Alcatel·Lucent

Alcatel-Lucent

1359 CorbaNBI | Release 1.6

Interface Specification

8BE 00097 0141 PTZZA Ed.2

8DG 34000 AAAA TQZZA

January 2010
Issue 4

CORBANBI 1.6 INTERFACE SPECIFICATION

Author: **CORBA NBI System team**

8DG 34000 AAAA TQZZA

Revision History

ED	Date	Comments	Author	Appraisal authority
1.1	2009-10-27	Creation	Liu Liu	Tang Tie
1.2	2009-10-30	Modification	Wu liang	Tang Tie
1.3	2009-11-1	Modification	Cui Kai	Tang Tie
1.4	2010-01-18	Modification	Wu liang	Tang Tie

Table of Contents

List of Figures

List of Tables

HISTORY

ED	Date	Author	Change Reason
1.1	2009-10-27	Liu Liu	add getAllSupportingEquipment, getAllSupportingEquipmentNames, getSupportingEquipment, getSupportingEquipmentNames, getSupportedEquipment, getSupportedEquipmentNames Add 1.6 supported to existing interfaces
1.2	2009-10-30	Wu Liang	▪ Add all PKTNBI1.6 step 1 interfaces. ▪ Replace the “NMNBI” to “EMLNBI”. ▪ Replace the “RMNBI” to SDHNBI”. ▪ Update SDHNBI interface:All operations of multiLayerSubnetwork interface. ▪ Update Common interface: → getAllActiveAlarms → getVersion → ping → endSession ▪ Update EMLNBI interface: → getAllActiveAlarms → getAllFixedCrossConnections → getGTP → getAllGTPs → getAllGTPNames → getContainingGTP → getPotentialFixedCCs
1.3	2009-11-1	Cui Kai	▪ Change “Revision History” table ▪ Add PKTNBI LayerRate table ▪ Add PKTNBI Alarm ObjectType table

Table 1 – History

OPEN ISSUES

This part of the document is used to record and track open issues still living when the document is distributed. As the document become stable, such issues will be solved and tracked in the list.

Ref.	Date	Issue	Description	Status

Table 2 – Open Issues

PURPOSE AND SCOPE

This handbook is to describe the parameter of the Corbal NBI IDL.

REFERENCED DOCUMENTS

[1] 8BU 93003 0000 ASZZA - OND Product Life Cycle
http://dedom002.stgl.sel.alcatel.de/smapsR2/3CK/00131/0001/QRZZA03/3CK001310001QRZZA.pdf

[2] 8BE 00097 0042 QRZZA - OND NM SW Development and Validation
http://nm.ond.alcatel.it/ppo/docs/OND-NM-SW_DevVal_ed6.pdf

[3] 8BE 00097 0141 PTZZA - OND NM R&D Interface Specification (PB) Template Ed.2
http://nm.ond.alcatel.it/ppo/docs/Templates/NM_PB_051214.dot

[4] 1350 MS Physical Topology Manager - Technical Functional Specification
3AL 00000 XXXX DSZZA

[5] ITU-T (International Telecommunication Union) – Generic Functional architecture of Transport network G.805 (03/2000)
http://aww.alcatel.de/pub/smis/itu/dms/pay/itu-t/rec/g/T-REC-G.805-200003-I!!MSW-E.doc

[6] ITU-T (International Telecommunication Union) – Functional architecture of connectionless layer networks G.809 (03/2003)
http://aww.alcatel.de/pub/smis/itu/dms/pay/itu-t/rec/g/T-REC-G.809-200303-I!!MSW-E.doc

[7] ITU-T (International Telecommunication Union) – Data Networks and Open System Communications X.731 (04/95)
http://aww.alcatel.de/pub/smis/itu/dms/pay/itu-t/rec/x/T-REC-X.731-199201-I!!MSW-E.doc

[8] Docs TMF 513, 517, 608, 814,

[9] MTNM Business Agreement NML-EML Interface Version 3.0 TMF 513
Member Evaluation Version 3.0 August 2003

[10] MTNM Information Agreement NML-EML Interface Version 3.0 TMF 608
Member Evaluation Version 3.0 August 2003

[11] MTNM Implementation Statement Templates and Guidelines NML-EML Interface Version 3.0 TMF 814A
Member Evaluation Version 3.0 August 2003

[12] MTOSI Working document TMF 517 v0.1
Member Evaluation January 2005

[13] MTNM Support for a Naming convention Version 3.0 TMF 608 v0.1 2003

[14] MTNM Business Agreement NML-EML Interface Version 3.5 TMF 513
Member Evaluation Version 3.5

[15] MTNM Information Agreement NML-EML Interface Version 3.5 TMF 608
Member Evaluation Version 3.5

1. GLOSSARY AND DEFINITIONS

1.1. Definitions

1.2. Abbreviations

CORBA	Common Object Request Broker Architecture
EMS	Element Management System
NMS	Network Management System

2. INTRODUCTION

2.1. Document scope

This document covers the NML-EML interface.

3. ARCHITECTURAL DESCRIPTION

4. INTERFACE DESCRIPTION

5. DESIGN CHOICES

6. DATA TYPE DESCRIPTION

6.1.1. AssignedSeverity_T

```
Enum AssignedSeverity_T
{
 AS_INDETERMINATE,
 AS_CRITICAL,
 AS_MAJOR,
 AS_MINOR,
 AS_WARNING,
 AS_NONALARMED,
 AS_FREE_CHOICE
};
```

General comment	
Alarm notifications include a severity. The severity is defined in notifications::PerceivedSeverity_T . The assigned severity is allocated to a probable cause using the alarm severity assignment profile (ASAP). The identified probable cause should be raised with the assigned severity. The assignment of severities is extended beyond notifications::PerceivedSeverity_T to cover the case where: no alarm should be raised: "AS_NONALARMED" the EMS/ME is free to make a choice based upon its local criteria: "AS_FREE_CHOICE"	
Attribute name	**Comment**
➢ AS_INDETERMINATE	If there is no local mechanism on the ME/EMS to determine severity then the severity "AS_INDETERMINATE" should be assigned.
➢ AS_CRITICAL	The severity of an alarm notification is critical.
➢ AS_MAJOR	The severity of an alarm notification is major.
➢ AS_MINOR	The severity of an alarm notification is minor.
➢ AS_WARNING	The severity of an alarm notification is warning.
➢ AS_NONALARMED	The EMS should not emit an alarm over the EML-NML interface.
➢AS_FREE_CHOICE	The ME/EMS are free to determine the severity.

6.1.2. AlarmSeverityAssignment_T

```
struct AlarmSeverityAssignment_T
 {
  string probableCause;
  string probableCauseQualifier;
  string nativeProbableCause;
  AssignedSeverity_T serviceAffecting;
  AssignedSeverity_T serviceNonAffecting;
  AssignedSeverity_T serviceIndependentOrUnknown;
 };
```

General comment

The AlarmSeverityAssignment_T structure provides three values for severity to cover the three cases of service affecting conditions, non-service affecting conditions and conditions where the service impact is unknown. The structure identifies the specific case of alarm that it applies to using three probable cause identifiers in combination.	
Attribute name	**Comment**
➢ probableCause	The probable cause of the alarm to which the severities should be applied. This is a defined string
➢ probableCauseQualifier	A qualifier of the probable cause that is used to achieve uniqueness in some cases where the probable cause is not sufficient. This is a free form string.
➢ nativeProbableCause	The probable cause used on the ME/EMS. This may also be used to qualify the probable cause to achieve uniqueness in some cases where the probable cause is not sufficient. This is a free form string.
➢ serviceAffecting	Severity assigned to the probable cause when service affecting.
➢ serviceNonAffecting	Severity assigned to the probable cause when not service affecting.
➢ serviceIndependentOrUnknown	Severity assigned to the probable cause when the service impact is not known or is known to be service independent.

6.1.3. AlarmSeverityAssignmentList_T

typedef sequence<AlarmSeverityAssignment_T> AlarmSeverityAssignmentList_T;

An alarm severity assignment list provides a listing of all abnormal conditions that may exist in instances of an object class.
Each element of this sequence specifies the probable cause (plus optionally the probableCauseQualifier and / or the nativeProbableCause) and the three severities to be assigned.

6.1.4. ASAP_T

```
struct ASAP_T
{
 globaldefs::NamingAttributes_T name;
 string userLabel;
 string nativeEMSName;
 string owner;
 boolean notModifiable;
 AlarmSeverityAssignmentList_T alarmSeverityAssignmentList;
 globaldefs::NVSList_T  additionalInfo;
};
```

General comment

An ASAP (Alarm Severity Assignment Profile) models the (flexible) severity assignment to specified probable causes. So the ASAP object includes a table, with each row specifying the probable cause (plus optionally the probableCauseQualifier and/or the nativeProbableCause) and the assigned severity for "service affecting", "non service affecting" and "service independent or unknown" alarms.

Probable cause, probableCauseQualifier and nativeProbableCause are the coordinates of the ASAP entries, so there must not be two ASAP entries with the same coordinates. It is the responsibility of the EMS to enforce this.

Alarms against entities that are not modeled by the interface are reported using the "AID" objectType. For these "AID" alarms the ASAP mechanism is not applicable. Of course this does not apply for 2nd class objects (which potentially emit alarms) whose type has been specified using the "ENUM extension" methodology (i.e. also using AID as explained for notifications::ObjectType_T)).

Typically the ASAP function is implemented in the OSS or NE depending upon where the alarm is originated. E.g. a LOS severity will usually be assigned directly by the ME, so it is up to the EMS to download the defined ASAP to the ME. Other alarms, e.g. on subnetwork connections, may be generated by the EMS.

When for PTP, CTP, FTP: the layer-specific transmission parameter AlarmReporting changes from "off" to "on"

When for SNC, TopologicalLink: the additional info parameter AlarmReporting changes from "off" to "on"

When for Equipment, EquipmentHolder:

the attribute alarmReportingIndicator changes from FALSE to TRUE

Alarms are always reportable for all other object types as these do not have any alarm reporting attribute.

When an alarm becomes reportable the ASAPs are applied (accessed via the aSAPpointer). The severity of the alarm is adopted from the ASAP entry that matches for all of the following:

same probableCause

same probableCauseQualifier (an empty string is a match)

same nativeProbableCause (an empty string is a match)

For example, if the reportable alarm has LOS probableCause and an ASAP entry is found with LOS probableCause and both probableCauseQualifier and nativeProbableCause are empty strings, then the search is successful.

If there is a matching ASAP, then the severities are adopted on the following basis:

the alarm is service affecting: it is assigned the severity specified in the serviceAffecting field

the alarm is service non affecting: it is assigned the severity specified in the serviceNonAffecting field

the alarm is service independent or ME/EMS is not equipped to determine

if the alarm affects service or not: it is assigned the severity specified in the serviceIndependentOrUnknown field

If the assigned severity specified in the ASAP is:

"AS_FREE_CHOICE" then the ME/EMS are free to determine the severity.

If there is no local mechanism on the ME/EMS to determine severity then the severity "AS_INDETERMINATE" should be assigned.

"AS_NONALARMED" then the EMS should not emit an alarm over the EML-NML interface

If there is no ASAP that matches then the ME/EMS are free to determine the severity. If there is no local mechanism on the ME/EMS to determine severity then the severity "AS_INDETERMINATE" should be assigned.

Once a severity has been assigned (other than for "AS_NONALARMED" and replacing "AS_FREE_CHOICE" by a proper severity), the alarm notification is emitted with the assigned severity.

Note:Any operation of alarm retrieval will not include "AS_NOTALARMED" alarms.

Attribute name	Comment
➢ name	The name represents the name of the ASAP, which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of EMS. It is a read-only attribute.
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service setUserLabel. It is a

	read/write attribute.
➢ nativeEMSName	The name represents how the ASAP is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through).
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service setOwner. It is a read/write attribute.
➢ notModifiable	If TRUE, then this ASAP instance is fixed, i.e. is an ASAPs which is defined at ME or EMS level and can be neither modified nor deleted through MTNM interface, but only assigned/de-assigned. If FALSE, otherwise. It is a read-only attribute.
➢ alarmSeverityAssignmentList	Each element of this sequence specifies the probable cause (plus optionally the probableCauseQualifier and / or the nativeProbableCause) and the three severities to be assigned.
➢ additionalInfo	This attribute allows the communication of additional information that is not explicitly modelled.

6.1.5. ASAPList_T

typedef sequence<ASAP_T> ASAPList_T;
Sequence of ASAP_T.

6.1.6. ASAPCreateModifyData_T

```
struct ASAPCreateModifyData_T
{
 string userLabel;
 boolean forceUniqueness;
 string owner;
 AlarmSeverityAssignmentList_T alarmSeverityAssignmentList;
 globaldefs::NVSList_T additionalInfo;
};
```

General comment	
ASAPCreateModifyData_T is used when ASAP object is either created or modified by the NMS.	
Attribute name	**Comment**
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service setUserLabel. It is a read/write attribute.
➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst ASAPs of the EMS. The operation will fail if userLabel is already in use.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service setOwner. It is a read/write attribute.
➢ alarmSeverityAssignmentList	Each element of this sequence specifies the probable cause (plus optionally the probableCauseQualifier and / or the

	nativeProbableCause) the three severities to be assigned.
➢ additionalInfo	This attribute allows the communication of additional information which is not explicitly modelled.

6.1.7. Capability_T

typedef globaldefs::NameAndStringValue_T Capability_T;

A Capability_T value is used to identify a functionality supported by the EMS across the NML-EML interface. It is a name value pair, in which the name represents the feature/capability name and the value represents the support or non-support of the specified feature/capability.

The EMS capabilities for this Release include individual IDL operation support. The feature/capability name part is used to identify an IDL operation using the following convention:
module_name::interface_name::operation_name".

There are a number of other specifiable capabilities in addition to the operation-oriented capabilities:
"Supports_CC_sharing" - defined for MultiLayerSubnetworkMgr_I indicates the EMS' SNC management mode of operation
"Supports_pending" - defined for MultiLayerSubnetworkMgr_I indicates the EMS' SNC management mode of operation.
"Supports_adjacent_termination_inclusion" - defined for MultiLayerSubnetworkMgr_I indicates that the EMS allows extension of SNCs to all G.805 TCP of CTPs, PTPs and FTPs.

The currently defined values are as follows:
"Supported": The specified feature/capability is fully or partially supported across the NML-EML interface; an operation may be partially supported if not all values of the parameters are supported.
"Unsupported": The specified feature/capability is not supported at all across the NML-EML interface.
Other capabilities may be added with the approval of the Specification

Authority, or through bilateral agreements.

6.1.8. CapabilityList_T

typedef sequence<Capability_T> CapabilityList_T;

Set of Capability_T. Used to represent the full set of capabilities of a manager. Any capability that is not listed is considered unsupported.

6.1.9. EMS_T

```
struct EMS_T
{
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        string emsVersion;
        string type;
        long    maxNEs;
        long    actualNEs;
        string physicalLocation;
        string ipAddress;
        string operationState;
        notifications::PerceivedSeverity_T alarmState;
        globaldefs::NVSList_T additionalInfo;
};
```

General comment	
Holds EMS identification information.	
Attribute name	**Comment**
➢ name	Represents the friendly name of the EMS and is constructed according to the following pattern: "CompanyName/EMSName" The EMSName must be unique relative to the CompanyName. It is up to each company to maintain this.It is a readonly attribute.
➢ userLabel	The userLabel is a friendly name that the operator wants to give to the EMS. Typical expectations of the operator are that the same name is seen on all operation systems. This is set by the NMS and could be displayed on the EMS based on each system's capabilities. THIS IS NOT A MANDATORY EXPECTATION, but is left to the implementation of the EMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	Represents how the EMS refers to itself on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through). May be a null string.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ emsVersion	Software version of the EMS. This is a free format string with no semantics attached to it for the NMS. Each EMS system models its software version independently. There is no standard way to represent the software version. Decision about support of a particular version by the NMS system is up to the NMS system. emsVersion may be an empty string. It is a readonly attribute.
➢type	Free format string indicating the type of EMS. The EMS type may be empty string. It is a readonly attribute.
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile. This may be an empty list.

6.1.10. managerNames_T

typedef sequence<string> managerNames_T;

Sequence of manager names.

6.1.11. EquipmentObjectType_T

typedef string EquipmentObjectType_T;

Type of equipment object in equipment holder. Values are vendor-specific.

6.1.12. EquipmentObjectTypeList_T

typedef sequence<EquipmentObjectType_T> EquipmentObjectTypeList_T;

Set of types of equipment objects that can be supported by an equipment holder.

6.1.13. ServiceState_T

```
enum ServiceState_T
 {
        IN_SERVICE,
        OUT_OF_SERVICE,
        OUT_OF_SERVICE_BY_MAINTENANCE,
        SERV_NA
  };
```

General comment	
Basic administration state equipment objects.	
Attribute name	**Comment**
➢ IN_SERVICE	IN_SERVICE means the entity has been put into operation and is operating as provisioned (completely or partially).
➢ OUT_OF_SERVICE	OUT_OF_SERVICE means the entity is entirely not capable of performing its provisioned functions and is not restricted by administrative actions.
➢OUT_OF_SERVICE_BY_MAINTENANCE	OUT_OF_SERVICE_BY_MAINTENANCE means that the entity has been taken intentionally out of service by a management action.
➢ SERV_NA	SERV_NA means that the service state is not applicable.

6.1.14. EquipmentHolderType_T

typedef string EquipmentHolderType_T;

Types of equipment holders. Valid values are: "rack", "shelf", "sub_shelf", "slot", and “sub_slot".

6.1.15. HolderState_T

```
enum HolderState_T
 {
        EMPTY,
        INSTALLED_AND_EXPECTED,
        EXPECTED_AND_NOT_INSTALLED,
        INSTALLED_AND_NOT_EXPECTED,
        MISMATCH_OF_INSTALLED_AND_EXPECTED,
        UNAVAILABLE,
        UNKNOWN
         };
```

Indicates the state of the equipment holder object with respect to its directly contained equipment. Applies when the equipment holder can contain equipment objects.

6.1.16. Equipment_T

```
struct Equipment_T
  {
   globaldefs::NamingAttributes_T    name;
   string                            userLabel;
   string                            nativeEMSName;
   string                            owner;
   boolean                           alarmReportingIndicator;
   ServiceState_T                    serviceState;
   EquipmentObjectType_T             expectedEquipmentObjectType;
   EquipmentObjectType_T             installedEquipmentObjectType;
   string                            installedPartNumber;
   string                            installedVersion;
   string                            installedSerialNumber;
   globaldefs::NVSList_T             additionalInfo;
   string                            protectionType;
   string                            protectionRole;
  };
```

General comment	
The equipment object represents physical resources managed by an EMS.An equipment is contained within an equipment holder, and may not itself contain other equipment. There is only one equipment struct for identifying replaceable and non-replaceable units. For each replaceable hardware unit there shall be an equipment object. The modelling of non-replaceable units as equipment objects is optional.	
Attribute name	**Comment**
➢ name	The name represents the name of the Equipment, which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement.
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	Represents how the equipment is referred to on EMS/NE displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS/NE displays (via GUI cut through). May be a NULL string.
➢ owner	Owner may be specified by the NMS. May be empty.
➢ alarmReportingIndicator	Provides an indication whether alarm reporting for this instance is active or not. It is a read/write attribute.
➢ serviceState	Current state of the equipment.
➢ expectedEquipmentObjectType	Defines the type of expected equipment. This is an empty string if there is no expected equipment.
➢ installedEquipmentObjectType	Defines the type of installed equipment. This is an empty string if there is no installed equipment. Not supported in this version.
➢ installedPartNumber	This is the resource part number (PN) of the installed equipment. If not available (or there is no installed equipment), an empty string shall be

	used. It is a read-only attribute. Not supported in this version.
➢ installedVersion	Firmware version of the installed equipment. If not available (or there is no installed equipment), an empty string shall be used. It is a read-only attribute.
➢ installedSerialNumber	Defines the serial number of the installed equipment. If not available (or there is no installed equipment), an empty string shall be used. It is a read-only attribute.
➢ additionalInfo	Supported additional information: Ports: integer Ports indicates the equipment itself supported number of ports. SupportedPortType: string SupportedPortType indicates the Ethernet board supported port type list. Valid values are: bOther, bAUI, b10base5, bFoirl, b10base2, b10baseT, b10baseFP, b10baseFB, b10baseFL, b10broad36, b10baseTHD, b10baseTFD, b10baseFLHD, b10baseFLFD, b100baseT4, b100baseTXHD, b100baseTXFD, b100baseFXHD, b100baseFXFD, b100baseT2HD, b100baseT2FD, b1000baseXHD, b1000baseXFD, b1000baseLXHD, b1000baseLXFD, b1000baseSXHD, b1000baseSXFD, b1000baseCXHD, b1000baseCXFD, b1000baseTHD, b1000baseTFD, b10GbaseX, b10GbaseLX4, b10GbaseR, b10GbaseER, b10GbaseLR, b10GbaseSR, b10GbaseW, b10GbaseEW, b10GbaseLW, b10GbaseSW, N/A
➢protectionType	"1+1", "1:1" or "N/A"
➢protectionRole	"Primary", "Secondary" or "N/A"

6.1.17. EQTCreateData_T

```
struct EQTCreateData_T
  {
  string                          userLabel;
  boolean                         forceUniqueness;
  string                          owner;
  EquipmentObjectType_T           expectedEquipmentObjectType;
  globaldefs::NamingAttributes_T  equipmentHolderName;
  globaldefs::NVSList_T           additionalInfo;
  };
```

General comment	
Represents the data necessary to create an equipment object.	
Attribute name	**Comment**
➢ userLabel	The userLabel is provisionable by the NMS.
➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst equipments of the EMS. The operation will fail if userLabel is already in use.
➢ owner	Owner may be specified by the NMS. May be empty.
➢ expectedEquipmentObjectType	Defines the type of expected equipment.
➢ equipmentHolderName	Represents the name of equipment holder that will contain the equipment.
➢ additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information, which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile. In this case the attribute may convey vendor specific data related to equipment. This list can be empty.

6.1.18. EquipmentHolder_T

```
struct EquipmentHolder_T
{
globaldefs::NamingAttributes_T   name;
string                           userLabel;
string                           nativeEMSName;
string                           owner;
boolean                          alarmReportingIndicator;
EquipmentHolderType_T            holderType;
globaldefs::NamingAttributes_T   expectedOrInstalledEquipment;
EquipmentObjectTypeList_T        acceptableEquipmentTypeList;
HolderState_T                    holderState;
globaldefs::NVSList_T            additionalInfo;
short                            number;
};
```

General comment

Represents the physical resource of a network element that is capable of holding other physical resources. Examples of resources are equipment racks, shelves, or slots. An equipment holder object may contain a number of instances of other equipment holder objects (for instance representing slots within a shelf, or shelves within a rack), and/or a single equipment object.	
Attribute name	**Comment**
➢name	An equipment holder is identified by a unique name. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement.
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	Represents how the equipment holder is referred to on EMS/NE displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS/NE displays (via GUI cut through). May be a NULL string.
➢ owner	May be specified by the NMS. May be empty.
➢ alarmReportingIndicator	Provides an indication whether alarm reporting for this instance is active or not. It is a read/write attribute. Not supported in this version.
➢ holderType	Indicates the type of equipment holder. Always set to slot.
➢ expectedOrInstalledEquipment	The equipment object expected or installed in the equipment holder, if any. NULL if the equipment holder is empty or if it only contains other equipment holders. Not supported in this version.
➢ acceptableEquipmentTypeList	Represents the types of equipment objects that can be supported directly by the equipment holder. This is an empty list if the equipment holder can only contain other equipment holders.
➢ holderState	Represents the state of the equipment holder.
➢ additionalInfo	Supported addtional informations are: LocationName: string LocationName indicates the location of the equipmentHolder,its default value is NE's location and can be set by operation "setAdditionalInfo" in Common_I. HighestAlarmSeverity: string HighestAlarmSeverity indicates the highest alarm severity of this holder (include sub equipment/holder). Valid values are:CRITICAL, MAJOR, MINOR, WARNING, INDETERMINATE, CLEARED
➢number	Location in subrack.

6.1.19. EquipmentTypeQualifier_T

```
enum EquipmentTypeQualifier_T
{
 EQT,
 EQT_HOLDER
  };
```

Distinguishes between equipment and equipment holders.

6.1.20. EquipmentOrHolder_T

```
union EquipmentOrHolder_T switch (EquipmentTypeQualifier_T)
  {
   case EQT:            Equipment_T          equip;
   case EQT_HOLDER:     EquipmentHolder_T    holder;
  };
```

UNION equipmentOrHolder.

6.1.21. EquipmentOrHolderList_T;

```
typedef sequence<EquipmentOrHolder_T> EquipmentOrHolderList_T;
```

List of EquipmentOrHolder_T.

6.1.22. NameAndStringValue_T

```
struct NameAndStringValue_T
  {
   string name;
   string value;
  };
```

General comment
The NameAndStringValue_T structure is provided here as a replacement of the NVList construct defined by the OMG. In consideration for performance and the cost associated with the marshaling of the any type, it is decided to use the type string for the value field instead of the any type. When used for name components the structure is equivalent to the CosNaming::NameComponent structure of the Naming Service.

6.1.23. NVSList_T

```
typedef sequence<NameAndStringValue_T> NVSList_T;
```

A list of (name=string, value=string) tuples. For example, the transmission parameters of a TerminationPoint and the additional info parameters of any managed object use this structure. A standardized naming scheme is adopted between the NMS and the EMS to identify the name and the value field.

6.1.24. NamingAttributes_T

```
typedef NVSList_T NamingAttributes_T;
```

The NamingAttributes_T structure is used as a naming scheme between the NMS and EMS interface. NamingAttributes_T is used to define identifiers for managed entities that are not instantiated as first class CORBA objects and thus do not have object identifiers (IORs). The NamingAttributes represent "the hierarchical name structure" of a second-class non-CORBA object. The structure of the name is hierarchical and reflects the containment relationship between objects in a simple way.

6.1.25. NamingAttributesList_T

typedef sequence<NamingAttributes_T> NamingAttributesList_T;

A list of NamingAttributes_T. It is a list of lists.

6.1.26. Time_T

typedef string Time_T;

Time_T is represented by a string holding a time string as defined in ITU-T Rec. X.208 "SPECIFICATION OF ABSTRACT SYNTAX NOTATION ONE (ASN.1)".

The format is "yyyyMMddhhmmss.s[Z|{+|-}HHMm]" where:

Yyyy	"0000"..	"9999"	year
MM	"01"..	"12"	month
Dd	"01"..	"31"	day
hh	"00"..	"23"	hour
mm	"00"..	"59"	minute
ss	"00"..	"59"	second
.s	".0"..	".9"	tenth of second (set to ".0" if EMS or ME cannot support this granularity)
Z	"Z"		indicates UTC (rather than local time)
{+|-}	"+" or	"-"	delta from UTC
HH	"00"..	"23"	time zone difference in hours
Mm	"00"..	"59"	time zone difference in minutes

"19851106210627.3Z" would be 6 minutes, 27.3 seconds after 9 p.m. on November 6th, 1985 indicating UTC time. "19851106210627.3" would be local time. "19851106210627.3+0500" would be local time specifying a +5 hour time difference from UTC. "19851106210627.3-0530" would be local time specifying a -5.5 hour difference from UTC.

6.1.27. ConnectionDirection_T

enum ConnectionDirection_T

```
{
 CD_UNI,
 CD_BI
};
```

General comment	
Direction of a subnetwork connection, cross-connects, or topological link.	
Attribute name	**Comment**
➢ CD_UNI	UNIdirectional, i.e. source TP to sink TP Note: creation of unidirectional connections is supported even when CTPs/FTPs are modelled as bidirectional.
➢ CD_BI	BIdirectional, i.e. bidirectional TP to bidirectional TP, a.k.a. two-way.

6.1.28. ExceptionType_T

enum ExceptionType_T

```
{
 EXCPT_NOT_IMPLEMENTED,
```

EXCPT_INTERNAL_ERROR,
EXCPT_INVALID_INPUT,
EXCPT_OBJECT_IN_USE,
EXCPT_TP_INVALID_ENDPOINT,
EXCPT_ENTITY_NOT_FOUND,
EXCPT_TIMESLOT_IN_USE,
EXCPT_PROTECTION_EFFORT_NOT_MET,
EXCPT_NOT_IN_VALID_STATE,
EXCPT_UNABLE_TO_COMPLY,
EXCPT_NE_COMM_LOSS,
EXCPT_CAPACITY_EXCEEDED,
EXCPT_ACCESS_DENIED,
EXCPT_TOO_MANY_OPEN_ITERATORS,
EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS,
EXCPT_USERLABEL_IN_USE
};

General comment	
Exception Definitions. As per CORBA policies agreement, only one exception object is defined to capture all of the possible exceptions defined in the ProcessingFailureException.	
Attribute name	**Comment**
➢ EXCPT_NOT_IMPLEMENTED	If some IDL operations are optional or not implemented in this release, then this value may be used for this purpose. If the operation itself is not supported, then errorReason shall be an empty string. If this exception is raised because of the values of specific parameters, then the names of these parameters shall be supplied in errorReason (separated by commas), unless otherwise specified in the operation description.
➢ EXCPT_INTERNAL_ERROR	To indicate an EMS internal error. Applies to all methods.
➢ EXCPT_INVALID_INPUT	If the format of a parameter is incorrect, e.g. if a TP name which is a 3 level namingAttribute is passed as a single level name, then this type will be used. Also if a parameter is out of range, this type will be used. The reason field will be filled with the parameter that was incorrect.
➢ EXCPT_OBJECT_IN_USE	To indicate an object already in use.
➢ EXCPT_TP_INVALID_ENDPOINT	To indicate that the specified TP does not exist or cannot be created. (e.g., attempt to create a VPL TP using an out of range VPI value). Note that if the TP is valid but is already terminated & mapped or cross- connected then EXCPT_OBJECT_IN_USE must be returned.
➢ EXCPT_ENTITY_NOT_FOUND	In general, if the NMS supplies an object name as a parameter to an operation and the EMS can not find the object with the given name then an exception of this type is returned. The reason field in the exception will be filled with the name that was passed in as parameter.
➢ EXCPT_TIMESLOT_IN_USE	To indicate a timeslot already in use when creating or activating an SNC.
➢ EXCPT_PROTECTION_EFFORT_NOT_MET	If the NMS requests an SNC with a protection effort that cannot be met by the EMS.

