, the system will return a unique identifier that is used to identify the test.

This operation deals with the test setup only. Once the test is executed, an alarm will be raised in case of a failure.

The operation signature for the **initializeContinuityCheck** is shown below:

```
CCSetUpIdType initializeContinuityCheck(
                  in UserIdType testRequestorId,
                  in ManagedEntityIdType sourceCtp,
                  in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
                  in GeneralizedTimeType stopTime,
                  in ServiceInstanceIdType serviceInstanceId)
                  raises (AccessDenied, CommFailure, UnknownManagedEntity,
                  NotAvailableForTest, InvalidStartTime, InvalidStopTime,
                  InvalidDirection);
```

The input parameter **testRequestorId** is used to identify the initiator of the test. The input parameter **sourceCtp** and **sinkCtp** is used to identify the A point for the continuity test. The input parameter **aTMContinuityCheckInfo** is used to provide specific continuity test information. The directionality must be set to 'BothDirections' and segmentCellInd is set to 'true' for segment continuity test, and 'false' for end-to-end. The input parameter **stopTime** provides information on how long the test should be executed. The input parameter **serviceInstanceId** identifies the possible service instance associated with the loopback test request.

The return value is of type **CCSetUpIdType** which uniquely identifies the test that was set up. The CCSetUpId for the setup test exists until the stopTime is reached, or until it is explicitly cancelled using the terminateContinuityCheck operation.

9.15.1.3 terminateContinuityCheck

This operation is used to take down the ATM continuity test environment. If the continuity test has already been initiated (i.e., its startTime is reached), the continuity test will stop executing and will then be removed.

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

```
void terminateContinuityCheck(
                  in CCSetUpIdType cCSetUpId)
                  raises (AccessDenied, CommFailure, UnknownTest);
```

The input parameter **CCSetUpId** is used to identify the test to terminate.

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

9.15.1.4 scheduleResourceSelfTest

This operation is used to schedule a resource self test. This operation is used by the OMS to set up self tests on resources to be executed at a regular basis. Having a scheduler object setup is a prerequisite for initiating this operation.

The operation signature for the **scheduleResourceSelfTest** is shown below:

```
TestTrackingObjectIdType scheduleResourceSelfTest(
               in UserIdType testRequestorId,
               in ManagedEntityIdType targetNE,
               in unsigned long timeOutPeriod, //In seconds.
               in ResourceSelfTestInfoSeqType specificTestInfo,
               in UserLabelType schedulerName)
               raises (AccessDenied, UnknownNE, UnknownScheduler,
               InvalidScheduler, InvalidTimeoutPeriod, InvalidTestOperations);
```

The input parameter **testRequestorId** is used to identify the initiator of the test. The input parameter **targetNE** identifies the network element to perform self test. The input parameter **timeOutPeriod** identifies the maximum time period that the system will allow the test to run on the resource. The input parameter **specificTestInfo** is used to provide supplier-specific self test information regarding each diagnostic test type to be run; this information is provided to the operator via system documentation. The input parameter **schedulerName** is used to reference the applicable scheduler for this test.

The return value of type **TestTrackingObjectIdType** uniquely identifies the scheduled testing. The Test Tracking Object exists until it is explicitly cancelled using the terminateScheduledResourceSelfTest operation or if the scheduler end time is reached.

The self test results are logged. The **ResourceSelfTestResultSeqType** data type defines part of the information that is logged.

9.15.1.5 modifyResourceSelfTestSchedule

This operation is used to modify the schedule for a regularly conducted resource self test. If successful, the resource self test initiation is changed with the next iteration.

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

```
void   modifyResourceSelfTestSchedule (
             in TestTrackingObjectIdType testTrackingObjectId,
             in UserLabelType newSchedulerName)
             raises (UnknownTest, UnknownScheduler, InvalidScheduler,
             AccessDenied);
```

The input parameter **testTrackingObjectId** is used to identify the scheduled test. The input parameter **testTrackingObjectId** is used to identify the new schedule.

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

9.15.1.6 cancelScheduledResourceSelfTest

This operation is used to cancel a regularly scheduled resource self test. If successful, this operation cancels the test prior to its initiation with the next trigger time.

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

```
void  cancelScheduledResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

The input parameter **testTrackingObjectId** is used to identify the scheduled test to terminate.

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

9.15.1.7 conductResourceSelfTest

This operation is used to initiate a resource self test following the identification of a system fault or a subscriber service complaint. The test results are logged in the Supplier Management System.

The operation signature for the **conductResourceSelfTest** is shown below:

```
TestTrackingObjectIdType conductResourceSelfTest (
              in UserIdType testRequestorId,
              in ManagedEntityIdType targetNE,
              in unsigned long timeOutPeriod, //In seconds.
              in ResourceSelfTestInfoSeqType specificTestInfo)
              raises (AccessDenied, CommFailure, UnknownNE,
              InvalidTimeoutPeriod, InvalidTestOperations);
```

The input parameter **testRequestorId** is used to identify the initiator of the test. The input parameter **targetNE** identifies the network element to perform self test. The input parameter **timeOutPeriod** identifies the time period that the system will try to initiate the test. The input parameter **specificTestInfo** is used to provide specific self test information providing details on each diagnostic test type to run.

The return value is of type **TestTrackingObjectId** and provides a mechanism to allow the operator to terminate a controlled resource self test. If the resource self test is uncontrolled, the Supplier Management System returns the value 0.

9.15.1.8 terminateResourceSelfTest

This operation terminates a resource self test in progress.

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

```
ResourceSelfTestResultSeqType terminateResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

The input parameter **testTrackingObjectId** is used to identify the test to terminate.

The return value of type **ResourceSelfTestResultSeqType** provides whatever interim test results are available.

9.15.1.9 initiateLoopback

This operation is used to initiate a loopback for a service. For example, a DS1 NearEndLineLoopback [could we support far end] or a SONET FacilityLoop will be performed.

The operation signature for the **initiateLoopback** is shown below:

```
LoopbackTrackingObjectIdType   initiateLoopback (
                         in UserIdType testRequestorId,
                         in ManagedEntityIdType loopingCtp,
                         in long duration, //In minutes.
                         in DirectionalityType directionality,
                         in LoopbackTestType loopbackTest
                         in ServiceInstanceIdType serviceInstanceId)
                         raises (AccessDenied, CommFailure, UnknownManagedEntity,
                         NotAvailableForTest);
```

The input parameter **testRequestorId** is used to identify the initiator of the test. The input parameter **loopingCtp** is used to identify the CTP to perform the loopback on. The input parameter **duration** defined the amount of time in seconds that the loopback should be active. The input parameter **directionality** indicates if the loopback should be performed for ingress traffic, egress traffic or for traffic in both directions. The input parameter **loopbackTest** is used to identify the specific loopback test type. The input parameter **serviceInstanceId** is used to specify the service associated with this loopback operation.

The return value of type **LoopbackTrackingObjectIdType** uniquely identifies the loopback test that is initiated. Once a loopback completes its running cycle (i.e., the test duration elapses), the object will not longer be available.

9.15.1.10 terminateLoopback

This operation is used to cancel a running loopback.

The operation signature for the **terminateLoopback** is shown below:

```
void terminateLoopback  (
                        in LoopbackTrackingObjectId loopbackTrackingObjectId)
                        raises (UnknownTest, AccessDenied);
```

The input parameter **loopbackTrackingObjectId** is used to identify the loopback to terminate.

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

9.15.1.11 getLoopbackInfo

This operation is used to retrieve the loopback information of a particular connection termination point.

The operation signature for the **getLoopbackInfo** is shown below:

```
LoopbackInfoType getLoopbackInfo (
                        in ManagedEntityIdType cTP)
                        raises (UnknownManagedEntity, AccessDenied);
```

The input parameter **cTP** is used to identify the possible loopback location.

The return value is of type **LoopbackInfoType** and it specifies the loopback type and directionality.

9.15.1.12 getLoopbackInfoByNE

This operation is used to retrieve the location and details of every connection point in an NE that are in loopback mode.

The operation signature for the **getLoopbackInfoByNE** is shown below:

```
LoopbackInfoSeqType getLoopbackInfoByNE    (
                        in ManagedEntityIdType nEId)
                        raises (UnknownManagedEntity, AccessDenied);
```

The input parameter **nEId** is used to identify the network resource.

The return value is of type **LoopbackInfoSeqType** providing a listing of loopback locations, loopback types and directionality.

9.15.1.13 getTestStatus

This operation is used to retrieve the status of a loopback in progress.

The operation signature for the **getTestStatus** is shown below:

```
StatusValueType getTestStatus (
                   in LoopbackTrackingObjectIdType id)
                   raises (AccessDenied, UnknownTest);
```

The input parameter **id** specifies the test setup of interest.

The return value is of type **StatusValueType** and indicates the status of the loopback action.

9.15.1.14 scheduledTestNEListGet

This operation is used to retrieve the list of all scheduled NE testing.

The operation signature for the **scheduledTestNEListGet** is shown below:

```
ScheduledTestNESeqType scheduledTestNEListGet ()
                    raises (AccessDenied);
```

There are no input parameters.

The return value is of type **ScheduledTestNESeqType** and provides the listing desired.

9.15.1.15 testHistoryByManagedEntity

This operation is used to retrieve the test history associated with a managed entity. This history should be retained as long as required by the operator.

The operation signature for the **testHistoryByManagedEntity** is shown below:

```
TestHistorySeqType testHistoryByManagedEntity (
                    in ManagedEntityIdType managedEntityId)
                    raises (AccessDenied, UnknownManagedEntity);
```

The input parameter **managedEntityId** specifies the reference managed entity.

The return value is of type **TestHistorySeqType** and provides the listing desired.

9.15.1.16 testHistoryByServiceInstance

This operation is used to retrieve the test history associated with a service instance. This history should be retained as long as required by the operator.

The operation signature for the **testHistoryByServiceInstance** is shown below:

```
TestHistorySeqType testHistoryByServiceInstance (
                    in ServiceInstanceIdType serviceInstanceId)
                    raises (AccessDenied, UnknownServiceInstance);
```

The input parameter **serviceInstanceId** specifies the reference service instance.

The return value is of type **TestHistorySeqType** and provides the listing desired.

9.15.1.17 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 **InvalidDirection** is raised when the specified test directionality is invalid for the given test.

The exception **InvalidLocationId** is raised when the LLID specified is not valid.

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

The exception **InvalidStartTime** is raised when the specified start time is inconsistent with the current trigger time matrix or the stop time.

The exception **InvalidStopTime** is raised when the specified stop time is inconsistent with the current trigger time matrix or the start time.

The exception **InvalidTestOperations** is raised when the requested self test operation is not valid.

The exception **InvalidTimeOutPeriod** is raised when the designated timeout period violates definition of valid values.

The exception **NotAvailableForTest** is raised when the sourceCTP is unable to establish a continuity check test setup with the sinkCTP.

The exception **UncontrolledTestInProgress** is raised when the self test cannot be cancelled because of an uncontrolled test.

The exception **UnknownNE** is raised when the NE mentioned in the request is unknown to the Supplier Management System.

The exception **UnknownManagedEntity** is raised when the specified ManagedEntity is unknown to the Supplier Management System.

The exception **UnknownServiceInstance** is raised when the specified Service Instance is unknown to the Supplier Management System.

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

The exception **UnknownTest** is raised when the test specified by the trackingId is not known in the Supplier Management system.

9.16 FileTransfer module

This module deals with the management of non-real time transfer records stored in any short-term archive in the Supplier Management System. It supports the subsequent tracking and monitoring of the file transfer process through the use of TransferTrackingObjectId. The Supplier Management System logs the successful or unsuccessful results of the file transfer. Any request for file transfer is accompanied by security credentials whereby the Supplier Management System is allowed to communicate with the destination server. The file transfer can be scheduled in advance. Transfer Tracking Objects are automatically deleted by the Supplier Management System once the associated file transfer has finished and results (whether successful or unsuccessful) are recorded in the completion log.

9.16.1 TransferMgr interface

9.16.1.1 fileTransfer

The file transfer is initiated immediately by this operation.

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

```
TransferTrackingObjectIdType fileTransfer(
                    in ManagedEntityIdType recordSetId,
                    in DCNAddressType destinationServerAddr,
                    in UserIdType userId,
                    in PasswordType password,
                    in FilenameType destinationFile,
                    in boolean overwriteExistingFile)
                    raises (AccessDenied, CommFailure, UnknownRecordSet,
                    UnknownDestinationServer );
```

The input parameter **recordSetId** identifies the short-term archive from which the data is to be extracted for file transfer. The input parameter **destinationServerAddr** identifies the data communication networking address for the server that is the destination of the file transfer. The input parameters **userId** and **password** provide the login mechanism to the destination server (assuming such security credentials are needed). The input parameter **destinationFile** provides a full directory location for the transferred file. Finally, the parameter **overwriteExistingFile** indicates whether or not the file transfer should allow the overwriting of a pre-existing file with the same destination directory location.

The return value of type **TransferTrackingObjectIdType** provides a correlation key to be used when attempting to track the status of the non-real time transfer of data from the short-term archive at some later point.

9.16.1.2 scheduleFileTransfer

The file transfer is scheduled for future initiation by this operation. All the contents of the identified short-term archive are extracted for file transfer. No assumption is made concerning the purging of the archive based on successful file transfer of data. Instead, the archive is purged, based on agreements between the supplier and operator concerning retention policy of archived information.

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

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

The input parameter **recordSetId** identifies the short-term archive from which the data is to be extracted for file transfer. The input parameter **destinationServerAddr** identifies the data communication networking address for the server that is the destination of the file transfer. The input parameters **userId** and **password** provide the login mechanism to the destination server (assuming such security credentials are needed). The input parameter **destinationFile** provides a full directory location for the transferred file. The parameter **overwriteExistingFile** indicates whether or not the file transfer should allow the overwriting of a pre-existing file with the same destination directory location. Finally, the input parameter **schedulerName** is the schedule information associated with the file transfer, based on which the file transfer takes place.

The return value of type **TransferTrackingObjectIdType** provides a correlation key to be used when attempting to track the status of the non-real time transfer of data from the short-term archive at some later point.

9.16.1.3 modifyFileTransferSchedule

The file transfer schedule is modified by this operation. If successful, the new schedule is applied with the next iteration.

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

```
void modifyFileTransferSchedule (
                    in TransferTrackingObjectIdType transferTrackingObjectId,
                    in UserLabelType newSchedulerName)
                    raises (AccessDenied, UnknownTransferProcess,
                    UnknownScheduler, InvalidScheduler);
```

The input parameter **transferTrackingObjectId** identifies the scheduled file transfer activity. The input parameter **newSchedulerName** is the new schedule information to be associated with the file transfer.

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

9.16.1.4 cancelScheduledFileTransfer

The file transfer schedule is cancelled by this operation. If successful, the activity is cancelled with the next iteration.

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

```
void cancelScheduledFileTransfer (
              in TransferTrackingObjectIdType transferTrackingObjectId)
              raises (AccessDenied, UnknownTransferProcess);
```

The input parameter **transferTrackingObjectId** identifies the scheduled file transfer activity.

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

9.16.1.5 getStatus

This operation allows the client to check the status of a transfer before its completion using a key.

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

```
StatusValueType getStatus (
              in TransferTrackingObjectIdType transferTrackingObjectId)
              raises (UnknownTransferProcess, AccessDenied);
```

The input parameter **transferTrackingObjectId** identifies the key of a particular file transfer process. The operator specifies this information and finds the status information of that file transfer process.

The return value is of type **StatusValueType** and provides the status of the file transfer process.

9.16.1.6 fileTransferHistoryListGet

This operation is used to retrieve the list of all completed file transfers for the Supplier Management System. This list is maintained in the Supplier Management System as a wraparound log.

The operation signature for the **fileTransferHistoryListGet** is shown below:

```
FileTransferHistorySeqType fileTransferHistoryListGet ()
                  raises (AccessDenied);
```

There are no input parameters.

The return value is of type **FileTransferHistorySeqType** and provides the listing desired.

9.16.1.7 scheduledFileTransferListGet

This operation is used to retrieve the names of all existing scheduled file transfers defined for the Supplier Management System.

The operation signature for the **scheduledFileTransferListGet** is shown below:

```
ScheduledFileTransferSeqType scheduledFileTransferListGet ()
                  raises (AccessDenied);
```

There are no input parameters.

The return value is of type **ScheduledFileTransferSeqType** and provides the listing desired.

9.16.1.8 Exceptions

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

The exception **CommFailure** is raised when there is a communication failure between the destination server and the Supplier Management System.

The exception **UnknownDestinationServer** is raised when the specified destination server cannot be accessed by the transfer agent.

The exception **UnknownRecordSet** is raised when the target file cannot be found.

The exception **UnknownScheduler** is raised when the specified scheduler cannot be accessed by the transfer agent.

The exception **UnknownTransferProcess** is raised when the specified **TransferTrackingObjectId** cannot be identified.

10 Compliance statement

An implementation claiming conformance to any of the interfaces defined in this Recommendation shall implement the complete behaviour associated with all the operations of the interface, as well as behaviour of referenced definitions in the module q834_4::Q834Common.

Annex A

Data dictionary

Table A.1 provides a listing of all data elements (data types) used in the interface specification found within this Recommendation. The listing includes management information interpretation, syntax, and any qualifying comments for each. Data elements are listed in alphabetical order. If a data type and a second data type constructed as a sequence of the first are both present in the interface specification, then only the first data element is defined. The interpretation of the second is obvious. In these cases, the name of the sequence is the name of the original data element with the characters "Seq" inserted before the ending of "Type".

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
AAL1PMHistoryDataType	This data element provides the items in this record type.	struct	
AAL1ProfileType	This data element provides the values for a profile of kind AAL1.	struct	
AAL2PMHistoryDataType	This data element provides the items in this record type.	struct	
AAL2ProfileType	This data element provides the values for a profile of kind AAL2.	struct	
AAL2PVCProfileType	This data element provides the values for a profile of kind AAL2PVC.	struct	
AAL5PMHistoryDataType	This data element provides the items in this record type.	struct	
AAL5ProfileType	This data element provides the values for a profile of kind AAL5.	struct	
AALModeType	This data element indicates which mode the AAL for the supporting VCC is employed.	enum	
ActivityLevelType	Specifies the permission level of access afforded to a user for an activity.	enum	monitorOnly, allowedToExecute, noAccess

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`ActivityType`	Specifies the type or category of user activity.	`short`	Defined as constants within the interface `q834_4::AccessControl::AccessControlMgr`
`AdministrationDomainType`	The identifier provided by the OMS or operator during registration to indicate the administration domain to which the NE belongs.	`UserLabelType`	
`AdministrativeStateType`	Is used to activate (unlock), de-activate (lock), or shutdown (shuttingdown) the functions of the associated managed entity.	`enum`	Defined in ITU-T Rec. X.780.
`AggregateATMLoopbackResultType`	Specifies the results of an ATM loopback test.	`struct`	
`AlarmLogRecordType`	This data element provides the items in this record type.	`struct`	
`AlarmStatusSeqType`	This data element provides all relevant values for the status variable alarm status.	`enum`	Valid `enum` values are `AS_UnderRepair`, `AS_Critical`, `AS_Major`, `AS_Minor`, `AS_AlarmOutstanding`
`AnnouncementType`	This data element provides the announcement to the customer going off-hook when no call has been attempted.	`enum`	
`APONPMHistoryDataType`	This data element provides the items in this record type.	`struct`	
`AppIdType`	This data element specifies the protocol combinations used between the Inter-Working Functions found in the Voice Gateway function and the ONT or NT.	`enum`	
`ATMContinuityCheckInfoType`	Specifies the input for ATM continuity check.	`struct`	
`ATMLoopbackInfoType`	Specifies the ATM loopback test info.	`struct`	
`ATMNetworkAccessProfileType`	Profiles an instance of profile kind ATMNetworkAccess	`struct`	
`ATMOverbookingFactorType`	This data element provides the value for ATM Overbooking.	`struct`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`AudioServIndType`	This boolean data element indicates whether or not audio service is transported, where the value TRUE implies the presence of this service.	`boolean`	
`AutoDetectionIndType`	This boolean data element identifies whether or not data rate auto-detection is enabled.	`boolean`	
`AvailableSysBandwidthSeqType`	This is a listing of available system bandwidth by port.	sequence of `PortBandwidthType`	
`AvailabilityStatusSetType`	This data element provides all relevant values for the status variable availability status.	sequence of availability status values	Defined in ITU-T Rec. X.780.
`BackedUpStatusType`	This data element indicates whether or not the alarm emitting managed entity has an operational backup unit.	`boolean`	Defined in ITU-T Rec. X.780.
`BridgedLANServiceProfileType`	This data element provides the values for a profile of kind Bridged LAN Service.	`struct`	
`BridgePriorityType`	This boolean data element indicates whether or not the learning functions of the bridge are enabled. The value TRUE means enabled.	`short`	
`BRISignallingType`	This data element selects which signalling format should be used for Basic Rate ISDN.	`enum`	
`BufferedCDVToleranceType`	This data element represents the duration of user data that must be buffered by the CES interworking entity to offset cell delay variation. This timing will be in 10 microsecond increments.	`long long`	
`CableLengthType`	This data element provides the length of twisted pair cable from the physicalPathTP of type "DS1" interface to the DSX1 cross-connect point (if applicable).	`long long`	
`CASType`	This data element selects which AAL1 format should be used. It applies to structured interfaces only. For unstructured interfaces this value, if present, must be set to the default of basic.	`enum`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`CASIndType`	This boolean data element indicates whether or not Channel Associated Signaling is enabled on the connection, where the value TRUE implies it is enabled.	`boolean`	
`CBRRateType`	This data element represents the rate of the CBR service supported by the AAL.	`enum`	
`CCSetUpIdType`	Specifies a unique Id for CC test set up.	`long long`	
`CDVTPCREgressType`	Cell Delay Variation Tolerance – this parameter is required for all service categories. It is applied to CLP = 0 flow for ABR and apply to CLP = 0 + 1 flows otherwise.	`long long`	
`CDVTPCRIngressType`	Cell Delay Variation Tolerance – this parameter is required for all service categories. It is apply to CLP = 0 flow for ABR and apply to CLP = 0 + 1 flows otherwise.	`long long`	
`CellLossIntegration PeriodType`	This data element represents the time in milliseconds for the cell loss integration period. If cells are lost for this period of time, the associated interworking vcCTPF entity will generate a cell starvation alarm.	`long long`	
`CESServiceProfileType`	This data element provides the values for a profile of kind CES Service.	`struct`	
`ClockRecoveryType`	This data element indicates whether the clock recovery type is derived from the physical interface.	`enum`	
`CMDataIndType`	This boolean data element indicates whether or not Circuit Mode Data is carried on this connection, where the value TRUE implies its presence.	`boolean`	
`CMMultiplierNumType`	This data element provides the *N* value in *N* × 64 kbit/s circuit mode data.	`short`	
`ConformanceDefType`	Indicates the type of conformance as defined in ATM-Forum TM 4.0.	`enum`	
`ControlStatusSetType`	This data element provides all relevant values for the status variable control status.	`Sequence of enum`	Defined in ITU-T Rec. X.780.