➢ EXCPT_NOT_IN_VALID_STATE	Used if the client tries to delete an active SNC for example.
➢ EXCPT_UNABLE_TO_COMPLY	The value EXCPT_UNABLE_TO_COMPLY value is used as a generic value when a server cannot respond to the request.
➢ EXCPT_NE_COMM_LOSS	The value EXCPT_NE_COMM_LOSS value is used as a generic value when a server cannot communicate with the NE and that prevents the successful completion of the operation. All operations that involve communication with the NE may throw this particular exception type.
➢ EXCPT_CAPACITY_EXCEEDED	Raised when an operation will result in resources being created or activated beyond the capacity supported by the NE/EMS.
➢ EXCPT_ACCESS_DENIED	Raised when an operation results in a security violation.
➢ EXCPT_TOO_MANY_OPEN_ITERATORS	Raised when an EMS exceeds its internal limit of the number of iterators it can support.
➢ EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS	Raised when an EMS does not support the routing constraints specified as input
➢ EXCPT_USERLABEL_IN_USE	Raised when the userLabel uniqueness constraint can not be met.

6.1.29. ProcessingFailureException

```
exception  ProcessingFailureException
  {
   ExceptionType_T exceptionType;
   string errorReason;
  };
```

General comment	
A coarse grain approach is adopted for capturing exceptions as well. This has the advantage of making the catching of exceptions fairly generic. Since CORBA does not allow as in the Java language to subclass exceptions, it is recommended to reduce the number of exceptions a client may catch. On the down side, a client may need to write explicit code when an exception is thrown by the server (i.e. a client may have a switch statement on the ExceptionType parameter).	
Attribute name	**Comment**
➢ exceptionType	See ExceptionType
➢ errorReason	A string indicating further details about the exception. It is a free format string filled by the EMS Server.

6.1.30. ServerLaunchCapability_T

```
enum ServerLaunchCapability_T
  {
    CLIENT_LAUNCH_ONLY,
    SERVER_LAUNCH_CAPABLE
  };
```

General comment
Describes the capability for server launch; either the EMS supports only a client launch, or the EMS supports both a client launch and a server launch.

Attribute name	Comment
➢ CLIENT_LAUNCH_ONLY	The EMS supports only a client launch.
➢ SERVER_LAUNCH_CAPABLE	EMS supports server launch.

6.1.31. GuiCutThroughData_T

```
struct GuiCutThroughData_T
  {
    string gctScope;
    string gctContext;
    string gctCommand;
    globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
This struct is used to represent the GCT window data for one window. The guiCutThroughData_T provides all the information needed by the NMS to launch a specific GUI Cut-Through window outside of the EMS-NMS interface. The information in the guiCutThroughDataList attribute of a GCT profile information represents all the windows supported by the EMS. This is a readonly attribute, i.e. the NMS cannot configureGCT launch data but retrieve the GCT capabilities of the EMS.	
Attribute name	**Comment**
➢ gctScope	Scope of the window: "EMS", "ME". Only support of the "EMS" scope is mandatory.
➢ gctContext	Context of the window. Valid values are: "TOP_LEVEL" "FAULT" "CONFIGURATION_SOFTWARE" "CONFIGURATION_CONNECTION" "ACCOUNTING" "PERFORMANCE" "SECURITY" "SYSTEMS_MANAGEMENT" Only support of the "TOP_LEVEL" context is mandatory.
➢ gctCommand	The gctCommand contains the exact command needed by the EMS to launch its GUI so that the window for that EMS is launched. This string shall not be empty. Placeholders are used to indicate where values should be replaced by the NMS when performing the client launch.
➢ additionalInfo	This is a vendor specific attribute which contains additional specific information required for the GCT.

6.1.32. GuiCutThroughDataList_T

```
typedef sequence<GuiCutThroughData_T> GuiCutThroughDataList_T;
```

Set of GuiCutThroughData_T.

6.1.33. GCTProfileInfo_T

```
struct GCTProfileInfo_T
  {
    ServerLaunchCapability_T serverLaunchCapability;
    string gctHostname;
    string emsGctPlatform;
    GuiCutThroughDataList_T guiCutThroughDataList;
  };
```

General comment	
This struct is used to represent the GCT profile data for the EMS.	
Attribute name	**Comment**
➢ serverLaunchCapability	Indicates whether the EMS supports the server launchGCT operation. The default is to support client launch only.
➢ gctHostname	IP address (or host name which is mapped to the appropriate IP address) that allows the NMS to identify where to launch the EMS GUI client (NMS display). This string can be empty only in case the attribute emsGctPlatform is "local" or "web-based". For Citrix commands, this is the TSE (Terminal Server Environment) IP address.
➢ emsGctPlatform	Indicates the platform supported for the GUI Cut-Through Valid values are:"unix","windowsNT","local","web-based".
➢ guiCutThroughDataList	Indicates the supported window types and how to lauch them.

6.1.34. MaintenanceOperation_T

```
typedef string MaintenanceOperation_T;
```

Supported maintenance operations. Valid values are:
"FACILITY_LOOPBACK"
"TERMINAL_LOOPBACK"
"FACILITY_FORCED_AIS"
"TERMINAL_FORCED_AIS"
"FORCE_RDI"
"SET_AS_SEGMENT_END_POINT" (ATM)
"END_TO_END_LOOPBACK_OAM_CELL" (ATM)
"SEGMENT_LOOPBACK_OAM_CELL" (ATM)
"LOCAL_LOOP_QUALIFICATION" (DSL)
"DSL_LINE_SUPERVISION" (DSL)

6.1.35. MaintenanceOperationMode_T

```
enum MaintenanceOperationMode_T {
    MOM_OPERATE,
    MOM_RELEASE
  };
```

Describes the mode of the operation, i.e. operate the maintenance operation or release the maintenance operation.

6.1.36. CurrentMaintenanceOperation_T

```
struct CurrentMaintenanceOperation_T {
    globaldefs::NamingAttributes_T tpName;
    MaintenanceOperation_T maintenanceOperation;
    transmissionParameters::LayerRate_T layerRate;
    globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
This struct is used to represent a current persistent maintenance operation that has been invoked for the given TP, and layerRate if applicable.	
Attribute name	**Comment**
➢ tpName	The name of the TP to which the maintenance operation applies.The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ maintenanceOperation	Current maintenance operation that is invoked,released, or retrieved.
➢ layerRate	The layer to which the maintenance operation applies. Use LR_Optional if not required or not applicable.
➢ additionalInfo	Additional information on the maintenance operation, subject to bilateral agreement.

6.1.37. CurrentMaintenanceOperationList_T

```
typedef sequence<CurrentMaintenanceOperation_T> CurrentMaintenanceOperationList_T
```

Sequence of CurrentMaintenanceOperation_T.

6.1.38. CommunicationState_T

```
enum CommunicationState_T
  {
    CS_AVAILABLE,
    CS_UNAVAILABLE
  };
```

. This state reflects a communication state between the EMS and its ManagedElement.
The NMS user will have to go to the EMS to determine the exact reasons as to why the ManagedElement is unavailable.

6.1.39. ManagedElement_T

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

General comment	
The managedElement represents an abstraction of a set of co-located physical resources managed as a single entity by an EMS.	
Attribute name	**Comment**
➢ name	The name represents the name of the Managed Element, which is assigned by the EMS upon creation.The EMS is responsible for guaranteeing the uniqueness of the name. It is a readonly attribute.
➢ userLabel	The userLabel is a friendly name that the operator wants to place for the managedElement. Typical expectations of the operator is that the same name is seen on all operation systems. This is set by the NMS and could be displayed on the EMS (and/or the network element) based on each systems'capabilities. THIS IS NOT A MANDATORY EXPECTATION, but is left to the implementation of the EMS/ManagedElement. This is one way of implementing certain acronyms/naming convention that the operator may want to impose on all the systems. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	The name represents how the managedElement is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through). This is never set to a null string. If supported by the EMS, this attribute can be set by NMS through the Common_I interface service common::Common_I::setNativeEMSName(). It is a read/write attribute.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢location	The location is the geographical location of the Managed Element.This is a free format string and can be an empty string. The attribute has no impact on the operation of the system. It is a readonly attribute.
➢version	The active software version of the ManagedElement.This attribute is defined as a free format string with no semantics to allow for different vendor implementation. It is a readonly attribute.
➢productName	The productName identifies the managed element product/type name. This attribute is defined as a non-empty free format string with no semantics to allow for different vendor implementation. It is a readonly attribute.
➢communicationState	The communicationState of the ManagedElement describes the viability of EMS-ME messaging. It is a readonly attribute.
➢emsInSyncState	Indicates if the EMS is able to keep the current EMS data synchronized with the current NE data and generate all appropriate notifications. The EMS sets this attribute to false to indicate that it requires resynchronization with NE data and that it is not able to generate the appropriate notifications (such as OCs/ODs/AVCs) while doing so. The EMS sets this attribute back to true when the resynchronization is completed and when notifications can start being generated as appropriate.
➢supportedRates	This attribute is a list (possibly empty) of potential Cross Connection Rates at which it is possible to have cross-connections within the managed element.It is a readonly

	attribute.
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile. This may be an empty list This attribute can be set by the NMS through the Common_I interface service common::Common_I::setAdditionalInfo(). It is a read/write attribute.
➢manufacturer	This attribute presents the name of manufacturer of this product. It is a read only attribute.

6.1.40. ManagedElementList_T

typedef sequence<ManagedElement_T> ManagedElementList_T;

.

Sequence of ManagedElement_T.

6.1.41. Topology_T

```
enum Topology_T
  {
   TOPO_SINGLETON,
   TOPO_CHAIN,
   TOPO_PSR,
   TOPO_OPEN_PSR,
   TOPO_SPRING,
   TOPO_OPEN_SPRING,
   TOPO_MESH
  };
```

General comment	
Topology_T is used to describe the subnetwork configuration. The EMS shall be capable of providing a SubNetwork Connection through any physical termination point of any managed element that belongs to the Subnetwork.	
Attribute name	**Comment**
➢ TOPO_SINGLETON	TOPO_SINGLETON, which is used for a single NE (of any type) that is managed independently of its Topological Link connectivity to other NEs. It may for example be a member of a ring that is managed by a number of EMSes. It is acceptable for an EMS to represent all NEs as being in Singleton subnetworks regardless of the actual network configuration. A singleton subnetwork does not contain internal topological links.
➢ TOPO_CHAIN	TOPO_CHAIN, which is used to cover the case where two or more Nes are managed by the same EMS and are connected by Topological Links in a chain.
➢ TOPO_PSR	TOPO_PSR, which is used to cover the case where two or more Nes are managed by the same EMS and are connected by Topological Links in a ring that is capable of supporting subnetwork connection protection.
➢ TOPO_OPEN_PSR	TOPO_OPEN_PSR, which is used to cover the case where two or more Nes of a PS ring (but not the entire ring) are managed by the same EMS.
➢ TOPO_SPRING	TOPO_SPRING, which is used to cover the case where two or more Nes are managed by the same EMS and are connected by Topological Links in a complete ring that supports Shared Line Protection.

➢ TOPO_OPEN_SPRING	TOPO_OPEN_SPRING, which is used to cover cases where two or more NEs of an SP ring (but not the entire ring) are managed by one EMS.
➢ TOPO_MESH	TOPO_MESH, which is used to cover an arbitrary set of two or more Nes not covered by any other type. Composite subnetworks (i.e. containing other Subnetworks) are not supported in this release of the interface.

6.1.42. EMSFreedomLevel_T

```
enum EMSFreedomLevel_T
  {
   EMSFL_CC_AT_SNC_LAYER,
   EMSFL_TERMINATE_AND_MAP,
   EMSFL_HIGHER_ORDER_SNCS,
   EMSFL_RECONFIGURATION
  };
```

General comment	
Describes the NMS-specified EMS level of freedom when performing SNC operations.	
Attribute name	**Comment**
➢ EMSFL_CC_AT_SNC_LAYER	The EMS is allowed to create or delete cross- connections,at the layer of the SNC only , that are or will be directly used by it.
➢ EMSFL_TERMINATE_AND_MAP	In addition to EMSFL_CC_AT_SNC_LAYER, the EMS is allowed to terminate and map or unmap and unterminate CTPs to generate or eliminate CTPs that are or will be used by the SNC.
➢ EMSFL_HIGHER_ORDER_SNCS	In addition to EMSFL_TERMINATE_AND_MAP, the EMS is allowed to create or delete higher order SNCs that are or will be used to carry the SNC.
➢ EMSFL_RECONFIGURATION	The EMS is allowed to perform any operation that it considers relevant, which includes reorganizing any SNC or TP to allow the creation or activation of the SNC or to make the subnetwork more efficient.

6.1.43. MultiLayerSubnetwork_T

```
struct MultiLayerSubnetwork_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   Topology_T subnetworkType;
   transmissionParameters::LayerRateList_T supportedRates;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment

The MultiLayerSubnetwork structure is the abstraction offered by the EMS to the NMS to represent a Subnetwork that is managed by the EMS system. In this document specification, Subnetwork and MultiLayerSubnetwork are used interchangeably. It represents a logical grouping or partitioning of the managed elements in a way that is entirely determined by the EMS. A managed element may belong to more than one subnetwork, at different layer rates (e.g. SDH & ATM). However, subnetworks cannot overlap at the same layer rate. The NMS does not create or delete Subnetworks, they are managed by the EMS. The NMS has a handle to the Subnetworks managed by the EMSes and can request the establishment or the removal of connections within subnetworks. In this interface specification, the way the NMS requests services on the Subnetworks, including establishment and removal of subnetwork connections,is through the MultiLayerSubnetworkMgr_I.	
Attribute name	**Comment**
➢ name	The name represents the name of the MultiLayerSubNetwork which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the MultiLayerSubnetworkMgr_I. It is a readonly attribute.
➢ userLabel	The user label is seldom used on a singleton. However for rings and meshes, the operator may choose to assign some logical name for the subnetwork and the EMS may choose to display it on the GUI, so that the operator has a consistent view of the managed network. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	This name of the subnetwork on the EMS GUI.The nativeEMSName is defaulted to a NULL string. However, this could be used by the EMS for its implementation dependent purpose.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ subnetworkType	The subnetworkType gives a coarse view of the topology of the subnetwork. It is a readonly attribute.
➢ supportedRates	This attribute is a list (possibly empty) of potential Cross Connection Rates at which it is possible to make SNCs within the subnetwork. It is a readonly attribute.
➢ additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. This may be an empty list. It is a readonly attribute.

6.1.44. SubnetworkList_T

typedef sequence<MultiLayerSubnetwork_T> SubnetworkList_T;

Sequence of MultiLayerSubnetwork_T objects.

6.1.45. TPPoolCreateData_T

```
struct TPPoolCreateData_T
  {
   string userLabel;
   boolean forceUniqueness;
   string owner;
   multiLayerSubnetwork::MultiLayerSubnetwork_T containingMLSN;
   globaldefs::NamingAttributesList_T containedMembers;
   transmissionParameters::LayeredParameterList_T transmissionParams;
```

```
 string descriptionOfUse;
 globaldefs::NVSList_T additionalCreationInfo;
};
```

General comment	
The attributes required for the creation of a termation point pool on the EMS are packaged together in a TPPoolCreateData structure which the NMS will pass to the EMS at TPPool creation time.These are the read-create attributes of the TPPool.	
Attribute name	**Comment**
➢ userLabel	UserLabel may be specified by the NMS. May be empty.
➢ forceUniqueness	Specifies whether uniqueness of userLabel is required amongst TPPools of the prescribed MLSN. The operation will fail if userLabel is already in use.
➢ owner	The owner may be specified by the NMS. May be empty.
➢ containingMLSN	The subnetwork that shall contain the TPPool to be created.
➢ containedMembers	The list of TPs or GTPs that shall comprise the TPPool.
➢ transmissionParams	The common layers and transmission parameters the above-specified Contained TPs, or TPs contained in Contained GTPs, are required to have (e.g., ATM VP layer with prescribed traffic characteristics).
➢ descriptionOfUse	A description of the specific use of the TP pool, in particular how its members are collected and administered. It could include, for example, the name of a TP with the meaning that all TP pool members have to be collected from the set of potential client CTPs of this TP.
➢additionalCreationInfo	Some additional creation information may be specified by the NMS. This information may or may not become a part of the TPPool's additionalInfo attribute. The list may be empty.

6.1.46. ProtectionSchemeState_T

```
enum ProtectionSchemeState_T
 {
  PSS_UNKNOWN,
  PSS_AUTOMATIC,
  PSS_FORCED_OR_LOCKED_OUT
 };
```

The protection scheme state identifies the state in which the protection scheme is in. PSS_FORCED_OR_LOCKED_OUT indicates that the entire group is locked; partial locking is indicated by PSS_AUTOMATIC.Individual locks can be reported through transmission parameters on the appropriate TPs or equipment instances.

6.1.47. ProtectionType_T

```
enum ProtectionType_T
 {
  PT_MSP_APS,
  PT_SNCP
 };
```

The protection type identifies whether a protection switch is an MS protection switch or an SNCP protection switch.It should be noted that although the term MSP was chosen as the original specific protection scheme to which the related behaviour applied was Multiplex Section Protection, the label is now more generally applied to any 1+1 or 1:N Trail protection scheme.

6.1.48. SwitchReason_T

```
enum SwitchReason_T
  {
   SR_NA,
   SR_RESTORED,
   SR_SIGNAL_FAIL,
   SR_SIGNAL_MISMATCH,
   SR_SIGNAL_DEGRADE,
   SR_AUTOMATIC_SWITCH,
   SR_MANUAL
  };
```

General comment	
The switch reason reflects the reason why a switch occurred.	
Attribute name	**Comment**
➢ SR_NA	SR_NA is used upon retrieval of switch data for non-revertive groups, if a more precise value is not available.
➢ SR_RESTORED	SR_RESTORED is used for revertive groups to indicate a return to the normal state.
➢ SR_SIGNAL_MISMATCH	SR_SIGNAL_MISMATCH is used in the case the signal is ok, but is identified as coming from an incorrect source: TRAIL_TRACE_IDENTIFIER_MISMATCH, Signal Label Mismatch, etc.
➢ SR_AUTOMATIC_SWITCH	SR_AUTOMATIC_SWITCH is used when the exact switch reason is unknown, in retrievals of switch data if a protection switch is currently active or in protection switch notifications.
➢ SR_MANUAL	SR_MANUAL indicates a switch that was requested by the operator and includes forced switches.

6.1.49. ESwitchReason_T

```
typedef string ESwitchReason_T;
```

The equipment switch reason reflects the reason why a switch occurred. EswitchReason is a string that can take the following values."SR_NA" is used, if a more precise value is not available.
"SR_E_FAILURE" is used when an instance of equipment has failed. "SR_MANUAL" indicates a switch that was requested by the operator and includes forced switches.

6.1.50. ProtectionCommand_T

```
enum ProtectionCommand_T
  {
   PC_CLEAR,
   PC_LOCKOUT,
   PC_FORCED_SWITCH,
   PC_MANUAL_SWITCH,
   PC_EXERCISER
  };
```

This type identifies the possible protection switch commands.See ITU-T Recommendation G.841 for definitions.

6.1.51. ProtectionGroupType_T

```
enum ProtectionGroupType_T
   {
   PGT_MSP_1_PLUS_1,
```

```
 PGT_OSNCP_1_PLUS_1,
 PGT_MSP_1_FOR_N,
 PGT_2_FIBER_BLSR,
 PGT_4_FIBER_BLSR
};
```

The protection group type identifies the type of the protection Group.It should be noted that although the term MSP was chosen as the original specific protection scheme to which the related behaviour applied was Multiplex Section Protection, the label is now more generally applied to any 1+1 or 1:N Trail protection scheme.

To indicate the Alcatel WDM OSNCP protection group, PGT_OSNCP_1_PLUS_1 is added to extend the standard.

6.1.52. EProtectionGroupType_T

typedef string EProtectionGroupType_T

The equipment protection group type is a string that identifies the type of equipment protection. The string can take on the following value:

"M_FOR_N."

M: N equipment protection means that M pieces of equipment protect N pieces of equipment.

6.1.53. ReversionMode_T

```
enum ReversionMode_T
  {
   RM_UNKNOWN,
   RM_NON_REVERTIVE,      // i.e. non revertive
   RM_REVERTIVE // when the failed resource is repaired,
              // it reverts to the preferred resource.
  };
```

Reversion mode is used to indicate whether, after repair of a failed resource, an additional switch should be made to revert to the preferred resource. Revertive modes may require a wait to restore (WTR) time setting.

6.1.54. ProtectionGroup_T

```
struct ProtectionGroup_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   ProtectionGroupType_T protectionGroupType;
   ProtectionSchemeState_T protectionSchemeState;
   ReversionMode_T reversionMode;
   transmissionParameters::LayerRate_T rate;
   globaldefs::NamingAttributesList_T pgpTPList;
   globaldefs::NVSList_T pgpParameters;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
The struct ProtectionGroup_T represents a protection group, which is used to model any 1+1 or 1:N Trail protection (for example MS layer protection in an MSSPRING).	
Attribute name	**Comment**
➢ name	The name represents the name of the Protection Group which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement.

	It is a readonly attribute.
➢ userLabel	The userLabel is provisionable by the NMS and would typically represent the ring-id of a ring. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	The native name of the protection group is the AID of the protection group itself. In case of the 4-Fiber BLSR (MSSPRING) group, the managed element usually has only one group, whereas it is represented as three groups in this interface. Therefore the native name of the 4-Fiber group will be the AID of the 4-Fiber group. The native name of the component APS (MSP) groups is set to some useful name chosen by the EMS. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through). This is never set to a null string. It should be noted that although the term MSP was chosen as the original specific protection scheme to which the related behaviour applied was Multiplex Section Protection, the label is now more generally applied to any 1+1 or 1:N Trail protection scheme.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ protectionGroupType	Defines the type of scheme this group represents (e.g. 2f,4f blsr etc). It is a readonly attribute.
➢protectionSchemeState	Identifies the current protection scheme state. It is a readonly attribute.
➢reversionMode	Defines whether the protection scheme is revertive or not. It is a readonly attribute.
➢rate	Line rate of the PTPs/FTPs of the protection group. It is a readonly attribute.
➢pgpTPList	This is the list of TPs that belong in the protection group.The list of TPs is partially ordered. The protecting TP always trails its worker TPs. The East TPs are always contiguous in the list, as well as West TPs.
➢pgpParameters	The pgpParameters contains a name value list for the known parameters of the protection group. Non-applicable parameters and parameters for which the value is unknown to the EMS may be left out. It is a readonly attribute.
➢additionalInfo	WaitToRestoreTime: integer This information indicates the waitToRestoreTime in seconds. AlarmDuration: integer This information indicates the alarm duration time in ms.

6.1.55. ProtectionGroupList_T;

typedef sequence <ProtectionGroup_T> ProtectionGroupList_T;

Sequence of ProtectionGroup_T.

6.1.56. EProtectionGroup_T

```
struct EProtectionGroup_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   EProtectionGroupType_T eProtectionGroupType;
```

```
  ProtectionSchemeState_T protectionSchemeState;
  ReversionMode_T reversionMode;
  globaldefs::NamingAttributesList_T protectedList;
  globaldefs::NamingAttributesList_T protectingList;
  globaldefs::NVSList_T ePgpParameters;
  globaldefs::NVSList_T additionalInfo;
};
```

General comment	
The struct EProtectionGroup_T represents an equipment protection group, which is used to model equipment protection.	
Attribute name	**Comment**
➢ name	The name represents the name of the Equipment Protection Group, which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. It is a readonly attribute.
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	The native name of the equipment protection group.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ eProtectionGroupType	Defines the type of scheme this group represents (so far, only M:N equipment protection has been identified). It is a readonly attribute.
➢ protectionSchemeState	Identifies the current protection scheme state. It is a readonly attribute.
➢ reversionMode	Defines whether the protection scheme is revertive or not. It is a readonly attribute.
➢ protectedList	This is provides a list of the protected equipment instances. For M: N equipment protection, this list would have N elements.
➢ protectingList	This is provides a list of the protecting equipment instances. For M:N equipment protection, this list would have M elements.
➢ ePgpParameters	The epgpParameters contains a name value list of the associated parameters for the equipment protection group. No epgpParameters have been identified. It is a readonly attribute.
➢ additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information, which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile. This may be an empty list.

6.1.57. EProtectionGroupList_T

typedef sequence <EProtectionGroup_T> EProtectionGroupList_T;

Sequence of EProtectionGroup_T.

6.1.58. SwitchData_T

```
struct SwitchData_T
  {
  ProtectionType_T protectionType;
  SwitchReason_T switchReason;
  transmissionParameters::LayerRate_T layerRate;
```

```
  globaldefs::NamingAttributes_T groupName;
  globaldefs::NamingAttributes_T protectedTP;
  globaldefs::NamingAttributes_T switchToTP;
  globaldefs::NVSList_T additionalInfo;
};
```

General comment	
This structure is used to respond to queries regarding the current protection switch status of a protection group or an SNC.	
Attribute name	**Comment**
➢ protectionType	The type of protection.
➢ switchReason	The reason the last switch occurred.
➢ layerRate	The layer which this switch is relevant to.
➢ groupName	Identifies the protectionGroup for which protection switch status is being reported. NULL if protectionType is SNCP.
➢ protectedTP	This is the TP that is protected. The protected (worker) and protecting TPs are fixed in a revertive protection group, the worker can be active or not at any point in time. In a non-revertive protection group, there is no fixed worker/protecting distinction. The worker TP is indeed always the active TP; after a switch, the worker and protecting TPs exchange their roles. For an SNCP, this is always the reliable TP. For a retrieval of a 2F BLSR, each TP is protected, and two SwitchData_T are returned.For a retrieval of a 4FMSSPR, each worker TP is protected, and two SwitchData_T are returned. For a retrieval of a 1:N MSP, each worker TP is protected, and N SwitchData_T are returned. For a revertive 1+1 MSP, this is always the worker TP. For a retrieval of a non-revertive 1+1 MSP switch, this is the active TP. Note: The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ switchToTP	Identifies the TP, which is being switched to. This identifies the TP that is the active source after the switch, or currently active if no protection switch is currently active. Note: The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information, which is not explicitly modelled. This may be an empty list.

6.1.59. SwitchDataList_T

typedef sequence<SwitchData_T> SwitchDataList_T;

Sequence of SwitchData_T.

6.1.60. ESwitchData_T

```
struct ESwitchData_T
  {
   EProtectionGroupType_T eProtectionGroupType;
   ESwitchReason_T eSwitchReason;
   globaldefs::NamingAttributes_T ePGPName;
   globaldefs::NamingAttributes_T protectedE;
   globaldefs::NamingAttributes_T switchToE;
   globaldefs::NVSList_T additionalInfo;
```

```
};
```

General comment	
This structure is used to respond to queries regarding the current protection switch status of an equipment protection group.	
Attribute name	**Comment**
➢ eProtectionGroupType	The type of protection.
➢ eSwitchReason	The reason that the last switch occurred.
➢ ePGPName	Identifies the equipment protection group for which protection switch status is being reported.
➢ protectedE	This attributes identifies the protected equipment. For a retrieval of an M: N group, protectedE always identifies a worker equipment instance. In this case, N ESwitchData_T is returned as a result of retrieveESwitchData (one for each worker equipment instance).
➢ switchToE	This identifies the equipment instance that is working after the switch, or currently working if no protection switch is currently active.
➢ additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information, which is not explicitly modelled. This may be an empty list.

6.1.61. ESwitchDataList_T

typedef sequence<ESwitchData_T> ESwitchDataList_T;

Sequence of ESwitchData_T.

6.1.62. BackupId_T

```
struct BackupId_T {
   globaldefs::NamingAttributes_T meName ;
   globaldefs::Time_T backupTime;
  };
```

General comment	
This data structure provides an identifier for a backup on an EMS. The ME name and the time when the backup was taken uniquely identifies the backup. The EMS may decide to ignore the seconds/sub-seconds field in the Time_T parameter. The EMS is expected to administer time adjustments and ME name changes in such a way as to ensure that no two backups have the same identifier.	
Attribute name	**Comment**
➢ meName	The name of the Managed Element from which the backup was taken.
➢ backupTime	The time at which the backup was taken.

6.1.63. BackupIdList_T

typedef sequence <BackupId_T> BackupIdList_T;

This is a list of Database backups.

6.1.64. Current_OperationStatus_T

```
enum Current_OperationStatus_T {
   COS_Idle,
   COS_Pending,
   COS_InProgress,
   COS_Completed,
   COS_Aborted
  };
```

General comment	
This enum identifies the status of a managed element with respect to current database backup operation. Initially when the EMS is started the Operational status will be set to COS_Idle.	
Attribute name	**Comment**
➢ COS_Idle	No database backup operation has been performed since EMS last (re)started (boot).
➢ COS_Pending	A backup operation has been requested but has not yet started.
➢COS_InProgress	A backup operation is being performed.
➢COS_Completed	Last backup operation was successful.
➢COS_Aborted	Last backup operation failed.

6.1.65. BackupStatus_T

```
struct BackupStatus_T {
   Current_OperationStatus_T opStatus;
   string failureReason;
  };
```

General comment	
This data structure identifies the status of backup operation for a managed element. The failure reason should be present if the operation status indicates a failure i.e. in Abort state.	
Attribute name	**Comment**
➢ opStatus	Indicates the current operational status of the backup.
➢ failureReason	A free form text string provided if the opStatus value is COS_Aborted to explain the reason for the abort (e.g. "Comms loss with NE".

6.1.66. StaticProtectionLevel_T

```
enum StaticProtectionLevel_T
  {
   PREEMPTIBLE,
   UNPROTECTED,
   PARTIALLY_PROTECTED,
   FULLY_PROTECTED,
   HIGHLY_PROTECTED
  };
```

The static protection level is a statement of the internal resiliency of the SNC (internal to the subnetwork). The more resilient an SNC is, the more bandwidth it will consume.
The protection level does not have any bearing on the externally visible shape and traffic flows of the SNC (in non-failure cases).

6.1.67. ProtectionEffort_T

```
enum ProtectionEffort_T
  {
   EFFORT_WHATEVER,
   EFFORT_SAME_OR_BETTER,
   EFFORT_SAME_OR_WORSE,
   EFFORT_SAME
  };
```

The protection effort is a statement of the requirement of the static protection level. For example if EFFORT_SAME_OR_WORSE is specified for a 3-ended FULLY_PROTECTED connection, a 3-ended PARTIALLY_PROTECTED connection is acceptable, but a 2-ended connection (ST_SIMPLE) is not acceptable i.e. ProtectionEffort_T does not have any bearing on the externaly visible shape of the SNC. EFFORT_WHATEVER indicates that the specified static protection level is preferred, but that any other level is acceptable.

6.1.68. SNCState_T

```
enum SNCState_T
  {
   SNCS_NONEXISTENT,
   SNCS_PENDING,
   SNCS_ACTIVE,
   SNCS_PARTIAL
  };
```

General comment	
The SNCState enum is used to represent the various states that an SNC may take. The following states may be supported by the EMS, depending on the SNC management mode of operation used by the EMS.	
Attribute name	**Comment**
➢ SNCS_PENDING	The SNC has been created by an NMS and has not been activated by any NMS; or the SNC has been successfully deactivated by an NMS. That state has no relationship with the network state of the cross-connects of the SNC. It is allowable for an EMS to not support the SNCS_PENDING state and to reject any operation that attempts to put an SNC into SNCS_PENDING state.
➢ SNCS_ACTIVE	The SNC is not in pending state, a route has been assigned to the SNC and all cross-connects for the SNC are active in the network.
➢SNCS_PARTIAL	The SNC is not in pending state, and either a route has not been assigned to the SNC, or not all of the cross-connects of the SNC are active in the network. This may or may not include activated SNCs for which there are currently no active cross-connects in the network, depending on the SNC management mode of operation.It is possible that, in some EMSes, this state be unreachable.
➢SNCS_NONEXISTENT	This is not an SNC state per se, as it applies to "non-existent SNCs". It is used in the interface to report SNCs that have been deleted.

6.1.69. GradesOfImpact_T

```
enum GradesOfImpact_T
  {
   GOI_HITLESS,
   GOI_MINOR_IMPACT,
   GOI_MAJOR_IMPACT
```

```
};
```

Grades of maximum tolerable disruption to traffic as a result of the operation that this parameter is describing. The following relates to design intent: GOI_HITLESS, GOI_MINOR_IMPACT <= 50ms,GOI_MAJOR_IMPACT > 50ms.

6.1.70. TPData_T

```
struct TPData_T
  {
   globaldefs::NamingAttributes_T tpName;
   terminationPoint::TerminationMode_T tpMappingMode;
   transmissionParameters::LayeredParameterList_T transmissionParams;
   globaldefs::NamingAttributes_T ingressTrafficDescriptorName;
   globaldefs::NamingAttributes_T egressTrafficDescriptorName;
  };
```

General comment	
The TPData struct contains termination point data that is settable by the NMS.	
Attribute name	**Comment**
➢ tpName	The name of the termination point to which this data applies. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ tpMappingMode	The mapping mode to put the TP in.
➢ transmissionParams	When used as input to the EMS, this is a "delta" list that needs to be applied to the specified TP. Only a subset of the parameters may be specified in the list, and only those should be applied in the NE. Note that this could be used to convey the pointer to the alarm severity assignment profile during SNC creation/activation. If the list is empty then this means do nothing. To remove a parameter from the list, "-" should be specified in the value part of the structure. When a BLSR connection is created, the SPRING_NodeId parameter should be provided for the aEnd and zEnd CTPs/FTPs.
➢ ingressTrafficDescriptorName	A connection termination point may have an optional reference to an ingress (incoming) Traffic Descriptor or Transmission Descriptor. The Descriptor name will be empty if there is no associated Descriptor. * It is expected that Traffic Descriptors and Transmission Descriptors are not mixed on a single TP. Therefore if the Ingress Descriptor name is that of a Traffic Descriptor then the Egress Descriptor name should either be that of a Traffic Descriptor or a null value. Likewise if the Ingress Descriptor name is that of a Transmission Descriptor then the Egress Descriptor name should either be that of a Trasmission Descriptor or a null value.
➢ egressTrafficDescriptorName	A connection termination point may have an optional reference to an egress (outgoing)Traffic Descriptor or Transmission Descriptor. The Traffic Descriptor name will be empty if there is no associated Descriptor. It is expected that Traffic Descriptors and Transmission Descriptors are not mixed on a single TP. Therefore if the Egress Descriptor name is that of a Traffic Descriptor then the Ingress Descriptor name should either be that of a Traffic Descriptor or a null value. Likewise if the Egress Descriptor name is that of a Transmission Descriptor then the Ingress

	Descriptor name should either be that of a Trasmission Descriptor or a null value.

6.1.71. TPDataList_T

typedef sequence<TPData_T> TPDataList_T;

Sequence of TPData_T.

6.1.72. SNCType_T

```
enum SNCType_T
  {
   ST_SIMPLE,
   ST_ADD_DROP_A,
   ST_ADD_DROP_Z,
   ST_INTERCONNECT,
   ST_DOUBLE_INTERCONNECT,
   ST_DOUBLE_ADD_DROP,
   ST_OPEN_ADD_DROP,
   ST_EXPLICIT
  };
```

The SNC Type describes the connection based on the signal flows. In case of an ATM SNC (VP or VC), ST_SIMPLE is used to specify a 'plain' (hard or regular) PVC.

6.1.73. Reroute_T

```
enum Reroute_T
  {
   RR_NA,
   RR_NO,
   RR_YES
  };
```

General comment	
The Reroute_T type indicates if the EMS/MEs are allowed and/or required to reroute this SNC if there is a failure on this SNC, periodically to optimize the routes, of for any other reason.It is an EMS/ME implementation whether this is done using network routing protocols or if the EMS/MEs detect the failure and take appropriate action to attempt to fix the SNC. The RR_NO value means that the EMS/MEs are not allowed to reroute the SNC. The RR_YES value means that the EMS/MEs are allowed to reroute the SNC and required to attempt to reroute it upon failure. The RR_NA value is used when the NMS does not want to specify the exact EMS behaviour. In this case it is left up to the EMS to decide whether rerouting will be provided. It is also used if the EMS/MEs are allowed to reroute the SNC but not required to attempt to reroute it upon failure.	
Attribute name	**Comment**
➢ RR_NO	The RR_NO value means that the EMS/MEs are not allowed to reroute the SNC.
➢ RR_NA	The RR_NA value is used when the NMS does not want to specify the exact EMS behaviour. In this case it is left up to the EMS to decide whether rerouting will be provided. It is also used if the EMS/MEs are allowed to reroute the SNC but not required to attempt to reroute it upon failure.
➢ RR_YES	The RR_YES value means that the EMS/MEs are allowed to reroute the SNC and required to attempt to reroute it upon failure.

6.1.74. NetworkRouted_T

```
enum NetworkRouted_T
  {
   NR_NA,
   NR_NO,
   NR_YES
  };
```

Network routed, indicates if the route must be or is computed and implemented at the network level. NR_YES indicates that the route must be / is computed at the network. NR_NO indicates that the route must not be / is not computed at the network. NR_NA indicates that the route can be computed anywhere.
If rerouting is allowed, this attribute will indicate who last rerouted the SNC, the network (NR_YES) or the EMS (NR_NO).

6.1.75. RerouteChangeEvent_T

```
typedef string RerouteChangeEvent_T;
```

The route change goes through different stages during a reroute of a SubnetworkConnection. These stages are a part of the ROUTE_CHANGE notification.
These are:

"RerouteStarted"
"RerouteCompleted"
"RerouteFailed"
"RouteAdded", raised only when the addRoute operation is successful
"RouteRemoved", raised only when the removeRoute operation is successful
"RouteActivated", raised only when the switchRoute operation is successful
"RouteDeactivated", raised only when the switchRoute is successful
"RouteSetToIntended", raised only when the setIntendedRoute operation is successful

6.1.76. SubnetworkConnection_T

```
struct SubnetworkConnection_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   SNCState_T sncState;
   globaldefs::ConnectionDirection_T direction;
   transmissionParameters::LayerRate_T rate;
  StaticProtectionLevel_T staticProtectionLevel;
   SNCType_T sncType;
   TPDataList_T aEnd;
   TPDataList_T zEnd;
   Reroute_T rerouteAllowed;
   NetworkRouted_T networkRouted;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
A subnetwork connection represents a connection between TPs (in any combination). It may also represent Network Connections between the G.805 TCP of any TP.	
Attribute name	**Comment**
➢ name	The name represents the name of the SubnetworkConnection, which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the MultiLayerSubnetworkMgr_I. It is a readonly attribute.

➢ userLabel	The user label of the subnetwork connection is NMS data (typically end-to-end trail data). This could be used by the EMS to display to the user (to associate SNCs/cross-connects to the NMS data), but this is not a requirement on the EMS to display on its GUI.This attribute can be set by the NMS through the Common_I interface service common::Common_I::setUserLabel() or through the createSNC and createAndActivateSNC operations. It is a read/write attribute.
➢nativeEMSName	The name represents how the SNC is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through). The native name is defaulted to a NULL string. However, this could be used by the EMS for its implementation dependent purpose.
➢owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner() or through the createSNC and createAndActivateSNC operations. It is a read/write attribute.
➢sncState	Alcatel RM: Implemented, Commissioned --->SNC_ACTIVE Alcatel RM: Partially Implem --->SNC_PARTIAL Alcatel RM: Defined, Allocated --->SNC_PENDING
➢direction	Specifies the direction of the subnetwork connection. It is a readonly attribute.
➢rate	The rate of the SubnetworkConnection is obtained by reading this attribute. The EMS sets this attribute at creation of the SNC. The EMS is allowed to choose a Layer Rate equivalent to that of one of the TPs. The EMS is expected to reflect any equivalent layer rate in any connection retrievals (i.e. the connection rate of the SNC retrieved from the EMS does not have to be the same as that requested by the NMS so long as it is an equivalent rate).
➢staticProtectionLevel	The static protection level of the SNC. Alcatel RM: In Rings --->HIGHLY_PROTECTED Alcatel RM: SNCP --->FULLY_PROTECTED Alcatel RM: D&C SNCP --->PARTIALLY_PROTECTED Others: UNPROTECTED
➢sncType	Alcatel RM: pointToPoint --->ST_POINT_TO_POINT Alcatel RM: broadcast --->ST_BROADCAST Others: ST_POINT_TO_POINT
➢aEnd	The list of Aend termination points TP where this SNC terminates. This is a readonly attribute.
➢zEnd	The list of Zend termination points (CTP/FTP) where this SNC terminates. This is a readonly attribute.
➢rerouteAllowed	This attribute indicates if the EMS/MEs are allowed and/or required to reroute this SNC if there is a failure on this SNC, periodically to optimize the routes, of for any other reason. It is an EMS/ME implementation whether this is done using network routing protocols or if the EMS/MEs detect the failure and take appropriate action to attempt to fix the SNC. This attribute is not supported currently. This attribute is not supported currently.
➢networkRouted	This attribute specifies if the route for this SNC is computed by the network. This attribute is not supported currently.
➢additionalInfo	LcasFlag="Not Available" | "Enabled" |"Disabled" |"Fixed Enabled" "Not Available": LCAS functionality is not available for the SNC. "Enabled": LCAS functionality is enabled for the SNC. "Disabled": LCAS functionality is disabled for the SNC.

	"Fixed Enabled": LCAS functionality is always enabled and can't be disabled. SncRmProtectionType="None" |"In Rings" |"SNCP" |"D&C SNCP"

6.1.77. SubnetworkConnectionList_T

typedef sequence<SubnetworkConnection_T> SubnetworkConnectionList_T;

Sequence of SubnetworkConnection_T.

6.1.78. CrossConnect_T

```
struct CrossConnect_T
  {
   boolean active;
   globaldefs::ConnectionDirection_T direction;
   SNCType_T ccType;
   globaldefs::NamingAttributesList_T aEndNameList;
   globaldefs::NamingAttributesList_T zEndNameList;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
A crossConnect represents a connection within a single managed element. This structure is primarily used in the specification of routes.	
Attribute name	**Comment**
➢ active	Indicates if the cross-connect is active in the ME.
➢ direction	Directionality of the cross connection.
➢ccType	Alcatel NM: point to point --->CCT_SIMPLE Alcatel NM: multicast --->CCT_SIMPLE Alcatel NM: sncp --->CCT_ADD_DROP_A Alcatel NM: D&C --->CCT_ADD_DROP_A Alcatel RM: Bridge --->CCT_ADD_DROP_Z Alcatel RM: Bridge & Switch --->CCT_EXPLICIT Alcatel RM: Point To Point --->CCT_SIMPLE Alcatel RM: Switch --->CCT_ADD_DROP_A Alcatel RM: Open SNCP --->CCT_ADD_DROP_A Alcatel RM: Drop & Continue --->CCT_INTERCONNECT Alcatel RM: 2N D&C --->CCT_DOUBLE_INTERCONNECT Alcatel RM: Enhanced SNCP --->CCT_DOUBLE_ADD_DROP
➢aEndNameList	Names of CTPs/FTPs/GTPs at the aEnd of the cross connection.
➢zEndNameList	Names of CTPs/FTPs/GTPs at the zEnd of the cross connection.
➢additionalInfo	Possible information: crossConnectionId=<string> This information indicates the crossConnection Id, which is unique in single NE. isFixed=<true|false> This information indicates whether the cross connection is fixed or flexible (just in EMLNBI).

6.1.79. Route_T

typedef sequence<CrossConnect_T> Route_T;

A route for an SNC is defined as a partially ordered list of cross-connects.
This structure can deal with any arbitrarily complex protection paths made up from connection types.
The cross-connects is listed from the NE on which the SNC starts (first entry) to the NE on which the SNC ends (last entry) and the aEndNameList and zEndNameList will name CTPs/FTPs of the same or equivalent layerRate only. There is no mandatory order in the cross-connects listed in between the first one and the last one.

6.1.80. CrossConnectList_T

typedef sequence<CrossConnect_T> CrossConnectList_T;

Arbitrary sequence of cross-connects.

6.1.81. Resource_T

typedef globaldefs::NamingAttributes_T Resource_T;

An inclusion/exclusion constraint for an SNC is defined as a list of resources A resource is defined as an ME, TL, CTP, PTP, FTP or SNC.

6.1.82. ResourceList_T

typedef sequence<Resource_T> ResourceList_T;

Set of Resources_T, used as an inclusion/exclusion constraint for SNC creation.There is no mandatory order for the resources listed.

6.1.83. RouteDescriptor_T

```
struct RouteDescriptor_T
  {
   string id;
   string intended;
   string actualState;
   string administrativeState;
   string inUseBy;
   string exclusive;
   Route_T routeXCs;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
A route is a partially ordered list of cross connections. A SNC always has one route and may have more than one: always 1 intended route, i.e. the preferred, or default route ,0..n backup / alternative route. A route belongs to only one SNC. However XCs/CTPs can be shared by routes of different SNCs.	
Attribute name	**Comment**
➢ id	Unique identifier within the SNC name, with format up to EMS.
➢ intended	Can assume only the following values: "y", "n". The intended route (value "y") could be defined as the preferred, or default route for a given service. Practically, the intended route could be simply the first time-provisioned route, or the preferred route for a number of factors, from network engineering to intrinsic media reliability. The backup route (value "n") is partly or totally different from intended route (but with same end points), and it is

	useful mainly for restoration and maintenance purposes.
➢ actualState	it is the summary state of the actual state of XCs in the network, regardless the SNC such XCs are currently serving; it can assume only the following values: inactive: None of its XCs is active in the network Active: all its XCs are active in the network. So it is the route where SNC traffic is currently carried. There can be at most one active route per SNC. inUseBy shall be always "n". Partial: one or more, but not all the XCs are active in the network. If the route was unsuccessfully activated, then inUseBy shall be always "n". If the route was unsuccessfully deactivated, then inUseBy could be "y".Note: a capability to activate segments of a backup route for more rapid switch over would be of value. It was agreed that this would not be included in the current release but that it could be considered for a later release.
➢ administrativeState	This state refers to the belonging SNC, and has not any relationship with the actual state. It can assume only the following values: locked: the route is not allowed to be active. This state is changeable by: multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::setRoutes AdminState() if all the routes of a given SNC are set to locked, the SNC transits in PENDING state. multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::createSNC () creates one locked intended route multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::deactivate SNC() locks all routes multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::addRoute() creates one locked backup route multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::createModi fiedSNC() modifies the addressed route, which transits to locked. Note for createModifySNC(): the SNC will transit to PARTIAL state, because the just modified route was never unlocked/activated before, and the old route is still ACTIVE in the network. So the SNC PARTIAL state means that an activateSNC() or unlockRoute() operation is needed. Moreover, it is not possible to determine the actual route of the SNC in the network once the old route was modified. unlocked: the route is allowed to be active. This state is changeable by: multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::setRoutes AdminState().If a route was modified by createModifySNC(), then setting it to unlocked implies the activation of the new route. If the activation is successful, (route actual state is ACTIVE) then also the SNC will transit to ACTIVE. If a locked route of a PENDING SNC is set to UNLOCKED, the SNC will transit to either ACTIVE or PARTIAL. multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::createAnd ActivateSNC() creates and unlocks the intended route multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::activateNC () unlocks all routes multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::switchRout e() is a "manual" switch, so no route is locked or unlocked. multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::modifySNC () modifies and unlocks the addressed route multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::deleteSNC () fails if at least one route is unlocked
➢ inUseBy	With value "y" if at least one of its XCs or CTPs is carrying traffic of another SNC, "n" otherwise.

➢ exclusive	The value "y" means not any routes of other SNCs can share any of its XCs or CTPs, even in locked state, "n" otherwise.
➢routeXCs	The partially ordered list of cross-connects that forms the route.
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information, which is not explicitly modelled.

6.1.84. RouteList_T

typedef sequence<RouteDescriptor_T> RouteList_T;

Arbitrary sequence of routes.

6.1.85. RouteCreateData_T

```
struct RouteCreateData_T {
   string  intended;
   string  exclusive;
   CrossConnectList_T ccInclusions;
   ResourceList_T neTpInclusions;
   boolean fullRoute;
   ResourceList_T neTpSncExclusions;
   globaldefs::NVSList_T additionalCreationInfo;
  };
```

General comment	
RouteCreateData_T structure is used by the NMS to pass to the EMS when a route is added to a SNC.	
Attribute name	**Comment**
➢ intended	Can assume only the following values: "y", "n".
➢ exclusive	The value "y" means not any routes of other SNCs can share any of its XCs or CTPs, even in locked state.
➢ ccInclusions	Specifies a list of cross-connects that must be used by the route. The list must be empty if no cross-connect constraints are required. If the EMS cannot fully satisfy the constraints, then the request will be rejected.
➢ neTpInclusions	Specifies a list of MEs/TPs/GTPs that must be used by the route. The list must be empty if no ME/TP/GTP constraints are required. If the EMS cannot fully satisfy the constraints, then the request will be rejected.
➢ fullRoute	Specifies if the ccInclusions and neTpInclusions constraints describe the full route details (as opposed to only a partial constraint). When no inclusions constraints are specified, false must be used.
➢ neTpSncExclusions	Specifies a list of MEs/TPs/GTPs and/or "SNC + routes" to be excluded. The route to be created must not use any of the MEs/TPs/GTPs specified, nor any resource used by the "SNC + routes" specified. Specifying both inclusion and exclusion constraints is not supported, therefore this list must be empty if ccInclusions or neTpInclusions is non empty. "SNC + route" is described by the SNC name followed by the string ("/routeId=") and the route id. If only the SNC name is specified, then its intended route is considered.
➢ additionalCreationInfo	This attribute allows the communication from the EMS to the NMS of additional information, which is not explicitly modelled.

6.1.86. RouteNameAndAdminState_T

```
struct RouteNameAndAdminState_T  {
   string  id;
   string administrativeState;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
This structure is used by the NMS to lock or unlock the route of an SNC.	
Attribute name	**Comment**
➢ id	Unique identifier within the SNC name, with format up to EMS.
➢ administrativeState	This state refers to the belonging SNC, and has not any relationship with the actual state. It can assume only the following values: locked: the route is not allowed to be active. unlocked: the route is allowed to be active.
➢ additionalInfo	To allow the communication of additional information which is not explicitly modelled.

6.1.87. RouteNameAndAdminStateList_T

```
typedef sequence <RouteNameAndAdminState_T> RouteNameAndAdminStateList_T;
```

Arbitrary sequence of RouteNameAndAdminState.

6.1.88. SNCCreateData_T

```
struct SNCCreateData_T
  {
   string userLabel;
   boolean forceUniqueness;
   string owner;
   globaldefs::ConnectionDirection_T direction;
   StaticProtectionLevel_T staticProtectionLevel;
   ProtectionEffort_T protectionEffort;
   Reroute_T rerouteAllowed;
   NetworkRouted_T networkRouted;
   SNCType_T sncType;
   transmissionParameters::LayerRate_T layerRate;
   CrossConnectList_T ccInclusions;
   ResourceList_T neTpInclusions;
   boolean fullRoute;
   ResourceList_T neTpSncExclusions;
   globaldefs::NamingAttributesList_T aEnd;
   globaldefs::NamingAttributesList_T zEnd;
   globaldefs::NVSList_T additionalCreationInfo;
  };
```

General comment	
The read-create attributes required for the creation of a subnetworkConnection on the EMS are packaged together in an SNCCreateData structure, which the NMS will pass to the EMS at SNC creation time. These are the read-create attributes of the SNC.	
Attribute name	**Comment**
➢ userLabel	UserLabel may be specified by the NMS. May be empty.

➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst SNCs of the EMS. The operation will fail if userLabel is already in use.
➢ owner	Owner may be specified by the NMS. May be empty.
➢direction	The connection directionality must be specified by the NMS.
➢staticProtectionLevel	The NMS specify the requested staticProtectionLevel
➢protectionEffort	The NMS must specify the protectionEffort
➢rerouteAllowed	This attribute indicates if the EMS/MEs are allowed and/or required to reroute this SNC if there is a failure on this SNC, periodically to optimize the routes, of for any other reason. It is an EMS/ME implementation whether this is done using network routing protocols or if the EMS/MEs detect the failure and take appropriate action to attempt to fix the SNC. There is no requirement for the reroutes to respect the constraints specified in the creation request (ccInclusions, neTpInclusions, fullRoute, neTpSncExclusions).
➢networkRouted	This attribute specifies if the network is allowed/required to route this SNC.
➢sncType	The NMS specify the sncType
➢layerRate	Identifies the layer at which the SNC is to be made.
➢ccInclusions	Specifies a list of cross-connects that must be used by the SNC.The list must be empty if no cross-connect constraints are required. If the EMS cannot fully satisfy the constraints, then the request will be rejected
➢neTpInclusions	Specifies a list of MEs/TLs/TPs/GTPs that must be used by the SNC. The list must be empty if no ME/TL/TP/GTP constraints are required. If the EMScannot fully satisfy the constraints, then the request will be rejected.
➢fullRoute	Specifies if the ccInclusions and neTpInclusions constraints describe the full route of the SNC (as opposed to only a partial constraint). When no inclusions constraints are specified, false must be used.
➢neTpSncExclusions	Specifies a list of MEs/TLs/TPs/GTPs/SNCs to be excluded. The SNC to be created must not use any of the MEs/TLs/TPs/GTPs/SNCs specified, nor any resource used by the SNCs specified. Specifying both inclusion and exclusion constraints is not supported,therefore this list must be empty if ccInclusions or neTpInclusions is not empty.
➢aEnd	The NMS specify the aEnd TP.Is also used to indicate the Source TP when adding a leg to an existing broadcast system. If the TP is an FTP the NMS is allowed to specify a generic end point within a ME, the EMS will choose the appropriate TP instance.
➢zEnd	The NMS specify the zEnd TP.Is also used to indicate the Sink TP when adding a leg to an existing broadcast system. If the NMS supplies an invalid combination of TPs in aEnd and zEnd, then the EMS will throw an INVALID_INPUT exception.Aend and Zend TPs have to be on the same subnetwork If the TP is an FTP the NMS is allowed to specify a generic end point within a ME, the EMS will choose the appropriate TP instance.
➢additionalCreationInfo	Some additional creation information may be specified by the NMS. For example: for a system which cannot use the routing constraints a BLSR case it may supply the pairs: “BLSRDirection" taking values "EAST", "WEST" and "NA" "Timeslot" taking values "1".."nnnnn".this could be used to carry the "PotentialFutureSetupIndicator" of the aEnd in the case

	where it has an impact on the creation of the SNC.this could be used to indicate the intended role of one or more aEnd or zEnd TPs by using the name conventions name = "A Role" or "Z Role", and value = "CMEndPoint" or "LCEndPoint", where refers to the index of the TP within the aEnd or zEnd list.this could be used to convey the pointer to the alarm severity assignment profile.The list may be empty.

6.1.89. SNCModifyData_T

```
struct SNCModifyData_T
  {
   string userLabel;
   boolean forceUniqueness;
   string owner;
   globaldefs::ConnectionDirection_T direction;
   string modifyType;
   boolean retainOldSNC;
   boolean modifyServers_allowed;
   StaticProtectionLevel_T staticProtectionLevel;
   ProtectionEffort_T protectionEffort;
   Reroute_T rerouteAllowed;
   NetworkRouted_T networkRouted;
   SNCType_T sncType;
   transmissionParameters::LayerRate_T layerRate;
   RouteList_T addedOrNewRoute;
   RouteList_T removedRoute;
   ResourceList_T neTpInclusions;
   boolean fullRoute;
   ResourceList_T neTpSncExclusions;
   globaldefs::NamingAttributesList_T aEnd;
   globaldefs::NamingAttributesList_T zEnd;
   globaldefs::NVSList_T additionalCreationInfo;
  };
```

General comment	
The read-create attributes required for the modification of a subnetworkConnection on the EMS are packaged together in an SNCModifyData structure which the NMS will pass to the EMS in a request to modify an existing SNC.	
Attribute name	**Comment**
➢ userLabel	UserLabel may be specified by the NMS. May be empty.
➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst SNCs of the EMS. The operation will fail if userLabel is already in use.
➢ owner	Owner may be specified by the NMS. May be empty.
➢ direction	The connection directionality must be specified by the NMS.
➢ modifyType	Class of modification: "rerouting", "add_protection" or * "remove_protection".
➢ retainOldSNC	Request for the EMS to keep the old SNC in pending state.
➢ modifyServers_allowed	Allow to modify the server layers to fulfil the protection constraint.
➢ staticProtectionLevel	The NMS specify the requested staticProtectionLevel.
➢ protectionEffort	The NMS specify the protectionEffort
➢layerRate	Identifies the layer at which the SNC is to be made.