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`CorrelatedNotification Type`	This data element lists the reference numbers for other event notifications that have a relationship to this event notification.	`Sequence of Notification IdentifierType`	
`CreationModeType`	This data element indicates how the record set was created.	`enum`	Choice between operator-defined or instantiated as part of Supplier Management System application installation.
`CurrentListingType`	Specifies a current event listing type that can be synchronized between the Supplier Management System and the NE.	`short`	Values specified as constants.
`CurrentSizeType`	This data element describes the current size of a record set.	`unsigned long long`	In the same units as `MaxSizeType`.
`DataRateType`	This data element provides the data rate for the Ethernet connection. The valid values are 10 Mbit/s or 100 Mbit/s.	`enum`	
`DayOfMonthType`	Specifies the day of the month.	`short`	0 is interpreted to mean unspecified.
`DayOfWeekType`	Specifies the day of the week in the schedule.	`enum`	Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Unspecified
`DCNAddressType`	Provides the address for the NE or system server on the Data Communications Network of the Operator. Used for routing of messages.	`string`	Normally IP Address. Examples include items labelled as `softwareSource Addr`, `destinationServer Addr`, `sourceServerAddr`, `nEDCNAddress`, and `newNEDCNAddress`.
`DefaultSSCSParameter Profile1PtrType`	This data element identifies the default values for the service specific convergence service profile associated with channels carrying control and management plane traffic.	`Name`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`DefaultSSCSParameter Profile2PtrType`	This data element identifies the default values for the service specific convergence service profile associated with channels carrying media streams (e.g., POTS or ISDN B-channels).	`Name`	
`DiagnosticType`	Specifies the diagnostic test type.	`short`	Supplier specific
`DirectionalityType`	Specifies the directionality of the test.	`enum`	Values are `Egress, Ingress, BothDirections`
`DirectionType`	Specifies the direction of the optical wavelength associated with a port.	`enum`	Values are unidirection or bidirection.
`DownloadStatusSeqType`	This data element provides the status of a software download activity.	`struct` comprised of the Id of a target followed by the status for delivery, distribution, commitment, and activation	
`DS1EncodingType`	This data element identifies the type of DS1 line encoding.	`enum`	
`DS1FramingType`	This data element identifies the type of framing employed.	`enum`	
`DS1PMHistoryDataType`	This data element provides the items in this record type.	`struct`	
`DS1ProfileType`	This data element provides the values for a profile of kind DS1.	`struct`	
`DS3ApplicationType`	This data element identifies the type of DS3 signal.	`enum`	
`DS3PMHistoryDataType`	This data element provides the items in this record type.	`struct`	
`DS3ProfileType`	This data element provides the values for a profile of kind DS3.	`struct`	
`DS3EncodingType`	This data element identifies the DS3 line encoding.	`enum`	
`DTEDCEType`	This data element indicates whether the Ethernet Interface wiring is DTE or DCE.	`boolean`	
`DTMFIndType`	This boolean data element indicates whether or not Dual Tone Multi-Frequency dialled digits is transported on the connection, where the value TRUE implies its presence.	`boolean`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
DuplexType	This data element indicates whether full duplex (=TRUE) or half duplex mode (=FALSE) is employed.	boolean	
E1PMHistoryDataType	This data element provides the items in this record type.	struct	
E3PMHistoryDataType	This data element provides the items in this record type.	struct	
EchoCancellationInd Type	This data element indicates the presence of echo cancellation circuitry.	boolean	
ELCPIndType	This boolean data element indicates whether or not Emulated Loop Control Protocol is in use.	boolean	
EncapsulationProtocol Type	This data element identifies the encapsulation protocol used for bridging LAN over ATM.	short	
EncProfileIndexType	This data element indicates the specific predefined encoding profile used.	short	
EncSrcTypeType	This data element indicates the source for the encoding profile format. Valid values include but are not limited to "ITU-T" and "ATM Forum".	enum	
EndPointType	This specifies the characteristics of an end point of a connection	struct	
EquipmentHolderAddress Type	This data element identifies the equipment location for a circuit pack.	struct whose components include the shelf number (short) and the port number (short)	
EquipmentHolderFType	This data structure identifies the equipmentHolderF item named in the autodiscovery event.	struct listing the attribute values for the equipment HolderF Managed Entity	
EquipmentType	This data element identifies the inventory item named in the autodiscovery event.	short	
EthernetPMHistoryData Type	This data element provides the items in this record type.	struct	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`EthernetProfileType`	This data element provides the values for a profile of kind Ethernet.	`struct`	
`ExternalTimeType`	This data element establishes the local time to be associated with the network resource.	`Generalized TimeType`	
`FaxDemodIndType`	This boolean data element indicates whether or not a FAX demodulation is present, where the value TRUE implies its presence.	`boolean`	
`FilenameType`	Identifies the full pathname of a file. This file can contain configuration data uploaded from an NE, a location to transfer a file of records, or a location for an NE software generic or patch.	`string`	
`FileTransferHistory Type`	This data element provides a record concerning file transfer still logged on the Supplier Management System.	`struct`	
`FilterType`	This data element is defined by CosNotifyFilter::Filter.		
`FMDataIndType`	This boolean data element indicates whether or not Frame Mode Data is carried on this connection, where the value TRUE implies its presence.	`boolean`	
`FMMaxFrameLenType`	This data element the maximum length of a frame mode data unit.	`long long`	
`ForwardDelayType`	This data element gives the time (in hundredths of a second) that the bridge on the Ethernet card in the ONT (as a member of the community of all bridges in the Bridged Local Area Network) retains a packet before forwarding it.	`short`	
`ForwardErrorCorrection Type`	This data element indicates the FEC method.	`enum`	
`FullActionType`	Specifies the behaviour of the record set when it reaches its maximum size.	`enum`	Choice between wrap and halt.
`GeneralizedTimeType`	Provides a normalized metric of time. Used to avoid ambiguities involving time zones.	`string`	Only one allowed format YYYYMMDDHH MMSS.fffZ (i.e., Greenwich Mean Time).

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`HelloTimeType`	This data element provides the time interval (in hundredths of a second) between hello packets. It is the time interval, in hundredths of a second, that a bridge advertises its presence while a root or attempting to become a root.	`short`	
`HistoryDataType`	Identifies a record type comprised of performance monitoring statistics.	`RecordKind Type`	
`HourlyDailyWeekly MonthlyIndType`	Specifies whether the schedule has an Hourly, Daily, Weekly or Monthly time cycle	`enum`	Hourly, Daily, Weekly, Monthly
`IDLCCallProcessing ProfileType`	This data element provides the values for a profile of kind IDLC Call Processing.	`struct`	
`JitterBufferMaxType`	This data element provides the maximum depth of the jitter buffer associated with this service. Units are in milliseconds.	`long long`	
`JitterTargetType`	This data element provides the target value of the jitter buffer. The system will try to maintain the jitter buffer at the target value. Units are in milliseconds.	`long long`	
`LANType`	This data element provides information on the type of LAN employed, e.g., ethernet, token-ring, etc.	`short`	
`LESProfileType`	This data element provides the values for a profile of kind LES.	`struct`	
`LocalMaxNumVCC SupportedType`	This data element identifies the number of VCCs that can be supported by the ATM NE at this end of the interface.	`long long`	
`LocalMaxNumVCIBitsType`	This data element identifies the maximum number of allocated bits of the VCI sub-field that can be supported by the FSAN NE at this end of the interface.	`short`	
`LocalMaxNumVPC SupportedType`	This data element identifies the number of VPCs that can be supported by the OLT at this end of the interface.	`long long`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
LocalMaxNumVPIBitsType	This data element identifies the maximum number of allocated bits of the VPI sub-field that can be supported by the FSAN NE at this end of the interface.	short	
LoopbackCodeType	This data element identifies the type of in-band loopback code supported.	enum	
LoopbackInfoType	The data element specifies the loopback type, directionality, and location.	struct	
LoopbackInfoSeqType	Provides information concerning one loopback condition in the network resource.	struct	
LoopbackLocationIdSeq Type	Specifies the unique location Id for a ctp.	Sequence of octets of length 16	
LoopbackLocCodeType	This data element provides the code that identifies incoming ATM layer OAM loopback cells that are to be looped-back at this UNI.	string	
LoopbackTestType	This data element identifies the type of loopback test setup to be used or in use.	unsigned short	Possible values are defined in PhysicalLayerLoop back interface of Q834::Common.
LoopbackTrackingObject IdType	This data element identifies current loopback condition.	Tracking ObjectIdType	
MACBridgePortPMHistory DataType	This data element provides the items in this record type.	struct	
MACBridgePMHistoryData Type	This data element provides the items in this record type.	struct	
MACBridgeService ProfileType	This data element provides the values for a profile of kind MACBridgeService.	struct	
ManagedEntityIdType	This data element provides a unique name for a managed entity. The name includes an indication if the name represents a fine-grain managed object name, a façade object name, or just references a data structure.	struct of an enum and an RDNType	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`MaxAgeType`	This data element indicates the maximum age (in seconds) for an entry in the spanning tree listing. It indicates the maximum age in seconds of received protocol information before it is discarded.	`short`	
`MaxBSEgressType`	Maximum Burst Size - This parameter is required for real-time and non-real-time VBR traffic and for GFR traffic. It applies to CLP = 0 + 1 traffic flow for VBR.1, GFR.1, and GFR.2, and applies to CLP = 0 traffic flow for VBR.2 and VBR.3.	`long long`	
`MaxBSIngressType`	Maximum Burst Size - This parameter is required for real-time and non-real-time VBR traffic and for GFR traffic. It applies to CLP = 0 + 1 traffic flow for VBR.1, GFR.1, and GFR.2, and applies to CLP = 0 traffic flow for VBR.2 and VBR.3.	`long long`	
`MaxCPCSSDUSizeType`	This multi-valued data element represents the maximum CPCS_SDU size that will be transmitted over the connection in both the incoming (forward) and outgoing (backward) direction of transmission.	`struct`	
`MaxCPS_SDULengthType`	This data element provides the maximum allowed length of the Common Part Sublayer Service Data Unit (or CPS SDU) that will be allowed over the connection in either the upstream or downstream direction of transmission.	`long long`	
`MaxFrameSizeType`	This data element denotes the maximum allowed frame size to be transmitted across this interface.	`long long`	
`MaximumChanIdType`	This data element provides the maximum value for the Channel Id allowed for the channel within the connection.	`short`	
`MaxNumChannelsType`	This data element provides the maximum number of channels that can be carried by the VC Trail associated with the interworking vcCTP.	`short`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`MaxNumCIDsType`	This data element specifies the maximum number of channels within the VCC that can be active.	`short`	
`MaxPacketLengthType`	This data element specifies the maximum packet length.	`long long`	
`MaxSizeType`	Specifies the maximum size of the Record Set.	`unsigned long long`	In units to be determined by mutual agreement between supplier and operator.
`MaxSSSARSDULengthType`	This data element provides the maximum length allowed for an SSSAR-SDU of the Segmentation and Reassembly Service Specific Convergence sublayer.	`long long`	
`MFR1IndType`	This boolean data element indicates whether or not Multi-Frequency R1 dialled digits is transported on the connection, where the value TRUE implies its presence.	`boolean`	
`MFR2IndType`	This boolean data element indicates whether or not Multi-Frequency R2 dialled digits is transported on the connection, where the value TRUE implies its presence.	`boolean`	
`MFSEgressType`	Maximum Frame Size – This parameter is required for GFR traffic only.	`long long`	
`MFSIngressType`	Maximum Frame Size – This parameter is required for GFR traffic only.	`long long`	
`MinimumChanIdType`	This data element provides the minimum value for the Channel Id allowed for any channel within the connection.	`short`	
`MonitoredParameterSeqType`	This data element identifies a monitored parameter.	`string`	Monitored parameters are defined in `Q834Common::MonitoredParameter`
`MonitoringKindType`	This data element identifies the type of performance monitoring specified.	`string`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`MonitoringPoint ThresholdsType`	This data element provides a listing of monitoring point instances and the associated threshold data for a specified network resource.	sequence of `struct`	
`NameType`	This data element provides a CORBA name for a profile object.	`CosNaming:: Name`	
`NEFSANType`	This data structure identifies the generic equipment inventory item named in the autodiscovery event.	`struct` listing the attribute values for the `NEFSAN Managed Entity`	
`NEKindType`	This data element identifies the type of network elements that can be constructed.	`enum`	The choice is between `BPONOLT`, `BPONONT`, `BPONONU`, and `BPONNT` (Note).
`NotificationIdentifier Type`	Uniquely identifies the Notification.	`long long`	
`NTType`	This data structure identifies the NT inventory item named in the autodiscovery event.	`struct` listing the attribute values for the `NT Managed Entity`	
`OLTType`	This data structure identifies the OLT inventory item named in the autodiscovery event.	`struct` listing the attribute values for the `OLT Managed Entity`	
`ONTType`	This data structure identifies the ONT inventory item named in the autodiscovery event.	`struct` listing the attribute values for the `ONT Managed Entity`	
`ONUType`	This data structure identifies the ONU inventory item named in the autodiscovery event.	`struct` listing the attribute values for the `ONU Managed Entity`	
`OperationalStateType`	Specifes the operational state (enabled or disabled) of the managed entity.	`enum`	Defined in ITU-T Rec. X.780.
`OpticalWaveLengthArray SeqType`	Specifies the wavelength (in nanometers) and direction for a wavelength multiplexed on this optical port.	`struct`	
`ParameterSettingSeq Type`	This data element identifies a monitoring parameter together with its associated sliding window parameters.	`struct`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`PartiallyFilledCells Type`	This boolean data element identifies the number of leading octets in use.	`long long`	
`PasswordType`	Provides a security credential for access to the Supplier Management System application or servers.	`string`	A password supplied with a user Id must confirm to the passwordPolicy enforced by the Supplier Management System.
`PasswordPolicyType`	Specifies the Password Policy which is enforced by the Supplier Management System. It is composed of two components: UserLoginPolicyType and SessionPolicyType.	`struct`	
`PCMEncTypeType`	This data element indicates the type of PCM coding. Valid values include but are not limited to "mu-law PCM coding" and "alpha-law PCM coding".	`short`	
`PCREgressType`	Peak Cell Rate – This parameter is required for traffic of all service categories. It is applied to CLP = 0 flow for ABR and applied to CLP = 0 + 1 flow otherwise.	`long long`	
`PCRIngressType`	Peak Cell Rate – This parameter is required for traffic of all service categories. It is applied to CLP = 0 flow for ABR and applied to CLP = 0 + 1 flow otherwise.	`long long`	
`PerceivedSeverityType`	Defined by ITU-T Rec. X.780.	`enum`	
`PIDType`	This data element identifies the media type values that can be used in ATM encapsulation (defined in RFC 1483).	`short`	
`PlugInUnitFType`	This data structure identifies the PlugInUnitF inventory item named in the autodiscovery event.	`struct` listing the attribute values for the `PlugInUnit Type` Managed Entity.	
`PlugInUnitType`	Implementation specific, supplier provided name of circuit pack type.	`string`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`PortBandwidthSeqType`	This data element gives bandwidth by provisioned port.	`struct`	Listing examples include `ReservedBandwidthSeqType` and `AvailableSysBandwidthSeqType`
`POTSSignallingType`	This data element selects which signalling format should be used for POTS service.	`enum`	
`ProbableCauseType`	Data for probable cause.	`unsigned short`	Values are defined in `Q834Common::ProbableCause` and ITU-T Rec. X.780
`ProceduralStatusSetType`	Provides the procedural status of a non-terminated activity.	Sequence of `enum`	Defined in ITU-T Rec. X.780.
`ProfileInfoType`	This data element identifies the profile type and its attribute values.	`struct`	
`ProfileKindType`	This data element identifies the type of profile named.	`unsigned short`	Values provided in `Q834Common.`
`ProtectionParameterType`	This data element describes the protection switching parameters associated with a protection grouping of managed entities. Parameters include protection switching ratio, allowed protection switching mechanisms, revertive indicator, and wait time to revert.	`struct`	
`ProtectionUnitType`	This data element associates protected and protecting network resources.	`struct`	
`RASTimerType`	This data element provides the reassembly time (in seconds) of the Segmentation and Reassembly Service Specific Convergence sublayer for ITU-T Rec. I.366.1.	`short`	
`RateControlIndType`	This boolean data element indicates whether or not rate control is transported on the connection, where the value TRUE implies its presence.	`boolean`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`RecordType`	This is an item stored in a Record Set.	`any`	The value is a struct based in the `RecordKindType` defined in `Q834Common::RecordSetType`. Only one type of RecordKind can be stored in a record set.
`RecordSetIdType`	The ManagedEntityId of the Record Set.	`ManagedEntityIdType`	
`RecordSetStatusType`	Specifies the current status of the RecordSet.	`struct` of `currentSizeType, OperationalStateType,MaxSizeType, SizeThresholdType, filterName, Administra-tiveStateType, RecordKindType,` and `RecordSetUserLabel`	
`RecordKindType`	Identifies the type of the record.	`unsigned short`	Values defined in `Q834Common::RecordSetType`
`RecordsSeqType`	This data element provides a set of records grouped together by type.	sequence of `any`	See above.
`RemainingPasswordValidity`	Specifies the password validity in number of days.	`long`	Value ≥ 1
`ReservationIdType`	An identifier that correlates bandwidth reserved on a PON or within an OLT system with a service instance.	`string`	
`ReservationInfoType`	This data element provides a complete accounting of the reservation information and associated service connection information.	`struct`	
`ReservedBandwidthSeqType`	Quantifies the amount of bandwidth associated with a reservation Id for a port.	sequence of `PortBandwidthType`	
`ResourceSelfTestInfoType`	Provides one testing diagnostic to be used in the resource self test.	`short`	Values are determined by supplier implementation.

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
ResourceSelfTestResult Type	Specifies one result for Resource self test.	struct	
ScheduledFileTransfer Type	Identifies a pending scheduled file transfer activity for the Supplier Management System.	struct	
ScheduledSynchNEType	This data element associates an NE with its schedule for synchronisation.	struct	
ScheduledTestNESeqType	This data element associates an NE with a scheduled test.	struct	
SchedulerType	Identifies an instance of Scheduler within the Supplier Management System.	struct	
SCREgressType	Sustainable Cell Rate – This parameter applies to real-time and non-real-time VBR. It applies to CLP = 0 + 1 traffic flow for VBR.1 and applies to CLP = 0 traffic flow VBR.2 and VBR.3.	long long	
SCRIngressType	Sustainable Cell Rate – This parameter applies to real-time and non-real-time VBR. It applies to CLP = 0 + 1 traffic flow for VBR.1 and applies to CLP = 0 traffic flow VBR.2 and VBR.3.	long long	
SecurityAlarmLogRecord Type	This data element provides the items in this record type.	struct	
SegmentEndpointIndType	Indicates whether or not the constructed vpNetworkCTP is to be viewed as a segment or endpoint for ATM OAM cell loopback testing.	enum	choice of segment, endpoint, or none
SegmentLengthType	This data element provides the length of segment for the Segmentation and Reassembly Service Specific Convergence sublayer. It ranges from 0 to the maximum value provided by MaxCPS_SDULen data element.	long long	
SerialNumType	Provides the unique hardware Id used in the ranging activity defined in ITU-T Rec. Q.983.1.	string	
ServiceAffectingType	This data element indicates whether or not a failure condition is affecting service if it can be determined.	enum	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`ServiceCategoryType`	Indicates the service category as defined in ATM-Forum TM 4.0. Valid values are CBR, rt-VBR, nrt-VBR, UBR, ABR, or GFR.	`enum`	
`ServiceCatType`	This data element indicates the type of service category provided by AAL2. Valid values include but are not limited to "Audio" and "Multirate".	`enum`	
`ServiceInstanceIdType`	Operator-provided label associated by the Supplier Management System with connections.	`string`	
`ServiceOutageRecord Type`	This data element provides the items in this record type.	`struct`	
`SessionPolicyType`	Specifies the rules governing GUI client sessions interconnected to the Supplier Management System.	`struct`	
`SizeThresholdType`	Specifies the threshold in the record set to raise alarms.	`unsigned short`	The value is interpreted as a whole number percentage (0-100%) of `MaxSize`.
`SlotAssignmentType`	Provides an association of a slot to a provisioning assignment.	`struct` showing the slot number and plug-in type	
`SoftwareDownload TrackingObjectIdType`	Identifies one software download activity in progress or have failed to complete successfully.	`TrackObjectId Type`	
`SONETSDHLinePMHistory Data`	Provides the record structure itemizing the performance data collected in a 15-minute interval for monitoring point rsTTPF.	`struct`	
`SONETSDHSection AdaptationPM HistoryData`	Provides the record structure itemizing the performance data collected in a 15-minute interval for monitoring points for au3CTPF or au4CTPF.	`struct`	
`SONETSDHSectionPathPM HistoryData`	Provides the record structure itemizing the performance data collected in a 15-minute interval for monitoring points for msTTPF, vc3CTPF or vc4CTPF.		

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`SpanningTreeIndType`	This boolean data element indicates whether or not a spanning tree algorithm is enabled. The value TRUE means enabled.	`boolean`	
`SpecificProblems`	This data element provides a listing of code values for all specific problems associated with a failure condition and alarm.	sequence of `long`	Values defined by supplier.
`SSADTIndType`	This boolean data element indicates whether or not the assured data transfer mechanism has been selected, with value TRUE indicating selection.	`boolean`	
`SSCSParameterProfile1 Type`	This data element provides the values for a profile of kind SSCS Parameter Profile 1.	`struct`	
`SSCSParameterProfile2 Type`	This data element provides the values for a profile of kind SSCS Parameter Profile 2.	`struct`	
`SSCSTypeType`	This data element identifies the SSCS type for the AAL.	`enum`	
`SSTEDIndType`	This boolean data element indicates whether or not the transmission error detection mechanisms have been selected, with value TRUE indicating selection.	`boolean`	
`StateChangeDefinition Type`	This data element provides a listing of all state and status variable changes triggered by a failure condition causing an alarm.	`AttributeChan geSetType`	Imported from ITU-T Rec. X.780.
`StatusAttributeType`	A pair showing the StatusValueType and completion percentage for the backup or restore of configuration data associated with an NE.	`struct`	
`StatusValueType`	Provides a choice of values between ProceduralStatusSetType and OtherStatus values indicating completion aspects.	`union`	
`StructuredDataTransfer Type`	This boolean data element indicates whether Structured Data Transfer (SDT) has been configured at the AAL.	`boolean`	
`SubType`	This data element identifies the AAL subtype.	`enum`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`SupplyPowerIndType`	This data element indicates whether or not the associated port provides power.	`boolean`	
`SWPValueType`	Shows a monitoring point and its associated sliding window parameters.	`struct`	
`SynchChangeIndType`	This boolean data element indicates whether or not synchronization of change in SSCS operation is transported on the connection, where the value TRUE implies its presence.	`boolean`	
`SystemTimingType`	Specifies the primary and secondary clock source for the NE.	`struct` of `struct`	
`T303Type`	This data element defines the maximum length of time in milliseconds that the ONT will wait for a reply to the "SETUP" message found at Layer 3 for the TMC or the CSC.	`enum`	
`T396Type`	This data element specifies the length of time that the ONT will wait for a reply to a "SETUP" message following the initial expiration of the timer T303.	`enum`	
`TargetType`	This data element provides a listing of network resources.	sequence of `struct` of `Managed EntityIdType` and `string`	Allows selection at system, NE, plug-in unit type, or specific slot level.
`TargetActivityType`	Provides an association of activityLevel, activityType and administrative domain to be associated with a user or a user group.	`struct`	
`TCAdaptionProtocol MonitoringPM HistoryData`	Provides the record structure itemizing the performance data collected in a 15-minute interval for monitoring points for tcAdaptorF.		
`TestHistoryType`	This data element shows archived information on one completed testing activity for the Supplier Management System.	`struct`	
`TestIterationNumType`	This data element specifies the number of times that a test is to be repeated.	`short`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`TestTrackingObjectIdType`	This data element identifies a test in progress or pending within the Supplier Management System.	`TrackingObjectId`	
`ThresholdDataProfileType`	This data element is a listing of pairs: a thresholdDataProfile instance and a monitoring point type.	sequence of `struct`	
`ThresholdDataType`	This data element lists names of performance parameter with a threshold value for each.	`struct`	
`ThresholdInfoType`	This data element supplies information for a quality of service alarm event and provides the identity of the performance parameter, its observed value, and the upper and lower values that defined its threshold value range.	`struct`	
`ThresholdsList`	This data element lists relationships between monitoring termination point types and their associated threshold data profile.	sequence of `struct` the first component indicating the type of monitoring point and the second giving the reference to a `threshold Data profile`	
`ThresholdsType`	This data element provides an association of monitoring point types associated with its threshold data profile name.	`struct`	
`TimerCULengthType`	This data element provides the value for the "combined use" timer Timer_CU.	`long long`	
`TimingReferenceType`	This data element defines how the internal timing is derived.	`enum`	
`TotalEgressBandwidthType`	This data element identifies the total amount of egress bandwidth for an ATM Interface.	`long long`	
`TotalIngressBandwidthType`	This data element identifies the total amount of ingress bandwidth for an ATM Interface.	`long long`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`TrackingObjectIdType`	This data element helps to identify the status of a requested activity whose time of completion takes longer than a few seconds.	`unsigned long`	Examples include `TransferTrackingObjectIdType`, `SoftwareDownloadTrackingObjectType`, and `TestTrackingObjectIdType`.
`TrafficDescriptorProfileType`	This data element provides the values for a profile of kind Traffic Descriptor.	`struct`	
`TransferTrackingObjectIdType`	Identifies either an outstanding file transfer activity in the Supplier Management System.	`string`	
`TriggerTimeMatrixType`	Specifies trigger times associated with a period schedule.	`struct`	
`UNIProfileType`	This data element provides the values for a profile of kind UNI.	`struct`	
`UPCNPCDisagreementPMHistoryDataType`	This data element provides the items in this record type.	`struct`	
`UPCNPCIndicatorType`	This boolean data element determines whether or not policing is performed for all connections at the interface.	`boolean`	
`UsageStateType`	This data element enumerates the value for the usage state variable.	`enum`	Valid enum values are idle, active, busy. Defined in ITU-T Rec. X.780.
`UserGroupId`	This is the operator-provided name for a grouping of users to the Supplier Management System.	`UserLabelType`	Cannot be the empty string.
`UserGroupType`	Provides the User Label for the user group, lists the users who are members of the group, and enumerates their target activities.	`struct` of `UserGroupIdType`, `UserIdSeqType`, and `TargetActivitySeqType`	
`UserIdType`	UserLabel assigned for users of the Supplier Management System.	`UserLabel`	
`UserLabelType`	Provides an identifier that is created and provided by the Operator or OMS to associate with a managed resource by the Supplier Management System.	`string`	
`UserLoginPolicyType`	Specifies the policy governing user login.	`struct`	

Table A.1/Q.834.4 – Data elements and definitions

Data element name	Definition	Syntax	Comments
`UserLoginPolicy ViolationReasonType`	Specifies one reason for rejecting the assignment of a password to a user.	`enum`	
`UserType`	Provides the user Id, the user groups to which the user belongs, and the users assigned target activities.	`struct` of `UserIdType`, `UserGroupId SeqType`, and `Target ActivitySeq Type`	
`VersionType`	Provides one version identifier of the hardware or software for the system by managedEntityIdType.	`struct`	
`VoicePMHistoryDataType`	This data element provides the items in this record type.	`struct`	
`VoiceServicesProfile AAL1Type`	This data element provides the values for a profile of kind Voice Services using AAL1.	`struct`	
`VoiceServicesProfile AAL2ProfileType`	This data element provides the values for a profile of kind Voice Services using AAL2.	`struct`	
`VPVCPMHistoryDataType`	This data element provides the items in this record type.	`struct`	
NOTE – If Q834Build module is considered for use in the management of other types of access technologies, then either the enumerated values should be expanded, or the syntax could be changed to short, with constants identifying the types of network elements to be constructed.			

Annex B

Exceptions

Table B.1 provides the list of all possible exceptions and circumstances under which they may be raised by one or more operations in this interface specification.

Table B.1/Q.834.4 – Exceptions

Exception raised	Description
`AccessDenied`	System is not granted access to this interface object.
`ActivationCompleted`	Indicates that software activation has been executed so that activation cannot be cancelled.
`ActivationFailure`	Software activation process failure.
`ActivityCompleted`	The software activity has been executed and cannot be cancelled.
`ActivityInProgress`	This exception is raised when the software activity has been initiated and cannot be cancelled.

Table B.1/Q.834.4 – Exceptions

Exception raised	Description
`AddressLabelMismatch`	The identified NE does not have the current DCN Address provided in the request.
`APONLayerFailure`	There was a APON protocol ranging failure between the OLT and the designed subtending node.
`BackupInProgress`	This exception is raised when the request is issued while the backup is in progress.
`CannotAssignManagedEntityId`	The Supplier Management System was unable to set the ManagedEntityId for the OLT, thus indicating that the Supplier Management System is unable to manage the OLT.
`CannotRetrieveUserLabel`	The Supplier Management System was unable to read the User Label provisioned on the OLT.
`CollectionLimitation`	The Supplier Management System cannot collect data for the given time duration and granularity period due to implementation restrictions.
`CollectionPeriodPast`	When period end time is lesser than or equal to the current time.
`CommFailure`	There was a DCN link failure between the NE and the Supplier Management System.
`ConnectionAlreadyExists`	There already exists a subnetwork connection with the same endpoints.
`ConnectionCountExceeded`	The maximum number of connections for the OLT or PON port has been exceeded with this request for service provisioning.
`DCNTimeout`	The DCN communications link between at least one of the NEs and the Supplier Management System is so congested that current state or status information cannot be transferred within a system defined synch time.
`DeniedAccess`	System is not granted access to the NE.
`DuplicateProfileName`	This exception is raised when the new profile name duplicates an existing profile name.
`DuplicateSerialNumber`	There exists other equipment of the same type with this serial number.
`DuplicateUserGroupId`	The id is already used for another User Group.
`DuplicateUserId`	Access control profile has already been established for this User Id.
`DuplicateUserLabel`	The User Label provided in the request has been used to label another managed entity of the same time.
`EquipmentFailure`	The NE currently has a failure condition preventing the requested transaction from being completed.
`HWServicesMismatch`	The replacement NE cannot perform the provisioned services.
`InstallationFailure`	Software installation process failure
`InsufficientBW`	The CAC algorithm indicates that requested service requires too much bandwidth for the OLT.
`InsufficientMemory`	There is insufficient memory on the NE to load the software unit.
`InsufficientPONBW`	The ONT or ONU cannot be ranged due to insufficient bandwidth on the APONLink.

Table B.1/Q.834.4 – Exceptions

Exception raised	Description
`InterfaceSpeedNotChangeable`	The physical port cannot support the new interface speed or if the speed in not configurable.
`IntervalCountTooLarge`	This exception is raised when the requested intervals exceed the maximum supported by the Supplier Management System. The exception indicates the maximum allowed monitoring intervals supported by the Supplier Management System.
`InvalidDCNAddress`	The specified DCN address is not valid.
`InvalidEquipmentCode`	The equipment code does not conform to syntax.
`InvalidExternalTime`	External Time specified is not valid.
`InvalidLocationId`	The LLID specified is not valid.
`InvalidPort`	The specified PON port is not valid.
`InvalidProtectionScheme`	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.
`InvalidScheduler`	The Scheduler parameters values are outside defined scope.
`InvalidSerialNumSyntax`	Syntax of the serial number provided does not match definition rules.
`InvalidSlotAssignmentList`	Expected slot provisioning rules are violated by slot assignment provided.
`InvalidSoftwareTracking Object`	The referenced software tracking object is not the most recent associated with the installation of a software load on the NE.
`InvalidStartTime`	The start time is inconsistent with the current time, the current trigger time matrix, or the new stop time.
`InvalidStopTime`	The new stop time is inconsistent with the current trigger time matrix, the current time, or the new start time.
`InvalidTestOperations`	The requested self test operation is not valid.
`InvalidTimeoutPeriod`	Designated timeout period violates definition of valid values.
`InvalidTrigger`	The specified Trigger has values that cannot be interpreted by the Scheduler.
`InvalidUserLabelSyntax`	The specified UserLabel does not follow established rules for userLabel.
`LockedAlready`	The administrative state of the named managed entity is locked already and it has not been performing its normal function.
`MaxSubtendingNodesExceeded`	The maximum engineered number of subtending nodes for the identified PON interface has been exceeded with this request for service provisioning or reservation.
`NoResponse`	The ONT or ONU could not be ranged and the failure was due to something other than a detected problem with the serial number syntax, the APON Layer protocol, or duplicate or invalid User Labels.
`NoSuchRecords`	No records among the designated Record Sets match the selection criteria.
`NoSynchInProgress`	The exception is raised if there is no synchronisation process in progress.

Table B.1/Q.834.4 – Exceptions

Exception raised	Description
`NotAvailableForTest`	The specified CTP is not available for this test.
`ParameterViolation`	This exception is raised when the endpoint parameters do not match the protocol characteristics of the port, or when the values are out of range or invalid duplicates.
`ProfileInUse`	This exception is raised when the profile may not be deleted because it is still being used to characterize managed entities within the management jurisdiction of the Supplier Management System.
`ProfileSuspended`	The named profile(s) in the invocation have been suspended for use within the Supplier Management System by the OMS or operator.
`RecordSetExists`	The record set defined by the parameters of the creation request already exists in the Supplier Management System.
`RemainingContainedManaged Entities`	Contained circuit packs or equipment holders have not been deleted yet.
`RemainingReservations`	The node cannot be deleted as resource reservations still exist.
`RemainingSubnetwork Connections`	The node of plug-in unit cannot be deleted as there are remaining subnetwork connections.
`ScheduleInUse`	There are still activities scheduled by the named Scheduler.
`SlotAlreadyAssigned`	The requested slot is already provisioned.
`SoftwareLoadHardwareMismatch`	The former NE configuration data could not be downloaded to the NE because there were changes made to the NE hardware that caused an incapability.
`SoftwareLoadHWMismatch`	The designated software may not be loaded onto the equipment hardware since the version of the hardware cannot accept the software load.
`SoftwareNotYetInstalled`	The software may not be activated since it has not been installed yet.
`SoftwareTrackingObjectInUse`	There are outstanding software activities tracked by this object and it may not be deleted.
`SourceUnreachable`	The server holding the software load to be downloaded could not be reached by the OLT.
`SynchNotScheduled`	This exception is raised when the schedule modification references a NE for which no synchronization schedule has previously been established.
`Timeout`	The process duration reached a system-defined timeout before the process could complete.
`TooManyNEs`	The Supplier Management System cannot manage one more OLT.
`TooManyRecords`	The number of records selected for retrieval produces a response to the request that exceeds a predetermined size.
`UncontrolledTestInProgress`	The self test cannot be cancelled because of a uncontrolled test.
`UnknownBackupProcess`	The named transfer-tracking object identifying the file transfer process is unknown to the Supplier Management System.

Table B.1/Q.834.4 – Exceptions

Exception raised	Description
`UnknownConnection`	The subnetwork connection is not known by the Supplier Management System.
`UnknownDestinationServer`	The identified destination server cannot be accessed by the transfer agent.
`UnknownHistoryDataType`	The history data type is unknown in the Supplier Management System.
`UnknownManagedEntity`	The specified Managed Entity is unknown to the Supplier Management System.
`UnknownMonitoringPointTypes`	The monitoringPointType is unknown to the Supplier Management System.
`UnknownNE`	Identified NE is unknown to the Supplier Management System.
`UnknownOption`	The given value of the archive administrativeState is "ShuttingDown".
`UnknownParameters`	The given monitored parameter is unknown in the Supplier Management System.
`UnknownPort`	The identified port is unknown to the Supplier Management System.
`UnknownProfiles`	This exception is raised if the profile name provided is unknown to the Supplier Management System and cannot be retrieved from the profile object repository.
`UnknownRecordSet`	Record set identified in the request is unknown to the Supplier Management System.
`UnknownReservationId`	The Supplier Management System does not recognize this Reservation Id.
`UnknownRestoreProcess`	The named transfer-tracking object identifying the file transfer process is unknown to the Supplier Management System.
`UnknownScheduler`	The named scheduler is unknown to the Supplier Management System.
`UnknownService`	The described Service is unknown to the Supplier Management System.
`UnknownServiceInstance`	The Service Instance is unknown to the Supplier Management System.
`UnknownSlot`	The requested slot is unknown in the NE.
`UnknownSoftwareDownloadTrackObject`	The named Software Unit (refers to software distributed to an NE) is unknown to the Supplier Management System.
`UnknownSoftwareLoad`	The specified software unit cannot be found.
`UnknownSourceServer`	The Supplier Management System and/or OLT cannot communicate with the Source Server. The DCN address is unknown or access is blocked.
`UnknownSystemTimingSource`	External time source unknown to the Supplier Management System.
`UnknownTargets`	The list of target activities is unknown to the Supplier Management System.

Table B.1/Q.834.4 – Exceptions

Exception raised	Description
`UnknownTest`	The test specified by the Test Tracking Object Id is not known in the Supplier Management System.
`UnknownTransferProcess`	The status of the identified transfer process could not be checked because it is unknown to the Supplier Management System.
`UnknownUserGroupId`	User Group is unknown to the Supplier Management System.
`UnknownUserIds`	This exception is raised when any userId is unrecognized.
`UnrecognisedVersion`	Equipment version provided does not match known values.
`UserGroupNotEmpty`	Non-empty User Group cannot be deleted.
`UserLoginPolicyViolation`	The specified assignment of a new password to a user violates the user login policy currently enforced by the Supplier Management System.

Annex C

IDL files

If the following files are saved as text files, then any subset of them can be compiled successfully using any OMG IDL compiler conforming to CORBA Specification 2.1 or later provided that the subset includes Q834Common.idl, and the standardized CosNaming.idl, CosNotifyFilter.idl, CosNotification.idl, X780.idl, and CosNotifyComm.idl.[11]

C.1 Q834AccessControl.idl

```
#ifndef __Q834_4_ACCESSCONTROL_DEFINED
#define __Q834_4_ACCESSCONTROL_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module AccessControl {
// Begin definitions from other idl files

// From Q834Common

    typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
    typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
    typedef Q834Common::AdministrationDomainSeqType
                AdministrationDomainSeqType;
```

[11] See references [18] and [19].

```
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;

#define AccessDenied Q834Common::AccessDenied

// End definitions from other idl files

// Local data types

     struct UserLoginPolicyType {
          short minUserId; // Minimum length of userid
          short minPassword; // Minimum length of password
          short passwordReuse;
          short loginAttempts;
          long passwordValidity;
          boolean alphanumeric;//?should password contain alphanumeric mixture
          boolean specialCharacters;
              //?should password contain special characters
          boolean repeatingCharacters; //?should password contain repeating
          boolean disallowUserId; //disallow username in password
     };

     struct SessionPolicyType {
          short sessionInactiveTime;
          short inactiveUserIdDisableTime;
          short multipleActiveLogins;
     };

     struct PasswordPolicyType {
          UserLoginPolicyType userLoginPolicy;
          SessionPolicyType sessionPolicy;
     };


     typedef sequence<UserIdType> UserIdSeqType;

     enum ActivityLevelType {
          monitorOnly,  // read
          allowedToExecute,  // write
          noAccess
     };

     typedef short ActivityType;



     struct TargetActivityType {
          ActivityType type;
          ActivityLevelType activityLevel;
          AdministrationDomainSeqType AdministrationDomainSeq;
     };

     typedef sequence<TargetActivityType> TargetActivitySeqType;

     enum UserLoginPolicyViolationReasonType {
          minUserId,
          minPassword,
          passwordReuse,
          loginAttempts,
          passwordValidity,
          alphanumeric,
          specialCharacters,
          repeatingCharacters,
          disallowUserId
     };



     typedef sequence<UserLoginPolicyViolationReasonType>
UserLoginPolicyViolationReasonSeqType;
          typedef sequence<UserLabelType> UserGroupIdSeqType;

     struct UserType {
          UserIdType userId;
          UserGroupIdSeqType userGroupIdSeq;
          TargetActivitySeqType TargetActivitySeq;
     };

     struct UserGroupType {
          UserLabelType userGroupId;
          UserIdSeqType userIdSeq;
          TargetActivitySeqType TargetActivitySeq;
     };

     typedef sequence<UserType> UserSeqType;
     typedef sequence<UserGroupType> UserGroupSeqType;


     // Local exceptions

     exception UnknownUserIds {
          UserIdSeqType userIdSeq;
     };
     exception DuplicateUserId {};
     exception UnknownUserGroupId {};
     exception DuplicateUserGroupId {};
     exception UnknownTargets {
          TargetActivitySeqType unknownTargetActivities;
     };
     exception UserGroupNotEmpty {};
     exception UserLoginPolicyViolation {
          UserLoginPolicyType userLoginPolicy;
          UserLoginPolicyViolationReasonSeqType reason;
     };
     // End local definitions

     valuetype AccessControlMgrValueType: itut_x780::ManagedObjectValueType {

          public PasswordPolicyType passwordPolicy; // GET
          public UserSeqType userList; // GET
          public UserGroupSeqType userGroupList; // GET
     };


     interface AccessControlMgr : itut_x780::ManagedObject {

     // define the activities
          const short ALL_ACTIVITIES = 0;
          const short ACCESS_CONTROL_MANAGEMENT = 1;
          const short ALARM_EVENT_CONFIGURATION_MANAGEMENT = 2;
          const short SCHEDULE_ACTIVITY = 3;
          const short SOFTWARE_DOWNLOAD = 4;
          const short TEST_CONTROL = 5;
          const short SYNCHRONISE_CURRENT_EVENT_LIST = 6;
          const short SYNCHRONISE_NE = 7;
          const short RANGE_NE = 8;
          const short REGISTER_SYSTEM = 9;
```

```
      const short RESERVE_RESOURCES = 10;
      const short PROFILE_MANAGEMENT = 11;
      const short PROVISION_NE = 12;
      const short PROVISION_TELEPHONY_SERVICE = 13;
      const short PROVISION_PACKETISED_DATA_SERVICES = 14;
      const short PROVISION_VIDEO_SERVICE = 15;
      const short PROVISION_LEASED_LINE_SERVICE = 16;
      const short BULK_TRANSFER = 17;
      const short HISTORY_DATA_COLLECTION = 18;
      const short CONTROL_ARCHIVING = 19;
      const short CONTROL_PERFORMANCE_MONITORING = 20;
      const short CONFIGURATION_BACKUP_RESTORE = 21;


   // See 9.1.1.1 for the description of the behaviour of this operation

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

   // See 9.1.1.2 for the description of the behaviour of this operation

      PasswordPolicyType passwordPolicyGet()
           raises (AccessDenied);

   // See 9.1.1.3 for the description of the behaviour of this operation

      UserSeqType userListGet ()
           raises (AccessDenied);

   // See 9.1.1.4 for the description of the behaviour of this operation

      UserGroupSeqType userGroupListGet ()
           raises (AccessDenied);

   // See 9.1.1.5 for the description of the behaviour of this operation

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

   // See 9.1.1.6 for the description of the behaviour of this operation

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


   // See 9.1.1.7 for the description of the behaviour of this operation

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


   // See 9.1.1.8 for the description of the behaviour of this operation

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


 // See 9.1.1.9 for the description of the behaviour of this operation

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



 // See 9.1.1.10 for the description of the behaviour of this operation

      void addUsersToGroup (
           in UserLabelType userGroupId,
           in UserIdSeqType userIdList )
           raises (AccessDenied,
                UnknownUserGroupId); // duplicate users are ignored


 // See 9.1.1.11 for the description of the behaviour of this operation

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


 // See 9.1.1.12 for the description of the behaviour of this operation

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


 // See 9.1.1.13 for the description of the behaviour of this operation

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

 // See 9.1.1.14 for the description of the behaviour of this operation

      void createUser (
           in UserIdType userId,
           in PasswordType password,
           in TargetActivitySeqType targetAdditions )
           raises (DuplicateUserId,
                UnknownTargets,
                AccessDenied,
                UserLoginPolicyViolation);
     // See 9.1.1.15 for the description of the behaviour of this operation

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


     // See 9.1.1.16 for the description of the behaviour of this operation

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

     }; // interface AccessControlMgr

}; // module AccessControl

}; // module q834_4

#endif
```