➢ rerouteAllowed	This attribute indicates if the EMS/MEs are allowed and/or required to reroute this SNC if there is a failure on this SNC, periodically to optimize the routes, of for any other reason. It is an EMS/ME implementation whether this is done using network routing protocols or if the EMS/MEs detect the failure and take appropriate action to attempt to fix the SNC. There is no requirement for the reroutes to respect the constraints specified in the creation request (ccInclusions, neTpInclusions, fullRoute, neTpSncExclusions).
➢ networkRouted	Sp This attribute specifies if the network is allowed/required to route this SNC.
➢ sncType	The NMS specify the sncType.
➢ layerRate	Identifies the layer at which the SNC is to be made.
➢ addedOrNewRoute	Depending on the modifyType, AddedOrNewRoute describes the route of a new protection leg or the whole SNC. When it describes a segment to be added, either the SNCP cross-connects or the switch TPs that will be changed in the segment may be specified by the NMS. The EMS then chooses the missing segments. Alternatively, the NMS may specify the full route.
➢ removedRoute	RemovedRoute describes dropping of a protection leg from the original SNC. Either the last cross-connects (that contain the SNCP) are specified by the NMS or the full route may be specified. This parameter can be used in conjunction with addedOrNewRoute only to reroute a segment.
➢ neTpInclusions	Specifies a list of MEs, TLs, PTPs, FTPs and/or CTPs that must be used by the SNC when carrying out a full reroute or adding a routing leg. The list must be empty if no ME/TL/PTP/CTP/FTP constraints are required.If the EMScannot fully satisfy the constraints, then the request will be rejected.
➢fullRoute	Specifies if the neTpInclusions constraints describe the full route of the SNC or routing leg (as opposed to only a partial constraint).When no inclusions constraints are specified, false must be used.
➢neTpSncExclusions	Specifies a list of MEs, TLs, PTPs, FTPs, CTPs, and/or SNCs to be excluded. This is applicable for adding the leg or doing a full reroute and the route must not use any of the NEs, TLs, PTPs, FTPs and CTPs specified, nor any resource used by the SNCs specified. Specifying both inclusion and exclusion constraints is not supported, therefore this list must be empty if neTpInclusions is not empty.
➢aEnd	The NMS specify the aEnd (CTP/FTP).Is also used to indicate the Source TP when adding a leg to an existing broadcast system. If the TP is an FTP the NMS is allowed to specify a generic end point within a ME, the EMS will choose the appropriate TP instance.
➢zEnd	The NMS specify the zEnd (CTP/FTP).Is also used to indicate the Sink TP when adding a leg to an existing broadcast system.If the NMS supplies an invalid combination of TPs in aEnd and zEnd, then the EMS will throw an INVALID_INPUT exception. Aend and Zend TPs have to be on the same subnetwork If the TP is an FTP the NMS is allowed to specify a generic end point within a ME, the EMS will choose the appropriate TP instance.
➢additionalCreationInfo	Some additional creation information may be specified by the NMS.For example: for a system which cannot use the routing constraints for a BLSR case it may supply the pairs: "BLSRDirection" taking values "EAST", "WEST" and "NA" "Timeslot" taking values "1".."nnnnn". this could be used to carry the

	"PotentialFutureSetupIndicator" of the aEnd in the case where it has an impact on the creation of the SNC.this could be used to indicate the intended role of one or more aEnd or zEnd TPs by using the name conventions name = "A Role" or "Z Role", and value = "CMEndPoint" or "LCEndPoint", where refers to the index of the TP within the aEnd or zEnd list.this could be used to convey the pointer to the alarm severity assignment profile.The list may be empty.

6.1.90. Directionality_T

```
enum Directionality_T
  {
    D_NA,
    D_BIDIRECTIONAL,
    D_SOURCE,
    D_SINK
  };
```

General comment	
Direction for a TerminationPoint. The directionality of PTPs is defined from an external point of view, while the directionality of CTPs is defined from an internal point of view. Consequently, sink PTPs generate source CTPs, and sink CTPs form source PTPs. Note that ATM termination points (LR_ATM_NI, LR_ATM_VP and LR_ATM_VC) are always bi-directional even though the traffic may be asymmetric (and eventually null in one of the two directions).	
Attribute name	**Comment**
➢ D_NA	Used when the directionality specification is not necessary.
➢ D_BIDIRECTIONAL	Source and sink (transmit and receive).
➢ D_SOURCE	Source (transmit).
➢ D_SINK	sink (receive).

6.1.91. TPConnectionState_T

```
enum TPConnectionState_T
  {
    TPCS_NA,
    TPCS_SOURCE_CONNECTED,
    TPCS_SINK_CONNECTED,
    TPCS_BI_CONNECTED,
    TPCS_NOT_CONNECTED
  };
```

A CTP/FTP may be involved in zero, one, or many connections. The value TPConnectionState indicates the degree to which a CTP/FTP is used. The values TPCS_SOURCE_CONNECTED and TPCS_SINK_CONNECTED reflect the presence of a one-way connection. The value TPCS_BI_CONNECTED means that the TP is both sink and source connected. When this attribution is used in conjunction with a GTP, it indicates the TPConnectionState of the contained CTPs. All CTPs within a GTP shall have the same TPConnectionState.If an EMS cannot report whether the source of the TP is connected or the sink is connected, TPCS_BI_CONNECTED may be reported by that EMS. The value TPCS_NA is used for PTPs and TPPools.

6.1.92. TPType_T

```
enum TPType_T
  {
    TPT_PTP,
    TPT_CTP,
    TPT_TPPool
  };
```

As the interface is coarse grained, TPs are modeled as pure data objects and do not appear as first class CORBA objects at the interface between the NMS and EMS.
The PTP or Physical Termination Point represents a single port of an NE. The PTP is an aggregate of G.805 TCPs, G.805 Termination Functions and G.805 CPs etc at many layers. The PTP approach is used for performance and interface simplification.
A CTP in this model may correspond directly to a single G.805 CP or may represent an aggregate of G.805 TCPs, G.805 Termination Functions and G.805 CPs etc at many layers. A CTP may also include the G.805 adaptation function of IM (Inverse Multiplexing) in the aggregation or may include the fragment TCP of IM.
A CTP may be involved in SNCs on its server side to its aggregated TCP or to its aggregated CP depending upon its structure and relationship to the containing FTP/PTP
The FTP or Floating Termination Point represents a set of G.805 termination functions and G.805 connection points that are not associated directly with a physical port of an NE. The FTP is an aggregate of G.805 TCPs, G.805 Termination Functions and G.805 CPs etc at many layers.
The FTP always contains one or more client CTPs (same as a PTP). Unlike a PTP the FTP may contain server CTPs (strictly a CTP may also contain server CTPs). An FTP may also include the G.805 adaptation function of IM (inverse multiplexing) in the aggregation. Like a CTP the FTP may be involved in an SNC on its server side (i.e. takes the role of a CTP). An FTP that is capable of being involved in an SNC on its server side does not contain server CTPs, an FTP that is not capable of being involved in an SNC on its server side must contain server CTPs. Like the PTP, the FTP is not contained in another TP and it effectively takes the role of a PTP from a containment perspective.
The FTP is not represented in the TPType_T enum explicitly, but instead is identified as a TPT_PTP based upon its role in containment. This approach was taken in release 3.0 to maximise compatibility with release 2.0.
A TPPool is a set of Termination Points (CTPs, PTPs or FTPs in any mix). This type has been initially introduced to support the concept of administrative partitioning of an ATM Network Interface (a VP TPPool is defined as a set of VP CTPs).

6.1.93. TerminationMode_T

```
enum TerminationMode_T
  {
    TM_NA,
    TM_NEITHER_TERMINATED_NOR_AVAILABLE_FOR_MAPPING,
    TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING
  };
```

General comment

For ATM SNCs, the Terminated and Mapped parameter of a VP or VC CTP is used to model a VPC or a VCC terminating within the Managed Element (i.e.internal VP or VC Trails). Such a terminated ATM connection is used as a trail acting as a server for upper layer protocols (e.g., VCCs in case of a VPC TP, Frame Relay in case of a VCC TP). In the two switching layer ATM model, the VP layer acts as the infrastructure on which VC Connections (either PVC, SPVC or SVC) are routed and switched. This capability allows the operator to build a logical partitioning (VP overlay) of the physical ATM network by configuring VPCs (or VP "tunnels"), which are terminated inside the subnetwork. Such overlay VP network allows operators to aggregate and segregate VCs according to their traffic management policy.
In addition, the use of an intra-subnetwork VPC may provide for enhanced protection of the VC traffic by using 1+1 VP protection (VP protection is not supported in this release of the EMS-NMS interface). In order to be able to make routing decisions at VC level (e.g., explicit route constraint), the NMS needs to know the VP topology available from the ATM network.
Note that requesting the operator to explicitly set up an overlay VP network as a pre-requisite for passing any VC traffic may be cumbersome and in some case results in a sub-optimal use of the ATM links. For that reason, most ATM NEs provide for an alternative which allows for each ATM NE to act as a VC switch without having to explicitly configure VPs (i.e., each ATM link acts as an internal VP link between two adjacent nodes)

Attribute name	Comment
➢ TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING	CTP: The CTP can be mapped and currently is (i.e., it has contained actual CTPs). This indicates that the G.805 CPs adapted from the corresponding G.805 Termination Function within the CTP aggregate are available to provide client layer capacity (e.g. STS1 terminated and mapped to VT1.5 i.e. channelized). In this state the TCP will be sourcing and sinking traffic. This is only a valid value for CTPs that support client adaptation. FTP: The FTP can be mapped on the server side and currently is (i.e., it has contained actual CTPs on the server side). This indicates that the FTP supports IM (Inverse Multiplexing) and the server side contained CTPs (if any) are available for connection as the FTP is actively assembling the fragments of the IM. This value is not valid for an FTP that does not support IM. PTP: This value is not valid for a PTP.
➢ TM_NEITHER_TERMINATED_NOR_AVAILABLE_FOR_MAPPING	CTP: The CTP can be mapped but currently is not (i.e., it does have contained potential CTPs, but currently has no contained actual CTPs). This indicates that the G.805 CP aggregated in the CTP is not connected to a TCP (that would be aggregated in the sameCTP) and therefore the CTP is available for SNC/crossconnect connectivity (e.g. STS1 not terminated and not mapped to VT1.5i.e. not channelized). FTP: The FTP can be mapped on the server side but currently is not (i.e., it does have contained potential CTPs on the server side, but currently has no contained actual CTPs on the server side). This indicates that the FTP supports IM (Inverse Multiplexing) but the IM function is not active and the TCP within the FTP is available for external SNC/crossconnect connectivity. PTP: This value is not valid for a PTP.
➢ TM_NA	CTP: The CTP cannot be mapped (i.e., it has no contained potential CTPs). This indicates that an SNC/crossconnect can be used to connect the G.805 TCP of the CTP to the connection point of another CTP/FTP. This only applies to CTPs that connect on their server side and are related to the containing PTP/FTP on their client side. FTP: The FTP cannot be mapped on the server side (i.e., it has no contained potential CTPs on the server side). This indicates that an SNC/crossconnect can be used to connect the G.805 TCP of the FTP to the connection point of another CTP/FTP. This only applies to FTPs that connect on their server side and do not support IM (Inverse Multiplexing).

	PTP: This is the only legal value for a PTP For ATM SNCs, the Terminated and Mapped parameter of a VP or VC CTP is used to model a VPC or a VCC terminating within the Managed Element (i.e.internal VP or VC Trails). Such a terminated ATM connection is used as a trail acting as a server for upper layer protocols (e.g., VCCs in case of a VPC TP,Frame Relay in case of a VCC TP). In the two switching layer ATM model, the VP layer acts as the infrastructure on which VC Connections (either PVC, SPVC or SVC) are routed and switched. This capability allows the operator to build a logical partitioning (VP overlay) of the physical ATM network by configuring VPCs (or VP "tunnels") which are terminated inside the subnetwork. Such overlay VP network allows operators to aggregate and segregate VCs according to their traffic management policy. In addition, the use of an intra-subnetwork VPC may provide for enhanced protection of the VC traffic by using 1+1 VP protection (VP protection is not supported in this release of the EMS-NMS interface). In order to be able to make routing decisions at VC level (e.g., explicit route constraint), the NMS needs to know the VP topology available from the ATM network. Note that requesting the operator to explicitly set up an overlay VP network as a pre-requisite for passing any VC traffic may be cumbersome and in some case results in a sub-optimal use of the ATM links. For that reason, most ATM NEs provide for an alternative which allows for each ATM NE to act as a VC switch without having to explicitly configure VPs (i.e., each ATM link acts as an internal VP link between two adjacent nodes).

6.1.94. TPProtectionAssociation_T

```
enum TPProtectionAssociation_T
  {
   TPPA_NA,
   TPPA_PSR_RELATED
  };
```

TpProtectionAssociation expresses constraints on PTPs/CTPs/FTPs for PSR connection management.
In a multi-layer subnetwork, say 'a', 'b', 'c' are edge points. Suppose, for example, a three-ended connection is sought from 'a' to 'b', where 'b' is one of the endpoints. If 'c' is the constrained choice for 'b' as the other end of the three-ended connection,then 'b' and 'c' are said to be associated by a protectionAssociation.
The tpProtectionAssociation is set to TPPA_PSR_RELATED in 'b' and 'c', and getAssociatedTP(b) returns c and getAssociatedTP(c) returns b.
In all other cases, tpProtectionAssociation is set to TPPA_NA.
The multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::getAssociatedTP() service must be used to obtain the related TP.

6.1.95. TerminationPoint_T

```
struct TerminationPoint_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   globaldefs::NamingAttributes_T ingressTrafficDescriptorName;
   globaldefs::NamingAttributes_T egressTrafficDescriptorName;
   TPType_T type;
   TPConnectionState_T connectionState;
   TerminationMode_T tpMappingMode;
```

```
Directionality_T direction;
transmissionParameters::LayeredParameterList_T transmissionParams;
TPProtectionAssociation_T tpProtectionAssociation;
boolean edgePoint;
globaldefs::NVSList_T additionalInfo;
string transmissionType;
transmissionParameters::LayerRate_T layerRate;
};
```

General comment
A TP is modeled as a data structure to avoid a great number of CORBA objects across the EMS/NMS interface. Internally in the EMS, these data structures can map to any desired architecture. This is an abstract class that encapsulates the data and behavior that is common to the different types of end points. For instance, a TP may be Sink (Receive), Source (Transmit) or Bi-directional (Transmit and Receive), has a LayerRate, a name and a userLabel. Each TP has an associated set of attributes that represent transmission parameters.For a PTP/FTP/CTP the transmission parameters are at various LayerRates that are aggregated to form the PTP/FTP/CTP). Termination points in this model are either bidirectional or unidirectional. Whereas bidirectional SNCs can be supported by bidirectional endpoint TPs only, unidirectional SNCs can be supported by both, bidirectional and unidirectional endpoint TPs where the TPConnectionState indicates the TP is either source connected (a_end of an SNC) or a sink connected (z_end of an SNC).In either case, the intermediate CTPs used by the SNC may be unidirectional or bidirectional.The location of free unidirectional resources may be determined by testing for source or sink connected termination points. Termination points that are bidirectional cannot be assumed to be associated with bidirectional SNCs except by checking the connection direction on the SNC(s)using the Termination point. A CTP is a (Subnetwork) Connection Termination Point. A CTP is a potential end point of a subnetwork connection. If a CTP is part of an active subnetwork connection, then the CTP entity must exist on an NE (in terms of TL1 the termination is entered), otherwise the model does not specify if the CTP exists on the NE or not. A PTP (Physical (Topological Link) Termination Point) is an end-point of a Physical Link. Examples of PTPs are T1 ports, T3 ports, OC-N optical ports, etc. PTPs have a containment relation with CTPs. A FTP (Floating Termination Point) it may be an end-point of a (logical) Topological Link. It may be involved in an SNC. It will have contained CTPs. A TPPool is a TP Pool Termination Point. It is used to logically group TPs for administrative purposes. An example is the partitioning of VP CTPs for bandwidth management. PTPs forming an APS pair are related.

Attribute name	Comment
➢ name	The name represents the name of the Termination Point, which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. The naming for CTPs,PTPs and FTPs is deterministic. Note that the naming of FTPs and PTPs differ to allow them to be distinguished. Note that the termination point name must be explicit (a generic endpoint specification may not be used in this case). It is a readonly attribute.
➢ userLabel	The user label of the TP is set with NMS data (typically the end to end trail data). This can be set via the setUserLabel operation. The user label may be cleared when the TP is deleted or when the model does not specify if the CTP exists (see above). The EMS is expected to place this data in the PM Data file transfers if the EMS supports the Performance Manager services. It is a read-write attribute.
➢nativeEMSName	The name represents how the TP is refered to on EMS displays. Its aim is to provide a "nomenclature bridge" the aid relating information presented on NMS displays to EMS displays (via GUI cut through). It is never set to NULL string.

➢owner	The owner attribute of the TerminationPoint indicates the ownership of the TP so that adminstrativeState can be managed. Since the administrative state is not used, there is no use for this now. The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). The owner may be cleared when the TP is deleted or when the model does not specify if the CTP exists (see above). It is a read/write attribute.
➢ingressTrafficDescriptorName	A connection termination point may have an optional reference to an ingress (incoming) Traffic Descriptor or Transmission Descriptor. The Descriptor name will be empty if there is no associated Descriptor. It is expected that Traffic Descriptors and Transmission Descriptors are not mixed on a single TP. Therefore if the Ingress Descriptor name is that of a Traffic Descriptor then the Egress Descriptor name should either be that of a Traffic Descriptor or a null value. Likewise if the Ingress Descriptor name is that of a Transmission Descriptor then the Egress Descriptor name should either be that of a Trasmission Descriptor or a null value. Not supported in this version.
➢egressTrafficDescriptorName	A connection termination point may have an optional reference to an egress (outgoing) Traffic Descriptor or Transmission Descriptor. The Traffic Descriptor name will be empty if there is no associated Descriptor. It is expected that Traffic Descriptors and Transmission Descriptors are not mixed on a single TP. Therefore if the Egress Descriptor name is that of a Traffic Descriptor then the Ingress Descriptor name should either be that of a Traffic Descriptor or a null value. Likewise if the Egress Descriptor name is that of a Transmission Descriptor then the Ingress Descriptor name should either be that of a Trasmission Descriptor or a null value. Not supported in this version.
➢type	Because the TerminationPoint can either be a PTP, CTP or TPPool. it is necessary to identify its type. As noted earlier, FTPs use the same type as PTPs for compatibility reasons. An FTP may be distinguished from a PTP by examining the name. It is a readonly attribute.
➢connectionState	This attribute is only applicable to CTPs and FTPs. If the source is connected to one entity and the sink is connected to another entity then the value of this attribute is TPCS_BI_CONNECTED. This is a readonly attribute. Not supported in this version.
➢tpMappingMode	Within the definition, the CTP/FTP can act as an aggregate of associated G.805 TCPs, G.805 Termination Functions and G.805 CPs at one or more LayerRates. The CTP is contained within a PTP or FTP. The TerminationMode attribute indicates and controls the connection of the named CP at a specified LayerRate to the dedicated G.805 TCP and associated G.805 Termination Function at the same LayerRate within the CTP/FTP. The TerminationMode is used, for example, to control the Termination and mapping to VT1.5 of an STS1 Trail within an OC3 port. This capability could potentially be used to terminate an STS1 backbone Trail within a lower order mux that has the capability to alternatively cross-connect the STS1 unterminated. The same capability is used in SDH and is potentially applicable to any LayerRate. Not supported in this version.
➢direction	The direction of the termination point. It is a readonly attribute.
➢transmissionParams	A list of transmission parameters which can be set and/or retrieved on the TP at a specified layer. From EMLNBI1.3, below parameters are supported: OscCentralFrequency=<string>

	OscFrequencySpacing=<string> MaxNumberOCh=<integer> From EMLNBI1.5, following parameters are supported: SDH TTI information: TrailTraceExpectedRx=<string> TrailTraceReceivedRx=<string> TrailTraceActualTx=<string> examples: name=TrailTraceExpectedRx , value=123456789ABCDEF name=TrailTraceExpectedRx , value=disabled name=TrailTraceReceivedRx , value=123456789ABCDEF name=TrailTraceActualTx , value=123456789ABCDEF
➢tpProtectionAssociation	The associated TP indication.The NMS is responsible to invoke the multiLayerSubnetwork::MultiLayerSubnetworkMgr_I::getAssociatedTP() service to obtain the related TP if any. Not supported in this version.
➢edgePoint	Indicates if the TP is an edge point of at least one subnetwork, i.e. if it is an end point of a potential inter-subnetwork topological link. Not supported in this version.
➢additionalInfo	Supported additional informations: q3MOI=<string> This information indicateds the object instance from Q3 interface. fdnMapperLabel=<string> This information indicateds the object name from alcatel RM. OGPI,OTS ptp support this in NBI1.4. ttpIndicator="YES"|"NO" This information indicates the tp is whether ttp. ethernetMappingApplied ="gFPnullExtensionNoFCS | gFPnullExtensionWithFCS | gFPlinearExtensionNoFCS | gFPlinearExtensionWithFCS | gFPpacketConcBasic | gFPpacketConcExtended | x86 | hDLC | unknown | gFPnullExtensionNoFCSTransparent | gFPnullExtensionWithFCSTransparent | gFPlinearExtensionNoFCSTransparent | gFPlinearExtensionWithFCSTransparent" This information indicates encapsulate protocol of XVirtualTTp. frequency=<String> This information indicates the optical wave frequency on ochCTP,OGPITTP. AutoProvisioning=<String> If the autoProvisioning field is present alcTargetPower value can be set either by the management system (in this case the autoProvisioning field has value FALSE) or by the network element itself. in this case the autoProvisioning field has value TRUE). TargetPower=<String> This attribute contains the target power for the ALC procedure. usedFlag="Used" | "Idle" "Used": indicates that the whole tp or some time slot of the tp is currently in used. "Idle": indicates that the whole tp is not in used currently. crossConnected="true"|"false" This information indicates whether the TP is involved in any crossConnections.crossConnectedDirection="sink"|"source"|"bidirectional"|"notConnected"

	This intofmation indicateds the TP directional in crossConnection, if it is involved in any crossConnections. crossConnectionId=<string> This information indicateds the TP associated crossConnection's Id, if it is involved in any crossConnections. protectionRole="protecting"|"protected" This information indicates the TP protection role , if it is involved in any protection scheme. TTIexpected=<string> This information indicates the Trail Trace Identifier the TP expected. TTIAccepted=<string> This information indicates the Trail Trace Identifier the TP received. TTISend=<string> This information indicates the Trail Trace Identifier the TP send, if it config to send TTI. ALSStatus="[<LaserId>=]<LASERSTATUS> ..." This information indicates the Automatic Laser Shutdown status of the PTP. <LaserId> is Laser's Id. <LASERSTATUS> value: "0": The ALS procedure is enabled. "1": The ALS procedure is disabled and the laser is forced to ON. "2": The ALS procedure is disabled and the laser is forced to OFF. "3": The ALS procedure is disabled and the laser is free. CI=<string> (EMLNBI1.3 supported) This information indicates the charicator information of the SDH/PDH TP. Valid values are: opticalSTM1SPICI, opticalSTM4SPICI, opticalSTM16SPICI, opticalSTM64SPICI, opticalSTM256SPICI, electricalSTM1SPICI, rsSTM1SPICI, rsSTM4SPICI, rsSTM16SPICI, rsSTM64SPICI, rsSTM256SPICI, msSTM1SPICI, msSTM4SPICI, msSTM16SPICI, msSTM64SPICI, msSTM256SPICI, au3TU3VC3CI, au4VC4CI, tu11VC11CI, tu12VC12CI, tu2VC2CI, tu12VC11CI, au4VC4Cont4CI, au4VC4Cont16CI, au4VC4Cont64CI, e0CI, e1CI, e2CI, e3CI, e4CI, N/A waveLength=<string> (EMLNBI1.3 supported) This information indicates the OGPI physical interface's wave length. Valid values are: wl1310, wl1550, coloured, wl820, multimode, N/A. regeneratorType=<string> (EMLNBI1.3 supported) This information indicates the regenerator of the WDM TP. Valid values are: threeR, twoR, N/A maxTransDistance=<integer> (EMLNBI1.3 supported) This information indicates the max transmission distance of the WDM OTS physical interface. nIType="UNI" | "NNI" | "Unknow" (EMLNBI1.3 supported) "UNI": User-Network Interface side, added on OGPI ttp. "NNI": Network-Network Interface side, added on OGPI ttp, rs ctp or otu ctp. channelSpace=<integer> (EMLNBI1.4 supported) This information indicates the channel space on OCH ctp and OMS ttp of alcatel wdm ne. maximum</i>=<integer> (EMLNBI1.4 supported)

	This information indicates the maximum frequence value of the WDM OMS ttp. minimum=<integer> (EMLNBI1.4 supported) This information indicates the minimum frequence value of the WDM OMS ttp. potentialOchTotalNumber=<integer> (EMLNBI1.4 supported) This information indicates the potential Och total number of the WDM OMS ttp. potentialBundleTotalNumber</i>=<integer> (EMLNBI1.4 supported) This information indicates the potential bundle total number of the WDM OMS ttp. supervisoryChannelType=<string> (EMLNBI1.3 supported) This information indicates the OSC type of WDM OTS Port. Valid values are: inband, outOfBand, N/A maximumChannel=<string> (EMLNBI1.4 supported) This information indicates the och number of the OMSTTP contains. equipmentType=<string> (EMLNBI1.4 supported) This information indicates the equipment's type which contains the OGPI and OTS ttp. daughter_equipmentType=<string> (EMLNBI1.4 supported) This information indicates the daughter equipment's type which contains the OGPI and OTS ttp. clientSignalRate=<string> (EMLNBI1.4 supported) This information indicates the signal rate on OGPI TTP. neAddress=<string> (EMLNBI1.4 supported) This information indicates network address on virtual ttp. flowControl=<string> (EMLNBI1.3 supported) This information indicates the Ethernet TP's flow control configuration. Valid values are: disabled, asymmetric-PAUSE, symmetric-PAUSE, N/A. This information can be modified by operation setTPData(). workingMode=<string> (EMLNBI1.3 supported) This information indicates the Ethernet TP's working mode. Valid values are: isPromiscuousMode, notPromiscuousMode, N/A. Mtu=<integer> (EMLNBI1.3 supported) This information indicates the Ethernet TP's MTU. vlanTag=<integer> (EMLNBI1.3 supported) This information indicates the Ethernet TP's vlan tag. This information can be modified by operation setTPData(). defaultVlanID=<integer> (EMLNBI1.3 supported) This information indicates the Ethernet TP's default vlan ID. This information can be modified by operation setTPData() incomingMonitor=<string> (EMLNBI1.3 supported) This information indicates the Ethernet TP's incoming monitor's status. Valid values are: true, false, N/A. MPLSRole=<string> (EMLNBI1.3 supported) This information indicates the Ethernet TP's MPLS role. Valid values are: P, PE, N/A. EncapProt=<string> (EMLNBI1.3 supported) This information indicates the Ethernet TP's encapsulation protocol. Valid values are: ethernetV2, llc-snap, HDLC, GFP, LAPS, PPP, N/A.

➢transmissionType	"Optical" or "Electric"
➢layerRate	supported layer rate

6.1.96. TerminationPointList_T

typedef sequence<TerminationPoint_T> TerminationPointList_T;

Sequence of TerminationPoint_T.

6.1.97. GTP_T

```
struct GTP_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   boolean alarmReportingIndicator;
   globaldefs::NamingAttributesList_T listOfTPs;
   TPConnectionState_T gtpConnectionState;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
A GTP is simply a list of TPs in the same ME. GTPs can be cross connected. Either the EMS or the NMS can create a GTP. In the case of EMS initiated creation, the NMS would receive an Object Creation notification. The TPs comprising a GTP need not be contiguous and need not to be at the same layer rate. A GTP is modeled as a data structure. All the TP in a GTP are either bidirectional or unidirectional. GTP names are set by the EMS and must be unique within an ME. A GTP is contained under an ME which is contained under an EMS. So, a GTP name has three components, i.e., the EMS name-value pair, the ME name-value pair and the GTP name-value pair. A GTP can serve as an end point as well as an interior point of an SNC. However, a GTP is not a CTP.	
Attribute name	**Comment**
➢ name	The name uniquely identifies a GTP within the context of a managed element. A GTP's name is assigned by the EMS upon creation of the GTP. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. It is a readonly attribute.
➢ userLabel	The user label of the GTP is set with NMS data (typically the end to end trail data). This can be set via the setUserLabel operation. The user label may be cleared when the GTP is deleted It is a read-write attribute.
➢ nativeEMSName	The native EMS name represents how the GTP is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" help relate information displayed by the EMS to information displayed by the NMS. It is never set to NULL string.
➢ owner	The owner attribute of the GTP indicates the ownership of the GTP so that adminstrativeState can be managed. Since the administrative state is not used, there is no use for this now. The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). The owner may be cleared

	when the GTP is deleted. It is a read/write attribute.
➢ alarmReportingIndicator	Provides an indication whether alarm reporting for this GTP is active or not. It is a read/write attribute.
➢ listOfTPs	This is the list of CTP names that comprise the GTP. It is a read/write attribute.
➢ gtpConnectionState	When this attribution is used in conjunction with a GTP, it indicates the TPConnectionState of the contained CTPs. All CTPs within a GTP shall have the same TPConnectionState
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile. This may be an empty list.

6.1.98. GTPlist_T

typedef sequence<GTP_T> GTPlist_T;

Sequence of GTP_T.

6.1.99. GTPEffort_T

```
enum GTPEffort_T
  {
   EFFORT_WHATEVER,
   EFFORT_SAME
  };
```

The GTP effort is a statement of the requirement of the list of CTPs that a new GTP. If EFFORT_SAME is specified then the EMS must create the GTP with the exact same list of CTPs as provided with the GTP creation request. Further, if the NMS uses EFFORT_SAME, it is assume that the EMS will not modify the CTP list comprising the GTP at some later point (unless requested by the NMS via the modifyGTP operation). If EFFORT_WHATEVER is specified then the EMS may comply with the total bandwidth requirement by using a different set of CTPs.
If the NMS requests a GTP with gtpEffort equal to EFFORT_SAME, this also implies that the EMS will not modify on its own, i.e., the EMS will only modify the GTP if requested by the NMS via the modifyGTP operation.

6.1.100. TopologicalLink_T

```
struct TopologicalLink_T
  {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   globaldefs::ConnectionDirection_T direction;
   transmissionParameters::LayerRate_T rate;
   globaldefs::NamingAttributes_T aEndTP;
   globaldefs::NamingAttributes_T zEndTP;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment

The TopologicalLink structure provides information to the NMS to allow it to present and interpret a topological view of the network managed by the EMSes.
The rate of a TopologicalLink describes the layer at which the link is modelled. E.g. in SDH links could be modelled at the RS layer and the MS layer.
A TopologicalLink has a name and references to the two TPs. These TPs are any combination of CTP, FTP and PTP(i.e. can be a mix PTP-CTP etc)
For a TP that is connected outside of the EMS' span of control, if the EMS knows about the remote end, the EMS may provide this information via a single-ended topological link.
Note that an ATM link is an example of TopologicalLink terminated by CTPs defined at LR_ATM_NI layer rate. Such a link is used to transport VP traffic and 'VC without VP' traffic between ATM NEs. Another example would be a VP trail internal to an ATM subnetwork. The EMS should expose such a trail as a topological link since this VP is a component of the ATM infrastructure used for VC traffic (PVCCs or SVCCs). The trail has been created as a subnetwork connection between two ATM VP CTPs at the LR_ATM_VP Layer Rate, which has been 'extended' on both ends by two further ATM VP CTPs that are terminated and mapped. Each terminated CTP is attached to its connected counterpart CTP by a topological link that needs not be exposed at the interface. But the whole VP trail between the two terminated VP CTPs should be exposed as a topological link with layer rate LR_ATM_VP.