C.2 Q834Build.idl

```
#ifndef __Q834_4_BUILD_DEFINED
#define __Q834_4_BUILD_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module Build {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;

     typedef Q834Common::SlotAssignmentSeqType SlotAssignmentSeqType;
     typedef Q834Common::SerialNumType SerialNumType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ExternalTimeType ExternalTimeType;
     typedef Q834Common::SystemTimingType SystemTimingType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
     typedef Q834Common::AdministrationDomainType AdministrationDomainType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::LoopbackLocationIdSeqType LoopbackLocationIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;
```

```
#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define UnknownProfiles Q834Common::UnknownProfiles
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define ParameterViolation Q834Common::ParameterViolation
#define ProfileSuspended Q834Common::ProfileSuspended
#define UnknownNE Q834Common::UnknownNE

// End definitions from other idl files

     // Local data types


     enum NEKindType {
          BPONOLT,
          BPONONT,
          BPONONU,
          BPONNT
     };




     struct ProtectionUnit
     {
           ManagedEntityIdType portId;
           boolean protectingInd; //TRUE = Protecting, FALSE = Protected
     };

     typedef sequence<ProtectionUnit> ProtectionUnitSeqType;

     enum ProtectionRatioType
     {
          oneForOne,
          oneForN, // N > 1
          mForN // M > 1
     };
               enum AllowedProtectionSwitchingType
     {
          automatic,
          manual,
          both
     };

        struct ProtectionParameterType
        {
          ProtectionRatioType protectionRatio;
          AllowedProtectionSwitchingType protectionSwitchingMethod;
          boolean revertiveInd; // TRUE = revertive
          long waitToRestore; // number of milliseconds before // reverting
        };

     enum SegmentEndpointIndType
     {
          segment,
          endpoint,
          none
     };
```

```
enum DirectionType
{
              unidirection,
              bidirection
};



struct OpticalWaveLengthArrayType {
              unsigned long wavelength;
              DirectionType direction;
     };
     typedef sequence <OpticalWaveLengthArrayType>
OpticalWaveLengthArraySeqType;


     // Local exceptions

     exception InvalidExternalTime {};
     exception UnrecognisedVersion {};
     exception DuplicateSerialNumber {};
     exception InvalidSlotAssignmentList {};
     exception RemainingContainedManagedEntities {
              ManagedEntityIdSeqType containedManagedEntities ;
     };
     exception UnknownNE {};
     exception UnknownSystemTimingSource {};
     exception RemainingReservations {
          ReservationIdSeqType remainingReservationList;
     };
     exception InvalidEquipmentCode {};
     exception SlotAlreadyAssigned {};
     exception UnknownSlot {};
     exception RemainingSubnetworkConnections {
          ManagedEntityIdSeqType containedManagedEntities ;
     };
          exception InterfaceSpeedNotChangeable {};
     exception InvalidProtectionScheme {};


     // End local definitions

     valuetype BuilderValueType: itut_x780::ManagedObjectValueType {

          public ManagedEntityIdSeqType createdNodes; // GET
     };

     interface Builder: itut_x780::ManagedObject {


     // See 9.2.1.1 for the description of the behaviour of this operation


ManagedEntityIdType buildNode (
              in NEKindType nEKind,
              in string supplierName,
              in string location,
              in VersionType hWVersion,
              in SerialNumType serialNum,
              in NameSeqType alarmSeverityProfiles,
              in NameSeqType thresholdDataProfiles,
              in SlotAssignmentSeqType slotAssignmentList,
              in ManagedEntityIdType port, // OLT PON 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);

     // See 9.2.1.2 for the description of the behaviour of this operation

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

     // See 9.2.1.3 for the description of the behaviour of this operation


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

     // See 9.2.1.4 for the description of the behaviour of this operation

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



     // See 9.2.1.5 for the description of the behaviour of this operation
     void modifyPort (
          in ManagedEntityIdType physicalPathTPId,
          in NameSeqType newAlarmSeverityProfiles,
          in NameSeqType newThresholdDataProfiles,
          in NameSeqType newPortProfiles,
          in string newFrameFormat,
          in AdministrativeStateType administrativeState,
          in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
          in LoopbackLocationIdSeqType newLoopbackLocationId,
          in unsigned long newInterfaceSpeed,
          in unsigned long aRCTimer)
          raises (UnknownManagedEntity,
               UnknownProfiles,
               AccessDenied,
               InterfaceSpeedNotChangeable,
               ProfileSuspended);

// See 9.2.1.6 for the description of the behaviour of this operation

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

// See 9.2.1.7 for the description of the behaviour of this operation

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



// See 9.2.1.8 for the description of the behaviour of this operation

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

// See 9.2.1.9 for the description of the behaviour of this operation

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

// See 9.2.1.10 for the description of the behaviour of this operation

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

// See 9.2.1.11 for the description of the behaviour of this operation

     void modifyProtectionUnitList(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionUnitSeqType deltaProtectionUnitList,
          in boolean addDeleteInd) // TRUE = add
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// See 9.2.1.12 for the description of the behaviour of this operation
     void deleteProtectionGrouping(
          in ManagedEntityIdType protectionGroupingId)
          raises (UnknownManagedEntity,
                  AccessDenied);

// See 9.2.1.13 for the description of the behaviour of this operation
     ManagedEntityIdType buildBridge(
          in NameType mACBridgeProfile,
          in ManagedEntityIdType uplinkPort,
          in ManagedEntityIdSeqType uNIPortList)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// See 9.2.1.14 for the description of the behaviour of this operation
     void modifyBridgeProfile(
          in ManagedEntityIdType bridgeId,
          in NameType newMACBridgeProfile)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// See 9.2.1.15 for the description of the behaviour of this operation
     void modifyBridgePortList(
          in ManagedEntityIdType bridgeId,
          in ManagedEntityIdSeqType deltaUNIPortList,
          in boolean addDeleteInd)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
               UnknownManagedEntity);
     // See 9.2.1.16 for the description of the behaviour of this operation
          void deleteBridge(
               in ManagedEntityIdType bridgeId)
               raises (AccessDenied,
                    RemainingSubnetworkConnections,
                    UnknownManagedEntity);

     // See 9.2.1.17 for the description of the behaviour of this operation
          ManagedEntityIdType buildVPNetworkCTP(
               in ManagedEntityIdType port,
               in short vPI,
               in NameType trafficDescriptorProfileName,
               in ATMOverbookingFactorType overbookingFactor,
               in UserLabelType userLabel,
               in SegmentEndpointIndType segmentEndpointInd)
               raises (AccessDenied,
                    UnknownManagedEntity,
                    UnknownProfiles,
                    ParameterViolation,
                    ProfileSuspended);

     // See 9.2.1.18 for the description of the behaviour of this operation
          void deleteVPNetworkCTP (
               in ManagedEntityIdType vPNetworkCTP)
               raises (AccessDenied,
                    RemainingSubnetworkConnections,
                    UnknownManagedEntity);

     // See 9.2.1.19 for the description of the behaviour of this operation

          ManagedEntityIdSeqType createdNodesGet ()
               raises (AccessDenied);


     }; // interface Builder

}; // module Build

}; // module q834_4

#endif
```

C.3 Q834Common.idl

```
#ifndef __Q834_4_COMMON_DEFINED
#define __Q834_4_COMMON_DEFINED

#include "CosNaming.idl"
#include "CosNotifyFilter.idl"
#include "itut_x780.idl"
#include "TimeBase.idl"


#pragma prefix "itu.Int"

module q834_4 {

module Q834Common {

// Begin definitions from other idl files

// From CosNaming
     typedef CosNaming::Name NameType;
```

```
// From CosNotifyFilter
     typedef CosNotifyFilter::Filter FilterType;

// From TimeBase
     typedef TimeBase::UtcT UtcT;

// From X780

     typedef itut_x780::ProbableCauseType ProbableCauseType;
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::OperationalStateType OperationalStateType;
     typedef itut_x780::ProceduralStatusSetType ProceduralStatusSetType;
     typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
     typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
     typedef itut_x780::UsageStateType UsageStateType;
     typedef itut_x780::ControlStatusSetType ControlStatusSetType;
     typedef itut_x780::SpecificProblemSetType SpecificProblemSetType;
     typedef itut_x780::BackedUpStatusType BackedUpStatusType;
     typedef itut_x780::AttributeChangeSetType AttributeChangeSetType;
     typedef itut_x780::SecurityAlarmCauseType SecurityAlarmCauseType;

// End definitions from other idl files

// Local data types

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

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

     enum IdType {
          none,
          x780_fineGrained,
          x780_coarseGrained
     };

     typedef RDNType MEIdType;

     struct ManagedEntityIdType {
          IdType id;
          MEIdType mEId;
     };

     typedef sequence<ManagedEntityIdType> ManagedEntityIdSeqType;
     typedef string UserLabelType;
     typedef string SerialNumType;
     typedef string VersionType;
     typedef string PlugInUnitType;
     typedef sequence<UserLabelType> UserLabelSeqType;

     typedef UserLabelType AdministrationDomainType;
     typedef sequence<AdministrationDomainType> AdministrationDomainSeqType;

     typedef string DCNAddressType;

     typedef unsigned short RecordKindType;


     typedef   string    PlugInType;
```

```
// SlotAssignmentList
struct SlotAssignmentType {
     short number;
     PlugInType plugIn; // empty string implies that the slot is not
                   // provisioned for any particular type of plug-in unit
};

typedef sequence <SlotAssignmentType> SlotAssignmentSeqType;
typedef string ReservationIdType;
typedef sequence<ReservationIdType> ReservationIdSeqType;
typedef string ServiceInstanceIdType;
typedef sequence<ServiceInstanceIdType> ServiceInstanceIdSeqType;

typedef UtcT GeneralizedTimeType;
typedef GeneralizedTimeType ExternalTimeType;

enum SystemTimingSourceType {
     internalTimingSource, // freerun
     remoteTimingSource, // external
     slaveTimingTerminationSignal
};

typedef ManagedEntityIdType TimingSourceIdType ;

struct SystemTimingComponentType {
     SystemTimingSourceType type;
     TimingSourceIdType  id;
};

struct SystemTimingType {
     SystemTimingComponentType Primary;
     SystemTimingComponentType Secondary;
};


typedef UserLabelType UserIdType;
typedef string PasswordType;

/*
All zeros indicate a loopback request directed at all connection
points having an LLID in the flow. All ones indicate a loopback
request directed at the endpoint (segment or connection endpoint).
'x6A'H indicates no designated CP for loopback, and therefore no
loopback should be performed. All other values for LLID indicate
a loopback request directed at a specific LLID location.
*/
typedef   sequence<octet,16> LoopbackLocationIdSeqType;

typedef string FilenameType; // specifies the complete path

enum StatusType {
     procedural_status,
     other
};

enum OtherStatusType {
     completed_success,
     completed_failure,
     activityInProgress,
     unknownstatus
};
```

```
     union StatusValueType switch (StatusType) {
          case procedural_status: ProceduralStatusSetType proceduralStatusSet;
          case other: OtherStatusType otherStatus;
     };

     struct StatusAttributeType {
          StatusValueType valueOfStatus;
          ManagedEntityIdType ne;
          short percentComplete;
     };

     typedef sequence<StatusAttributeType> StatusAttributeSeqType;

     typedef   unsigned long TrackingObjectIdType;

     typedef TrackingObjectIdType TransferTrackingObjectIdType;

     typedef any RecordType; // based on recordType (Values defined in
Q834Common::RecordSetType) Need definition of recordType and individual type of
records
     typedef sequence<RecordType> RecordSeqType;


     typedef unsigned long long NotificationIdentifierType; // will be replaced
                        // by NotifIDType defined in X.780 when that definition
is changed from "long" to "long long"

     typedef sequence<NotificationIdentifierType> NotificationIdentifierSeqType;


     typedef stringMonitoredParameterType; // Values defined in
Q834Common::MonitoringParameter
     typedef unsigned short MonitoringKindType; // values defined in
Q834Common::PMCategory interface

     struct EndPointType {
          ManagedEntityIdType portId;
          any endPointParameters;
          NameSeqType serviceCharacteristicsProfiles;
     };

     /*
     endPointParameters: service specific parameters, structures to be
     provided as part of implementation. An ATM connection for example
     would have the VPI, VCI parameters.
     */


     enum AlarmStatusComponentType {
          AS_UnderRepair,
          AS_Critical,
          AS_Major,
          AS_Minor,
          AS_AlarmOutstanding
     };

     typedef sequence<AlarmStatusComponentType> AlarmStatusSeqType;

     struct EquipmentHolderAddressType {
          short shelfNumber;
          short slotNumber;
     };

     /*
     If the equipment holder is a shelf, the slot number is 0. If the
     equipment holder is a slot, then both the slot number and shelf numbers
     are greater than or equal to 1.
     */

     /*
     The supplier provides a customization of the DiscoveryEventSupplier
     interface by defining the const string plugInUnit names. It will be up to
     the operator to enforce consistency and non-duplication across multiple
     supplier's solutions.
     */

     typedef sequence<PlugInUnitType> PlugInUnitSeqType;



struct NEFSANType {
          ManagedEntityIdType managedEntityId;
          AdministrativeStateType administrativeState;
          OperationalStateType operationalState;
          GeneralizedTimeType externalTime;
          string locationName;
          string supplierName;
          VersionType hardwareVersion;
          VersionSeqType softwareVersions;
          string serialNumber;
          NameSeqType alarmSeverityAssignmentProfileNames;
          AlarmStatusSeqType alarmStatusValues;
          NameSeqType thresholdDataNames;
          ManagedEntityIdSeqType supportedByManagedEntityList;
          UserLabelType userLabel;
     };
     /*
     SupportedByManagedEntityList should include the instance of Software
     controlling the NE.
     */

     /*
     Next, structs are defined that form the basis for equipment
     inventory information discovered at installation or with equipment
     rearrangement actions. Any one of these structs is subsequently
     identified as Mestruct in the DiscoveryEventSupplier interface
     specification.
     */

     struct OLTType {
          NEFSANType nEFSAN;
          ManagedEntityIdSeqType subtendingNEFSANList;
     };

     struct ONTType {
          NEFSANType nEFSAN;
          ManagedEntityIdType upstreamNEFSAN;
     };

     struct ONUType {
          NEFSANType nEFSAN;
          ManagedEntityIdType upstreamNEFSAN;
          ManagedEntityIdSeqType subtendingNEFSANList;
     };
```

```
struct NTType {
     NEFSANType nEFSAN;
     ManagedEntityIdType upstreamNEFSAN;
};

struct EquipmentHolderFType {
     ManagedEntityIdType equipmentHolderFId;
     ManagedEntityIdType containingNEId;
     EquipmentHolderAddressType equipmentHolderAddress;
     boolean slotStatus;
     PlugInUnitSeqType expectedPlugInUnits;
     string SoftwareLoad;
     NameType alarmSeverityAssignmentProfileName;
     AlarmStatusSeqType alarmStatusValues;
     OperationalStateType operationalState;
};

struct PlugInUnitFType {
     ManagedEntityIdType plugInUnitFId;
     ManagedEntityIdType containingNEId;
     EquipmentHolderAddressType containingSlotAddress;
     AdministrativeStateType administrativeState;
     AvailabilityStatusSetType availabilityStatus;
     OperationalStateType operationalState;
     string modelCode;
     string functionCode;
     string supplierName;
     VersionType hardwareVersion;
     string serialNumber;
     short portCount;
     NameSeqType alarmSeverityAssignmentProfileNames;
     NameSeqType thresholdDataNames;
     UserLabelType circuitPackUserLabel;
     ManagedEntityIdSeqType supportedByManagedEntityList;
};

enum ServiceCategoryType {
     CBR,
     UBR,
     RTVBR,
     NRTVBR,
     AdaptiveBR,
     GFR
};

enum ConformanceDefType {
     CBR1,
     UBR1,
     UBR2,
     VBR1,
     VBR2,
     VBR3,
     ABR,
     GFR1,
     GFR2
};

struct ATMOverbookingFactorType {
     ServiceCategoryType serviceCategory;
     ConformanceDefType conformanceDef;
     short overbookingFactor; // percentage: 100 = no overbooking
};

struct AlarmLogRecordType {
     long long recordId;
     ManagedEntityIdType mEId;
     GeneralizedTimeType loggingTime;
     short probableCause;
     PerceivedSeverityType severity;
     GeneralizedTimeType eventTime;
     ManagedEntityIdType backupEntityId;
     boolean backedupStatus;
     boolean serviceAffectingInd;
     ServiceInstanceIdSeqType affectedServices;
     NotificationIdentifierType notificationId;
     NotificationIdentifierSeqType correlatedNotifications;
     string monitoredParameter;
     long long thresholdValue;
     long long observedValue;
     string additionalText;
};

struct SecurityAlarmLogRecordType {
     long long recordId;
     ManagedEntityIdType mEId;
     GeneralizedTimeType loggingTime;
     short securityAlarmCause; // values defined in X.780
     GeneralizedTimeType eventTime;
     ManagedEntityIdType securityAlarmDetector;
     ManagedEntityIdType serviceUser;
     ManagedEntityIdType serviceProvider;
     NotificationIdentifierType notificationId;
     NotificationIdentifierSeqType correlatedNotifications;
     string additionalText;
};

struct ServiceOutageRecordType {
     long long recordId;
     ServiceInstanceIdType affectedService;
     GeneralizedTimeType loggingTime;
     GeneralizedTimeType outageStartTime;
     GeneralizedTimeType outageEndTime;
     NotificationIdentifierSeqType correlatedNotifications;
};

struct AAL1PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long HeaderErrors;
     unsigned long long LostCells;
     unsigned long long CellMisinsertion;
     unsigned long long BufferUnderflows;
     unsigned long long SequenceViolations;
     unsigned long long SDTPtrReframes;
     unsigned long long SDTPtrParityCheckFailures;
};

struct AAL2PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId; unsigned long long CPSInPkts;
     unsigned long long CPSOutPkts;
     unsigned long long BufferUnderflow;
```

```
          unsigned long long BufferOverflow;
          unsigned long long ParityErrors;
          unsigned long long SeqNumErrors;
          unsigned long long CPS_OSFMismatchErrors;
          unsigned long long CPS_OSFErrors;
          unsigned long long CPSHECErrors;
          unsigned long long OversizedSDUErrors;
          unsigned long long ReassemblyErrors;
          unsigned long long HECOverlapErrors;
          unsigned long long UUIErrors;
          unsigned long long CIDErrors;
     };

     struct AAL5PMHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
          GeneralizedTimeType periodEndTime;
          boolean suspectIntervalFlag;
          NameType ThresholdDataId;
          unsigned long long SumOfInvalidCSFieldErrors;
          unsigned long long CRCViolations;
          unsigned long long BufferOverflows;
          unsigned long long EncapProtocolErrors;
     };

     struct APONPMHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
          GeneralizedTimeType periodEndTime;
          boolean suspectIntervalFlag;
          NameType ThresholdDataId;
          unsigned long long ES;
          unsigned long long FEES;
     };

     struct ATMTrafficLoadHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
          GeneralizedTimeType periodEndTime;
          boolean suspectIntervalFlag;
          NameType ThresholdDataId;
          unsigned long long CellsReceived;
          unsigned long long CellsTransmitted;
     };

     struct DS1PMHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
          GeneralizedTimeType periodEndTime;
          boolean suspectIntervalFlag;
          NameType ThresholdDataId;
          unsigned long long ESP;
          unsigned long long BESP;
          unsigned long long SESP;
          unsigned long long UASP;
          unsigned long long ESPFE;
          unsigned long long BESPFE;
          unsigned long long SESPFE;
          unsigned long long UASPFE;
     };

     struct DS3PMHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESL;
     unsigned long long SESL;
     unsigned long long CVCPorCVPP;
     unsigned long long ESCPPorESPP;
     unsigned long long SESCPPorSESPP;
     unsigned long long UASCPPorUASPP;
};

struct E1PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESP;
     unsigned long long BESP;
     unsigned long long SESP;
     unsigned long long UASP;
     unsigned long long ESPFE;
     unsigned long long BESPFE;
     unsigned long long SESPFE;
     unsigned long long UASPFE;
};

struct EthernetHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long SingleCollisionFrame;
     unsigned long long MultipleCollisionFrames;
     unsigned long long SQE;
     unsigned long long DeferredTransmission;
     unsigned long long LateCollision;
     unsigned long long ExcessiveCollision;
     unsigned long long InternalMACTransmitError;
     unsigned long long CarrierSenseError;
     unsigned long long BufferOverflows;
     unsigned long long AlignmentError;
     unsigned long long FrameTooLongs;
     unsigned long long FCSErrors;
     unsigned long long InternalMACReceiveError;
};

struct MACBridgePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long NumberofSuppressedIntervals;
     unsigned long long BridgeLearningEntryDiscard;
};

struct MACBridgePMPortHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ForwardedFrame;
     unsigned long long DelayExceededDiscard;
     unsigned long long MTUExceededDiscard;
     unsigned long long ReceivedFrame;
     unsigned long long ReceivedAndDiscarded;
};

struct UpcNpcDisagreementPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long DiscardedCells;
     unsigned long long DiscardedCLP_0Cells;
     unsigned long long TaggedCLP_0Cells;
};

struct VoicePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long IncomingCallAttempts;
     unsigned long long OutgoingCallAttempts;
     unsigned long long VoicePortBufferOverflows;
     unsigned long long VoicePortBufferUnderflows;
};

struct VpVcPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long Lost0plus1UserInformationCells;
     unsigned long long Lost0UserInformationCells;
     unsigned long long MisinsertedUserInformationCells;
     unsigned long long Transmitted0plus1UserInformationCells;
     unsigned long long Transmitted0UserInformation;
     unsigned long long ImpairedBlock;
};

struct SONETSDHLinePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
     unsigned long long SeverelyErroredSecondsP;
     unsigned long long BackgroundBlockErrorP;
     unsigned long long OutOfFrameSecondsP;
     unsigned long long UnavailableSecondsP;
};

struct SONETSDHSectionPathPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
```

```
          unsigned long long SeverelyErroredSecondsP;
          unsigned long long BackgroundBlockErrorP;
          unsigned long long OutOfFrameSecondsP;
          unsigned long long UnavailableSecondsP;
          unsigned long long FailureCountP;
          unsigned long long ErroredSecondsTypeAP;
          unsigned long long ErroredSecondsTypeBP;
          unsigned long long ErroredSecondsPFE;
          unsigned long long SeverelyErroredSecondsPFE;
          unsigned long long BackgroundBlockErrorPFE;
          unsigned long long OutOfFrameSecondsPFE;
          unsigned long long UnavailableSecondsPFE;
          unsigned long long FailureCountPFE;
          unsigned long long ErroredSecondsTypeAPFE;
          unsigned long long ErroredSecondsTypeBPFE;
     };

     struct SONETSDHSectionAdaptationPMHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
          GeneralizedTimeType periodEndTime;
          boolean suspectIntervalFlag;
          NameType ThresholdDataId;
          unsigned long long PointerJustificationHighCountP;
          unsigned long long PointerJustificationLowCountP;
     };

     struct TCAdaptationProtocolMonitoringPMHistoryDataType {
          long long recordId;
          ManagedEntityIdType monitoringPoint;
          GeneralizedTimeType periodEndTime;
          boolean suspectIntervalFlag;
          NameType ThresholdDataId;
          unsigned long long DiscardedCellsHECViolationP;
          unsigned long long ErroredCellsHECViolationP;
     };

//Common exceptions

     exception AccessDenied {};
     exception CommFailure {};
     exception ConnectionCountExceeded {};
     exception DuplicateUserLabel {};
     exception EquipmentFailure {};
     exception InsufficientBW {};
     exception InsufficientPONBW {
          ManagedEntityIdType ponPORT;
     };
     exception InvalidSerialNumSyntax {};
     exception InvalidUserLabelSyntax {};
     exception MaxSubtendingNodesExceeded {};
     exception Timeout {};
     exception UnknownManagedEntity {
          ManagedEntityIdType managedEntityId;
     };
     exception UnknownNE {
          ManagedEntityIdSeqType unknownNEs;
     };
     exception UnknownPort {};
     exception UnknownProfiles {};
     exception UnknownReservationId {};
     exception UnknownScheduler {};
     exception InvalidScheduler {};
     exception DCNTimeout {};
     exception UnknownDestinationServer {};
     exception UnknownServiceInstance {};
     exception DeniedAccess {};
     exception BackupInProgress {};
     exception InvalidStartTime {};
     exception InvalidStopTime {};
     exception ParameterViolation {};
     exception UnknownRecordSet {};
     exception SynchInProgress {};
     exception ProfileSuspended {
          NameSeqType profilesSuspended;
     };


     module ProbableCauseConst {

          const string moduleName = "q834_4::Q834Common::ProbableCauseConst";

          interface ProbableCause {
          /*
          For X.780 probableCause refer to itut_x780::ProbableCauseConst.
          */

          // Additional ProbableCause values needed for BPON

               const unsigned short LOSS_OF_PATH_POINTER = 1;
               const unsigned short STS_PAYLOAD_LABEL_MISMATCH = 2;
               const unsigned short STS_PATH_UNEQUIPPED = 3;
               const unsigned short ALARM_INDICATION_SIGNAL = 4;
               const unsigned short REMOTE_FAILURE_INDICATION = 5;
               const unsigned short REMOTE_ALARM_INDICATION = 6;
               const unsigned short ALARM_INDICATION_SIGNAL_PATH = 7;
               const unsigned short
ALARM_INDICATION_SIGNAL_CUSTOMER_INSTALLATION = 8;
               const unsigned short LOSS_OF_SIGNAL_TO_ALL_ONU_ONU = 9;
               const unsigned short LOSS_OF_SIGNAL_ONU_OR_ONT = 10;
               const unsigned short LOSS_OF_ACKNOWLEDGEMENT_ONU_OR_ONT = 11;
               const unsigned short PLOAMCL_ONU_OR_ONT = 12; //Physical Layer OAM
Cell Loss
               const unsigned short LOCD_ONU_OR_ONT = 13; // Loss of Cell
Delineation
               const unsigned short CPHE_ONU_OR_ONT = 14; // Cell Phase Error
               const unsigned short PEE_ONU_OR_ONT = 15;
               const unsigned short RF_ONU_OR_ONT = 16; // Ranging Failure
               const unsigned short BED_ONU_OR_ONT = 17; // Block Error Detection
               const unsigned short SD_ONU_OR_ONT = 18; // Signal Degraded
               const unsigned short REI_ONU_OR_ONT = 19; // Remote Error
Indication
               const unsigned short UM_ONU_OR_ONT = 20; // Unknown Message
               const unsigned short LM_ONU_OR_ONT = 21; // Link Mismatch
               const unsigned short REMOTE_DEFECT_INDICATION = 22; // Signal
Degraded
               const unsigned short MAJOR_POWER_FAILURE = 23;
               const unsigned short
REMOTE_ALARM_INDICATION_FAR_END_CUSTOMER_INSTALLATION = 24;
               const unsigned short LOSS_OF_ATM_CELL_DELINEATION = 25;
               const unsigned short LOW_BATTERY_THRESHOLD = 26;
               const unsigned short DIAGNOSTIC_TEST_FAILURE = 27; // See below
comment
               const unsigned short LOSS_OF_DCN_LINK = 28;
               const unsigned short CELL_STARVATION = 29;
               const unsigned short UNEXPECTED_PLUGIN = 30;
               const unsigned short IMPROPER_CARD_REMOVAL = 31;
               const unsigned short SLOT_CARD_MISMATCH = 32;
               const unsigned short LOS_LAN = 33; // Loss of carrier on Bridge
LAN port
               const unsigned short PERSISTENT_IMPAIRMENT = 34;
          }; // interface ProbableCause
     }; // module ProbableCauseConst

     interface MonitoringParameter {
          /*
          Names for monitored performance and traffic parameters
          */

          const string allParameters = "AllParameters";
          const string aAL1HeaderErrors = "AAL1HeaderErrors";
          const string allTypesCellsDiscarded = "AllTypesCellsDiscarded";
          const string aTMProtocolErrors = "ATMProtocolErrors";
          const string bESFEP = "BESFEP";
          const string bESP = "BESP";
          const string bufferOverflows = "BufferOverflows";
          const string bufferUnderflows = "BufferUnderflows";
          const string cellDelineationAnomalies = "CellDelineationAnomalies";
          const string cellMisinsertion = "CellMisinsertion";
          const string cRCViolations = "CRCViolations";
          const string cVCP = "CVCP";
          const string cVL = "CVL";
          const string cVPP = "CVPP";
          const string cVS = "CVS";
          const string discardedAllTypeCellsduetoNPC =
"DiscardedAllTypeCellsduetoNPC";
          const string discardedAllTypeCellsduetoUPC =
"DiscardedAllTypeCellsduetoUPC";
          const string discardedPriorityCellsduetoNPC =
"DiscardedPriorityCellsduetoNPC";
          const string discardedPriorityCellsduetoUPC =
"DiscardedPriorityCellsduetoUPC";
          const string encapProtocolErrors = "EncapProtocolErrors";
          const string eSCPP = "ESCPP";
          const string eSPONT = "ESPONT";
          const string eSL = "ESL";
          const string eSP = "ESP";
          const string eSPP = "ESPP";
          const string eSS = "ESS";
          const string excessiveCollisions = "ExcessiveCollisions";
          const string fCSErrors = "FCSErrors";
          const string frameTooLongs = "FrameTooLongs";
          const string hECViolations = "HECViolations";
          const string impairedBlocks = "ImpairedBlocks";
          const string lateCollisions = "LateCollisions";
          const string lostCells = "LostCells";
          const string lostPriorityUserInformationCells =
"LostPriorityUserInformationCells";
          const string lostUserInformationCells = "LostUserInformationCells";
          const string misinsertedUserInformationCells =
"MisinsertedUserInformationCells";
          const string priorityCellsDiscarded = "PriorityCellsDiscarded";
          const string sDTPointerReframes = "SDTPointerReframes";
          const string sDTPointerParityCheckFailures =
"SDTPointerParityCheckFailures";
          const string sequenceViolations = "SequenceViolations";
          const string sESCPP = "SESCPP";
          const string sESPONT = "SESPONT";
          const string sESL = "SESL";
          const string sESP = "SESP";
          const string sESPP = "SESPP";
          const string sESS = "SESS";
          const string sumOfInvalidCSFieldErrors = "SumOfInvalidCSFieldErrors";
          const string uASPONT = "UASPONT";
          const string uASCPP = "UASCPP";
          const string uASP = "UASP";
          const string uASPP = "UASPP";
          const string incomingCallAttempts = "IncomingCallAttempts";
          const string outgoingCallAttempts = "OutgoingCallAttempts";
          const string voicePortBufferOverflows = "VoicePortBufferOverflows";
          const string voicePortBufferUnderflows = "VoicePortBufferUnderflows";
          const string cPSInPkts = "CPSInPkts";
          const string cPSOutPkts = "CPSOutPkts";
          const string bufferUnderflow = "BufferUnderflow";
          const string bufferOverflow = "BufferOverflow";
          const string parityErrors = "ParityErrors";
          const string seqNumErrors = "SeqNumErrors";
          const string cPS_OSFMismatchErrors = "CPS_OSFMismatchErrors";
          const string cPS_OSFErrors = "CPS_OSFErrors";
          const string cPSHECErrors = "CPSHECErrors";
          const string oversizedSDUErrors = "OversizedSDUErrors";
          const string reassemblyErrors = "ReassemblyErrors";
          const string hECOverlapErrors = "HECOverlapErrors";
          const string uUIErrors = "UUIErrors";
          const string cIDErrors = "CIDErrors";

     }; // interface MonitoringParameter

     interface PMCategory {
          const unsigned short DS1_PM = 1;
          const unsigned short DS3_PM = 2;
          const unsigned short E1_PM = 3;
          const unsigned short E3_PM = 4;
          const unsigned short VP_PM = 5;
          const unsigned short VC_PM = 6;
          const unsigned short ETHERNET_PM = 7;
          const unsigned short TC_ADAPTOR_PM = 8;
          const unsigned short VOICE_PM = 9;
          const unsigned short APON_PM = 10;
          const unsigned short MACBRIDGE_PM = 11;
          const unsigned short MACBRIDGEPORT_PM = 12;
          const unsigned short ATMTRAFFICLOAD_PM = 13;
          const unsigned short AAL1_PM = 14;
          const unsigned short AAL2_PM = 15;
          const unsigned short AAL5_PM = 16;
          const unsigned short UPCNPC_PM = 17;
          const unsigned short DBA_PM = 18;
          const unsigned short SONET_SDH_LINE_PM = 19;
          const unsigned short SONET_SDH_SECTION_PATH_PM = 20;
          const unsigned short SONET_SDH_SECTION_ADAPTATION_PM = 21;
          const unsigned short CALL_STATS_PM = 22;
          const unsigned short VIDEO_PM = 23;

     }; // interface PMCategory

     interface RecordSetType {

     // Beginning of Values for RecordKindType
     // Values 1-99 reserved for HistoryDataType

          const unsigned short DS1PMHISTORYDATA = 1;
          const unsigned short DS3PMHISTORYDATA = 2;
          const unsigned short E1PMHISTORYDATA = 3;
          const unsigned short E3PMHISTORYDATA = 4;
          const unsigned short VPVCPMHISTORYDATA = 5;
          const unsigned short AAL1PMHISTORYDATA = 6;
```

```
          const unsigned short AAL2PMHISTORYDATA = 7;
          const unsigned short AAL5PMHISTORYDATA = 8;
          const unsigned short UPCNPCDISAGREEMENTPMHISTORYDATA = 9;
          const unsigned short ETHERNETPMHISTORYDATA = 10;
          const unsigned short VOICEPMHISTORYDATA = 11;
          const unsigned short MACBRIDGEPORTPMHISTORYDATA = 12;
          const unsigned short MACBRIDGEPMHISTORYDATA = 13;
          const unsigned short APONPMHISTORYDATA = 14;
          const unsigned short SONETSDHLINEPMHISTORYDATA = 15;
          const unsigned short SONETSDHSECTIONADAPTATIONPMHISTORYDATA = 16;
          const unsigned short SONETSDHSECTIONPATHPMHISTORYDATA = 17;
          const unsigned short TCADAPTATIONPROTOCOLMONITORINGPMHISTORYDATA = 18;
          const unsigned short ALARMLOGRECORD = 100;
          const unsigned short SECURITYALARMLOGRECORD = 101;
          const unsigned short SERVICEOUTAGERECORD = 102;

     // End of Values for RecordKindType

     }; // interface RecordSetType

interface PhysicalLayerLoopback {

     // Beginning of Values for LoopbackTestType

          const unsigned short LINELOOPBACK = 1;
          const unsigned short PAYLOADLOOPBACK = 2;
          const unsigned short INWARDLOOPBACK = 3;
          const unsigned short DUALLOOPBACK = 4;
          const unsigned short FACILITYLOOPBACK = 5;
          const unsigned short TERMINALLOOPBACK = 6;

     // End of Values for LoopbackTestType

     }; // interface  PhysicalLayerLoopback

};   // module Q834Common

}; // module q834_4
#endif
```

C.4 Q834ControlArchive.idl

```
#ifndef __Q834_4_CONTROLARCHIVE_DEFINED
#define __Q834_4_CONTROLARCHIVE_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ControlArchive {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::FilterType FilterType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::RecordType RecordType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;
     typedef Q834Common::RecordKindType RecordKindType;


#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define Timeout Q834Common::Timeout
#define UnknownRecordSet Q834Common::UnknownRecordSet

// End definitions from other idl files

// Local data types

     enum FullActionType {
          halt, // indicates that the log should stop writing any more records
if the log is full
          wrap // indicates that the log should delete old records if the log is
full.
     };

     typedef unsigned long long MaxSizeType;
     typedef unsigned short SizeThresholdType; // 0-100%
     typedef unsigned long long CurrentSizeType;

     struct RecordSetStatusType {
          CurrentSizeType currentSize;
          OperationalStateType operationalState;
          MaxSizeType maxSize;
          SizeThresholdType sizeThreshold;
          NameType filterName;
          FullActionType fullAction;
          AdministrativeStateType administrativeState;
          RecordKindType recordKind;
          UserLabelType recordSetUserLabel;
     };


     enum CreationModeType {
          operatorDefined,
          initialList,
          either
     };

// Local exceptions

     exception RecordSetExists {ManagedEntityIdType recordSetId;};
     exception LockedAlready {};
     exception UnknownOption {};
     exception NoSuchRecords {};
     exception TooManyRecords {};

// End local definitions



     interface RecordSetMgr : itut_x780::ManagedObject {


     // See 9.4.1.1 for the description of the behaviour of this operation

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

// See 9.4.1.2 for the description of the behaviour of this operation

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

// See 9.4.1.3 for the description of the behaviour of this operation


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

// See 9.4.1.4 for the description of the behaviour of this operation


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

// See 9.4.1.5 for the description of the behaviour of this operation

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

// See 9.4.1.6 for the description of the behaviour of this operation

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

// See 9.4.1.7 for the description of the behaviour of this operation

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


// See 9.4.1.8 for the description of the behaviour of this operation

     RecordSeqType selectRecords (
          in FilterType SelectionFilter,
          in ManagedEntityIdType recordSetId)
          raises (UnknownRecordSet,
               Timeout,
               NoSuchRecords,
               AccessDenied,
               TooManyRecords);
     // See 9.4.1.9 for the description of the behaviour of this operation

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


     // See 9.4.1.10 for the description of the behaviour of this operation

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

     }; // interface RecordSetMgr

}; // module ControlArchive

}; // module q834_4

#endif
```

C.5 Q834SoftwareDownload.idl

```
#ifndef __Q834_4_SOFTWAREDOWNLOAD_DEFINED
#define __Q834_4_SOFTWAREDOWNLOAD_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SoftwareDownload {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::PlugInUnitType PlugInUnitType;
     typedef Q834Common::FilenameType  FilenameType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownNE Q834Common::UnknownNE
#define DeniedAccess Q834Common::DeniedAccess
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define Timeout Q834Common::Timeout
```

```
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStartTime Q834Common::InvalidStartTime

// End definitions from other idl files

// Local data types

     typedef TrackingObjectIdType SoftwareDownloadTrackingObjectIdType;

     typedef sequence<SoftwareDownloadTrackingObjectIdType>
SoftwareDownloadTrackingObjectIdSeqType;
     typedef sequence<FilenameType> FilenameSeqType;

     struct VersionsType {
          ManagedEntityIdType resourceId;
          VersionType softwarePrimary;
          VersionType softwareStandBy;
          VersionType hardware;
     };

     typedef sequence<VersionsType> VersionsSeqType;

     struct TargetType {
          ManagedEntityIdType containingSystem;
          ManagedEntityIdType containingNE; // empty sequence means everything
          string plugInUnitType; // empty string here implies to any suitable
plugInUnitType
          ManagedEntityIdType slot; // empty sequence here means any slot.
     }; //string is supplied by the supplier with Release Notes.

     struct DownloadStatusType
     {
             ManagedEntityIdType targetId;
             StatusValueType deliveryStatus;
             StatusValueType commitStatus;
             StatusValueType activationStatus;
             StatusValueType revertStatus;
     };

        typedef sequence<DownloadStatusType> DownloadStatusSeqType;

// Local exceptions


     exception InvalidExternalTime {};
     exception UnrecognisedTarget {};
     exception InsufficientMemory {};
     exception SoftwareLoadHWMismatch {};
     exception SourceUnreachable {};
     exception UnknownSoftwareLoad {};
     exception InstallationFailure {};
     exception UnknownSoftwareDownloadTrackingObject {};
     exception SoftwareNotYetInstalled {};
     exception ActivationFailure {};
     exception ActivationCompleted {};
     exception ActivityCompleted {};
     exception ActivityInProgress {};
     exception InvalidSoftwareTrackingObject {};
     exception SoftwareTrackingObjectInUse {};

// End local definitions

     valuetype DownloadMgrValueType: itut_x780::ManagedObjectValueType {
          public SoftwareDownloadTrackingObjectIdSeqType
ScheduledSoftwareDownloadTrackingObjectList; // GET
          public SoftwareDownloadTrackingObjectIdSeqType
OnDemandSoftwareDownloadTrackingObjectList; // GET

     };

     interface DownloadMgr : itut_x780::ManagedObject {

     // See 9.5.1.1 for the description of the behaviour of this operation

          SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
               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);


     // See 9.5.1.2 for the description of the behaviour of this operation

          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);

     // See 9.5.1.3 for the description of the behaviour of this operation

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

     // See 9.5.1.4 for the description of the behaviour of this operation

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

     // See 9.5.1.5 for the description of the behaviour of this operation
          void activate (
               in SoftwareDownloadTrackingObjectIdType id,
               in TargetType activateTarget)
               raises (UnknownSoftwareDownloadTrackingObject,
                    SoftwareNotYetInstalled,
                    ActivationFailure,
                    AccessDenied,
                    UnrecognisedTarget);

     // See 9.5.1.6 for the description of the behaviour of this operation

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

     // See 9.5.1.7 for the description of the behaviour of this operation

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

     // See 9.5.1.8 for the description of the behaviour of this operation

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

     // See 9.5.1.9 for the description of the behaviour of this operation

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

     // See 9.5.1.10 for the description of the behaviour of this operation

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

     // See 9.5.1.11 for the description of the behaviour of this operation

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

     // See 9.5.1.12 for the description of the behaviour of this operation

          SoftwareDownloadTrackingObjectIdSeqType
     scheduledSoftwareDownloadTrackingObjectList ()
               raises (AccessDenied);

     // See 9.5.1.13 for the description of the behaviour of this operation

          SoftwareDownloadTrackingObjectIdSeqType
     onDemandSoftwareDownloadTrackingObjectList ()
               raises (AccessDenied);


     }; // interface DownloadMgr



     interface VersionRepository : itut_x780::ManagedObject {

     // See 9.5.2.1 for the description of the behaviour of this operation

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

     // See 9.5.2.2 for the description of the behaviour of this operation

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

     // See 9.5.2.3 for the description of the behaviour of this operation

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

}; // interface VersionRepository

}; // module SoftwareDownload

}; // module q834_4

#endif
```