Attribute name	Comment
➢ name	The name represents the name of the Topological Link which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the EMS. It is a readonly attribute.
➢ userLabel	The user label of the topological link is NMS Data (typically end-to-end trail data). This could be used by the EMS to display to the user (to associate topological links to the NMS data), but this is not a requirement on the EMS to display on its GUI. The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	The name represents how the link is refered to on EMS displays. Its aim is to provide a "nomenclature bridge" the aid relating information presented on NMS displays to EMS displays (via GUI cut through). The native name is defaulted to a NULL string. However, this could be used by the EMS for its implementation dependent purpose.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ direction	Direction of the topological link. A topological link can be unidirectional even if both its ends are bidirectional TPs.
➢ rate	The layer of the topological link.
➢ aEndTP	Name of A end TP (PTP/CTP/FTP). The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ zEndTP	Name of Z end TP (PTP/CTP/FTP). The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile. This may be an empty list.

6.1.101. TopologicalLinkList_T

typedef sequence<TopologicalLink_T> TopologicalLinkList_T

Sequence of TopologicalLink_T.

6.1.102. TLCreateData_T

```
struct TLCreateData_T
  {
   string userLabel;
   boolean forceUniqueness;
   string owner;
   globaldefs::ConnectionDirection_T direction;
   transmissionParameters::LayerRate_T rate;
   globaldefs::NamingAttributes_T aEndTP;
   globaldefs::NamingAttributes_T zEndTP;
   globaldefs::NVSList_T additionalCreationInfo;
  };
```

General comment	
The read-create attributes required for the creation of a toplogicalLink on the EMS are packaged together in an TLCreateData structure which the NMS will pass to the EMS at topological link creation time. These are the read-create attributes of the topological link.	
Attribute name	**Comment**
➢ userLabel	userLabel may be specified by the NMS. May be empty. The user label of the topological link is NMS Data (typically end-to-end trail data). This could be used by the EMS to display to the user (to associate topological links to the NMS data),but this is not a requirement on the EMS to display on its GUI. Note: The userLabel is provisionable by the NMS and can be modified after creation by NMS through the Common_I interface service common::Common_I::setUserLabel().
➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst topological links of the EMS. The operation will fail if userLabel is already in use.
➢ owner	owner may be specified by the NMS. May be empty. Note: The owner is provisionable by the NMS and can be modified after creation by NMS through the Common_I interface service common::Common_I::setOwner().
➢ direction	Direction of the topological link. A topological link can be unidirectional even if both its ends are bidirectional TPs.
➢ rate	The layer of the topological link.
➢ aEndTP	Name of A end TP. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ zEndTP	Name of Z end TP. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. Additional info is used to convey the pointer to the alarm severity assignment profile.This may be an empty list

6.1.103. ServiceCategory_T

```
enum ServiceCategory_T {
   SC_CBR,
   SC_VBRRT,
   SC_VBRNRT,
   SC_ABR,
   SC_UBR,
   SC_GFR,
```

```
 SC_NA
};
```

General comment	
Represents Service Categories used by Traffic Descriptors. The current Service Categories are defined as follows (note that Guaranteed Bit Rate (GBR) is not defined because it is generally not used)	
Attribute name	**Comment**
➢ SC_CBR	SC_CBR = Constant Bit Rate. For connections that require a sstatic amount of bandwidth that is continuously available during the lifetime of the connection.
➢ SC_VBRRT	SC_VBRRT = Variable Bit Rate - Real-Time. For connections that require tightly constrained delay and delay variation.
➢ SC_VBRNRT	SC_VBRNRT = Variable Bit Rate - NonReal-Time. For connections that have bursty traffic.
➢ SC_ABR	SC_ABR = Available Bit Rate. For connections that do not require bounding the delay or delay variation. Not intended for real-time applications.
➢ SC_UBR	Unspecified Bit Rate. For connections that do not require tightly constrained delay and delay variation.
➢ SC_GFR	Guaranteed Frame Rate. For non-real-time applications that may require a minimum rate guarantee and can benefit from accessing additional bandwidth dynamically available in the network.
➢ SC_NA	SC_NA = Not Applicable

6.1.104. TrafficParameterList_T

typedef globaldefs::NVSList_T TrafficParameterList_T;

TrafficParametersList_T is used to specify the parameters used in Traffic Descriptors. The list is a sequence of name/value pairs (NVSList_T).

6.1.105. TrafficDescriptor_T

```
struct TrafficDescriptor_T {
   globaldefs::NamingAttributes_T name;
   string userLabel;
   string nativeEMSName;
   string owner;
   ServiceCategory_T serviceCategory;
   TrafficParameterList_T trafficParameters;
   string conformanceDefinition;
   globaldefs::NVSList_T additionalInfo;
};
```

General comment	
A Traffic Descriptor is a collection of attributes which are used to define bandwidth and Quality of Service characteristics on a TP. Traffic Descriptors can be created by the NMS or the EMS.	
Attribute name	**Comment**
➢ name	The name represents the name of the Traffic Descriptor which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of EMS. It is a read-only attribute.

➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ nativeEMSName	The name represents how the Traffic Descriptor is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through).
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ serviceCategory	The Service Category contains quality of Service characteristics of the Traffic Descriptor.
➢ trafficParameters	The Traffic Parameter name/values associated with the Traffic Descriptor.
➢ conformanceDefinition	The conformance definition for the Traffic Descriptor from UNI 4.1, UNI 4.0 and UNI 3.1 standards. This field is nullable if the conformance definition does not apply.
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled.

6.1.106. TDCreateData_T

```
struct TDCreateData_T {
   string userLabel;
   boolean forceUniqueness;
   string owner;
   ServiceCategory_T serviceCategory;
   TrafficParameterList_T trafficParameters;
   string conformanceDefinition;
   globaldefs::NVSList_T additionalInfo;
  };
```

General comment	
TDCreateData_T is used when traffic descriptors are created by the NMS.	
Attribute name	**Comment**
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst TDs of the EMS. The operation will fail if userLabel is already in use.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ serviceCategory	The Service Category contains quality of Service characteristics of the Traffic Descriptor.
➢ trafficParameters	The Traffic Parameter name/values associated with the Traffic Descriptor.
➢ conformanceDefinition	The conformance definition for the Traffic Descriptor from standards. See TrafficDescriptor_T.
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modelled. It is a readonly attribute.

6.1.107. TrafficDescriptorList_T

typedef sequence<TrafficDescriptor_T> TrafficDescriptorList_T;

A list of Traffic Descriptors.

6.1.108. ServiceCategory_T

```
enum ServiceCategory_T {
   SC_CBR,
   SC_VBRRT,
   SC_VBRNRT,
   SC_ABR,
   SC_UBR,
   SC_GFR,
   SC_NA
  };
```

General comment	
Represents Service Categories used by Transmission Descriptors. The current Service Categories are defined as follows (note that Guaranteed Bit Rate (GBR) is not defined because it is generally not used).	
Attribute name	**Comment**
➢ SC_CBR	SC_CBR = Constant Bit Rate. For connections that require a sstatic amount of bandwidth that is continuously available during the lifetime of the connection.
➢ SC_VBRRT	SC_VBRRT = Variable Bit Rate - Real-Time. For connections that require tightly constrained delay and delay variation
➢ SC_VBRNRT	SC_VBRNRT = Variable Bit Rate - NonReal-Time. For connections that have bursty traffic.
➢ SC_ABR	SC_ABR = Available Bit Rate. For connections that do not require bounding the delay or delay variation. Not intended for real-time applications.
➢ SC_UBR	Unspecified Bit Rate. For connections that do not require tightly constrained delay and delay variation.
➢ SC_GFR	Guaranteed Frame Rate. For non-real-time applications that may require a minimum rate guarantee and can benefit from accessing additional bandwidth dynamically available in the network.
➢ SC_NA	SC_NA = Not Applicable

6.1.109. TrafficParameterList_T

typedef globaldefs::NVSList_T TrafficParameterList_T;

TrafficParametersList_T is used to specify the parameters used in Transmission Descriptors. The list is a sequence of name/value pairs (NVSList_T).

6.1.110. TransmissionDescriptor_T

```
struct TransmissionDescriptor_T {
    globaldefs::NamingAttributes_T name;
    string userLabel;
    string nativeEMSName;
    string owner;
    transmissionParameters::LayeredParameterList_T transmissionParams;
    globaldefs::NVSList_T additionalTPInfo;
    globaldefs::NamingAttributes_T containingTMDName;
```

```
  string externalRepresentationReference;
  globaldefs::NVSList_T additionalInfo;
};
```

General comment	
A Transmission Descriptor (TMD) is a collection of attributes which are used to multi-layered transmission parameters are contained by the EMS and additional info parameters on a TP. Transmission Descriptors can be created by the NMS or the EMS.	
Attribute name	**Comment**
➢ name	The name represents the name of the Transmission Descriptor which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the EMS context. It is a read-only attribute.
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read-write attribute.
➢ nativeEMSName	The nativeEMSName represents how the Transmission Descriptor is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through). If supported by the EMS, this attribute can be set by the NMS through the Common_I interface service common::Common_I::setNativeEMSName() . It is a read-write attribute.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setOwner() . It is a read-write attribute.
➢ transmissionParams	A list of transmission parameters which can be set and retrieved at a specified layer on a TP having this TMD assigned as egress or ingress TMD. It is a read-only attribute.
➢ additionalTPInfo	Additional info parameters which can be set and/or retrieved on a TP having this TMD assigned as egress or ingress TMD. It is a read-only attribute.
➢ containingTMDName	The name of another TMD which is considered to contain this TMD. The containment semantics is that this TMD inherits the layered transmission parameters and additional TP information from the containing TMD. It is a read-only attribute.
➢externalRepresentationReference	A means to store at the EMS a reference to the external representation of the TMD (e.g., an XML file name). The contents of this attribute is opaque at the NML-EML interface and not utilized. It is a read-only attribute.
➢additionalInfo	This allows the communication from the EMS to the NMS, and vice versa, of additional information that isn't explicitly modelled, except that some parameter names and values may be pre-defined. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setAdditionalInfo() . It is a read-write attribute but some parameters may be read-only.

6.1.111. TMDCreateData_T

```
struct TMDCreateData_T {
   string userLabel;
   boolean forceUniqueness;
   string owner;
   transmissionParameters::LayeredParameterList_T transmissionParams;
   globaldefs::NVSList_T additionalTPInfo;
```

```
 globaldefs::NamingAttributes_T containingTMDName;
 string externalRepresentationReference;
 globaldefs::NVSList_T additionalCreationInfo;
};
```

General comment	
The attributes required for the creation of a transmission descriptor on the EMS are packaged together in a TMDCreateData structure which the NMS will pass to the EMS at TMD creation time. These are the read-create attributes of the TMD.	
Attribute name	**Comment**
➢ userLabel	The userLabel is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel() . It is a read/write attribute and may be empty.
➢ forceUniqueness	Specifies whether uniqueness of the userLabel is required amongst TMDs of the EMS. The operation will fail if userLabel is already in use.
➢ owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner() . It is a read/write attribute and may be empty.
➢ transmissionParams	A list of transmission parameters which can be set and retrieved at a specified layer on a TP having this TMD assigned as egress TMD or ingress TMD.
➢ additionalTPInfo	Additional info parameters which can be set and retrieved on a TP having this TMD assigned as egress TMD or ingress TMD.
➢ containingTMDName	The name of another TMD which is considered to contain this TMD. The semantics of the containment is that the TMD to be created shall inherit the layered transmission parameters and additional TP information from the containing TMD.
➢ externalRepresentationReference	A means to store at the EMS a reference to the external representation of the TMD (e.g., an XML file name). The contents of this attribute is opaque at the NML-EML interface and not utilized.
➢ additionalCreationInfo	Some additional creation information may be specified by the NMS. This information may or may not become a part of the TMD's additionalInfo attribute. The list may be empty.

6.1.112. TransmissionDescriptorList_T

typedef sequence<TransmissionDescriptor_T> TransmissionDescriptorList_T;

A list of Transmission Descriptors.

6.1.113. LayerRate_T

typedef short LayerRate_T;

The LayerRate_T value is used to identify: the Layer of a TTP/CTP, the characteristic information of a PTP/FTP, the Layer/Rate of a connection.
Each LayerRate_T may be used in conjunction with a CTP/TTP/FTP of that layer. In the SDH definitions, the G.805 CP and G.805 Termination Function layer names differ, for example TU12 is used for the G.805 CP and VC12 for the corresponding G.805 Termination Function. In this interface definition the TU and VC definitions have been combined into a single composite layer and this has been named to include the equivalent SONET layer, e.g. LR_VT2_and_TU12_VC12.
Certain specific LayerRates have been included to allow for description of Ports (Physical Termination Points - PTPs) in SONET equipment.
Any extension to the list of integers defined here will be agreed upon through a formal process. They will be added at the end of the range. The type of the LayerRate has been made a 'short' rather than an enum to allow new rates to be added without changing the IDL interface. The value for any new rates would have to

be agreed by client and server and will be included in the next IDL release for documentation. Other layerRates may be added with the approval of the Specification Authority.
The interface definition allows for failed specification of layer in the connection creation service. This is to allow the EMS to make the choice of specific layer of the connection to support the requested signal flow. The layer specified must be supported by the physical termination identified. The layer chosen by the subnetwork will have the same rate as that requested or will have a greater capacity.
For example, LR_T1_and_DS1_1_5M may be used in place of LR_VT1_5_and_TU11_VC11 where the NMS client does not want to dictate to the EMS the actual solution to a connection request. In this case the EMS may choose a VT1.5 connection to join the two CTPs/FTPs identified in the request. The EMS may reject a request for a connection at a layer that it can not specifically support.

6.1.114. LayerRateList_T

typedef sequence<LayerRate_T> LayerRateList_T;

Set of LayerRate_T.

6.1.115. LayeredParameters_T

```
struct LayeredParameters_T
  {
    LayerRate_T layer;
    globaldefs::NVSList_T transmissionParams;
  };
```

General comment	
The LayeredParameters_T struct will include the layer rate with the applicable list of transmissions parameters.	
Attribute name	**Comment**
➢ layer	Represents the layer to which the parameters apply.
➢ transmissionParams	Name value pair list

6.1.116. LayeredParameterList_T

typedef sequence<LayeredParameters_T> LayeredParameterList_T;

Set of LayeredParameters_T.
LayeredParameterList_T is used in the TP definition and as a consequence is available for use in a connection specification where a TP is specified.

6.1.117. Destination_T

typedef string Destination_T;

Destination specification for FTP transfer of history PM data.
In the request for retrieval of history PM Data requests with PM data transfer using FTP, the NMS may specify the target destination for the PM Data file (operation getHistoryPMData()). Within the destination field, the hostname of the destination machine and the full path name of the target file (including the directory name and the file name) within the destination machine are specified. Hostname and full path name are separated by a colon (:). The '/' or '\' in path name are dependent on the destination and managed by the client.

6.1.118. Granularity_T

typedef string Granularity_T;

Count period (interval) for which PM data may be collected (and subsequently retrieved).
The format is one of:

"nmin" (representing n minute granularity)
for values of n that are not multiples of 60
"nh" (representing n hour granularity)
"NA" (representing instantaneous measurements)

Standard values are:
"15min"
"24h"
"NA" (for current instantaneous measurements)

6.1.119. GranularityList_T

typedef sequence<Granularity_T> GranularityList_T;

Sequence of Granularity_T.

6.1.120. PMLocation_T

typedef string PMLocation_T;

PM parameters may relate to measurements taken on receive or transmit traffic either at the named TP (PML_NEAR_END_Rx/Tx) or at the TP at the far end of the trail (circuit/path) connected to the named TP (PML_FAR_END_Rx/Tx). Alternatively, the PM parameters may be bidirectional (PML_BIDIRECTIONAL) e.g. resulting from a second by second summation and evaluation of both far and near TPs .

Valid values are:
"PML_NEAR_END_Rx"
"PML_FAR_END_Rx"
"PML_NEAR_END_Tx"
"PML_FAR_END_Tx"
"PML_BIDIRECTIONAL"
"PML_CONTRA_NEAR_END_Rx"
"PML_CONTRA_FAR_END_Rx"

6.1.121. PMLocationList_T

typedef sequence<PMLocation_T> PMLocationList_T;

A set of PMLocation_T. In PM related operations, PM parameters may be specified per nearEnd and/or farEnd category, or bidirectional category.

6.1.122. PMParameterName_T

typedef string PMParameterName_T;

Holds the name of Performance Measurements. It has been defined as a string to accommodate backward compatibility and proprietary extension.

6.1.123. PMParameterNameList_T

typedef sequence<PMParameterName_T> PMParameterNameList_T;

List of PMParameterName_T.

6.1.124. PMParameter_T

```
struct PMParameter_T {
        PMParameterName_T  pmParameterName;
        PMLocation_T pmLocation;
};
```

General comment	
Used to represent the PM parameter name of the PM measure qualified by its location.	
Attribute name	**Comment**
➢ pmParameterName	Represents the name of the PM Measure.
➢ pmLocation	Represents the location of the PM Measure.

6.1.125. PMParameterList_T

typedef sequence <PMParameter_T> PMParameterList_T ;

List of PMParameter_T.

6.1.126. PMThresholdType_T

```
enum PMThresholdType_T
        {
                TWM_HIGHEST,
                TWM_HIGH,
                TWM_LOW,
                TWM_LOWEST
        };
```

Describes threshold types (watermark levels) for TCA parameters.

The TWM_HIGH and TWM_HIGHEST types are used for TCAs that are raised when the measured value goes above the threshold. The TWM_LOW and TWM_LOWEST types are used for TCAs that are raised when the measured value goes below the threshold (only apply to gauges).

When there is only one level of TCA trigger, only TWM_HIGH and/or TWM_LOW are used. When there are two levels of TCA triggers, TWM_HIGHEST and/or TWM_LOWEST are used in addition.

6.1.127. PMThresholdValue_T

```
struct PMThresholdValue_T
        {
                PMParameterName_T  pmParameterName;
                PMLocation_T       pmLocation;
                PMThresholdType_T  thresholdType;
                boolean            triggerFlag;
                float              value;
                string             unit;
        };
```

General comment	
Holds a PM threshold value.	
Attribute name	**Comment**
➢ pmParameterName	Holds the name of the PM Measure. It has been defined as a string to accomodate backward compatibility and proprietary extensions.
➢ pmLocation	Represents the location of the PM Measure.
➢thresholdType	Represents the type of threshold value.
➢triggerFlag	Indicates if the threshold is for the trigger (true) or the clear (false).
➢value	Threshold value for the PM parameter (float used to incorporate gauge PM parameters)

➢unit	Free format string describing the units for the threshold value.

6.1.128. PMThresholdValueList_T

typedef sequence<PMThresholdValue_T> PMThresholdValueList_T ;

Sequence of PMThresholdValue_T.

6.1.129. TCAParameter_T

```
struct TCAParameter_T  {
        performance::PMParameterName_T pmParameterName;
        performance::Granularity_T granularity;
        performance::PMLocation_T pmLocation;
        performance::PMThresholdType_T thresholdType;
        boolean triggerFlag;
        float value;
        string unit;
    };
```

General comment	
The TCAParameter_T struct contains the TCA parameters assigned to a TCA ParameterProfile. They can be retrieved and modified by the NMS through the operations getTCAParameterProfile and setTCAParameterProfile provided by the PerformanceManagementMgr_I interface.	
Attribute name	**Comment**
➢ pmParameterName	Holds the name of the Performance Parameters. It has been defined as a string to accomodate proprietary extension.
➢ granularity	This attribute defines the granularity of the TCA Parameter. For counters two values "15min" or "24h" are supported. For current instantaneous measurements the granularity does not apply which is indicated by the value "N/A".
➢ pmLocation	Represents the location of the PM measure.
➢ pmThresholdType	Represents the type of threshold value.
➢ triggerFlag	Indicates if the threshold is for the trigger (true) or the clear (false).
➢ value	Threshold for parameter (float used to incorporate gauge PM parameters).
➢unit	Free format string describing the units for the threshold value.

6.1.130. TCAParameterList_T

typedef sequence<TCAParameter_T> TCAParameterList_T;

Sequence of TCAParameter_T.

6.1.131. TCAParameterProfile_T

```
struct TCAParameterProfile_T  {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        globaldefs::NVSList_T additionalInfo;
        transmissionParameters::LayerRate_T layerRate;
```

```
        TCAParameterList_T tcaParameterList;
};
```

General comment	
This struct is used to represent the TCA (threshold crossing alert)Parameter Profile (TCAPP) object class containing per layer rate a set of PM parameter/threshold value pairs per granularity (e.g. VC12: ES / 50;15min). A profile may be created and deleted by EMS or NMS. Instances of this class may be associated to TPs. A list of references to all associated TPs will be maintained in the profile.	
Attribute name	**Comment**
➢ name	The name represents the name of the TCA Parameter Profile which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. It is a readonly attribute.
➢ userLabel	The user label of the TCA Parameter Profile can be set by EMS or NMS upon creation. After creation the user label can be changed by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read-write attribute.
➢ nativeEMSName	The name represents how the TCA Parameter Profile is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through).After a TCA Parameter Profile has been created, the nativeEMSName may be changed by the NMS, if the EMS supports this functionality, using the common::Common_I::setNativeEMSName() operation of the Common_I interface service. It is a read/write attribute and never set to NULL string.
➢ owner	The owner attribute of the TCA Parameter Profile indicates the ownership. The owner is provisionable by the NMS. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢ additionalInfo	This attribute allows the communication from the EMS to the NMS (and vice versa from NMS to EMS) of additional information which isn't explicitly modelled. This may be an empty list. It is a read/write attribute.
➢ layerRate	Defines the layer rate of the Profile to which the thresholds apply. It is a read only attribute.
➢ tcaParameterList	Provides the list of Performance Parameters. threshold value pairs including location and granularity. It is a read/write attribute.

6.1.132. TCAParameterProfileList_T

typedef sequence< TCAParameterProfile_T> TCAParameterProfileList_T;

Sequence of TCAParameterProfile_T.

6.1.133. PMMeasurement_T

```
struct PMMeasurement_T
        {
                PMParameterName_T pmParameterName;
                PMLocation_T pmLocation;
                float value;
                string unit;
                string intervalStatus;
```

```
};
```

General comment	
Holds a PM measurement and description of validity.	
Attribute name	**Comment**
➢ pmParameterName	Represents the name of the PM Measure.
➢ pmLocation	Represents the location of the PM Measure.
➢ value	Value of PM parameter (float used to incorporate gauge PM parameters).
➢ unit	Free format string describing the units for the PM measurement value.
➢ intervalStatus	The following string values, to indicate the relationship of measurement value and measurement interval, are allowed: "Valid" - valid data, "Incomplete" - data not available for the complete interval, "Invalid" - data available but marked as invalid for the interval (when the EMS can not distinguish incomplete measurements from invalid measurements, "Invalid" will be used) "Unavailable" - no data available at all for this interval, "Zero-suppressed" - zero-suppressed intervals.

6.1.134. PMMeasurementList_T

typedef sequence<PMMeasurement_T> PMMeasurementList_T ;

Sequence of PMMeasurement_T.

6.1.135. PMData_T

```
struct PMData_T {
        globaldefs::NamingAttributes_T tpName;
        transmissionParameters::LayerRate_T layerRate;
        Granularity_T granularity;
        globaldefs::Time_T retrievalTime;
        PMMeasurementList_T pmMeasurementList;
};
```

General comment	
This struct is used to pass current PM data for one collection bin from the EMS to the NMS. It is also used to pass historic PM data across the NML-EML interface when the file transfer capability via FTP is not preferred.	
Attribute name	**Comment**
➢ tpName	The name of the termination point. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ layerRate	Layer of the collected PM data.
➢ granularity	Identifies the granularity of the data.
➢ retrievalTime	Point in time at which PM measurement was obtained from the ME.
➢ pmMeasurementList	PM measurement values.

6.1.136. PMDataList_T

typedef sequence<PMData_T> PMDataList_T;

Sequence of PMData.

6.1.137. PMTPSelect_T

```
struct PMTPSelect_T {
        globaldefs::NamingAttributes_T name;
        transmissionParameters::LayerRateList_T layerRateList;
        PMLocationList_T pMLocationList;
        GranularityList_T granularityList;
            };
```

General comment	
Used to specify the scope/target for a PM operation.	
Attribute name	**Comment**
➢ name	The name of the object to which this selection applies. (i.e. the target of the PM operation). This may be: ManagedElement: The selection applies to all termination points contained within the ManagedElement. TerminationPoint: The selection applies only to the named termination point which will be a PTP, an FTP or a CTP. Unlike the case for ManagedElement, the operation will not apply to any "contained" TPs when a PTP, an FTP or CTP is specified. Note: As stated in the requirements (TMF513), the possible scopes do not currently include "SNC" and do not include "TL".
➢ layerRateList	Identifies the set of layers to which the selection applies. An empty list means all layers supported by the equipment.
➢ granularityList	Identifies set of granularities to which the selection applies. An empty list means all granularities supported by the equipment.

6.1.138. PMTPSelectList_T

typedef sequence<PMTPSelect_T> PMTPSelectList_T;

Sequence of PMTPSelect. Used by several methods to identify sets of PM parameters to which to apply the operation.

6.1.139. HoldingTime_T

```
struct HoldingTime_T {
        short storeTime24hr;
        short storeTime15min;
    };
```

General comment	
By means of this struct, the EMS informs the NMS about the minimum time it holds 24h PM data records and 15min PM data records. This minimum time refers to the time after the particular collection bins are closed.	
Attribute name	**Comment**
➢ storeTime24hr	Minimum store time in hours for 24h data.
➢ storeTime15min	Minimum store time in hours for 15min data.

6.1.140. TCAParameters_T

```
struct TCAParameters_T  {
        transmissionParameters::LayerRate_T layerRate;
        Granularity_T granularity;
        PMThresholdValueList_T tcaTypeValues;
    };
```

General comment	
The TCAParameters structure is used to get or set one or more PM Threshold values for a given TP/layer/granularity measurement point.	
Attribute name	**Comment**
➢ layerRate	Identifies the layer to which tcaTypeValues applies.
➢ granularity	Identifies the granularity of the data. For gauges, this should be "NA".
➢tcaTypeValues	Identifies threshold values. for PM parameters, PM locations and threshold types, and indicates Trigger/Clear properties.

6.1.141. AdministrativeState_T

```
enum AdministrativeState_T {
        AS_Locked,
        AS_Unlocked
    };
```

AdministrativeState_T indicates whether a specified function is enabled (unlocked) or disabled (locked).

6.1.142. PMThreshold_T

```
struct PMThreshold_T {
        PMThresholdType_T thresholdType;
        boolean triggerFlag;
        float value;
        string unit;
    };
```

General comment	
This structure represents a Performance Monitoring Threshold.	
Attribute name	**Comment**
➢ thresholdType	Represents the type of threshold value.
➢ triggerFlag	Indicates if the threshold is for the trigger (TRUE) or the clear (FALSE).
➢ value	Threshold value (float used to incorporate gauge PM parameters).
➢unit	Free format string describing the units for the threshold value.

6.1.143. PMThresholdList_T

```
typedef sequence <PMThreshold_T> PMThresholdList_T;
```

Sequence of Performance Monitoring Thresholds.

6.1.144. PMParameterWithThresholds_T

```
struct PMParameterWithThresholds_T {
        PMParameterName_T pmParameterName;
        PMThresholdList_T pmThresholdList;
};
```

General comment	
The PMParameterWithThresholds_T structure describes the list of thresholds associated with a particular PM Parameter.	
Attribute name	**Comment**
➢ pmParameterName	Represents the name of the PM Measure.
➢ pmThresholdList	Holds the list of PM thresholds associated with the named PM parameter.

6.1.145. PMParameterWithThresholdsList_T

```
typedef sequence <PMParameterWithThresholds_T>  PMParameterWithThresholdsList_T;
```

Sequence of Performance Monitoring parameters with their respective thresholds.