C.6 Q834EventPublisher.idl

```
#ifndef __Q834_4_EVENTPUBLISHING_DEFINED
#define __Q834_4_EVENTPUBLISHING_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module q834_4
{

module EventPublisher
{

// begin definitions from other idl files - filterable data value types


// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::ProbableCauseType ProbableCauseType; // Values defined
in Q834Common::ProbableCauseConst
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::NotificationIdentifierSeqType
NotificationIdentifierSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::EquipmentHolderFType EquipmentHolderFType;
     typedef Q834Common::PlugInUnitFType PlugInUnitFType;
     typedef Q834Common::UsageStateType UsageStateType;
     typedef Q834Common::ControlStatusSetType ControlStatusSetType;
     typedef Q834Common::SpecificProblemSetType SpecificProblemSetType;
     typedef Q834Common::BackedUpStatusType BackedUpStatusType;
     typedef Q834Common::AttributeChangeSetType AttributeChangeSetType;
     typedef Q834Common::SecurityAlarmCauseType SecurityAlarmCauseType;

// End definitions from other idl files

// Local data types




     typedef NotificationIdentifierSeqType CorrelatedNotificationType;

     typedef AttributeChangeSetType StateChangeDefinitionType;


     struct ThresholdInfoType {
          MonitoredParameterType monitoredParameter;
          unsigned long long observedValue;
          unsigned long long thresholdValueLow;
          unsigned long long thresholdValueHigh;
     };
     /*
     Values for monitoredParameter are provided in
     Q834Common::MonitoringParameter.
     Currently only counters are supported for performance or traffic monitoring
     in BPON technology, consequently observedValue, and thresholdValueHigh are
     restricted to non-negative integer values, and thresholdValueLow is 0.
     */

     typedef   unsigned short EquipmentType;




     enum ServiceAffectingType {
          serviceAffecting,
          nonServiceAffecting,
          unableToDetermine
     };


//End local data type definitions

// Local exceptions

// End local exceptions


     interface AlarmEventSupplier : itut_x780::ManagedObject {

          /* Structured event fixed header mappings:
          domain_type is set to "telecommunications",
          type_name is set to "Alarm", and
          event_name is set to one of the following constant strings provided
          below.
          */

          const string communicationAlarm = "CommunicationAlarm";
          const string equipmentAlarm = "EquipmentAlarm";
          const string environmentalAlarm = "EnvironmentalAlarm";
          const string processingErrorAlarm = "ProcessingErrorAlarm";
          const string qualityOfServiceAlarm = "QualityOfServiceAlarm";

          /*
          Filterable data names for populating the filterable event body in the
          structured event are listed in order below.
          */

          const string alarmEmittingMEId = "AlarmEmittingMEId";
          const string eventTime = "EventTime";
          const string probableCause = "ProbableCause";
          const string specificProblems = "SpecificProblems";
          const string perceivedSeverity = "PerceivedSeverity";
          const string backUpStatus = "BackUpStatus";
          const string backUpManagedEntityId = "BackUpManagedEntityId";
          const string thresholdInfo = "ThresholdInfo";
          const string notificationIdentifier = "NotificationIdentifier";
          const string correlatedNotifications = "CorrelatedNotifications";
          const string stateChangeDefinition = "StateChangeDefinition";
          const string monitoredAttributes = "MonitoredAttributes";
          const string additionalText = "AdditionalText";
          const string serviceAffectingInd = "ServiceAffectingInd";

          /*
          Remaining filterable data values.
```

```
    /* Names for State and Status Variables for StateChangeDefinition
    */

    const string administrativeState = "AdministrativeState";
    const string operationalState = "OperationalState";
    const string usageState = "UsageState";
    const string availabilityStatus = "AvailabilityStatus";
    const string proceduralStatus = "ProceduralStatus";
    const string controlStatus = "ControlStatus";
    const string alarmStatus = "AlarmStatus";


    /*
    Mapping to filterable data within the structured event is provided
    below for communication alarm, equipment alarm, processing error
    alarm, environmental alarm, and quality of service alarm.

    {
    {"AlarmEmittingMEId", any (ManagedEntityIdType)},
    {"EventTime", any (GeneralizedTimeType)},
    {"ProbableCause", any (ProbableCauseType)},
    {"SpecificProblems", any (SpecificProblemSetType)},
    {"PerceivedSeverity", any (PerceivedSeverityType)},
    {"ServiceAffectingInd", any (ServiceAffectingType)},
    {"BackUpStatus", any (BackedUpStatusType)},
    {"BackUpManagedEntityId", any (ManagedEntityIdType)},
    {"ThresholdInfo", any (ThresholdInfoType)},
    {"NotificationIdentifier", any (NotificationIdentifierType)},
    {"CorrelatedNotifications", any (CorrelatedNotificationType)},
    {"StateChangeDefinition", any (StateChangeDefinitionType)},
    {"AdditionalText", any (string)}
    }

    */

}; // interface AlarmEventSupplier


interface SecurityEventSupplier : itut_x780::ManagedObject {

    /*  Structured event fixed header mappings:
    domain_type is set to "telecommunications",
    type_name is set to "SecurityEvent", and
    event_name is set to one of the following constant strings
    provided below.
    */

    const string integrityViolation = "IntegrityViolation";
    const string operationalViolation = "OperationalViolation";
    const string physicalViolation = "PhysicalViolation";
    const string securityEventViolation = "SecurityEventViolation";
    const string timeDomainViolation = "TimeDomainViolation";



    const string eventEmittingMEId = "EventEmittingMEId";
    const string eventTime = "EventTime";
    const string securityAlarmCause = "SecurityAlarmCause";
    const string securityAlarmDetector = "SecurityAlarmDetector";
    const string serviceUser = "ServiceUser";
    const string serviceProvider = "ServiceProvider";
    const string securityEventNotificationIdentifier =
          "NotificationIdentifier";
     const string correlatedNotifications = "CorrelatedNotifications";
     const string additionalText = "AdditionalText";
     /*
     Mapping to filterable data within the structured event is provided
     below for Integrity Violations, Operational Violations, Physical
     Violations,Security Event Violations, Time Domain Violations.


     {
     {"EventEmittingMEId", any (ManagedEntityIdType)},
     {"EventTime", any (GeneralizedTimeType)},
     {"SecurityAlarmCause", any (SecurityAlarmCauseType)},
     {"SecurityAlarmDetector", any (ManagedEntityIdType)},
     {"ServiceUser", any (ManagedEntityIdType)},
     {"ServiceProvider", any (ManagedEntityIdType)},
     {"NotificationIdentifier", any (NotificationIdentifierType)},
     {"CorrelatedNotifications", any (CorrelatedNotificationType)},
     {"AdditionalText", any (string)}
     }

     */

}; // interface SecurityEventSupplier


interface DiscoveryEventSupplier : itut_x780::ManagedObject {

     /*  Structured event fixed header mappings:
     domain_type is set to "telecommunications",
     type_name is set to "DiscoveryEvent", and
     event_name is set to one of the following constant strings
     provided below. Only installed equipment changes are declared
     through this interface.
     */

     const string managedEntityCreation = "ManagedEntityCreation";
     const string managedEntityDeletion = "ManagedEntityDeletion";


     const string managedEntityType = "ManagedEntityType";
     const string managedEntityAttributeValues =
     "ManagedEntityAttributeValues";
     const string discoveryNotificationIdentifier =
     "NotificationIdentifier";
     const string correlatedNotifications = "CorrelatedNotifications";
     const string additionalText = "AdditionalText";"


     // The following items are equipment types that are discovered by the
     // Supplier Management System and automatically revealed to the OMS.
     // The data structure passed on the notification is defined in
     // Q834Common.idl and the mapping below refers to it as MEstruct. More
     // specifically, here is the list of data structures currently
     // supported: OLTType, ONUType,ONTType, NTType, EquipmentHolderType,
     and PlugInUnitFType.

     const unsigned short OLT_NE = 1;
     const unsigned short ONT_NE = 2;
     const unsigned short ONU_NE = 3;
     const unsigned short NT_NE = 4;
```

```
        const unsigned short EQUIPMENT_HOLDER_F = 5;
        const unsigned short PLUG_IN_UNIT_F = 6;


        /*
        Mapping to filterable data within the structured event is provided
        below for a discovery event that involves the creation of a managed
        entity.


        {
        {"ManagedEntityType", any (EquipmentType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ManagedEntityAttributeValues", any (MEstruct)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"AdditionalText", any (string)}
        }

        */

    }; // interface DiscoveryEventSupplier

}; // module EventPublisher

}; // module q834_4

#endif
```

C.7 Q834MIBTransfer.idl

```
#ifndef __Q834_4_MIBTRANSFER_DEFINED
#define __Q834_4_MIBTRANSFER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module MIBTransfer {
// Begin definitions from other idl files


// From Q834Common

    typedef Q834Common::UserLabelType UserLabelType;
    typedef Q834Common::DCNAddressType DCNAddressType;
    typedef Q834Common::FilenameType FilenameType;
    typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
    typedef Q834Common::TransferTrackingObjectIdType
    TransferTrackingObjectIdType;
    typedef Q834Common::UserIdType UserIdType;
    typedef Q834Common::PasswordType PasswordType;
    typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define FileExists Q834Common::FileExists
```

```
#define CannotCreateFile Q834Common::CannotCreateFile
#define DeniedAccess Q834Common::DeniedAccess

// End definitions from other idl files

// Local data types

     typedef sequence<TransferTrackingObjectIdType>
TransferTrackingObjectIdSeqType;

// Local exceptions

     exception UnknownBackupProcess {};
     exception UnknownSourceServer {};
     exception UnknownRestoreProcess {};
     exception SoftwareLoadHardwareMismatch {};


// End local definitions

     valuetype MIBMoverValueType: itut_x780::ManagedObjectValueType {

          public TransferTrackingObjectIdSeqType transferTrackingObjectIdList;
// GET

     };

     interface MIBMover : itut_x780::ManagedObject {

     // See 9.7.1.1 for the description of the behaviour of this operation

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

     // See 9.7.1.2 for the description of the behaviour of this operation

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


     // See 9.7.1.3 for the description of the behaviour of this operation

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

       // See 9.7.1.4 for the description of the behaviour of this operation

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

       // See 9.7.1.5 for the description of the behaviour of this operation

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

       // See 9.7.1.6 for the description of the behaviour of this operation

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

       // See 9.7.1.7 for the description of the behaviour of this operation

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

       // See 9.7.1.8 for the description of the behaviour of this operation

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

```
     // See 9.7.1.9 for the description of the behaviour of this operation

          TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
               raises (AccessDenied);

     }; // interface MIBMover

}; // module MIBTransfer

}; // module q834_4

#endif
```

C.8 Q834PerformanceManager.idl

```
#ifndef __Q834_4_PERFORMANCEMANAGER_DEFINED
#define __Q834_4_PERFORMANCEMANAGER_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module PerformanceManager {

     // begin definitions from other idl files

     // From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;
     typedef Q834Common::RecordKindType RecordKindType;


     #define AccessDenied Q834Common::AccessDenied
     #define UnknownNE Q834Common::UnknownNE
     #define UnknownManagedEntity Q834Common::UnknownManagedEntity
     #define UnknownProfiles Q834Common::UnknownProfiles
     #define UnknownServiceInstance Q834Common::UnknownServiceInstance
     #define UnknownScheduler Q834Common::UnknownScheduler
     #define CommFailure Q834Common::CommFailure
     #define InvalidScheduler Q834Common::InvalidScheduler
     #define EquipmentFailure Q834Common::EquipmentFailure
     #define ProfileSuspended Q834Common::ProfileSuspended

     // End definitions from other idl files

     // Local data types


     struct  MonitoringPointKindAndThresholdsType {
          MonitoringKindType monitoringType;
          NameType thresholdDataProfileName;
     };
     /*
     NOTE – ITU-T Recs Q.834.1 and Q.834.2 describe the relationships between
     ThresholdData and specific monitoring point types.
     */

     typedef sequence<MonitoringPointKindAndThresholdsType> ThresholdsSeqType;

     struct MonitoringPointAndThresholdsType {
          ManagedEntityIdType monitoringPoint;
          NameType  thresholdDataProfileName;
     };

     typedef sequence<MonitoringPointAndThresholdsType>
MonitoringPointThresholdsSeqType;

     typedef sequence<MonitoredParameterType> MonitoredParameterSeqType;

     typedef sequence<RecordSeqType> RecordsSeqType; // Implicitly grouped by
record type.

     typedef RecordKindType  HistoryDataType; // Values defined in
Q834Common::RecordSetType

     typedef ManagedEntityIdSeqType MonitoringPointSeqType;

     struct SWPValueType {
          ManagedEntityIdType monitoringPoint;
          short totConsecutiveIntvls;
          short persistenceMinimum;
     };

     typedef sequence< SWPValueType > SWPValueSeqType;


     struct ParameterSettingType {
          MonitoredParameterType monitoredParameter;
          short totConsecutiveIntvls;
          short persistenceMinimum;
     };

     typedef sequence<ParameterSettingType> ParameterSettingSeqType;

     // Local exceptions

     exception UnknownParameters {
          MonitoredParameterSeqType monitoredParameterList;
     };
     exception IntervalCountTooLarge {};
     exception UnknownMonitoringPointTypes {
          MonitoringPointSeqType monitoringPointKindList;
     };
     exception InvalidAssociation {};
     exception CollectionPeriodPast {};
     exception CollectionLimitation {};
     exception UnknownHistoryDataType {};

     // End local definitions


     interface ImpairmentPersistence : itut_x780::ManagedObject {

          // See 9.8.1.1 for the description of the behaviour of this operation

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

     // See 9.8.1.2 for the description of the behaviour of this operation

     void setSpecificSlidingWindowParameters (
          in ManagedEntityIdType nEManagedEntityId,
          in ManagedEntityIdType monitoringPoint,
          in MonitoredParameterSeqType monitoredParameterList,
          in short totConsecutiveIntvls,
          in short persistenceMinimum,
          in boolean allowGlobalOverwrite)
          raises (UnknownNE,
               UnknownParameters,
               IntervalCountTooLarge,
               AccessDenied,
               UnknownManagedEntity,
               CommFailure,
               EquipmentFailure);

     // See 9.8.1.3 for the description of the behaviour of this operation

     SWPValueSeqType getSpecificSlidingWindowParameters (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoredParameterType monitoredParameter)
          raises (UnknownNE,
               UnknownParameters,
               CommFailure);

     // See 9.8.1.4 for the description of the behaviour of this operation

     void setThreshold (
          in ManagedEntityIdType nEManagedEntityId,
          in ManagedEntityIdType monitoringPoint,
          in NameType thresholdDataProfileName)
          raises (UnknownNE,
               AccessDenied,
               UnknownManagedEntity,
               UnknownProfiles,
               InvalidAssociation,
               CommFailure,
               ProfileSuspended);

     // See 9.8.1.5 for the description of the behaviour of this operation

     void setThresholds (
          in ManagedEntityIdType nEManagedEntityId,
          in boolean sysScopeInd,
          in ThresholdsSeqType thresholdsList)
          raises (UnknownNE,
               UnknownProfiles,
               AccessDenied,
               UnknownMonitoringPointTypes,
               InvalidAssociation,
               CommFailure,
               ProfileSuspended);
     // See 9.8.1.6 for the description of the behaviour of this operation

     MonitoringPointThresholdsSeqType getThresholdValues (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoringKindType monitoringType)
          raises (UnknownNE,
               UnknownMonitoringPointTypes,
               CommFailure);

     // See 9.8.1.7 for the description of the behaviour of this operation

     ThresholdsSeqType getSystemThresholdsSetting(
          in ManagedEntityIdType nEManagedEntityId)
          raises (AccessDenied, UnknownManagedEntity);

     // See 9.8.1.8 for the description of the behaviour of this operation

     ParameterSettingSeqType getSystemSWSettings(
          in ManagedEntityIdType nEManagedEntityId)
          raises (AccessDenied, UnknownManagedEntity);

}; // interface ImpairmentPersistence


interface ReportController : itut_x780::ManagedObject {

     // See 9.8.2.1 for the description of the behaviour of this operation

     void addCustomerMonitoringReporting  (
          in ManagedEntityIdType nEManagedEntityId,
          in ServiceInstanceIdType serviceInstanceId,
          in ManagedEntityIdType monitoringPoint,
          in GeneralizedTimeType stopTime,
          in HistoryDataType historyData,
          in short granularityPeriod) //in minutes
          raises (UnknownServiceInstance,
               AccessDenied,
               UnknownNE,
               UnknownManagedEntity,
               CollectionPeriodPast,
               CollectionLimitation,
               InvalidAssociation,
               UnknownHistoryDataType,
               CommFailure);

     // See 9.8.2.2 for the description of the behaviour of this operation

     void removeCustomerMonitoringReporting  (
          in ServiceInstanceIdType serviceInstanceId)
          raises (UnknownServiceInstance,
               AccessDenied,
               CollectionPeriodPast,
               CommFailure);

     // See 9.8.2.3 for the description of the behaviour of this operation

     RecordsSeqType selectByServiceInstance (
          in ServiceInstanceIdType serviceInstanceId,
          in GeneralizedTimeType intervalStartTime,
          in GeneralizedTimeType intervalEndTime)
          raises (UnknownServiceInstance,
               AccessDenied );
```

```
// See 9.8.2.4 for the description of the behaviour of this operation

MonitoringPointSeqType displayActiveReporting (
     in ServiceInstanceIdType serviceInstanceId)
     raises (UnknownServiceInstance,
          AccessDenied);

// See 9.8.2.5 for the description of the behaviour of this operation

void addNewMonitoringReporting (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType stopTime,
     in HistoryDataType historyData,
     in short granularityPeriod) //in minutes
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionPeriodPast,
          CollectionLimitation,
          InvalidAssociation,
          UnknownHistoryDataType,
          CommFailure);

// See 9.8.2.6 for the description of the behaviour of this operation

RecordsSeqType selectByMonitoringPoint (
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType intervalStartTime,
     in GeneralizedTimeType intervalEndTime)
     raises (UnknownManagedEntity,
          AccessDenied);


// See 9.8.2.7 for the description of the behaviour of this operation

void createReportingSchedule (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in HistoryDataType historyData,
     in ServiceInstanceIdType serviceInstance,
     in short granularityPeriod, //in minutes
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionLimitation,
          UnknownScheduler,
          InvalidAssociation,
          UnknownHistoryDataType,
          InvalidScheduler);

// See 9.8.2.8 for the description of the behaviour of this operation

void modifyReportingSchedule (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in HistoryDataType historyData,
     in UserLabelType newSchedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionLimitation,
          UnknownScheduler,
                    InvalidAssociation,
                    UnknownHistoryDataType,
                    InvalidScheduler);

          // See 9.8.2.9 for the description of the behaviour of this operation

          void cancelReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType schedulerName)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownManagedEntity,
                    UnknownScheduler,
                    InvalidAssociation,
                    UnknownHistoryDataType);

     }; // interface ReportController

}; // module PerformanceManager

}; // module q834_4

#endif
```

C.9 Q834ProfileManager.idl

```
#ifndef __Q834_4_PROFILEMANAGER_DEFINED
#define __Q834_4_PROFILEMANAGER_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module  q834_4
{

module ProfileManager
{

     // begin definitions from other idl files - filterable data value types
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::ProbableCauseType ProbableCauseType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;

#define UnknownProfiles Q834Common::UnknownProfiles
#define AccessDenied Q834Common::AccessDenied

     // End definitions from other idl files

     // Local data types

     typedef unsigned short ProfileKindType;
```

```
struct ProfileInfoType {
     ProfileKindType profileKind;
     any  attributeValuestruct;
};

/*
Next, structs (and sequence of structs) are defined having the attribute
values for the newly created profile objects. Any one of these structs is
subsequently identified as ProfileStruct in the Profile Event Consumer
interface specification.
*/

enum SubType  {
     NULL,
     VoicebandBasedOn64kbps,
     SynchronousCircuitEmulation,
     AsynchronousCircuitEmulation,
     HighQualityAudio,
     Video
};

enum CBRRateType {
     br44736kbps,
     br1544kbps,
     br2048kbps,
     brE3kbps,
     br64kbps,
     br2x64kbps,
     br3x64kbps,
     br4x64kbps,
     br5x64kbps,
     br6x64kbps,
     br7x64kbps,
     br8x64kbps,
     br9x64kbps,
     br10x64kbps,
     br11x64kbps,
     br12x64kbps,
     br13x64kbps,
     br14x64kbps,
     br15x64kbps,
     br16x64kbps,
     br17x64kbps,
     br18x64kbps,
     br19x64kbps,
     br20x64kbps,
     br21x64kbps,
     br22x64kbps,
     br23x64kbps,
     br24x64kbps,
     br25x64kbps,
     br26x64kbps,
     br27x64kbps,
     br28x64kbps,
     br29x64kbps,
     br30x64kbps,
     br31x64kbps
};

enum ClockRecoveryType {
     PhysInterface,
     SRTS,
     AdaptiveClock,
     LocalOsc
};

enum ForwardErrorCorrectionType {
     NoFEC,
     FECforLossSensitiveSigTransport,
     FECforDelaySensSigTransport
};

typedef boolean StructuredDataTransferType; //TRUE = STD has been chosen
typedef long long PartiallyFilledCellsType;
typedef long long CellLossIntegrationPeriodType;

struct AAL1ProfileType {
     SubType aAL1subtype;
     CBRRateType cBRRate;
     ClockRecoveryType clockRecovery;
     ForwardErrorCorrectionType forwardErrorCorrection;
     StructuredDataTransferType structuredDataTransfer;
     PartiallyFilledCellsType partiallyFilledCells;
}; // ProfileStruct for profile type = 1


typedef stringDefaultSSCSParameterProfile1PtrType;

typedef stringDefaultSSCSParameterProfile2PtrType;

struct   AAL2ProfileType {
     DefaultSSCSParameterProfile1PtrType defaultSSCSParameterProfile1Ptr;
     DefaultSSCSParameterProfile2PtrType defaultSSCSParameterProfile2Ptr;
}; // ProfileStruct for profile type = 2


struct   MaxCPCSSDUSizeType {
     long long forwardDirectionCPCS_SDU;
     long long backwardsDirectionCPCS_SDU;
};

enum AALModeType {
     MessageAssured,
     MessageUnassured,
     StreamingAssured,
     StreamingUnassured
};

enum SSCSType {
     NoSSC,
     DataSSCSonSSCOPAssured,
     DataSSCSonSSCOPNotAssured,
     FrameRelaySSCS
};

struct AAL5ProfileType {
     // ManagedEntityIdType mEId; ??
     MaxCPCSSDUSizeType maxCPCSSDUSize;
     AALModeType aALMode;
     SSCSType sSCS;
}; // ProfileStruct for profile type = 4


enum AppIdType {
     LES_CAS_noELCP,
     LES_PSTN_noELCP,
         LES_PSTN_ELCP,
         LES_DSS1forBRI_noELCP,
         LES_DSS1forBRI_ELCP,
         LES_CASforPOTS_DSS1forBRI_noELCP,
         LES_PSTNforPOTS_DSS1forBRI_noELCP,
         LES_PSTNforPOTS_DSS1forBRI_ELCP,
         LES_otherCCS,
         UnspecifiedLES
    };

    typedef long long MaxNumChannelsType;
    typedef long long MinimumChanIdType;
    typedef long long MaximumChanIdType;
    typedef long long MaxCPS_SDULengthType;
    typedef long long TimerCULengthType;

    struct AAL2PVCProfileType {
         MaxNumChannelsType maxNumChannels;
         MinimumChanIdType minimumChanId;
         MaximumChanIdType maximumChanId;
         MaxCPS_SDULengthType maxCPS_SDULength;
         TimerCULengthType timerCULength;
    }; // ProfileStruct for profile type = 3

    struct AlarmSeverityAssignProfileType {
         string eventName; //As defined in AlarmEventSupplier
         ProbableCauseType probableCauseValue; //Ditto as above
         PerceivedSeverityType serviceAffectingSeverity;
         PerceivedSeverityType nonServiceAffectingSeverity;
    }; // ProfileStruct for profile type = 5

    typedef sequence<AlarmSeverityAssignProfileType>
AlarmSeverityAssignProfileSeqType;
    typedef long long LocalMaxNumVPCSupportedType;
    typedef long long LocalMaxNumVCCSupportedType;
    typedef short LocalMaxNumVPIBitsType;
    typedef short LocalMaxNumVCIBitsType;
    typedef long long TotalEgressBandwidthType;
    typedef long long TotalIngressBandwidthType;
    typedef boolean UPCNPCIndicatorType; //TRUE = policing is on

    struct ATMNetworkAccessProfileType {
         LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
         LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
         LocalMaxNumVPIBitsType localMaxNumVPIBits;
         LocalMaxNumVCIBitsType localMaxNumVCIBits;
         TotalEgressBandwidthType totalEgressBandwidth;
         TotalIngressBandwidthType totalIngressBandwidth;
         UPCNPCIndicatorType uPCNPCIndicator;
    }; // ProfileStruct for profile type = 6


    typedef short LANType;
    typedef short EncapsulationProtocolType;
    typedef short PIDType;

    struct BridgedLANServiceProfileValues {
         LANType lAN;
         EncapsulationProtocolType encapsulationProtocol;
         PIDType pID;
    }; // ProfileStruct for profile type = 7


    typedef long long BufferedCDVToleranceType;
```

```
enum CASType {
     basic,
     e1Cas,
     SfCas,
     ds1EsfCas,
     j2Cas
};

typedef long long CableLengthType;

struct CESServiceProfileValues {
     BufferedCDVToleranceType bufferedCDVTolerance;
     CASType cAS;
}; // ProfileStruct for profile type = 8


enum DS1FramingType {
     SuperFrame,
     ExtendedSuperFrame,
     Unframed
};

enum DS1EncodingType {
     AMI,
     B8ZS
};

enum LoopbackCodeType {
     SmartJack,
     SmartJack_ONTInband,
     COInband
};

typedef boolean SupplyPowerIndType;

struct DS1ProfileValues {
     DS1FramingType dS1Framing;
     DS1EncodingType dS1Encoding;
     LoopbackCodeType loopbackCode;
     SupplyPowerIndType supplyPowerInd;
     CableLengthType cableLength;
}; // ProfileStruct for profile type = 9


enum DS3ApplicationType {
     CBitParity,
     M23
};

enum DS3EncodingType {
     B3ZS,
     CCHAN
};

struct DS3ProfileType {
     DS3ApplicationType dS3Application;
     DS3EncodingType dS3Encoding;
     CableLengthType cableLength;
}; // ProfileStruct for profile type = 10

typedef boolean DuplexType; //TRUE = full, FALSE = half
typedef boolean AutoDetectionIndType;
```