6.1.146. PMP_T

```
struct PMP_T {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        PMParameterWithThresholdsList_T pmParameterWithThresholdsList;
        AdministrativeState_T monitoringState;
        AdministrativeState_T supervisionState;
        globaldefs::NVSList_T additionalInfo;
        Granularity_T granularity;
};
```

General comment	
This structure represents one performance monitoring point (PMP). It is determined by the containing PTP or CTP and by the layer rate, the PM location and the granularity the monitoring is done for. PMP objects are created by the EMSs only. They are contained in TPs and represent the PM capabilities of their containing TP(s).	
Attribute name	**Comment**
➢ name	The name represents the name of the PMP which is assigned by the EMS upon creation. It is a read only attribute. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the PerformanceManagementMgr_I. This is achieved by the MTNM naming hierarchy and by the following naming convention for the PMP name value: The value of the PMP name will follow ALCATEL name.
➢ userLabel	The userLabel is a friendly name that the operator wants to give to the PMP. Typical expectations of the operator is that the same name is seen on all operation systems. This is set by the NMS and could be displayed on the EMS based on each systems' capabilities. THIS IS NOT A MANDATORY EXPECTATION, but is left to the implementation of the EMS. This attribute can be set by NMS through the Common_I

	interface service common::Common_I::setUserLabel(). It is a read/write attribute.
➢nativeEMSName	Represents how the PMP refers to itself on PMP displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through). May be a null string.
➢owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
➢pmParameterWithThresholdsList	List of PM parameters supported by the PMP, e.g. PMP_ES, PMP_SES, etc. together with all their thresholds. Please note in this version, the threshold is not supported.
➢monitoringState	The monitoring state indicates whether performance monitoring is enabled or disabled for the PMP . In this version, always set to true(AS_Unlocked).
➢supervisionState	The supervision state indicates whether performance threshold supervision is enabled or disabled for the PMP In this version, always set to true(AS_Unlocked).
➢additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modeled. It is a read-only attribute.
➢granularity	This attributes presents the supported granularity of this PMP.

6.1.147. PMPList_T

typedef sequence<PMP_T> PMPList_T;

Sequence of PMP_T. Used to convey a batch of collected Performance Monitoring Points data.

6.1.148. PMMonitorOrReportStatus_T

```
enum PMMonitorOrReportStatus_T
        {
                Locked,
                Unlocked
        };
```

Describes PM data collection or report status. For collection status, Locked indicates the suspend status and For report status, Locked indicates that the collection plan data is not report to NMS automatically and Unlocked indicates that the collection plan data is report to NMS automatically.

6.1.149. PMCollectionPlan_T

```
struct PMCollectionPlan_T
        {
                string Id;
                PMTPSelect_T tpSelected;
                PMLocationList_T pMLocationList;
                globaldefs::Time_T beginTime;
                globaldefs::Time_T endTime;
                globaldefs::Time_T reportInterval;
                PMMonitorOrReportStatus_T pmMonitorStatus;
                PMMonitorOrReportStatus_T pmReportStatus;
        };
```

General comment

Describes PM data collection plan.	
Attribute name	**Comment**
➢ Id	An unique ID for the plan.
➢ tpSelected	This parameter specifies which PM data to collect (ME or TP name).
➢ pMLocationList	The PM location list.(currently not used.)
➢ beginTime	Specifies the start of the time window for collection (included).
➢ endTime	Specifies the end of the time window for collection (excluded).
➢ reportInterval	Specifies the report interval, must be multiple of 15 minutes.
➢ pmMonitorStatus	Specifies the plan is to resume or suspend.
➢ pmReportStatus	Specifies whether the plan data is to report to NMS automatically or not.

6.1.150. PMCollectionPlanList_T

typedef sequence<PMCollectionPlan_T> PMCollectionPlanList_T;

Sequence of PMCollectionPlan_T.

6.1.151. StpProtocolType_T

```
enum StpProtocolType_T
 {
  SPT_none,
  SPT_manualDisable,
  SPT_stp,
  SPT_rstp,
  SPT_pervlanstp,
  SPT_mstp,
  SPT_rstpPlus,
  SPT_mstpPlus
 };
```

General comment	
STP Type configuration.	
Attribute name	**Comment**
none	Not used
manualDisable	Disable the protocol for the bridge
stp	Enable STP and synchronously the Rapid STP is disabled, one stp, no vlan.
rstp	Enable Rapid STP and synchronously STP is disabled, one stp, no vlan.
pervlanstp	Enable per-VLAN RSTP, per vlan per STP.
mstp	Enable RSTP-based MSTP, multi-vlan by one STP.
rstpPlus	Enable optimization for RSTP.
mstpPlus	Enable optimization for MSTP.

6.1.152. BridgeType_T

```
enum BridgeType_T
 {
        BT_none,
        BT_bridge8021D,
        BT_bridge8021Q,
        BT_bridge8021ad
 };
```

General comment	
Indicates what types of mechanism this Bridge perfoms.	
Attribute name	**Comment**
none	This NE cannot perform any bridge mechanism.
bridge8021D	This Bridge performs 802.1D mechanism and can not support VLAN,Only SPT_stp, SPT_rstp is accepted in this type.
bridge8021Q	This Bridge performs 802.1Q mechanism, all protocol type is accepted type.
bridge8021ad	This Bridge performs 802.1ad mechanism all protocol type is accepted in the type.

6.1.153. BridgeLogicPort_T

```
struct BridgeLogicPort_T
 {
        unsigned long portNumber;
        globaldefs::NamingAttributes_T relatedPhysicalPort;
        string portType;
        string encapsulatedType;
        boolean portEnable;
 };
```

General comment	
The BridgeLogicPort_T structure indicates the logic ports included by virtual bridge.	
Attribute name	**Comment**
portNumber	It is logic port Id.
relatedPhysicalPort	It is the tp name which corresponds to logicPort.
portType	if it is edegeport, portType is "PE", otherwise "P"
portEnable	The enabled/disabled status of the port.

6.1.154. BridgeLogicPortList_T

```
typedef sequence<BridgeLogicPort_T> BridgeLogicPortList_T;
```

An bridge logic port list provides a listing of all port included by a certain virtual bridge.

6.1.155. VirtualBridgeParameter_T

```
struct VirtualBridgeParameter_T
```

```
{
        long vStpPriority;
        BridgeAddress vStpBridgeAddress;
        TimeTicks vStpTimeSinceTopologyChange;
        unsigned long vStpTopChanges;
        BridgeAddress vStpDesignatedRoot;
       long vStpRootCost;
       long vStpRootPortNumber;
       long vStpNextBestRootCost;
       long vStpNextBestRootPortNumber;
       long vStpMaxAge;
       long vStpHelloTime;
       long vStpHoldTime;
       long vStpForwardDelay;
       long vStpBridgeMaxAge;
       long vStpBridgeHelloTime;
       long vStpBridgeForwardDelay;
       long vStp1x1VlanNumber;
};
```

General comment	
This struct is used to represent the virtual bridge object class containing it's parameter.	
Attribute name	**Comment**
vStpPriority	the range is (0~65535) and the step is 4096.
vStpBridgeAddress	The bridge identifier of this spanning tree instanceas determined by the Spanning Tree Protocol.
vStpTimeSinceTopologyChange	The time (in hundredths of a second) since the last time a topology change was detected by this spanning tree instance.
vStpTopChanges	The total number of topology changes detected bythis spanning tree instance since the management entity was last reset or initialized.
vStpDesignatedRoot	The bridge identifier of the root of the spanning tree as determined by the Spanning Tree Protocol as executed by this instance. This value is used as the Root Identifier parameter in all Configuration Bridge PDUs originated by this node for this instance.
vStpRootCost	The cost of the path to the root as seen from this bridge for this spanning tree instance.
vStpRootPortNumber	The port ifindex of the port which offers the lowest cost path from this bridge to the root bridge for this spanning tree instance.
vStpNextBestRootCost	The cost of the path to the root through the next best root port as seen from this bridge for this spanning tree instance.
vStpNextBestRootPortNumber	The port ifindex of the next port which offers the lowestcost path from this bridge to the root bridge for this spanning tree instance. This port will become root port if the actual root port goes down.
vStpMaxAge	The maximum age of Spanning Tree Protocol information learned from the network on any port before it is discarded, in units of hundredths of a second. This is the actual value that this spanning tree instance is currently using.
vStpHelloTime	The amount of time between the transmission of Configuration bridge PDUs by this instance on any port when it is the root of the spanning tree or trying to become so, in units of hundredths of a second. This is the actual value that this instance is currently using.
vStpForwardDelay	This time value, measured in units of hundredths of a second, controls how fast a port changes its spanning state when moving towards the Forwarding state. The value determines how long the port stays in each of the Listening and Learning

	states, which precede the Forwarding state. This value is also used, when a topology change has been detected and is underway, to age all dynamic entries in the Forwarding Database.
vStpBridgeMaxAge	The value that all bridges use for MaxAge when this bridge is acting as the root. Note that 802.1D-1990 specifies that the range for this parameter is related to the value of vStpBridgeHelloTime. The granularity of this timer is specified by 802.1D-1990 to be 1 second. An agent may return a badValue error if a set is attempted to a value which is not a whole number of seconds. In case of enhanced Rapid Spanning tree the upperlimit of the admitted range changes from 4000 to 7700.
vStpBridgeHelloTime	The value that all bridges use for HelloTime when this bridge is acting as the root. The granularity of this timer is specified by 802.1D-1990 to be 1 second. An agent may return a badValue error if a set is attempted to a value which is not a whole number of seconds.
vStpBridgeForwardDelay	The value that all bridges use for ForwardDelay when this bridge is acting as the root. Note that 802.1D-1990 specifies that the range for this parameter is related to the value of vStpBridgeMaxAge. The granularity of this timer is specified by 802.1D-1990 to be 1 second. An agent may return a badValue error if a set is attempted to a value which is not a whole number of seconds.
vStp1x1VlanNumber	The Vlan number of the instance only if it is significant case of 1 x 1

6.1.156. VirtualBridge_T

```
struct VirtualBridge_T
  {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        boolean enableState;
        BridgeType_T  vBridgeType;
        StpProtocolType_T  stpProtocolType;
        BridgeLogicPortList_T logicPortList;
        VirtualBridgeParameter_T attrs;
  };
```

General comment	
This struct is used to represent the virtual bridge object class containing it's attributes	
Attribute name	**Comment**
name	The name represents the name of the virtual brideg. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. Naming rule is "bridge_<id>" It is a readonly attribute.
userLabel	The user label of the virtual bridge can be set by EMS or NMS upon creation. After creation the user label can be changed by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read-write attribute.
nativeEMSName	The name represents how the virtual bridge is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to

	aid relating information presented on NMS displays to EMS displays (via GUI cut through).After a virtual bridge has been created, the nativeEMSName may be changed by the NMS, if the EMS supports this functionality, using the common::Common_I::setNativeEMSName() operation of the Common_I interface service. It is a read/write attribute and never set to NULL string.
enableState	Provides the state of virtual bridge.
logicPortList	Represents the logic port list contained by vritual bridge. The "bridge_0" contains all ETB ports, others contains the ports through vlan's egree ports.

6.1.157. VirtualBridgeList_T

typedef sequence<VirtualBridge_T> VirtualBridgeList_T;
Sequence of VirtualBridge_T.

6.1.158. Vlan_T

```
struct Vlan_T
  {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        string vlanStatus;
        unsigned long vlanId;
        globaldefs::NamingAttributes_T vBridgeName;
        globaldefs::NamingAttributesList_T egressPortList;
        globaldefs::NamingAttributesList_T untaggedPortList;
  };
```

General comment	
This struct is used to represent the Vlan object class containing it's parameter.	
Attribute name	**Comment**
name	The name represents the name of the vlan. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. It is a readonly attribute.
userLabel	The user label of the vlan can be set by EMS or NMS upon creation. After creation the user label can be changed by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read-write attribute.
nativeEMSName	The name represents how the vlan is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through).After a virtual bridge has been created, the nativeEMSName may be changed by the NMS, if the EMS supports this functionality, using the common::Common_I::setNativeEMSName() operation of the Common_I interface service. It is a read/write attribute and never set to NULL string.
vlanStatus	Only support "other","permanent","dynamicGvrp"
vlanId	The VLAN-ID or other identifier referring to this VLAN.

egressPortList	The set of ports which are transmitting traffic for this VLAN as either tagged or untagged frames. 802.1AD/MSTP, egressPortList of vlan1 contains all ports. 802.1Q, egressPortList of vlan1 contains all ETB(Ethernet Bridge) ports.
untaggedPortList	The set of ports which are transmitting traffic forthis VLAN as untagged frames.

6.1.159. VlanList_T

typedef sequence<Vlan_T> VlanList_T;
Sequence of Vlan_T.

6.1.160. StpMode_T

```
struct StpMode_T
{
        StpProtocolType_T stpProtocolType;
        BridgeType_T bridgeType;
};
```

6.1.161. VStpPortEnable_T

```
enum VStpPortEnable_T
 {
        Enabled,
        Disabled
 };
```
The enabled/disabled status of the port.

6.1.162. VStpPortManualMode_T

```
enum VStpPortManualMode_T
 {
  No,
  Blocking,
  Forwarding
 };
```
The port's manual mode for this spanning tree instance. This mode defines the way the state of the port(vStpPortState) is managed. The mode can be dynamic (1) (managed by the Spanning Tree), or manual (blocking(2) or forwarding(3)). In manual mode,the port is not involved in the Spanning tree algorithm computation.

6.1.163. VStpPortRole_T

```
enum VStpPortRole_T
 {
        Root,
        Designated,
        Alternate,
        Backup,
        PortDisabled,
        Master
 };
```
The port's role for this spanning tree instance.

6.1.164. VStpPortAdminConnectionType_T

```
enum VStpPortAdminConnectionType_T
 {
```

```
        Admin_nopointtopoint,
        Admin_pointtopoint,
        Admin_autopointtopoint,
        Admin_edgeport
 };
```

The administrative port's connection type for this spanning tree instance. This parameter is used to optimise the fast Spanning Tree (6.4.3 of P802.1w).
The default value is nopointtopoint(1). The value pointtopoint(2) forces the port to be treated as if it is connected through a point-to-point LAN segment to another switch. Set this parameter to autopointtopoint(3) makes the point-to-point status to be determined automatically by the MAC entity. The value edgeport(4) indicates that the port is considered to be an edge port (18.3.3 of P802.1t).

6.1.165. VStpPortOperConnectionType_T

```
enum VStpPortOperConnectionType_T
 {
        Oper_nopointtopoint,
        Oper_pointtopoint,
        Oper_autopointtopoint,
        Oper_edgeport
 };
```

The operational port's connection type for this spanning tree instance.This parameter is used to optimise the fast Spanning Tree.

6.1.166. VstpPortRcvdInternal_T

```
enum VstpPortRcvdInternal_T
 {
        External,
        Internal,
        Unkown
 };
```

The operational port's connection type for this spanning tree instance.This parameter is used to optimise the fast Spanning Tree.

6.1.167. VStpPortState_T

```
enum VStpPortState_T
 {
        SPS_disabled,
        SPS_blocking,
        SPS_listening,
        SPS_learning,
        SPS_forwarding
 };
```

The port's current state as defined by application of the Spanning Tree Protocol. This state controls what action a port takes on reception of a frame. For ports which are disabled(see vStpPortEnable), this object will have a value of disabled(1). Falcon does not support the broken(6) state as defined in RFC1493.

6.1.168. StpPortParam_T

```
struct StpPortParam_T
 {
        unsigned long vStpPortNumber;
        unsigned long vStpPortPriority;
        VStpPortState_T vStpPortState;
        VStpPortEnable_T vStpPortEnable;
        unsigned long vStpPortPathCost;
        BridgeAddress vStpPortDesignatedRoot;
        unsigned long vStpPortDesignatedCost;
        BridgeAddress vStpPortDesignatedBridge;
```

```
        unsigned long vStpPortDesignatedPtPrio;
        unsigned long vStpPortDesignatedPtNumber;
        unsigned long vStpPortForwardTransitions;
        VStpPortManualMode_T vStpPortManualMode;
        VStpPortRole_T vStpPortRole;
        unsigned long vStpPortPrimaryPortNumber;
        VStpPortAdminConnectionType_T vStpPortAdminConnectionType;
        VStpPortOperConnectionType_T vStpPortOperConnectionType;
        BridgeAddress vStpPortCistRegionRootId;
        unsigned long vStpPortCistPathCost;
        unsigned long vStpPortHelloTime;
        unsigned long vStpPortBridgeHelloTime;
        VstpPortRcvdInternal_T vstpPortRcvdInternal;
 };
```

This struct is used to represent the Stp port object class containing it's parameter.

6.1.169. ForwardingRecordState_T

```
enum ForwardingRecordState_T
{
        FRS_other,
        FRS_invalid,
        FRS_permanent,
        FRS_deleteOnReset,
        FRS_deleteOnTimeout,
        FRS_learned,
        FRS_self,
        FRS_mgmt
};
```

6.1.170. ForwardingRecord_T

```
struct ForwardingRecord_T
  {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        unsigned long vlanId;
        sequence<unsigned long> portNumberList;
        MacAddress macAddressValue;
        ForwardingRecordState_T state;
  };
```

General comment	
This struct is used to represent the StaticMulticast/StaticUnicast/DynamicalMulticast/DynamicalUnicast forwarding record object class containing it's parameter.	
Attribute name	**Comment**
name	String (vlanId+portId+macAddress)
owner	Is empty when dynamicast forwarding record.
vlanId	If not any vlan, it is always 1.
portNumberList	If is static forwarding record, it is used to represent allowed to go to ports,allowed to go to ports means: The set of ports for which a frame with a specific unicast address will be flooded in the event that it has not been learned. If is dynamicast forwarding record, it is used to represent egress ports,egress ports means:

	The set of ports to which frames received from a specific port and destined for a specific Multicast or Broadcast MAC address must be forwarded, regardless of any dynamic information e.g. from GMRP.
macAddressValue	string ,example 00:34:ae:ff:89:90
state	This object indicates the status of this entry. other(1): this entry is currently in use but the conditions under which it will remain so differ from the following values. invalid(2): writing this value to the object removes the corresponding entry. permanent(3): this entry is currently in use and will remain so after the next reset of the bridge. deleteOnReset(4): this entry is currently in use and will remain so until the next reset of the bridge. deleteOnTimeout(5): this entry is currently in use and will remain so until it is aged out.

6.1.171. ForwardingRecordList_T

typedef sequence<ForwardingRecord_T> ForwardingRecordList_T;
Sequence of ForwardingRecord_T.

6.1.172. ObjectType_T

```
enum  ObjectType_T
  {
    OT_EMS,
    OT_MANAGED_ELEMENT,
    OT_MULTILAYER_SUBNETWORK,
    OT_TOPOLOGICAL_LINK,
    OT_SUBNETWORK_CONNECTION,
    OT_PHYSICAL_TERMINATION_POINT,
    OT_CONNECTION_TERMINATION_POINT,
    OT_TERMINATION_POINT_POOL,
    OT_EQUIPMENT_HOLDER,
    OT_EQUIPMENT,
    OT_PROTECTION_GROUP,
    OT_TRAFFIC_DESCRIPTOR,
    OT_AID
  };
```

"OobjectType" is a filterable field attribute of all notifications,except NT_PROTECTION_SWITCH, NT_EPROTECTION_SWITCH,NT_FILE_TRANSFER_STATUS, NT_PM_SC, and NT_BACKUP_STATUS.
The enum avoids any uncertainty in the type of object and allows and allows simple filtering.
A notification must be reported against the correct object if it this object is modelled.
For alarms, the objectType OT_AID (Alarm Identifier) is used to represent the EMS object types that are not modelled but can emit alarms. Other notifications types should not be reported against AIDs. This objectType value is also used for new object types from release 3.0 onwards in order to guarantee backward compatibility of the interface.
Due to the requirement of strict backward compatibility the release 2 enum ObjectType_T cannot be extended to include new object types. Therefore OT_AID is used as an "escape value" for the field "objectType". Thus OT_AID may also represent new object types. To identify which object type applies,the new filterable field "objectTypeQualifier" is introduced which is of type string and whose values are as follows:
"" - indicates an AID
"OT_EPROTECTION_GROUP" - equipment protection group
"OT_TCA_PARAMETER_PROFILE" - TCA parameter profile
"OT_PMP" - performance monitoring point
"OT_GTP" - group TP
"OT_ASAP" - alarm severity assignment profile
"OT_TRANSMISSION_DESCRIPTOR" - transmission descriptor

If "objectTypeQualifier" is not present but "objectType" has the value OT_AID we are dealing with a proper AID.

6.1.173. ObjectTypeQualifier_T

typedef string ObjectTypeQualifier_T;
The ObjectTypeQualifier_T is used to identify object types defined in v3.0 and beyond. It is needed because the ObjectType_T enum cannot be extended for backward compatibility reasons.

6.1.174. PerceivedSeverity_T

```
enum  PerceivedSeverity_T
  {
    PS_INDETERMINATE,
    PS_CRITICAL,
    PS_MAJOR,
    PS_MINOR,
    PS_WARNING,
    PS_CLEARED
  };
```

The PerceivedSeverity_T values are consistent with ITU-T X.733 definitions.

6.1.175. AcknowledgeIndication_T

```
enum AcknowledgeIndication_T
  {
    AI_EVENT_ACKNOWLEDGED,
    AI_EVENT_UNACKNOWLEDGED,
    AI_NA
  };
```

AcknowledgeIndication_T describes the event acknowledge state.
AI_EVENT_ACKNOWLEDGED - provided in case of manual or auto acknowledgement
AI_EVENT_UNACKNOWLEDGED - provided if the event has not been acknowledged but the EMS supports acknowledgement for this event or in the case that the alarm has been previously acknowledged and then unacknowledged
AI_NA provided in case the EMS does not support acknowledgement for this event or does not support acknowledgement at all

6.1.176. AlarmTypeQualifier_T

```
enum AlarmTypeQualifier_T
  {
    ALARM,
    TCA
  };
```

Used to distinguish TCA from alarm.

6.1.177. PerceivedSeverityList_T

typedef sequence<PerceivedSeverity_T> PerceivedSeverityList_T;
List of PerceivedSeverity_T values.

6.1.178. ServiceAffecting_T

```
enum ServiceAffecting_T
  {
   SA_UNKNOWN,
   SA_SERVICE_AFFECTING,
   SA_NON_SERVICE_AFFECTING
  };
```

General comment	
ServiceAffecting_T describes the impact of a fault on monitored entities.	
Attribute name	**Comment**
SA_UNKNOWN	The EMS cannot determine if the condition affects service or not.
SA_SERVICE_AFFECTING	The EMS determines that the condition affects service.
SA_NON_SERVICE_AFFECTING	The EMS determines that the condition doesn not affect service.

6.1.179. ProbableCauseList_T

typedef sequence<string> ProbableCauseList_T;
Set of probable causes.

6.1.180. NameAndAnyValue_T

```
struct NameAndAnyValue_T
  {
    string name;
    any value;
  };
```

The NameAndAnyValue_T structure is provided when an any value is needed.

6.1.181. NVList_T

typedef sequence<NameAndAnyValue_T> NVList_T;
A list of (name=string, value=any) tuples. The OMG standard NVList. This is used for AttributeValueChange and StateChange notifications.

6.1.182. FileTransferStatus_T

```
enum FileTransferStatus_T
  {
   FT_IN_PROGRESS,
   FT_FAILED,
   FT_COMPLETED
  };
```

Describe transfer status type. This is used for file transfer protocol notifications.

6.1.183. EventList_T

typedef sequence<CosNotification::StructuredEvent> EventList_T;
Sequence of CosNotification::StructuredEvent.

6.1.184. SpecificProblem_T

typedef string SpecificProblem_T;
Element of SpecificProblemList_T.

6.1.185. SpecificProblemList_T

typedef sequence <SpecificProblem_T> SpecificProblemList_T;
The optional "X.733::SpecificProblems" parameter uses that type.
When present in an alarm notification,it identifies further refinements to the probable cause of the alarm.This is consistent with the ITU-T X.733 definition.

6.1.186. NotifIDList_T

typedef sequence<string> NotifIDList_T;
List of notification IDs (field notificationId in the notifications).

6.1.187. CorrelatedNotifications_T

```
struct CorrelatedNotifications_T
  {
   globaldefs::NamingAttributes_T source;
   NotifIDList_T notifIDs;
  };
```

General comment	
Correlated notifications are identified by the object that emitted the notification and the notification IDs. Both are included in case the Notification IDs are not unique across objects. To use this structure, notification identifiers must be chosen to be unique across all notifications from a particular managed object throughout the time that correlation is significant.	
Attribute name	**Comment**
source	Reference to object that emitted the correlated notification. If empty, the correlated notifications are from the same source as the notification containing this data structure.
notifIDs	IDs of the correlated notifications.

6.1.188. CorrelatedNotificationList_T

typedef sequence<CorrelatedNotifications_T> CorrelatedNotificationList_T;
The optional "X.733::CorrelatedNotifications" parameter uses this type.
When present in an alarm notification, it contains a set of notification identifiers and, if necessary, their associated object names. This set is defined to be the set of all notifications to which this notification is considered to be correlated. This is consistent with the ITU-T X.733 definition.

6.1.189. ProposedRepairAction_T

typedef string ProposedRepairAction_T;
Element of ProposedRepairActionList_T.

6.1.190. ProposedRepairActionList_T

typedef sequence<ProposedRepairAction_T> ProposedRepairActionList_T;
The optional "X.733::ProposedRepairActions" parameter uses this type.
When present in an alarm notification, it indicates if the cause is known and the system being managed can suggest one or more solutions (such as switch in standby equipment, retry, replace media).This is consistent with the ITU-T X.733 definition.

6.1.191. AlarmId_T

```
struct AlarmId_T
  {
   globaldefs::NamingAttributes_T objectName;
   transmissionParameters::LayerRate_T layerRate;
   string probableCause;
   string probableCauseQualifier;
  };
```

General comment
AlarmId_T is used as a unique identifier of an alarm.

Attribute name	Comment
objectName	The name represents the name of the entity that gave rise to the alarm.
layerRate	Identifies the layerRate of the object raising the alarm. For objects where the layerRate is not applicable, such as EMS, the value is set to LR_Not_Applicable. LayerRate is applicable in alarms raised by objects such as PTPs.
probableCause	Probable cause identifies the type of alarm raised against the object.
probableCauseQualifier	Probable cause qualifier is used as the final component of the unique identification of the alarm and is left blank where the objectName, layerRate and probableCause alone provide a unique identification of the alarm.

6.1.192. TCAId_T

```
struct TCAId_T
  {
   globaldefs::NamingAttributes_T objectName;
   transmissionParameters::LayerRate_T layerRate;
   performance::PMParameterName_T pmParameterName;
   performance::PMLocation_T pmLocation;
   performance::Granularity_T granularity;
  };
```

General comment	
TCAId_T is used as a unique identifier of a threshold crossing alert.	
Attribute name	**Comment**
objectName	The name represents the name of the entity that gave rise to the TCA.
layerRate	Identifies the layerRate of the object raising the TCA. For objects where the layerRate is not applicable, such as EMS, the value is set to LR_Not_Applicable. LayerRate is applicable in alarms raised by objects such as PTPs.
pmParameterName	Hold the name of the performance measure.
pmLocation	Hold the location and orientation of the measure.
granularity	Count period of the measure for which the threshold alert was raised.

6.1.193. AlarmOrTCAIdentifier_T

```
union AlarmOrTCAIdentifier_T switch (AlarmTypeQualifier_T)
  {
   case ALARM:  AlarmId_T alarmId;
   case TCA:    TCAId_T tcaId;
  };
```

Structure used for component of a mixed list of Alarms and TCAs. The structure is switched on AlarmTypeQualifier_T. The contents is either an alarm id or a TCA id.

6.1.194. AlarmAndTCAIDList_T

```
typedef sequence<AlarmOrTCAIdentifier_T>  AlarmAndTCAIDList_T;
```

Sequence of identifiers for alarms and TCAs.

6.1.195. Destination_T

typedef string Destination_T;
Destination specification for FTP transfer of history PM data.
In the request for retrieval of history PM Data requests with PM data transfer using FTP, the NMS may specify the target destination for the PM Data file (operation getHistoryPMData()). Within the destination field, the hostname of the destination machine and the full path name of the target file (including the directory name and the file name) within the destination machine are specified.
Hostname and full path name are separated by a colon (:). The '/' or '\' in path name are dependent on the destination and managed by the client.

6.1.196. Granularity_T

typedef string Granularity_T;
Count period (interval) for which PM data may be collected (and subsequently retrieved).
The format is one of:
"n min" (representing n minute granularity) for values of n that are not multiples of 60
"n h" (representing n hour granularity)
"NA" (representing instantaneous measurements) Standard values are:
"15min"
"24h"
"NA" (for current instantaneous measurements)

6.1.197. GranularityList_T

typedef sequence<Granularity_T> GranularityList_T;
Sequence of Granularity_T.

6.1.198. PMLocation_T

typedef string PMLocation_T;
PM parameters may relate to measurements taken on receive or transmit traffic either at the named TP (PML_NEAR_END_Rx/Tx) or at the TP at the far end of the trail (circuit/path) connected to the named TP (PML_FAR_END_Rx/Tx). Alternatively, the PM parameters may be bidirectional(PML_BIDIRECTIONAL) e.g. resulting from a second by second summation and evaluation of both far and near TPs .Valid values are:

"PML_NEAR_END_Rx"
"PML_FAR_END_Rx"
"PML_NEAR_END_Tx"
"PML_FAR_END_Tx"
"PML_BIDIRECTIONAL"
"PML_CONTRA_NEAR_END_Rx"
"PML_CONTRA_FAR_END_Rx"

6.1.199. PMLocationList_T

typedef sequence<PMLocation_T> PMLocationList_T;
A set of PMLocation_T.
In PM related operations, PM parameters may be specified per nearEnd and/or farEnd category, or bidirectional category.

6.1.200. PMParameterName_T

typedef string PMParameterName_T;
Holds the name of Performance Measurements.
It has been defined as a string to accommodate backward compatibility and proprietary extension.

6.1.201. PMParameterNameList_T

typedef sequence<PMParameterName_T> PMParameterNameList_T;
List of PMParameterName_T.

6.1.202. PMParameter_T

```
struct PMParameter_T {
        PMParameterName_T  pmParameterName;
        PMLocation_T pmLocation;
        };
```

General comment	
Used to represent the PM parameter name of the PM measure qualified by its location.	
Attribute name	**Comment**
pmParameterName	Represents the name of the PM Measure.
pmLocation	Represents the location of the PM Measure.

6.1.203. PMParameterList_T

typedef sequence <PMParameter_T> PMParameterList_T ;
List of PMParameter_T.

6.1.204. PMThresholdType_T

```
enum PMThresholdType_T
{
        TWM_HIGHEST,
        TWM_HIGH,
        TWM_LOW,
        TWM_LOWEST
};
```

Describes threshold types (watermark levels) for TCA parameters.
The TWM_HIGH and TWM_HIGHEST types are used for TCAs that are raised when the measured value goes above the threshold. The TWM_LOW and TWM_LOWEST types are used for TCAs that are raised when the measured value goes below the threshold (only apply to gauges).
When there is only one level of TCA trigger, only TWM_HIGH and/or TWM_LOW are used. When there are two levels of TCA triggers,TWM_HIGHEST and/or TWM_LOWEST are used in addition.

6.1.205. PMThresholdValue_T

```
struct PMThresholdValue_T
        {
                PMParameterName_T  pmParameterName;
                PMLocation_T       pmLocation;
                PMThresholdType_T  thresholdType;
                boolean            triggerFlag;
                float              value;
                string             unit;
        };
```

General comment	
Holds a PM threshold value.	
Attribute name	**Comment**
pmParameterName	Holds the name of the PM Measure. It has been defined as a string to accomodate backward compatibility and proprietary extensions.
pmLocation	Represents the location of the PM Measure.
thresholdType	Represents the type of threshold value.
triggerFlag	Indicates if the threshold is for the trigger (true) or the clear(false).

value	Threshold value for the PM parameter (float used to incorporate gauge PM parameters).
unit	Free format string describing the units for the threshold value.

6.1.206. PMThresholdValueList_T

typedef sequence<PMThresholdValue_T> PMThresholdValueList_T ;
Sequence of PMThresholdValue_T.

6.1.207. TCAParameter_T

```
struct TCAParameter_T  {
        performance::PMParameterName_T pmParameterName;
        performance::Granularity_T granularity;
        performance::PMLocation_T pmLocation;
        performance::PMThresholdType_T thresholdType;
        boolean triggerFlag;
        float value;
        string unit;
        };
```

General comment	
The TCAParameter_T struct contains the TCA parameters assigned to a TCA ParameterProfile. They can be retrieved and modified by the NMS through the operations getTCAParameterProfile and setTCAParameterProfile provided by the PerformanceManagementMgr_I interface.	
Attribute name	**Comment**
pmParameterName	It has been defined as a string to accomodate proprietary extension.
granularity	This attribute defines the granularity of the TCA Parameter. For counters two values "15min" or "24h" are supported. For current instantaneous measurements the granularity does not apply which is indicated by the value "N/A".
pmLocation	Represents the location of the PM measure.
pmThresholdType	Represents the type of threshold value.
triggerFlag	Indicates if the threshold is for the trigger (true) or the clear (false).
value	Threshold for parameter (float used to incorporate gauge PM parameters).
unit	Free format string describing the units for the threshold value.

6.1.208. TCAParameterList_T

typedef sequence<TCAParameter_T> TCAParameterList_T;
Sequence of TCAParameter_T.

6.1.209. TCAParameterProfile_T

```
struct TCAParameterProfile_T  {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        globaldefs::NVSList_T additionalInfo;
        transmissionParameters::LayerRate_T layerRate;
        TCAParameterList_T tcaParameterList;
```

```
};
```

General comment	
This struct is used to represent the TCA (threshold crossing alert)Parameter Profile (TCAPP) object class containing per layer rate a set of PM parameter/threshold value pairs per granularity (e.g. VC12: ES / 50;15min). A profile may be created and deleted by EMS or NMS. Instances of this class may be associated to TPs. A list of references to all associated TPs will be maintained in the profile.	
Attribute name	**Comment**
name	The name represents the name of the TCA Parameter Profile which is assigned by the EMS upon creation. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the ManagedElement. It is a readonly attribute.
userLabel	The user label of the TCA Parameter Profile can be set by EMS or NMS upon creation. After creation the user label can be changed by the NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read-write attribute.
nativeEMSName	The name represents how the TCA Parameter Profile is referred to on EMS displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through).After a TCA Parameter Profile has been created, the nativeEMSName may be changed by the NMS, if the EMS supports this functionality, using the common::Common_I::setNativeEMSName() operation of the Common_I interface service. It is a read/write attribute and never set to NULL string.
owner	The owner attribute of the TCA Parameter Profile indicates the ownership. The owner is provisionable by the NMS. This attribute can be set by the NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
additionalInfo	This attribute allows the communication from the EMS to the NMS (and vice versa from NMS to EMS) of additional information which isn't explicitly modelled. This may be an empty list. It is a read/write attribute.
layerRate	It is a read only attribute.
tcaParameterList	It is a read/write attribute.

6.1.210. TCAParameterProfileList_T

typedef sequence< TCAParameterProfile_T> TCAParameterProfileList_T;
Sequence of TCAParameterProfile_T.

6.1.211. PMMeasurement_T

```
struct PMMeasurement_T
{
        PMParameterName_T pmParameterName;
        PMLocation_T pmLocation;
        float value;
        string unit;
        string intervalStatus;
        };
```

General comment

Holds a PM measurement and description of validity.	
Attribute name	**Comment**
pmParameterName	Represents the name of the PM Measure.
pmLocation	Represents the location of the PM Measure.
value	Value of PM parameter (float used to incorporate gauge PM parameters).
unit	Free format string describing the units for the PM measurement value.
intervalStatus	The following string values, to indicate the relationship of measurement value and measurement interval, are allowed: "Valid" - valid data, "Incomplete" - data not available for the complete interval, "Invalid" - data available but marked as invalid for the interval (when the EMS can not distinguish incomplete measurements from invalid measurements, "Invalid" will be used) "Unavailable" - no data available at all for this interval, "Zero-suppressed" - zero-suppressed intervals.

6.1.212. PMMeasurementList_T

typedef sequence<PMMeasurement_T> PMMeasurementList_T ;
Sequence of PMMeasurement_T.

6.1.213. PMData_T

```
struct PMData_T {
        globaldefs::NamingAttributes_T tpName;
        transmissionParameters::LayerRate_T layerRate;
        Granularity_T granularity;
        globaldefs::Time_T retrievalTime;
        PMMeasurementList_T pmMeasurementList;
        };
```

General comment	
This struct is used to pass current PM data for one collection bin from the EMS to the NMS. It is also used to pass historic PM data across the NML-EML interface when the file transfer capability via FTP is not preferred.	
Attribute name	**Comment**
tpName	The name of the termination point. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
layerRate	Layer of the collected PM data.
granularity	Identifies the granularity of the data.
retrievalTime	Point in time at which PM measurement was obtained from the ME.
pmMeasurementList	PM measurement values.

6.1.214. PMDataList_T

typedef sequence<PMData_T> PMDataList_T;
Sequence of PMData.

6.1.215. PMTPSelect_T

```
struct PMTPSelect_T {
        globaldefs::NamingAttributes_T name;
        transmissionParameters::LayerRateList_T layerRateList;
        PMLocationList_T pMLocationList;
        GranularityList_T granularityList;
        };
```

General comment	
Used to specify the scope/target for a PM operation.	
Attribute name	**Comment**
name	The name of the object to which this selection applies.(i.e. the target of the PM operation). This may be: ManagedElement: The selection applies to all termination points contained within the ManagedElement. TerminationPoint: The selection applies only to the named termination point which will be a PTP, an FTP or a CTP. Unlike the case for ManagedElement, the operation will not apply to any "contained" TPs when a PTP, an FTP or CTP is specified. Note: As stated in the requirements (TMF513), the possible scopes do not currently include "SNC" and do not include "TL".
layerRateList	Identifies the set of layers to which the selection applies. An empty list means all layers supported by the equipment.
pMLocationList	Identifies the set of locations to which the selection applies. An empty list means all locations supported by the equipment.
granularityList	Identifies set of granularities to which the selection applies. An empty list means all granularities supported by the equipment.

6.1.216. PMTPSelectList_T

typedef sequence<PMTPSelect_T> PMTPSelectList_T;
Sequence of PMTPSelect. Used by several methods to identify sets of PM parameters to which to apply the operation.

6.1.217. HoldingTime_T

```
struct HoldingTime_T {
        short storeTime24hr;
        short storeTime15min;
        };
```

General comment	
By means of this struct, the EMS informs the NMS about the minimum time it holds 24h PM data records and 15min PM data records. This minimum time refers to the time after the particular collection bins are closed.	
Attribute name	**Comment**
storeTime24hr	Minimum store time in hours for 24h data.
storeTime15min	Minimum store time in hours for 15min data.

6.1.218. TCAParameters_T

```
struct TCAParameters_T  {
                transmissionParameters::LayerRate_T layerRate;
                Granularity_T granularity;
                PMThresholdValueList_T tcaTypeValues;
        };
```

General comment	
The TCAParameters structure is used to get or set one or more PM Threshold values for a given TP/layer/granularity measurement point.	
Attribute name	**Comment**
layerRate	Identifies the layer to which tcaTypeValues applies.
granularity	Identifies the granularity of the data. For gauges, this should be "NA".
tcaTypeValues	Identifies threshold values. for PM parameters, PM locations and threshold types, and indicates Trigger/Clear properties.

6.1.219. AdministrativeState_T

```
enum AdministrativeState_T {
        AS_Locked,
        AS_Unlocked
    };
```

AdministrativeState_T indicates whether a specified function is enabled (unlocked) or disabled (locked).

6.1.220. PMThreshold_T

```
struct PMThreshold_T {
        PMThresholdType_T thresholdType;
        boolean triggerFlag;
        float value;
        string unit;
    };
```

General comment	
This structure represents a Performance Monitoring Threshold.	
Attribute name	**Comment**
thresholdType	Represents the type of threshold value.
triggerFlag	Indicates if the threshold is for the trigger (TRUE) or the clear (FALSE).
value	Threshold value (float used to incorporate gauge PM parameters).
unit	Free format string describing the units for the threshold value.

6.1.221. PMThresholdList_T

```
typedef sequence <PMThreshold_T> PMThresholdList_T;
```

Sequence of Performance Monitoring Thresholds.

6.1.222. PMParameterWithThresholds_T

```
struct PMParameterWithThresholds_T {
        PMParameterName_T pmParameterName;
        PMThresholdList_T pmThresholdList;
    };
```

General comment	
The PMParameterWithThresholds_T structure describes the list of thresholds associated with a particular PM Parameter.	
Attribute name	**Comment**

pmParameterName	Represents the name of the PM Measure.
pmThresholdList	Holds the list of PM thresholds associated with the named PM parameter.

6.1.223. PMParameterWithThresholdsList_T

typedef sequence <PMParameterWithThresholds_T> PMParameterWithThresholdsList_T;
Sequence of Performance Monitoring parameters with their respective thresholds.

6.1.224. PMP_T

```
struct PMP_T {
        globaldefs::NamingAttributes_T name;
        string userLabel;
        string nativeEMSName;
        string owner;
        PMParameterWithThresholdsList_T pmParameterWithThresholdsList;
        AdministrativeState_T monitoringState;
        AdministrativeState_T supervisionState;
        globaldefs::NVSList_T additionalInfo;
        Granularity_T granularity;
        };
```

General comment	
This structure represents one performance monitoring point (PMP). It is determined by the containing PTP or CTP and by the layer rate, the PM location and the granularity the monitoring is done for. PMP objects are created by the EMSs only. They are contained in TPs and represent the PM capabilities of their containing TP(s).	
Attribute name	**Comment**
name	The name represents the name of the PMP which is assigned by the EMS upon creation. It is a read only attribute. The EMS is responsible for guaranteeing the uniqueness of the name within the context of the PerformanceManagementMgr_I. This is achieved by the MTNM naming hierarchy and by the following naming convention for the PMP name value: The value of the PMP name will follow ALCATEL name.
userLabel	The userLabel is a friendly name that the operator wants to give to the PMP. Typical expectations of the operator is that the same name is seen on all operation systems. This is set by the NMS and could be displayed on the EMS based on each systems' capabilities. THIS IS NOT A MANDATORY EXPECTATION, but is left to the implementation of the EMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setUserLabel(). It is a read/write attribute.
nativeEMSName	Represents how the PMP refers to itself on PMP displays. Its aim is to provide a "nomenclature bridge" to aid relating information presented on NMS displays to EMS displays (via GUI cut through).May be a null string.
owner	The owner is provisionable by the NMS. This attribute can be set by NMS through the Common_I interface service common::Common_I::setOwner(). It is a read/write attribute.
pmParameterWithThresholdsList	List of PM parameters supported by the PMP, e.g. PMP_ES, PMP_SES,etc. together with all their thresholds. Please note in this version, the threshold is not supported.
monitoringState	The monitoring state indicates whether performance monitoring is enabled or disabled for the PMP . In this version, always set to true(AS_Unlocked).

supervisionState	The supervision state indicates whether performance threshold supervision is enabled or disabled for the PMP .In this version, always set to true(AS_Unlocked).
additionalInfo	This attribute allows the communication from the EMS to the NMS of additional information which is not explicitly modeled. It is a read-only attribute.
granularity	This attributes presents the supported granularity of this PMP.

6.1.225. PMPList_T

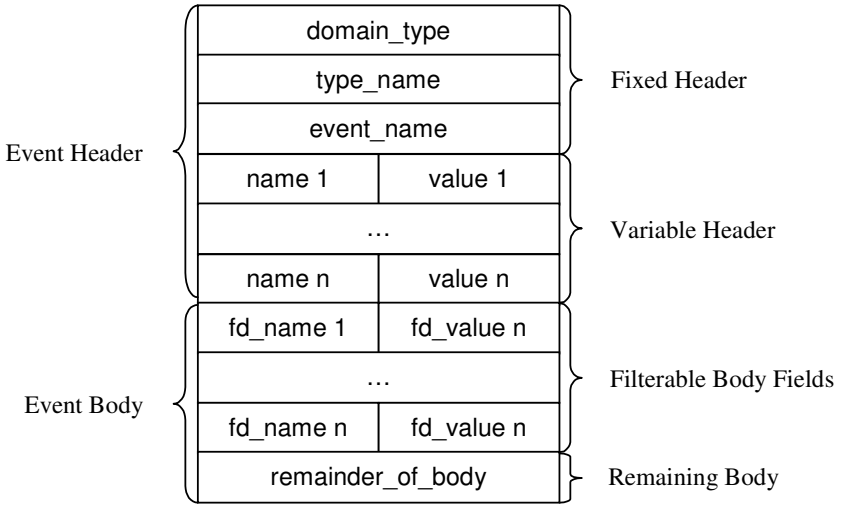
typedef sequence<PMP_T> PMPList_T;
Sequence of PMP_T. Used to convey a batch of collected Performance Monitoring Points data.

7. INTERFACE SET DESCRIPTION

7.1. Notifications

7.1.1. Structure of the notification

Each notification is a structured event that consists of header section and a filterable data section as shown in the figure below:



7.1.2. Supported notifications

See. Appendix A Notification Types

7.2. Common_I interface

The interface Common_I is a set of services and utilities that is inherited by every manager interface.

7.2.1. Operations

7.2.1.1. setUserLabel

```
void setUserLabel (
        in globaldefs::NamingAttributes_T objectName,
        in string userLabel,
        in boolean enforceUniqueness)
        raises(globaldefs::ProcessingFailureException);
```

General comment

The userLabel is owned by the NMSes. It is a string assigned by an NMS to an object. The difference between the userLabel and the NamingAttributes name is that the userLabel is an attribute of the objects that may be "set" by the NMS through well-defined interfaces (setUserLabel).
When an object is created by an NMS, the NMS specifies the userLabel for the object.
When an object is created by the EMS, the EMS sets the userLabel to the nativeEMSName.
Once an object is created, the userLabel may only be changed by an NMS through the setUserLabel operation.

Supported products: EMLNBI1.0, SDHNBI1.0
EMLNBI1.1, SDHNBI1.1
EMLNBI1.2, SDHNBI1.2
EMLNBI1.3, SDHNBI1.3
EMLNBI1.4, SDHNBI1.4
EMLNBI1.5, SDHNBI1.5
EMLNBI1.6, SDHNBI1.6

❑ Input / Output ➢	Comment
❑ objectName	Name of the object for which to change the userLabel.
❑ userLabel	New user label to assign to the object.
❑ enforceUniqueness	Specifies whether or not userLabel should be checked for uniqueness amongst objects of the same class within the EMS. If true, then the operation will fail if userLabel is already in use.

raises	ExceptionType
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - If EMS does not support this service *EXCPT_INTERNAL_ERROR* – Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - aised when objectName is incorrectly formed *EXCPT_ENTITY_NOT_FOUND* - Raised when objectName references object which does not exist. *EXCPT_UNABLE_TO_COMPLY* - Raised when the userLabel can not be set for the specified object *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost *EXCPT_USERLABEL_IN_USE* - Raised when the userLabel uniqueness constraint is not met

7.2.1.2. setOwner

```
void setOwner(
    in globaldefs::NamingAttributes_T objectName,
    in string owner)
   raises (globaldefs::ProcessingFailureException);
```

General comment

This service sets the owner attribute of the specified object.

Supported products: EMLNBI1.0, SDHNBI1.0
EMLNBI1.1, SDHNBI1.1
EMLNBI1.2, SDHNBI1.2
EMLNBI1.3, SDHNBI1.3
EMLNBI1.4, SDHNBI1.4
EMLNBI1.5, SDHNBI1.5
EMLNBI1.6, SDHNBI1.6

❑ Input / Output ➢	Comment

❑ objectName	Object name
❑ owner	Name of owner
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - If EMS does not support this service *EXCPT_INTERNAL_ERROR* – Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - aised when objectName is incorrectly formed *EXCPT_ENTITY_NOT_FOUND* - Raised when objectName references object which does not exist. *EXCPT_UNABLE_TO_COMPLY* - Raised when the userLabel can not be set for the specified object *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost

7.2.1.3. setAdditionalInfo

```
void setAdditionalInfo(
    in globaldefs::NamingAttributes_T objectName,
    inout globaldefs::NVSList_T additionalInfo)
   raises (globaldefs::ProcessingFailureException);
};
```

General comment	
This service sets the additional info attribute of the object identified by objectName. This operation should be used to set both vendor specific attributes as well as the attributes that are formally defined in this interface specification. As an input only the list of parameters to be changed, removed, or added shall be provided. If an entry is to be removed, "-" shall be specified as a value. If a parameter is specified that is currently not part of the additionalInfo attribute of the specified object that parameter is added by the EMS with the specified value. The EMS may reject removal and addition requests, however. The operation is best effort (except where specified otherwise for a particular parameter). The output specifies the values, which were actually set. **Supported products:** EMLNBI1.0, SDHNBI1.0 EMLNBI1.1, SDHNBI1.1 EMLNBI1.2, SDHNBI1.2 EMLNBI1.3, SDHNBI1.3 EMLNBI1.4, SDHNBI1.4 EMLNBI1.5, SDHNBI1.5 EMLNBI1.6, SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ objectName	the managed object whose additional info parameters are intended to get modified
❑➢ additionalInfo	Currently, only "locationName"parameter can be set through this operation.
raises	**ExceptionType**

ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - If EMS does not support this service *EXCPT_INTERNAL_ERROR* – Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when objectName is incorrectly formed,raised when an input parameter is syntactical incorrect. *EXCPT_ENTITY_NOT_FOUND* - Raised when objectName references object which does not exist. *EXCPT_UNABLE_TO_COMPLY* - Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" *EXCPT_NE_COMM_LOSS* - Raised when the communication to the managed element containing or hosting objectName is lost

7.3. EMSMgr_I interface

The EMSMgr_I is used to gain access to operations, which deal with the EMS itself.
A handle to an instance of this interface is gained via the emsSession::EmsSession_I::getManager() operation in managerInterface when the managerName "EMS" is used.

7.3.1. Operations

7.3.1.1. getEMSTime

```
void getEMSTime(
        out globaldefs::Time_T emsTime)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
Get EMS time. **Supported products:** EMLNBI1.2, SDHNBI1.2 EMLNBI1.3, SDHNBI1.3 EMLNBI1.4, SDHNBI1.4 EMLNBI1.5, SDHNBI1.5 EMLNBI1.6, SDHNBI1.6	
❑ Input / Output ➢	**Comment**
➢emsTime	The time retrieved.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.3.1.2. setEMSTime

```
void setEMSTime(
        in globaldefs::Time_T emsTime)
        raises(globaldefs::ProcessingFailureException);
```

General comment
Set EMS time. **Supported products:** EMLNBI1.2, SDHNBI1.2 EMLNBI1.3, SDHNBI1.3 EMLNBI1.4, SDHNBI1.4 EMLNBI1.5, SDHNBI1.5 EMLNBI1.6, SDHNBI1.6

❑ Input / Output ➢	Comment
❑emsTime	The time to be set.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal.

7.3.1.3. setEMSLocation

void setEMSLocation(in string emsLocation)
raises(globaldefs::ProcessingFailureException);

General comment	
Set EMS location. **Supported products:** EMLNBI1.2, SDHNBI1.2 EMLNBI1.3, SDHNBI1.3 EMLNBI1.4, SDHNBI1.4 EMLNBI1.5, SDHNBI1.5 EMLNBI1.6, SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **emsLocation**	The location to be set.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal.

7.3.1.4. getEMS

void getEMS(
out EMS_T emsInfo)
raises(globaldefs::ProcessingFailureException);

General comment	
This allows an NMS to request the EMS information. **Supported products:** EMLNBI1.0, SDHNBI1.0 EMLNBI1.1, SDHNBI1.1 EMLNBI1.2, SDHNBI1.2 EMLNBI1.3, SDHNBI1.3 EMLNBI1.4, SDHNBI1.4 EMLNBI1.5, SDHNBI1.5 EMLNBI1.6, SDHNBI1.6, PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
➢**emsInfo**	The EMS information.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.3.1.5. getAllTopLevelSubnetworks

void getAllTopLevelSubnetworks(

```
in unsigned long how_many,
out multiLayerSubnetwork::SubnetworkList_T sList,
out multiLayerSubnetwork::SubnetworkIterator_I sIt)
raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all levels of the Subnetworks that are under the control of this EMS. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ how_many	Maximum number of subnetworks to return in the first batch.
➢ sList	First batch of subnetworks.
➢ sIt	Iterator to retrieve the remaining subnetworks.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure. EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.3.1.6. getAllTopLevelSubnetworksNames

```
void getAllTopLevelSubnetworkNames(
    in  unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllTopLevelSubnetworks(),but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ how_many	Maximum number of subnetwork names to return in the first batch.
➢ nameList	First batch of subnetwork names.
➢ nameIt	Iterator to retrieve the remaining subnetwork names.

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure. EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.3.1.7. getAllTopLevelTopologicalLinks

```
void getAllTopLevelTopologicalLinks(
        in unsigned long how_many,
        out topologicalLink::TopologicalLinkList_T topoList,
        out topologicalLink::TopologicalLinkIterator_I topoIt)
        raises(globaldefs::ProcessingFailureException);
```

General comment
This allows an NMS to request all of the Topological Links that exist between multiLayerSubnetworks under the control of this EMS. This operation only return physical connections in alcatel RM by default; when the SDHNBI parameter "RETURN_SUBLINKCONNECT_AS_TL" set to true this operation will return both physical connections and sub link connections as TMF814 topological link..In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ **how_many**	Maximum number of topological links to return in the first batch.
➢ **topoList**	First batch of topological links.
➢ **topoIt**	Iterator to retrieve the remaining topological links.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.8. getAllTopLevelTopologicalLinkNames

```
void getAllTopLevelTopologicalLinkNames(
        in  unsigned long how_many,
        out globaldefs::NamingAttributesList_T nameList,
        out globaldefs::NamingAttributesIterator_I nameIt)
        raises(globaldefs::ProcessingFailureException);
```

General comment

This operation has exactly the same behaviour as getAllTopLevelTopologicalLinks, but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator.

Supported products: SDHNBI1.0
SDHNBI1.1
SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ **how_many**	Maximum number of topological link names to return in the first batch.
➢ **nameList**	First batch of topological link names.
➢ **nameIt**	Iterator to retrieve the remaining topological link names.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.9. getTopLevelTopologicalLink

```
void getTopLevelTopologicalLink(
        in globaldefs::NamingAttributes_T topoLinkName,
        out topologicalLink::TopologicalLink_T topoLink)
        raises(globaldefs::ProcessingFailureException);
```

General comment

This service returns a top-level topological link given its name. Some constraints refer to getAllTopLevelTopologicalLinks.

Supported products: SDHNBI1.0
SDHNBI1.1
SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ **topoLinkName**	Name of the top level topological link to retrieve.
➢ **topoLink**	Top-level topological link returned.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when topoLinkName does not reference a top level topological link object EXCPT_ENTITY_NOT_FOUND - Raised when topoLinkName references a top level topological link object that does not exist

7.3.1.10. filteredGetActiveAlarms

```
void filteredGetActiveAlarms(
                    in string constraintExp,
                    in unsigned long how_many,
                    out notifications::EventList_T eventList,
                    out notifications::EventIterator_I eventIt)
                    raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to retrieve active alarms by filter (constraint string). **Supported products:** EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ constraintExp	The filter constraint string, whose format is the same as Corba Notification Filter Constraint Expression string.
❑ how_many	Maximum number of events to report in the first batch.
➢ eventList	First batch of events.
➢eventIt	Iterator to retrieve the remaining events.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.3.1.11. getAllEMSAndMEActiveAlarms

```
void getAllEMSAndMEActiveAlarms(
    in notifications::ProbableCauseList_T excludeProbCauseList,
    in notifications::PerceivedSeverityList_T excludeSeverityList,
    in unsigned long how_many,
    out notifications::EventList_T eventList,
    out notifications::EventIterator_I eventIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the active alarms of the EMS. For Alcatel NM, the active alarms are all not cleared alarms in AS systems; For Alcatel RM, the active alarms are not cleared alarms from path, trail,physical connection, link connection in AS systems. The result of this operation is independent of the filtering set up by the NMS for the notification service. Alarms which ASAP assigned severity is "AS_NONALARMED" should not be reported by this operation. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**

❑ **excludeProbCauseList**	The list of probable causes to exclude (for which events should not be reported).
❑ **excludeSeverityList**	List of severities to exclude from the output event list.
❑ **how_many**	Maximum number of alarms and TCAs to report in the first batch.
➢**eventList**	First batch of alarms and TCAs.
➢**eventIt**	Iterator to retrieve the remaining alarms and TCAs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure. EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.3.1.12. getAllEMSSystemActiveAlarms

```
void getAllEMSSystemActiveAlarms(
    in notifications::PerceivedSeverityList_T excludeSeverityList,
    in unsigned long how_many,
    out notifications::EventList_T eventList,
    out notifications::EventIterator_I eventIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation is to request all EMS System alarms in alcatel EMS. The “EMS System” alarms can be customized by adding probableCause filters in AsEMSAlarmFilterFile.txt in CorbaNBI1.5 configuration directory. Default filter is empty, so this operation’s default behaviour is the same as **getAllEMSAndMEActiveAlarms.** In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **excludeSeverityList**	List of severities to exclude from the output event list.
❑ **how_many**	Maximum number of alarms and TCAs to report in the first batch.
➢ **eventList**	First batch of alarms and TCAs.
➢ **eventIt**	Iterator to retrieve the remaining alarms and TCAs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure. EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.3.1.13. createTopologicalLink

```
void createTopologicalLink(
    in  topologicalLink::TLCreateData_T    newTLCreateData,
    out topologicalLink::TopologicalLink_T newTopologicalLink)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation can be used to create top-level topological links as well as intra-MLSN topological links. The behaviour of this operation should be the same as if an EMS user were to attempt to create a topological link (or equivalent operation) at the EMS user interface. Therefore, the specific conditions under which the creation is rejected are left to the EMS implementation. **Supported products:** SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ newTLCreateDatat	Structure describing the topological link to be created.
➢ newTopologicalLink	Structure describing the newly created topological link reflecting the input data.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when a field of createData is invalid. EXCPT_OBJECT_IN_USE - Raised when the TL creation is rejected due to an aEndTP/zEndTP conflict EXCPT_ENTITY_NOT_FOUND - Raised if the aEndTP or zEndTP do not exist EXCPT_UNABLE_TO_COMPLY - Raised when the EMS is unable to execute the request because at least one of the parameters although valid could not be set and that parameter is identified as "not best effort" in the Additional Information Usage document. Raised if the creation is not possible because the EMS cannot comply for a reason different from the ones above. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is not met

7.3.1.14. deleteTopologicalLink

void deleteTopologicalLink(
in globaldefs::NamingAttributes_T topoLinkName)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation can be used to delete top-level topological links as well as intra-MLSN topological links. The behaviour of this operation should be the same as if an EMS user were to attempt to delete a topological link (or equivalent operation) at the EMS user interface. Therefore, the specific conditions under which the deletion is rejected are left to the EMS implementation. It is at the discretion of the EMS to delete topological links created by the NMS, as the owner of the topological link is the EMS, not the NMS. **Supported products:** SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ topoLinkName	The name of the topological link to be deleted.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when topoLinkName does not reference a topological link object EXCPT_ENTITY_NOT_FOUND - Raised if the topological link does not exist. EXCPT_UNABLE_TO_COMPLY when the deletion is rejected for EMS specific reasons

7.3.1.15. getAllEMSAndMEUnacknowledgedActiveAlarms

void getAllEMSAndMEUnacknowledgedActiveAlarms(
in notifications::ProbableCauseList_T excludeProbCauseList,
in notifications::PerceivedSeverityList_T excludeSeverityList,
in unsigned long how_many,
out notifications::EventList_T eventList,
out notifications::EventIterator_I eventIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation behave like getAllEMSAndMEActiveAlarms, but only return unacknowledged active alarms. The result of this operation is independent of the filtering set up by the NMS for the notification service. Alarms which ASAP assigned severity is "AS_NONALARMED" should not be reported by this operation. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ excludeProbCauseList	The list of probable causes to exclude (for which events should not be reported).
❑ excludeSeverityList	List of severities to exclude from the output event list.
❑ how_many	Maximum number of alarms and TCAs to report in the first batch.
➢eventList	First batch of alarms and TCAs.
➢eventIt	Iterator to retrieve the remaining alarms and TCAs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.16. getAllEMSSystemUnacknowledgedActiveAlarms

void getAllEMSSystemUnacknowledgedActiveAlarms(