```
enum DataRateType {
     TenBT,
     HundredBT,
     ThousandBT,
     otherrate
};

typedef long long MaxFrameSizeType; //Fixed for Ethernet

typedef boolean DTEDCEType; // TRUE = DTE setting; FALSE = DCE setting.

struct EthernetProfileValues {
     DuplexType duplex;
     AutoDetectionIndType autoDetectionInd;
     DataRateType dataRate;
     MaxFrameSizeType maxFrameSize;
     DTEDCEType dTEDCE;
}; // ProfileStruct for profile type = 11


enum T303Type {
     m700,
     m1200,
     m1700,
     m2200,
     m2700,
     m3200,
     m3700,
     m4200,
     m4700
};

enum T396Type {
     ms700,
     ms1700,
     ms2700,
     ms3700,
     ms4700,
     ms5700,
     ms6700,
     ms7700,
     ms8700,
     ms9700,
     ms10700,
     ms11700,
     ms12700,
     ms13700,
     ms14700
};

struct IDLCCallProcessingProfileType {
     T303Type t303;
     T396Type t396;
}; // ProfileStruct for profile type = 12

typedef boolean ELCPIndType;

enum POTSSignallingType {
     PSTN,
     ChAS,
     CCS,
     UNKNOWN
};
```

```
enum BRISignallingType {
     DSS1,
     OtherCCS
};

typedef long long MaxNumCIDsType;
typedef long long MaxPacketLengthType;
struct ChannelWithSSCSPtrType {
     long channelIndex;
     boolean ptr1orPtr2;
};

typedef sequence<ChannelWithSSCSPtrType> ChanAndSSCSParaPtrSeqType;

struct LESProfileType {
     ELCPIndType eLCPInd;
     POTSSignallingType pOTSSignalling;
     BRISignallingType bRISignalling;
     MaxNumCIDsType maxNumCIDs;
     MaxPacketLengthType maxPacketLength;
     ChanAndSSCSParaPtrSeqType chanAndSSCSParaPtrSeq;
}; // ProfileStruct for profile type = 13


typedef boolean SpanningTreeIndType;
typedef short BridgePriorityType;
typedef short MaxAgeType;
typedef short HelloTimeType;
typedef short ForwardDelayType;

struct MACBridgeServiceProfileType {
     SpanningTreeIndType spanningTreeInd;
     BridgePriorityType bridgePriority;
     MaxAgeType maxAge;
     HelloTimeType helloTime;
     ForwardDelayType forwardDelay;
}; // ProfileStruct for profile type = 14

typedef long long SegmentLengthType;
typedef short RASTimerType;
typedef long longMaxSSSARSDULengthType;
typedef boolean SSTEDIndType;
typedef boolean SSADTIndType;

struct SSCSParameterProfile1Type {
     SegmentLengthType segmentLength;
     RASTimerType rASTimer;
     MaxSSSARSDULengthType maxSSSARSDULength;
     SSTEDIndType sSTEDInd;
     SSADTIndType sSADTInd;
}; // ProfileStruct for profile type = 15

enum ServiceCatType {
     Audio,
     Multirate,
     UNKNCategory
};

enum EncSrcType {
     ITUT,
     ATMForum,
     Proprietary
};
```

```
typedef short EncProfileIndexType;
typedef boolean AudioServIndType;
typedef short PCMEncType;
typedef boolean CMDataIndType;
typedef short CMMultiplierNumType;
typedef boolean FMDataIndType;
typedef long long FMMaxFrameLenType;
typedef boolean CASIndType;
typedef boolean DTMFIndType;
typedef boolean MFR1IndType;
typedef boolean MFR2IndType;
typedef boolean RateControlIndType;
typedef boolean SynchChangeIndType;
typedef boolean FaxDemodIndType;

struct SSCSParameterProfile2Type {
     ServiceCatType serviceCat;
     EncSrcType encSrc;
     EncProfileIndexType encProfileIndex;
     AudioServIndType audioServInd;
     PCMEncType pCMEnc;
     CMDataIndType cMDataInd;
     CMMultiplierNumType cMMultiplierNum;
     FMDataIndType fMDataInd;
     FMMaxFrameLenType fMMaxFrameLen;
     CASIndType cASInd;
     DTMFIndType dTMFInd;
     MFR1IndType mFR1Ind;
     MFR2IndType mFR2Ind;
     RateControlIndType rateControlInd;
     SynchChangeIndType synchChangeInd;
     FaxDemodIndType faxDemodInd;
}; // ProfileStruct for profile type = 16


typedef long long PCRIngressType;
typedef long long PCREgressType;
typedef long long CDVTPCRIngressType;
typedef long long CDVTPCREgressType;
typedef long long CDVTSCRIngressType;
typedef long long CDVTSCREgressType;
typedef long long SCRIngressType;
typedef long long SCREgressType;
typedef long long MaxBSIngressType;
typedef long long MaxBSEgressType;
typedef long long MFSIngressType;
typedef long long MFSEgressType;

struct TrafficDescriptorProfileType {
     ServiceCategoryType serviceCategory;
     ConformanceDefType conformanceDef;
     PCRIngressTypepCRIngress;
     PCREgressType pCREgress;
     CDVTPCRIngressType cDVTPCRIngress;
     CDVTPCREgressType cDVTPCREgress;
     CDVTSCRIngressType cDVTSCRIngress;
     CDVTSCREgressType cDVTSCREgress;
     SCRIngressType sCRIngress;
     SCREgressType sCREgress;
     MaxBSIngressType maxBSIngress;
     MaxBSEgressType maxBSEgress;
     MFSIngressType mFSIngress;
     MFSEgressType mFSEgress;
}; // ProfileStruct for profile type = 17
```

```
     typedef string LoopbackLocCodeType;

     struct UNIProfileType {
          LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
          LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
          LocalMaxNumVPIBitsType localMaxNumVPIBits;
          LocalMaxNumVCIBitsType localMaxNumVCIBits;
          LoopbackLocCodeType loopbackLocCode;
     }; // ProfileStruct for profile type = 18


     enum AnnouncementType {
          Silence,
          ReorderTone,
          FastBusy,
          VoiceAnnouncement,
          OtherAnnouncementType,
          UNKNAnnouncementType
     };

     enum TimingReferenceType {
          NetworkTimingReference,
          AdaptiveVoice,
          FreeRun,
          OtherTimingReference
     };

     typedef boolean EchoCancellationIndType;

     struct VoiceServiceProfileAAL1Type {
          AnnouncementType announcement;
          TimingReferenceType timingReference;
          EchoCancellationIndType echoCancellationInd;
     }; // ProfileStruct for profile type = 19


     typedef long long JitterTargetType;
     typedef long long JitterBufferMaxType;

     struct VoiceServiceProfileAAL2Type {
          AnnouncementType announcement;
          TimingReferenceType timingReference;
          JitterTargetType jitterTarget;
          JitterBufferMaxType jitterBufferMax;
          EchoCancellationIndType echoCancellationInd;
     }; // ProfileStruct for profile type = 20

     struct ThresholdDataComponentType {
          MonitoredParameterType monitoredParameter; // Values defined in
          Q834Common::MonitoringParameter unsigned long long thresholdValue;
     };

     typedef sequence<ThresholdDataComponentType> ThresholdDataSeqType;

     struct ThresholdDataProfileType {
          MonitoringKindType monitoringType;
          ThresholdDataSeqType thresholdValues;
     }; // ProfileStruct for profile type = 21

     typedef sequence<ATMOverbookingFactorType> ATMOverbookingProfileType;
// ProfileStruct for profile type = 22
// Local exceptions
exception ProfileInUse {};
exception DuplicateProfileName {};

// End local definitions


interface ProfileConsumer : itut_x780::ManagedObject {
     /*
     Structured event fixed header mappings:
     domain_type is set to "telecommunications",
     type_name is set to "ProfileEvent", and
     event_name is set to the constant string
     provided below. Only new profiles are announced
     through this interface.
     */

     const string profileCreation = "ProfileCreation";

     /*
     Identification of remaining items in filterable data of structured
     event.
     */

     const string profileName = "ProfileName";
     const string profileType = "ProfileType";
     const string profileAttributeValues = "ProfileAttributeValues";

     /*
     Values identifying the types of profiles used is provided below.
     */

     const unsigned short AAL1Profile = 1;
     const unsigned short AAL2Profile = 2;
     const unsigned short AAL2PVCProfile = 3;
     const unsigned short AAL5Profile = 4;
     const unsigned short AlarmSeverityAssignmentProfile = 5;
     const unsigned short ATMNetworkAccessProfile = 6;
     const unsigned short BridgedLANServiceProfile = 7;
     const unsigned short CESServiceProfile = 8;
     const unsigned short DS1Profile = 9;
     const unsigned short DS3Profile = 10;
     const unsigned short EthernetProfile = 11;
     const unsigned short IDLCCallProcessingProfile = 12;
     const unsigned short LESProfile = 13;
     const unsigned short MACBridgeServiceProfile = 14;
     const unsigned short SSCSParameterProfile1 = 15;
     const unsigned short SSCSParameterProfile2 = 16;
     const unsigned short TrafficDescriptorProfile = 17;
     const unsigned short UNIProfile = 18;
     const unsigned short VoiceServiceProfileAAL1 = 19;
     const unsigned short VoiceServiceProfileAAL2 = 20;
     const unsigned short ThresholdData = 21;
     const unsigned short ATMOverbookingFactorProfile = 22;

     /*
     Mapping to filterable data within the structured event is provided for
     a consumer event that involves the creation of a profile object.

     {
     {"ProfileName", any (NameType)},
     {"ProfileType", any (ProfileKindType)},
     {"EventTime", any (GeneralizedTimeType)},
     {"ProfileAttributeValues", any (ProfileStruct)},
     {"NotificationIdentifier", any (NotificationIdentifierType)}
     }

     */


}; // interface ProfileConsumer


interface ProfileUsageMgr : itut_x780::ManagedObject {

     // See 9.9.2.1 for the description of the behaviour of this operation

     void reName (
          in NameType oldProfileName,
          in NameType newProfileName)
          raises (UnknownProfiles,
               AccessDenied,
               DuplicateProfileName
               );

     // See 9.9.2.2 for the description of the behaviour of this operation

     boolean inUse (
          in NameType  profileName)
          raises (UnknownProfiles,
          AccessDenied);

     // See 9.9.2.3 for the description of the behaviour of this operation

     void suspendUse (
          in NameType profileName)
          raises (UnknownProfiles,
               AccessDenied);

     // See 9.9.2.4 for the description of the behaviour of this operation

     void resumeUse (
          in NameType profileName)
          raises (UnknownProfiles, AccessDenied) ;

     // See 9.9.2.5 for the description of the behaviour of this operation

     void deleteProfile (
          in NameType profileName)
          raises (UnknownProfiles,
               AccessDenied,
               ProfileInUse);

}; // interface ProfileUsageMgr

/*
This object is instantiated on the actor called Profile Object Repository.
The Supplier Management System is the client.
*/


interface ProfileRetriever : itut_x780::ManagedObject {
```

```
          // See 9.9.3.1 for the description of the behaviour of this operation

          ProfileInfoType retrieve (
               in NameType profileName)
               raises (UnknownProfiles);

     }; // interface ProfileRetriever

}; // module ProfileManager

}; // module q834_4
#endif
```

C.10 Q834Registrar.idl

```
#ifndef __Q834_4_REGISTRAR_DEFINED
#define __Q834_4_REGISTRAR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Registrar
{

     // begin definitions from other idl files


     // From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::SerialNumType SerialNumType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::AdministrationDomainType AdministrationDomainType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define EquipmentFailure Q834Common::EquipmentFailure
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define DCNTimeout Q834Common::DCNTimeout
#define DeniedAccess Q834Common::DeniedAccess
#define BackupInProgress Q834Common::BackupInProgress
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define SynchInProgress Q834Common::SynchInProgress


     // End definitions from other idl files
// Local data types

// Local exceptions
exception TooManyNEs {};
exception InvalidDCNAddress {};
exception AddressLabelMismatch {};
exception APONLayerFailure {};
exception InvalidPort {};
exception HWServicesMismatch {};

// End local definitions

valuetype NERegistrarValueType: itut_x780::ManagedObjectValueType {

     public ManagedEntityIdSeqType NEList; // GET
};


interface NERegistrar : itut_x780::ManagedObject {


     // See 9.10.1.1 for the description of the behaviour of this operation

     ManagedEntityIdType registerNE (
          in DCNAddressType nEDCNAddress,
          in UserLabelType nEUserLabel,
          in AdministrationDomainType administrationDomain)
          raises (AccessDenied,
               DCNTimeout,
               AddressLabelMismatch,
               DuplicateUserLabel,
               TooManyNEs,
               InvalidDCNAddress,
               CanNotAssignManagedEntityId,
               CanNotRetrieveUserLabel,
               DeniedAccess,
               InvalidUserLabelSyntax);

     // See 9.10.1.2 for the description of the behaviour of this operation

     void modifyNEDCNAddress (
          in ManagedEntityIdType nEManagedEntityId,
          in DCNAddressType newNEDCNAddress)
          raises(AccessDenied,
               DeniedAccess,
               AddressLabelMismatch,
               DCNTimeout,
               CommFailure,
               UnknownNE,
               InvalidDCNAddress,
               BackupInProgress);



     // See 9.10.1.3 for the description of the behaviour of this operation

     ManagedEntityIdType rangeONTorONU (
          in ManagedEntityIdType oltManagedEntityId,
          in UserLabelType nEUserLabel,
          in SerialNumType serialNum,
          in ManagedEntityIdType port ) // OLT PON Port
          raises (AccessDenied,
               CommFailure,
               EquipmentFailure,
               UnknownNE,
               UnknownPort,
               MaxSubtendingNodesExceeded,
               InsufficientPONBW,
               InvalidSerialNumSyntax,
               APONLayerFailure,
               DuplicateUserLabel,
               InvalidUserLabelSyntax,
               BackupInProgress,
               SynchInProgress );

     // See 9.10.1.4 for the description of the behaviour of this operation

     ManagedEntityIdType rangeReplacementNE (
          in ManagedEntityIdType oldNEManagedEntityId,
          in UserLabelType newNEUserLabel,
          in SerialNumType replacementSerialNum )
          raises (AccessDenied,
               CommFailure,
               UnknownNE,
               InvalidSerialNumSyntax,
               APONLayerFailure,
               EquipmentFailure,
               InvalidUserLabelSyntax,
               DuplicateUserLabel,
               HWServicesMismatch,
               BackupInProgress,
               SynchInProgress);

     // See 9.10.1.5 for the description of the behaviour of this operation

     ManagedEntityIdType rangeUpgradeNE (
          in ManagedEntityIdType oldNEManagedEntityId,
          in ManagedEntityIdType newNEManagedEntityId,
          in UserLabelType newNEUserLabel,
          in SerialNumType newNESerialNum )
          raises (AccessDenied,
               CommFailure,
               UnknownNE,
               InvalidSerialNumSyntax,
               APONLayerFailure,
               EquipmentFailure,
               InvalidUserLabelSyntax,
               DuplicateUserLabel,
               HWServicesMismatch,
               BackupInProgress,
               SynchInProgress);

     // See 9.10.1.6 for the description of the behaviour of this operation

     ManagedEntityIdType moveONTorONU(
          in ManagedEntityIdType oldNEManagedEntityId,
          in ManagedEntityIdType newPONPort)
          raises (AccessDenied,
               CommFailure,
               UnknownPort,
               APONLayerFailure,
               EquipmentFailure,
               InsufficientPONBW,
               BackupInProgress,
               SynchInProgress,
               UnknownNE );
```

```
          // See 9.10.1.7 for the description of the behaviour of this operation

          ManagedEntityIdSeqType getSubtendingNEList(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, AccessDenied);

          // See 9.10.1.8 for the description of the behaviour of this operation

          ManagedEntityIdSeqType nEListGet ()
               raises (AccessDenied);


          // See 9.10.1.9 for the description of the behaviour of this operation

          void deRegisterNE(
               in ManagedEntityIdType nE)
               raises (UnknownNE,
                    AccessDenied);

          // See 9.10.1.10 for the description of the behaviour of this
          operation

          ManagedEntityIdType associateNE(
in ManagedEntityIdType preProvisionedNE,
               in ManagedEntityIdType discoveredNE)
               raises (UnknownManagedEntity,
                    AccessDenied);

     }; // interface NERegistrar

}; // module Registrar

}; // module q834_4

#endif
```

C.11 Q834ResourceAllocation.idl

```
#ifndef __Q834_4_RESOURCEALLOCATOR_DEFINED
#define __Q834_4_RESOURCEALLOCATOR_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ResourceAllocation {

     // Begin definitions from other idl files

     // From Q834Common
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::EndPointType EndPointType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
```

```
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define InsufficientBW Q834Common::InsufficientBW
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId Q834Common::UnknownReservationId
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended Q834Common::ProfileSuspended

    // End of definitions from other idl files

    // Local data types


    struct BandwidthInfoType {
         long upstreamBW;
         long downstreamBW;
    };

    struct PortBandwidthType {
         BandwidthInfoType portBandwidth;
         ManagedEntityIdType portId;
    };

    typedef sequence<PortBandwidthType> PortBandwidthSeqType;

typedef PortBandwidthSeqType AvailableSysBandwidthSeqType;
    typedef PortBandwidthSeqType ReservedBandwidthSeqType;

    struct ReservationInfoType {
         ReservationIdType reservationId;
         EndPointType endPointA;
         EndPointType endPointB;
         NameSeqType profileNameList; // servicetemplates
         ServiceInstanceIdType serviceInstanceId;
         ReservedBandwidthSeqType reservedBandwidth;
    };

    struct ReservationBandwidthType {
         ReservationIdType reservationId;
         ReservedBandwidthSeqType reservedBandwidth;
    };

    // Local exceptions

    // End local definitions


    interface ResourceAllocator : itut_x780::ManagedObject {


         // See 9.11.1.1 for the description of the behaviour of this operation

         ReservationBandwidthType reserveForService(
              in EndPointType endPointA,
              in EndPointType endPointZ,
              in NameSeqType networkCharacteristicsProfiles,
              in ServiceInstanceIdType serviceInstanceId )
              raises (UnknownNE,
                   UnknownPort,
                   UnknownProfiles,
                   InsufficientBW,
                   MaxSubtendingNodesExceeded,
                   ConnectionCountExceeded,
                   CommFailure,
                   AccessDenied,
                   ProfileSuspended );

         // See 9.11.1.2 for the description of the behaviour of this operation

         AvailableSysBandwidthSeqType cancelReservation (
              in ReservationIdType reservationId )
              raises (UnknownReservationId,
                   CommFailure,
                   AccessDenied);

         // See 9.11.1.3 for the description of the behaviour of this operation

         ReservationIdType getReservationId (
              in ServiceInstanceIdType serviceInstanceId)
              raises (UnknownServiceInstance, AccessDenied);

         // See 9.11.1.4 for the description of the behaviour of this operation

         ReservedBandwidthSeqType reportReservedResources (
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE, AccessDenied);

         // See 9.11.1.5 for the description of the behaviour of this operation

         ReservationIdSeqType getReservations(
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE, AccessDenied);

         // See 9.11.1.6 for the description of the behaviour of this operation

         AvailableSysBandwidthSeqType cancelAllRemainingReservations(
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE,
                   CommFailure,
                   AccessDenied);

         // See 9.11.1.7 for the description of the behaviour of this operation

         ReservationInfoType getReservation (
              in ReservationIdType reservationId)
              raises (UnknownReservationId,
                   AccessDenied);

         // See 9.11.1.8 for the description of the behaviour of this operation

         AvailableSysBandwidthSeqType getAvaliableSysBandwidth(
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE,
                   CommFailure,
                   AccessDenied);

    }; // interface ResourceAllocator


}; // module ResourceAllocation

}; // module q834_4

#endif
```

C.12 Q834SchedulerManagement.idl

```
#ifndef __Q834_4_SCHEDULERMGR_DEFINED
#define __Q834_4_SCHEDULERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SchedulerManagement {
     // Begin definitions from other idl files


     // From Q834Common

     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::OperationalStateType OperationalStateType;

#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime

     // End definitions from other idl files

     // Local data types


     enum HourlyDailyWeeklyMonthlyIndType {
          Hourly,
          Daily,
          Weekly,
          Monthly
     };

     enum DayofWeekType {
          Sunday,
          Monday,
          Tuesday,
          Wednesday,
          Thursday,
          Friday,
          Saturday,
          Unspecified
     };

     typedef short DayOfMonthType; // 0 is interpreted to mean unspecified.


     struct TriggerTimeType
     {
          long time; // trigger time in number of seconds
          DayofWeekType dayOfWeek;
          DayOfMonthType dayofMonth;
     };

     typedef sequence<TriggerTimeType> TriggerTimeMatrixSeqType;

     struct SchedulerType
```

```
    {
         UserLabelType schedulerName;
         GeneralizedTimeType startTime; // schedule start time
         GeneralizedTimeType stopTime; // schedule stop time
         HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd;
         TriggerTimeMatrixSeqType matrix;
         OperationalStateType operationalState;
         AdministrativeStateType administrativeState;
    };

    typedef sequence<SchedulerType> SchedulerSeqType;

    // Local exceptions

    exception MatrixSchedulerTypeMismatch {};
    exception InvalidTrigger {};
    exception ScheduleInUse {};

    // End local definitions


    valuetype SchedulerMgrValueType: itut_x780::ManagedObjectValueType {

         public SchedulerSeqType schedulerList; // GET

    };

    interface SchedulerMgr : itut_x780::ManagedObject {


         // See 9.12.1.1 for the description of the behaviour of this operation

         void makeScheduler (
              in UserLabelType schedulerName,
              in GeneralizedTimeType startTime,
              in GeneralizedTimeType stopTime,
              in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
              in TriggerTimeMatrixSeqType matrix)
              raises (InvalidStartTime,
                   InvalidStopTime,
                   DuplicateUserLabel,
                   MatrixSchedulerTypeMismatch,
                   AccessDenied,
                   InvalidTrigger );


         // See 9.12.1.2 for the description of the behaviour of this operation

         void suspendScheduler (
              in UserLabelType schedulerName)
              raises (UnknownScheduler,
                   AccessDenied );

         // See 9.12.1.3 for the description of the behaviour of this operation

         void resumeScheduler (in UserLabelType schedulerName) raises
(UnknownScheduler, AccessDenied );


         // See 9.12.1.4 for the description of the behaviour of this operation

         void modifyTime (
              in UserLabelType schedulerName,
              in GeneralizedTimeType newStartTime,
              in GeneralizedTimeType newStopTime)
              raises (InvalidStartTime,
                   InvalidStopTime,
                   UnknownScheduler,
                   AccessDenied );


         // See 9.12.1.5 for the description of the behaviour of this operation

         void changeSchedulerName (
              in UserLabelType oldSchedulerName,
              in UserLabelType newSchedulerName)
              raises (UnknownScheduler,
                   DuplicateUserLabel,
                   AccessDenied );

         // See 9.12.1.6 for the description of the behaviour of this operation

         void modifyTriggerTimes (
              in UserLabelType schedulerName,
              in HourlyDailyWeeklyMonthlyIndType newHourlyDailyWeeklyMonthly,
              in TriggerTimeMatrixSeqType newMatrix)
              raises (UnknownScheduler,
                   MatrixSchedulerTypeMismatch,
                   AccessDenied,
                   InvalidTrigger );

         // See 9.12.1.7 for the description of the behaviour of this operation

         void removeScheduler (
              in UserLabelType schedulerName)
              raises (UnknownScheduler,
                   AccessDenied,
                   ScheduleInUse );

         // See 9.12.1.8 for the description of the behaviour of this operation

         SchedulerType retrieveScheduler (in UserLabelType schedulerName)
              raises (UnknownScheduler,
                   AccessDenied) ;

         // See 9.12.1.9 for the description of the behaviour of this operation

         SchedulerSeqType schedulerListGet () raises (AccessDenied);

    }; // interface SchedulerMgr

}; // module SchedulerManagement

}; // module q834_4

#endif
```

C.13 Q834ServiceProvisioning.idl

```
#ifndef __Q834_4_SERVICEPROVISIONING_DEFINED
#define __Q834_4_SERVICEPROVISIONING_DEFINED


#include "Q834Common.idl"
#include "Q834ProfileManager.idl"

#pragma prefix "itu.Int"
```

```
module q834_4 {

module ServiceProvisioning {

// Begin definitions from other idl files

// From Q834Common
     typedef Q834Common::RDNType RDNType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::EndPointType EndPointType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define EquipmentFailure Q834Common::EquipmentFailure
#define InsufficientBW Q834Common::InsufficientBW
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId  Q834Common::UnknownReservationId
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended ProfileManager::ProfileSuspended
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime
#define ParameterViolation Q834Common::ParameterViolation
#define UnknownPort Q834Common::UnknownPort
#define ProfileSuspended Q834Common::ProfileSuspended


// End definitions from other idl files

// Local data types

// Local exceptions
     exception UnknownConnection {};
     exception ConnectionAlreadyExists {};

// End local definitions


     interface ServiceProvisioner : itut_x780::ManagedObject {


          // See 9.13.1.1 for the description of the behaviour of this operation

          ManagedEntityIdType provisionConnection(
               in EndPointType endPointA,
               in EndPointType endPointB,
               in NameSeqType networkCharacteristicsProfiles,
               in ServiceInstanceIdType serviceInstanceId,
               in AdministrativeStateType administrativeState)
               raises (UnknownNE,
                    UnknownProfiles,
                    UnknownPort,
                    InsufficientBW,
                    ConnectionCountExceeded,
                    CommFailure,
                    EquipmentFailure,
                    ParameterViolation,
                    AccessDenied,
                    InsufficientPONBW,
                    ConnectionAlreadyExists,
                    ProfileSuspended);

        // See 9.13.1.2 for the description of the behaviour of this operation

        ManagedEntityIdType provisionReservation(
              in ReservationIdType reservationId,
              in AdministrativeStateType administrativeState)
              raises (UnknownReservationId,
                    AccessDenied);

        // See 9.13.1.3 for the description of the behaviour of this operation

        void deleteConnection (
              in ManagedEntityIdType subnetworkConnectionId)
              raises (UnknownConnection,
                    CommFailure,
                    EquipmentFailure,
                    AccessDenied);

        // See 9.13.1.4 for the description of the behaviour of this operation

        ManagedEntityIdType modifyConnection (
              in ManagedEntityIdType subnetworkConnectionId,
              in ManagedEntityIdType portB,
              in NameSeqType newNetworkCharacteristicsProfiles)
              raises (UnknownConnection,
                    UnknownPort,
                    UnknownProfiles,
                    InsufficientBW,
                    AccessDenied,
                    ProfileSuspended);

        // See 9.13.1.5 for the description of the behaviour of this operation

        void suspendService (
              in ServiceInstanceIdType serviceInstanceId,
              in GeneralizedTimeType startTime,
              in GeneralizedTimeType stopTime)
              raises (UnknownServiceInstance,
                    InvalidStartTime,
                    InvalidStopTime,
                    AccessDenied);

        // See 9.13.1.6 for the description of the behaviour of this operation

        void resumeService(
              in ServiceInstanceIdType serviceInstanceId)
              raises (UnknownServiceInstance,
                    AccessDenied);

    }; // interface ServiceProvisioner

}; // module ServiceProvisioning

}; // module q834_4

#endif
```

C.14 Q834Synchroniser.idl

```
#ifndef __Q834_4_SYNCHRONISER_DEFINED
#define __Q834_4_SYNCHRONISER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Synchroniser
{

// begin definitions from other idl files


// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::UserLabelTypeUserLabelType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DCNTimeout Q834Common::DCNTimeout
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define BackupInProgress Q834Common::BackupInProgress
#define Timeout Q834Common::Timeout
#define SynchInProgress Q834Common::SynchInProgress


// End definitions from other idl files

// Local data types

     typedef sequence<short> CurrentListingSeqType;

     struct ScheduledSynchNEType {
          ManagedEntityIdType managedEntityId;
          UserLabelType schedulerName;
     };
     typedef sequence<ScheduledSynchNEType> ScheduledSynchNESeqType;

// Local exceptions
     exception NoSynchInProgress {};

// End local definitions

     valuetype SynchroniserValueType: itut_x780::ManagedObjectValueType {

          public ScheduledSynchNESeqType scheduledSynchNEList; // GET

     };


     interface NESynchroniser : itut_x780::ManagedObject {

          // Define constants for current event listings
```

```
const short CURRENT_ALARM = 1;
const short CURRENT_OPERATIONAL_STATE_DISABLED = 2;
const short CURRENT_ADMINSTRATIVE_STATE_LOCKED = 3;
const short CURRENT_PROTECTION_SWITCHING_EVENT = 4;
const short CURRENT_SERVICE_OUTAGE = 5;

// See 9.14.1.1 for the description of the behaviour of this operation

void synchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          EquipmentFailure,
          BackupInProgress,
          SynchInProgress);

// See 9.14.1.2 for the description of the behaviour of this operation


void abortSynchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          EquipmentFailure,
          NoSynchInProgress);

// See 9.14.1.3 for the description of the behaviour of this operation

void scheduleSynchNE(
     in ManagedEntityIdType nEManagedEntityId,
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler);

// See 9.14.1.4 for the description of the behaviour of this operation

void modifyNESynchSchedule(
     in ManagedEntityIdType nEManagedEntityId,
     in UserLabelType newSchedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler);

// See 9.14.1.5 for the description of the behaviour of this operation

void cancelScheduledSynchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          UnknownNE);

// See 9.14.1.6 for the description of the behaviour of this operation

void synchCurrentEventListings(
     in ManagedEntityIdType nEManagedEntityId,
     in CurrentListingSeqType currentListingTypeList)
     raises (AccessDenied,
          CommFailure,
          DCNTimeout,
          UnknownNE,
                    EquipmentFailure,
                    Timeout);

          // See 9.14.1.7 for the description of the behaviour of this operation

          ScheduledSynchNESeqType scheduledSynchNEListGet ()
               raises (AccessDenied);

     }; // interface NESynchroniser

}; // module Synchroniser

}; // module q834_4

#endif
```

C.15 Q834Test.idl

```
#ifndef __Q834_4_TEST_DEFINED
#define __Q834_4_TEST_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Test
{

     // begin definitions from other idl files

     // From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::LoopbackLocationIdSeqType
     LoopbackLocationIdSeqType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStopTime Q834Common::InvalidStopTime
#define InvalidStartTime Q834Common::InvalidStartTime
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance

     // End definitions from other idl files


     // Local data types
     enum DirectionalityType {
             Egress,
          Ingress,
          BothDirections
     };

     struct ATMLoopbackInfoType {
          DirectionalityType directionality;
          LoopbackLocationIdSeqType targetLLID;
          boolean segmentCellInd;
     };

     typedef  unsigned short TestIterationNumType;

     struct ATMLoopbackResultType {
          LoopbackLocationIdSeqType loopbackingLLID;
          unsigned long responseTime; //in microseconds
          boolean succeeded; //true or false
     };

     typedef sequence<ATMLoopbackResultType> ATMLoopbackResultSeqType;

     struct AggregateATMLoopbackResultType {
          unsigned short iterationSeqNum;
          ATMLoopbackResultSeqType iterationTestResults;
     };

     typedef sequence<AggregateATMLoopbackResultType>
     AggregateATMLoopbackResultSeqType;

     struct ATMContinuityCheckInfoType {
          DirectionalityType directionality;
          boolean segmentCellInd;
     };


     typedef short DiagnosticType; // Supplier specific.
     typedef DiagnosticType ResourceSelfTestInfoType;
     typedef sequence<ResourceSelfTestInfoType> ResourceSelfTestInfoSeqType;

     struct ResourceSelfTestResultType {
             DiagnosticType diagnostic;
             boolean testPassed;
     };

     typedef sequence<ResourceSelfTestResultType>
     ResourceSelfTestResultSeqType;
     typedef long long CCSetUpIdType;
     typedef TrackingObjectIdType TestTrackingObjectIdType;
     typedef long long LoopbackTrackingObjectIdType;


     typedef unsigned short LoopbackTestType;

     struct LoopbackInfoType {
          LoopbackTrackingObjectIdType trackingId;
          unsigned long remainingTime; // in seconds
ManagedEntityIdType ctpId;
          LoopbackTestType loopbackTest;
          DirectionalityType directionality;
     };

     typedef sequence<LoopbackInfoType> LoopbackInfoSeqType;
     typedef string SupportedTestType;

     typedef sequence<SupportedTestType> SupportedTestSeqType;

     struct ScheduledTestNEType {
            ManagedEntityIdType managedEntityId;
            SupportedTestType supportedTest;
            UserLabelType schedulerName;
     };
     typedef sequence<ScheduledTestNEType> ScheduledTestNESeqType;

     struct TestHistoryType {
          ManagedEntityIdType managedEntityId;
          SupportedTestType supportedTest;
          StatusValueType testStatus;
     };
     typedef sequence<TestHistoryType> TestHistorySeqType;

     // Local exceptions

     exception InvalidTimeoutPeriod {};
     exception NotAvailableForTest {};
     exception InvalidLocationId {};
     exception UnknownTest {};
     exception UncontrolledTestInProgress {};
     exception InvalidDirection {};
     exception InvalidTestOperations {};

     // End local definitions

     valuetype TestActionPerformerValueType: itut_x780::ManagedObjectValueType {

          public ScheduledTestNESeqType scheduledTestNEList; // GET

     };


     interface TestActionPerformer : itut_x780::ManagedObject {

          // See 9.15.1.1 for the description of the behaviour of this operation

          AggregateATMLoopbackResultSeqType aTMLoopback (
               in UserIdType testRequestorId,
               in ManagedEntityIdType ctp,
               in ATMLoopbackInfoType aTMLoopbackInfo,
               in TestIterationNumType testIterationNum,
               in ServiceInstanceIdType serviceInstanceId)
               raises (AccessDenied,
                   CommFailure,
                   UnknownManagedEntity,
                   NotAvailableForTest,
                   InvalidLocationId,
                   InvalidDirection);

          // See 9.15.1.2 for the description of the behaviour of this operation

          CCSetUpIdType initializeContinuityCheck(
               in UserIdType testRequestorId,
               in ManagedEntityIdType sourceCtp,
               in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
               in GeneralizedTimeType stopTime,
               in ServiceInstanceIdType serviceInstanceId)
               raises (AccessDenied,
                   CommFailure,
               UnknownManagedEntity,
               NotAvailableForTest,
               InvalidStartTime,
               InvalidStopTime,
               InvalidDirection);

     // See 9.15.1.3 for the description of the behaviour of this operation

     void terminateContinuityCheck(
          in CCSetUpIdType cCSetUpId)
          raises (AccessDenied,
               CommFailure,
               UnknownTest);

     // See 9.15.1.4 for the description of the behaviour of this operation

     TestTrackingObjectIdType scheduleResourceSelfTest(
          in UserIdType testRequestorId,
          in ManagedEntityIdType targetNE,
          in unsigned long timeOutPeriod, //In seconds.
          in ResourceSelfTestInfoSeqType specificTestInfo,
          in UserLabelType schedulerName)
          raises (AccessDenied,
               UnknownNE,
               UnknownScheduler,
               InvalidScheduler,
               InvalidTimeoutPeriod,
               InvalidTestOperations);

     // See 9.15.1.5 for the description of the behaviour of this operation

             void modifyResourceSelfTestSchedule(
               in TestTrackingObjectIdType testTrackingObjectId,
               in UserLabelType newSchedulerName)
               raises (AccessDenied,
                   UnknownTest,
                   UnknownScheduler,
                   InvalidScheduler);

     // See 9.15.1.6 for the description of the behaviour of this operation

          void cancelScheduledResourceSelfTest (
               in TestTrackingObjectIdType testTrackingObjectId)
               raises (AccessDenied,
                   UnknownTest);

     // See 9.15.1.7 for the description of the behaviour of this operation

     TestTrackingObjectIdType conductResourceSelfTest (
          in UserIdType testRequestorId,
          in ManagedEntityIdType targetNE,
          in unsigned long timeOutPeriod, //In seconds.
          in ResourceSelfTestInfoSeqType specificTestInfo)
          raises (AccessDenied,
               CommFailure,
               UnknownNE,
               InvalidTimeoutPeriod,
               InvalidTestOperations);

     // See 9.15.1.8 for the description of the behaviour of this operation

     ResourceSelfTestResultSeqType terminateResourceSelfTest (
          in TestTrackingObjectIdType testTrackingObjectId)
          raises (UnknownTest,
               UncontrolledTestInProgress,
               AccessDenied);

     // See 9.15.1.9 for the description of the behaviour of this operation

     LoopbackTrackingObjectIdType initiateLoopback (
          in UserIdType testRequestorId,
          in ManagedEntityIdType loopingCtp,
          in long duration, //In minutes.
          in DirectionalityType directionality,
          in LoopbackTestType loopbackTest,
          in ServiceInstanceIdType serviceInstanceId)
          raises (AccessDenied,
               CommFailure,
               UnknownManagedEntity,
               NotAvailableForTest);

     // See 9.15.1.10 for the description of the behaviour of this
     operation

     void terminateLoopback(
          in LoopbackTrackingObjectIdType testTrackingObjectId)
          raises (UnknownTest,
               AccessDenied);

     // See 9.15.1.11 for the description of the behaviour of this
     operation

     LoopbackInfoType getLoopbackInfo(
          in ManagedEntityIdType cTP)
          raises (UnknownManagedEntity,
               AccessDenied);

     // See 9.15.1.12 for the description of the behaviour of this
     operation

     LoopbackInfoSeqType getLoopbackInfoByNE(
          in ManagedEntityIdType nEId)
          raises (UnknownManagedEntity,
               AccessDenied);

     // See 9.15.1.13 for the description of the behaviour of this
     operation

     StatusValueType getTestStatus (
          in LoopbackTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownTest);

     // See 9.15.1.14 for the description of the behaviour of this
     operation

     ScheduledTestNESeqType  scheduledTestNEListGet()
          raises (AccessDenied);

     // See 9.15.1.15 for the description of the behaviour of this
     operation

     TestHistorySeqType testHistoryByManagedEntity (
          in ManagedEntityIdType managedEntityId)
          raises (UnknownManagedEntity,
               AccessDenied);
          // See 9.15.1.16 for the description of the behaviour of this
          operation

          TestHistorySeqType testHistoryByServiceInstance (
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                    AccessDenied);

     }; // interface TestActionPerformer

}; // module Test

}; // module q834_4

#endif
```

C.16 Q834Filetransfer.idl

```
#ifndef __Q834_4_TRANSFERMGR_DEFINED
#define __Q834_4_TRANSFERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module FileTransfer
{

// begin definitions from other idl files

// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::FilenameType FilenameType;
     typedef Q834Common::TransferTrackingObjectIdType
     TransferTrackingObjectIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownRecordSet Q834Common::UnknownRecordSet

// End definitions from other idl files

// Local data types



     struct FileTransferHistoryType {
       ManagedEntityIdType recordSetId;
       DCNAddressType destinationServerAddr;
       UserIdType  userId;
```

```
        FilenameType destinationFile;
        GeneralizedTimeType transferTime;
      };
      typedef sequence<FileTransferHistoryType> FileTransferHistorySeqType;

      struct ScheduledFileTransferType {
        ManagedEntityIdType recordSetId;
        UserLabelType schedulerName;
      };
      typedef sequence<ScheduledFileTransferType> ScheduledFileTransferSeqType;

// Local exceptions

      exception UnknownTransferProcess {};

// End local definitions


      valuetype TransferMgrValueType: itut_x780::ManagedObjectValueType {

           public FileTransferHistorySeqType fileTransferHistoryList; // GET
           public ScheduledFileTransferSeqType scheduledFileTransferList; // GET

      };

      interface TransferMgr : itut_x780::ManagedObject {

           // See 9.16.1.1 for the description of the behaviour of this operation

                TransferTrackingObjectIdType fileTransfer(
                     in ManagedEntityIdType recordSetId,
                     in DCNAddressType destinationServerAddr,
                     in UserIdType userId,
                     in PasswordType password,
                     in FilenameType destinationFile,
                     in boolean overwriteExistingFile)
                     raises (AccessDenied,
                          CommFailure,
                          UnknownRecordSet,
                          UnknownDestinationServer
                          );


           // See 9.16.1.2 for the description of the behaviour of this operation

                TransferTrackingObjectIdType scheduleFileTransfer(
                     in ManagedEntityIdType recordSetId,
                     in DCNAddressType destinationServerAddr,
                     in UserIdType userId,
                     in PasswordType password,
                     in FilenameType destinationFile,
                     in boolean overwriteExistingFile,
                     in UserLabelType schedulerName)
                     raises (AccessDenied,
                          UnknownRecordSet,
                          UnknownDestinationServer,
                          UnknownScheduler,
                          InvalidScheduler);

           // See 9.16.1.3 for the description of the behaviour of this operation

                   void modifyFileTransferSchedule(
                     in TransferTrackingObjectIdType transferTrackingObjectId,
                     in UserLabelType newSchedulerName)
                     raises (AccessDenied,
                         UnknownTransferProcess,
                         UnknownScheduler,
                         InvalidScheduler);

             // See 9.16.1.4 for the description of the behaviour of this operation

                 void cancelScheduledFileTransfer (
                     in TransferTrackingObjectIdType transferTrackingObjectId)
                     raises (AccessDenied,
                         UnknownTransferProcess);

             // See 9.16.1.5 for the description of the behaviour of this operation

                 StatusValueType getStatus(
                     in TransferTrackingObjectIdType transferTrackingObjectId)
                     raises (UnknownTransferProcess,
                         AccessDenied);

             // See 9.16.1.6 for the description of the behaviour of this operation

             FileTransferHistorySeqType fileTransferHistoryListGet ()
                 raises (AccessDenied);

             // See 9.16.1.7 for the description of the behaviour of this operation

             ScheduledFileTransferSeqType scheduledFileTransferListGet()
                 raises (AccessDenied);

       }; // interface TransferMgr

  }; // module FileTransfer

  }; // module q834_4

  #endif
```

Annex D

Example endpoint templates

Tables D.1 and D.2 provide example templates of how an endpoint might be defined by service type. Table D.1 shows examples for the case when the endpoint is part of an NNI port. Table D.2 shows examples for the case when the endpoint is part of a UNI port.

Table D.1/Q.834.4 – NNI port endpoints

Service type	Port	Parameter	Profiles
DS1	TDM DS3	Channel #	-
DS3	TDM DS3	-	-
DS1	ATM DS3	VPI/VCI	-
DS3	ATM DS3	VPI/VCI	-
DS1	TDM OC-n	STS #/VT1.5 #	-
DS1	ATM OC-n	VPI/VCI	-
DS3	ATM OC-n	VPI/VCI	-
Bridged LAN	ATM OC-n	VPI/VCI	-
Bridged LAN	ATM DS3	VPI/VCI	-
Bridged LAN	TDM DS3	Channel #	-
Bridged LAN	TDM OC-n	STS #/VT1.5 #	-
Voice	ATM DS3	VPI/VCI CID	TrafficDescriptorProfile (ingress) TrafficDescriptorProfile (egress)
Voice	-	Interface Group #/ Call Ref Value	-
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (ingress) TrafficDescriptorProfile (egress)

Table D.2/Q.834.4 – UNI port endpoints

Service type	Port	Parameter	Profiles
DS1	TDM DS3	Channel #	DS1Profile CESProfile AAL1Profile
DS3	TDM DS3	–	DS3Profile CESProfile AAL1Profile
DS1	ATM DS3	VPI/VCI	–
DS3	ATM DS3	VPI/VCI	–
DS1	TDM OC-n or STS-n	STS #/VT1.5 #	AAL1Profile CESProfile
DS1	ATM OC-n	VPI/VCI	–
DS3	ATM OC-n	VPI/VCI	–
Bridged LAN	Ethernet	Logical port #	BridgedLANServiceProfile AAL5Profile or AAL1Profile
Voice	RJ-11	–	VoiceServiceProfileAAL2 SSCSParameterProfile1 SSCSParameterProfile2 LESService
Voice	RJ-11	–	VoiceServiceProfileAAL1
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (ingress) TrafficDescriptorProfile (egress)
VLAN	Ethernet	VLAN tag	–

Printed in Switzerland
Geneva, 2004