```
in notifications::PerceivedSeverityList_T excludeSeverityList,
in unsigned long how_many,
out notifications::EventList_T eventList,
out notifications::EventIterator_I eventIt)
raises(globaldefs::ProcessingFailureException);
```

General comment
This operation behave like getAllEMSSystemActiveAlarms, but only return unacknowledged active alarms. The result of this operation is independent of the filtering set up by the NMS for the notification service. Alarms which ASAP assigned severity is "AS_NONALARMED" should not be reported by this operation. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6

❑ Input / Output ➢	Comment
❑ **excludeSeverityList**	List of severities to exclude from the output event list.
❑ **how_many**	Maximum number of alarms and TCAs to report in the first batch.
➢**eventList**	First batch of alarms and TCAs.
➢**eventIt**	Iterator to retrieve the remaining alarms and TCAs.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.17. createASAPUnderMe

```
void createASAPUnderMe(
    in globaldefs::NamingAttributes_T meName,
    in aSAP::ASAPCreateModifyData_T newASAPCreateData,
    out aSAP::ASAP_T newASAP,
    out globaldefs::NVSList_T additionalInfo)
    raises(globaldefs::ProcessingFailureException);
```

General comment
This operation creates a new ASAP object, named by the me, with the values described by the input parameters. The operation fails if the maximum number of ASAPs for EMS has been reached. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6

❑ Input / Output ➢	Comment
❑ **meName**	Name of ME.
❑ **newASAPCreateData**	Information about the ASAP to be created.For EMLNBI1.4, input parameter owner is meanlingless.
➢ **newASAP**	result of the creation.

➢ **additionalInfo**	to allow the communication of additional information which is not explicitly modelled.For EMLNBI 1.4,it's empty.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support creation of ASAPs via this interface. *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure. *EXCPT_INVALID_INPUT* - Raised if input parameters contains invalid data.

7.3.1.18. getAllASAPsUnderMe

void getAllASAPsUnderMe(
in globaldefs::NamingAttributes_T meName,
in unsigned long how_many,
out aSAP::ASAPList_T aSAPList,
out aSAP::ASAPIterator_I asapIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This allows an NMS to request all the ASAPs of this ne. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **meName**	Name of ME.
❑ **how_many**	Maximum number of ASAPs to return in the first batch.
➢ **aSAPList**	First batch of ASAPs.
➢ **asapIt**	Iterator to retrieve the remaining ASAPs.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS cannot support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_TOO_MANY_OPEN_ITERATORS* - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.19. getAllASAPNamesUnderMe

void getAllASAPNamesUnderMe(
in globaldefs::NamingAttributes_T meName,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment
This operation has exactly the same behaviour as getAllASAPsUnderMe, but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6

❑ Input / Output ➢	Comment
❑ **meName**	Name of ME.
❑ **how_many**	Maximum number of ASAPs to return in the first batch.
➢ **nameList**	First batch of ASAP names.
➢ **nameIt**	Iterator to retrieve the remaining ASAP names.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS cannot support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_TOO_MANY_OPEN_ITERATORS* - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.20. deleteASAP

void deleteASAP(
in globaldefs::NamingAttributes_T aSAPName,
inout globaldefs::NVSList_T additionalInfo)
raises(globaldefs::ProcessingFailureException);

General comment
This operation deletes the addressed ASAP. The operation fails if at least one resource is pointing to this ASAP. Moreover the EMS could refuse this operation, in case the addressed ASAP is fixed. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6

❑ Input / Output ➢	Comment
❑ **aSAPName**	The name of the ASAP object to be deleted.
❑ ➢**additionalInfo**	To allow the communication of additional information which is not explicitly modelled.Now for EMLNBI 1.4,it's empty.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support deletion of ASAPs via this interface *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when aSAPName does not refer to an ASAP object *EXCPT_ENTITY_NOT_FOUND* - Raised when aSAPName references an object that does not exist

7.3.1.21. assignASAP

void assignASAP(
in globaldefs::NamingAttributes_T aSAPName,
in globaldefs::NamingAttributes_T resourceName,
in transmissionParameters::LayerRate_T layerRate,
inout globaldefs::NVSList_T additionalInfo)
raises(globaldefs::ProcessingFailureException);

General comment

This operation assigns the addressed ASAP to the addressed resource,i.e. the aSAPpointer is updated accordingly. The formerly referenced ASAP, if any, is deassigned. For TPs, it is necessary to indicate also the layer rate the ASAP will apply.
This operation causes an alarm re-evaluation of the already detected defects according to the following rules.

Supported products: EMLNBI1.4
EMLNBI1.5
EMLNBI1.6

❑ Input / Output ➢	Comment
❑ **aSAPName**	The name of the ASAP object to be assigned.
❑ **resourceName**	The name of the resource to assign the ASAP to.For current version, it is only tp or equipment object.
❑ **layerRate**	The TP layer rate the ASAP object is assigned to. If the resource is not a TP, then the layerRate value shall be LR_Not_Applicable.Now for current version,it's meanlingless.
❑ ➢**additionalInfo**	to allow the communication of additional information which is not explicitly modelled.For current version,it is empty.

raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support assignment of ASAPs via this interface *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when aSAPName does not refer to an ASAP object, or layerRate is invalid for the addressed resource, i.e. it is not an encapsulated layerRate *EXCPT_ENTITY_NOT_FOUND* - Raised when aSAPName or resourceName reference an object that does not exist

7.3.1.22. deassignASAP

```
void deassignASAP(
    in globaldefs::NamingAttributes_T resourceName,
    in transmissionParameters:: LayerRate_T layerRate,
    inout globaldefs::NVSList_T additionalInfo)
  raises(globaldefs::ProcessingFailureException);
```

General comment

This operation deassigns the ASAP from the addressed resource, i.e. the aSAPpointer is updated to empty string. For TPs, it is necessary to indicate also the layer rate, in order to identify the ASAP to be deassigned.
This operation causes an alarm re-evaluation of the already detected defects according to same rules as ASAP assignment, considering as "newly assigned ASAP" the default alarm severity assignment.
If the addressed resource originates alarms from within the ME, then this operation could imply the "deactivation" of the ASAP instance and functionality over the proper ME. E.g. the EMS will remove the ASAP from a given ME only when the ASAP is no longer assigned to any resource of that ME.

Supported products: EMLNBI1.4
EMLNBI1.5
EMLNBI1.6

❑ Input / Output ➢	Comment
❑ **resourceName**	the name of the resource to deassign the ASAP from.For current version,it is only tp or equipment object.
❑ **layerRate**	the TP layer rate is necessary to identify the ASAP instance to be deassigned. If the resource is not a TP, then the layerRate value shall be LR_Not_Applicable.Now for current version,it's meanlingless.

❑ ➢**additionalInfo**	to allow the communication of additional information which is not explicitly modelled.For current version,it is empty.
raises	**ExceptionType**
ProcessingFailureException	For current version, if one resource asap is deassigned, the asap pointer of the resoure will point the default asap whose name is defined by key DEF_ASAP_NAME in param.cfg. *EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support deassignment of ASAPs via this interface *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when resourceName is an invalid reference, or layerRate is invalid for the addressed resource, i.e. it is not an encapsulated layerRate *EXCPT_ENTITY_NOT_FOUND* - Raised when resourceName references an object that does not exist

7.3.1.23. modifyASAP

void modifyASAP(
in globaldefs::NamingAttributes_T aSAPName,
in aSAP::ASAPCreateModifyData_T aSAPModifyData,
out aSAP::ASAP_T newASAP,
out globaldefs::NVSList_T additionalInfo)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation modifies the addressed ASAP according to the input parameters. This operation does not modify any current ASAP assignment. The specified alarmSeverityAssignmentList will completely replace the current one. EMS could refuse this operation in case the addressed ASAP is fixed. The modification of the ASAP does not trigger any alarm re-evaluation of the already detected defects. This because such re-evaluation process may involve a big number of managed resources across the me. If the addressed resource originates alarms from within the ME, then this operation could imply the modification of the ASAP instance and functionality over the proper ME. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **aSAPName**	the name of the ASAP object to be modified.
❑ **aSAPModifyData**	the data to be modified.For current version, owner can't be modified.
➢ **newASAP**	result of the modification.
➢**additionalInfo**	to allow the communication of additional information which is not explicitly modelled.For current version,it's empty.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support ASAP modification via this interface *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when aSAPName does not refer to an ASAP object *EXCPT_ENTITY_NOT_FOUND* - Raised when aSAPName reference an object that does not exist

7.3.1.24. getASAP

void getASAP(

```
 in globaldefs::NamingAttributes_T aSAPName,
 out aSAP::ASAP_T aSAP)
raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns an ASAP instance given its name. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ aSAPName	Name of the ASAP to retrieve.
➢ aSAP	ASAP returned.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS cannot support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when aSAPName does not reference an ASAP object *EXCPT_ENTITY_NOT_FOUND* - Raised when aSAPName references an ASAP object that does not exist

7.3.1.25. getASAPbyResource

```
void getASAPbyResource(
    in globaldefs::NamingAttributes_T resourceName,
    in transmissionParameters::LayerRateList_T layerRateList,
    in unsigned long how_many,
    out aSAP::ASAPList_T aSAPList,
    out aSAP::ASAPIterator_I asapIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation retrieves the ASAP(s) which are assigned to the addressed resource. The ASAP list can have zero or more elements, as all resources but TPs can refer to at most one ASAP. TPs can refer to more ASAPs, at most as many as the number of encapsulated layer rates.But for current version,one resource can only refert to one or zeor ASAP. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ resourceName	the name of the resource.For current version,it is only tp or equipment object.
❑ layerRateList	List of TP layer rates which assigned ASAPs are to be retrieved. If an empty list is specified, then all ASAPs assigned to the addressed resource will be replied. The list shall also be empty if the addressed resource is not a TP.For current version, it's meanlingless.
❑ how_many	Maximum number of ASAPs to return in the first batch.
➢aSAPList	First batch of the ASAP(s) assigned to the addressed resource.
➢asapIt	Iterator to retrieve the remaining ASAPs.
raises	**ExceptionType**

ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support this operation *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when layerRateList is invalid for the addressed resource, i.e. it is not an encapsulated layerRate *EXCPT_ENTITY_NOT_FOUND* - Raised when resourceName references an object that does not exist *EXCPT_TOO_MANY_OPEN_ITERATORS* - Raised when maximum number of iterators that the EMS can support has been reached

7.3.1.26. getASAPAssociatedResourceNames

```
void getASAPAssociatedResourceNames(
    in globaldefs::NamingAttributes_T aSAPName,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation retrieves all the resource names (could be TPs, Equipment) which point to the addressed ASAP instance. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ aSAPName	Name of the ASAP.
❑ how_many	Maximum number of resource names to return in the first batch.
➢ nameList	First batch of resource names.
➢ nameIt	Iterator to retrieve the remaining resource names.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if EMS cannot support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when aSAPName does not reference an ASAP object *EXCPT_ENTITY_NOT_FOUND* - Raised when aSAPName references an ASAP object that does not exist *EXCPT_TOO_MANY_OPEN_ITERATORS* - Raised when maximum number of iterators that the EMS can support has been reached

7.4. EmsSession_I Interface

A handle to an instance of this interface is gained via the emsSessionInterface parameter of the getEmsSession() operation in EmsSessionFactory_I.

7.4.1. Operations

7.4.1.1. getSupportedManagers

```
void getSupportedManagers(
    out managerNames_T supportedManagerList)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request the manager interfaces that the EMS implements. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
➢ supportedManagerList	The list of manager names supported by the EMS in the form ::managerName_T where ::managerName_T is one of the following defined manager strings: "EMS" (mandatory) "ManagedElement" (mandatory) "MultiLayerSubnetwork" (mandatory) "TrafficDescriptor" "PerformanceManagement" "Protection" "EquipmentInventory" "Maintenance" "softwareAndDataManager" "transmissionDescriptor" "GuiCutThrough" (mandatory) Additional managerName strings can be defined without changing this IDL.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_ACCESS_DENIED - Raised in case of security violation

7.4.1.2. getManager

```
void getManager(
      in string managerName,
      out common::Common_I managerInterface)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to gain access to the specified manager interface without using the OMG Naming Service. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managerName	The class or type of manager object that the client wants to access (see getSupportedManagers()).
➢managerInterface	The actual object returned will implement the specified manager interface. However it is returned as a Common_I object so that this operation can be generic. The client should narrow the returned object to the correct object type.

raises	ExceptionType
EXCPT_NOT_IMPLEMENTED	Attempting to gain access to the following manager interfaces may not raise EXCPT_NOT_IMPLEMENTED: EMS, ManagedElement, MultiLayerSubnetwork, GuiCutThrough
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support the manager EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_ACCESS_DENIED - Raised in case of security violation

7.4.1.3. getEventChannel

void getEventChannel(
out CosNotifyChannelAdmin::EventChannel eventChannel)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation allows an NMS to gain access to the event channel to receive notifications. It returns a reference to a NotifyChannel interface (which is an EventChannel).When the EMS supports the OMG Telecom Log service, this operation will return a reference to a NotifyLog interface (which is a NotifyChannel). **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
➢ **eventChannel**	The event channel(NotifyChannel or NotifyLog) to be used by the NMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_ACCESS_DENIED - Raised in case of security violation

7.4.1.4. modifyPassword

void modifyPassword(in string userName,
in string oldPassword,
in string newPassword)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation allows an NMS to modify password which is used to access EMS. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **userName**	The user name used by NMS.
❑ **oldPassword**	The old password used by NMS.
❑ **newPassword**	The new password NMS wanted.

7.5. EmsSessionFactory_I Interface

There is a single instance of the EmsSessionFactory_I. It is the entry point to the server/EMS.
This instance the object reference that the client uses to connect to the server.
This interface implements the version interface and will return the server IDL version when getVersion is called on it.

7.5.1. Operations

7.5.1.1. getEmsSession

```
void getEmsSession(
        in string user,
        in string password,
        in nmsSession::NmsSession_I client,
        out emsSession::EmsSession_I emsSessionInterface)
       raises(globaldefs::ProcessingFailureException);
};
```

General comment	
This operation allows the NMS to obtain the EmsSession_I object from which all managers of the EMS can be obtained. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ user	The (registered) user or application that is trying to access the server, can be empty string to indicate that no authentication mechanism is implemented by the server/EMS.
❑ password	The password of the user, can be empty string.
❑ client	A handle to the NmsSession_I object instantiated at the NMS, to which the returned EmsSession_I object has to be associated.
➢emsSessionInterface	A CORBA IOR for the EmsSession_I interface.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when client is invalid EXCPT_ACCESS_DENIED - Raised in case of security violation

7.6. MaintenanceMgr_I Interface

The MaintenanceMgr_I is used as a handle to gain access to the maintenance operation functionalities of the NML-EML interface.
A handle to an instance of this interface is gained via the emsSession::EmsSession_I::getManager() operation in the managerInterface parameter when the managerName "Maintenance" is used.

7.6.1. Operations

7.6.1.1. performMaintenanceOperation

```
void performMaintenanceOperation(
                in CurrentMaintenanceOperation_T maintenanceOperation,
                in MaintenanceOperationMode_T maintenanceOperationMode)
```

raises (globaldefs::ProcessingFailureException);

General comment	
This operation allows the NMS to operate and release the maintenance commands that are supported by a TP. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 EMLNBI1.4 EMLNBI1.5 SDHNBI1.5	
❑ **Input / Output ➢**	**Comment**
❑ maintenanceOperation	Information on the maintenance operation to perform.
❑ maintenanceOperationMode	Indicates whether the maintenance operation should be operated or released. For EMLNBI, the supported operation and its mapped alcatel operation are listed bellow: *FACILITY_LOOPBACK loopAndContinue & line *TERMINAL_LOOPBACK loopAndContinue & internal *FACILITY_FORCED_AIS non_loopAndContinue & line *TERMINAL_FORCED_AIS non_loopAndContinue & internal For SDHNBI, the supported operation and its mapped alcatel operation are listed bellow: *FACILITY_LOOPBACK InternalCont *TERMINAL_LOOPBACK LineCont *FACILITY_FORCED_AIS InternalAIS *TERMINAL_FORCED_AIS LineAIS
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. tpName does not reference a TP or maintenance operation is invalid) EXCPT_INVALID_INPUT - Raised when tpName does not reference an TP EXCPT_ENTITY_NOT_FOUND - Raised when tpName references TP object which does not exist EXCPT_UNABLE_TO_COMPLY - Raised when the operation is denied by the managed element (e.g. because of the current state of the object) EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.6.1.2. getActiveMaintenanceOperations

```
void getActiveMaintenanceOperations(
      in globaldefs::NamingAttributes_T tpOrMeName,
      in unsigned long how_many,
      out CurrentMaintenanceOperationList_T currentMaintenanceOpeationList,
      out CurrentMaintenanceOperationIterator_I cmoIt)
     raises (globaldefs::ProcessingFailureException);
  };
```

General comment

This operation allows the NMS to query the EMS to determine if any persistent maintenance commands have been invoked. This query is supported for the PTP, FTP, CTP, and ME objects. Supported products: EMLNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpOrMeName	The name of the PTP, FTP, CTP, or ME object. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
❑ how_many	Maximum number of maintenance operations to return in the first batch.
➢currentMaintenanceOperationList	First batch of maintenance operations.
➢cmoIt	Iterator to access the remaining maintenance operations.
raises *ProcessingFailureException*	**ExceptionType** *EXCPT_NOT_IMPLEMENTED* - Raised if EMS does not support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when tpOrMeName does not reference a valid object *EXCPT_ENTITY_NOT_FOUND* - Raised when tpOrMeName references an object that does not exist *EXCPT_NE_COMM_LOSS* - Raised when communications to the managed element is lost *EXCPT_TOO_MANY_OPEN_ITERATORS* - Raised when the maximum number of iterators that the EMS can support has been reached

7.7. ManagedElementMgr_I Interface

The managedElementManager is used to gain access to operations, which deal with managed elements and termination points.
A handle to an instance of this interface is gained via the emsSessionFactory::EmsSession_I::getManager() operation in Manager.

7.7.1. Operations

7.7.1.1. getAllManagedElements

```
void getAllManagedElements(
        in unsigned long how_many,
        out managedElement::ManagedElementList_T meList,
        out managedElement::ManagedElementIterator_I meIt)
        raises(globaldefs::ProcessingFailureException);
```

General comment
This allows an NMS to request details of all of the Managed Elements that are under the control of this ManagedElementMgr_I. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6

❑ Input / Output ➢	Comment
❑ how_many	Maximum number of MEs to report in the first batch.
➢ meList	First batch of Mes.
➢ meIt	Iterator to retrieve the remaining Mes.

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.7.1.2. getHighestLayerRateOfME

void getHighestLayerRateOfME(
in globaldefs::NamingAttributes_T managedElementName,
out transmissionParameters::LayerRate_T highestLayerRate)
raises(globaldefs::ProcessingFailureException);

General comment
This operation is used to get highest layer rate for a specified managed element. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.7.1.3. getHighestAlarmSeverityOfME

void getHighestAlarmSeverityOfME(
in globaldefs::NamingAttributes_T managedElementName,
out notifications::PerceivedSeverity_T highestPerceivedSeverity)
raises(globaldefs::ProcessingFailureException);

General comment
This operation is used to get highest alarm severity for a specified managed element. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.7.1.4. getAllGatewayManagedElements

void getAllGatewayManagedElements(
in unsigned long how_many,
out managedElement::ManagedElementList_T meList,

out managedElement::ManagedElementIterator_I meIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This allows an NMS to request details of all of the Gateway Managed Elements that are under the control of this ManagedElementMgr_I. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑how_many	Maximum number of MEs to report in the first batch.
➢meList	First batch of MEs.
➢meIt	Iterator to retrieve the remaining MEs.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.7.1.5. getAllManagedElementNames

void getAllManagedElementNames(
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllManagedElements(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑how_many	Maximum number of ME names to return in the first batch.
➢ nameList	First batch of ME names.
➢ nameIt	Iterator to retrieve remaining ME names.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.7.1.6. getAllGatewayManagedElementNames

```
void getAllGatewayManagedElementNames(
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllGatewayManagedElements(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑how_many	Maximum number of ME names to return in the first batch.
➢ nameList	First batch of ME names.
➢ nameIt	Iterator to retrieve remaining ME names.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.7.1.7. getAllPTPs

```
void getAllPTPs(
    in globaldefs::NamingAttributes_T managedElementName,
    in transmissionParameters::LayerRateList_T tpLayerRateList,
    in transmissionParameters::LayerRateList_T connectionLayerRateList,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment

This interface has been enhanced in version 3 to return both PTPs and FTPs (maximising inter-version compatibility). Two new operations have been added to getAllPTPsWithoutFTPs() and getAllFTPs() to allow more selective retrieval.
This allows an NMS to request all of the PTPs and FTPs on the specified Managed Element, that contain one or more of the NMS-specified PTP/FTP layer rates, and that are capable of supporting one or more of the NMS-specified connection layer rates.
If there are no PTPs/FTPs that match the layer constraints, an empty list is returned. A PTP/FTP will be returned regardless of connectivity to other managed elements and regardless of position in the subnetwork (both edgepoints of the subnetwork and the PTPs/FTPs that are internal to the subnetwork are reported).
In order to allow the NMS to deal with a large number of objects,this operation uses an iterator.

Supported products: EMLNBI1.0 SDHNBI1.0
EMLNBI1.1 SDHNBI1.1
EMLNBI1.2 SDHNBI1.2
EMLNBI1.3 SDHNBI1.3
EMLNBI1.4 SDHNBI1.4
EMLNBI1.5 SDHNBI1.5
EMLNBI1.6 SDHNBI1.6 PKTNBI1.6

❑ Input / Output ➢	Comment
❑ managedElementName	The name of the Managed Element for which to retrieve PTPs/FTPs.
❑ tpLayerRateList	List of PTP/FTP layer rates for which the PTPs/FTPs are to be fetched. A PTP/FTP must contain at least one of the layer rates specified to be reported. If the list is empty then all PTPs/FTPs (of all rates)are returned. Not supported in NBI 1.*.
❑connectionLayerRateList	List of connection layer rates for which the PTPs/FTPs are to be fetched. A PTP/FTP must support connections for at least one of the layer rates specified to be reported. If the list is empty then all PTPs/FTPs (for all connection rates) are returned. Not Supported in NBI 1.*.
❑how_many	Maximum number of PTPs/FTPs to report in the first batch.
➢ tpList	First batch of PTPs/FTPs.
➢ tpIt	Iterator to retrieve remaining PTPs/FTPs.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when managedElementName does not reference a managed element object, or tpLayerRateList or connectionLayerRateList contain undefined rates EXCPT_ENTITY_NOT_FOUND - Raised when managedElementName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.8. getAllPTPsWithoutFTPs

```
void getAllPTPsWithoutFTPs(
    in globaldefs::NamingAttributes_T managedElementName,
    in transmissionParameters::LayerRateList_T tpLayerRateList,
    in transmissionParameters::LayerRateList_T connectionLayerRateList,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This interface has been added in version 3 to return PTPs only. It allows more selective retrieval than the modified getAllPTPs() which now returns PTPs and FTPs. This operation has exactly the same behaviour as getAllPTPs() but instead of returning both PTPs and FTPs it returns solely PTPs **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ managedElementName	The name of the Managed Element.
❑ tpLayerRateList	List of PTP layer rates for which the PTPs are to be fetched. A PTP must contain at least one of the layer rates specified to be reported. If the list is empty then PTPs of all rates are returned.
❑ connectionLayerRateList	List of connection layer rates for which the PTPs are to be fetched. A PTP must support connections for at least one of the layer rates specified to be reported. If the list is empty then PTPs for all connection rates are returned.
❑ how_many	Maximum number of PTPs to return in the first batch.
➢ tpList	First batch of PTPs.
➢ tpIt	Iterator to retrieve remaining PTPs.
raises	**ExceptionType**
ProcessingFailureException	As for getAllPTPs().

7.7.1.9. getAllFTPs

```
void getAllFTPs(
    in globaldefs::NamingAttributes_T managedElementName,
    in transmissionParameters::LayerRateList_T tpLayerRateList,
    in transmissionParameters::LayerRateList_T connectionLayerRateList,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This interface has been added in version 3 to return FTPs only. It allows more selective retrieval than the modified getAllPTPs() which now returns PTPs and FTPs. **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ managedElementName	The name of the Managed Element.
❑ tpLayerRateList	List of FTP layer rates for which the FTPs are to be fetched. A FTP must contain at least one of the layer rates specified to be reported. If the list is empty then FTPs of all rates are returned.
❑ connectionLayerRateList	List of connection layer rates for which the FTPs are to be fetched. A FTP must support connections for at least one of the layer rates specified to be reported. If the list is empty then FTPs for all connection rates are returned.layer rates specified to be reported. If the list is empty then PTPs for all connection rates are returned.
❑ how_many	Maximum number of FTPs to return in the first batch.
➢ tpList	First batch of FTPs.
➢ tpIt	Iterator to retrieve remaining FTPs.

raises	ExceptionType
ProcessingFailureException	As for getAllPTPs().

7.7.1.10. getAllPTPNames

void getAllPTPNames(
in globaldefs::NamingAttributes_T managedElementName,
in transmissionParameters::LayerRateList_T tpLayerRateList,
in transmissionParameters::LayerRateList_T connectionLayerRateList,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This interface has been enhanced in version 3 to return both PTPs and FTPs (maximising inter-version compatibility). Two new operations have been added to getAllPTPNamesWithoutFTPs() and getAllFTPNames() to allow more selective retrieval. This operation has exactly the same behaviour as getAllPTPs(), but instead of returning the entire object structures, this operation returns their names. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ managedElementName	The name of the Managed Element.
❑ tpLayerRateList	List of PTP/FTP layer rates for which the PTPs/FTPs are to be fetched. A PTP/FTP must contain at least one of the layer rates specified to be reported. If the list is empty then PTPs/FTPs of all rates are returned. Not supported in NBI 1.*.
❑connectionLayerRateList	List of connection layer rates for which the PTPs/FTPs are to be fetched. A PTP/FTP must support connections for at least one of the layer rates specified to be reported. If the list is empty then PTPs/FTPs for all connection rates are returned. Not supported in NBI 1.*.
❑how_many	Maximum number of PTPs/FTPs to return in the first batch.
➢ nameList	First batch of PTPs/FTPs.
➢ nameIt	Iterator to retrieve the remaining PTPs/FTPs.
raises	**ExceptionType**
ProcessingFailureException	As for getAllPTPs().

7.7.1.11. getAllPTPNamesWithoutFTPs

void getAllPTPNamesWithoutFTPs(
in globaldefs::NamingAttributes_T managedElementName,
in transmissionParameters::LayerRateList_T tpLayerRateList,
in transmissionParameters::LayerRateList_T connectionLayerRateList,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

This interface has been added in version 3 to return PTP names only. It allows more selective retrieval than the modified getAllPTPNames() which now returns PTP and FTP names. This operation has exactly the same behaviour as getAllPTPsWithoutFTPs(), butinstead of returning the entire object structures, this operation returns their names. **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	The name of the Managed Element.
❑ tpLayerRateList	List of PTP layer rates for which the PTPs are to be fetched. A PTP must contain at least one of the layer rates specified to be reported. If the list is empty then PTPs of all rates are returned.
❑connectionLayerRateList	List of connection layer rates for which the PTPs are to be fetched. A PTP must support connections for at least one of the layer rates specified to be reported. If the list is empty then PTPs for all connection rates are returned.
❑how_many	Maximum number of PTPs to return in the first batch.
➢ nameList	First batch of PTPs.
➢ nameIt	Iterator to retrieve the remaining PTPs.
raises	**ExceptionType**
ProcessingFailureException	As for getAllPTPs().

7.7.1.12. getAllFTPNames

void getAllFTPNames (

in globaldefs::NamingAttributes_T managedElementName,

in transmissionParameters::LayerRateList_T tpLayerRateList,

in transmissionParameters::LayerRateList_T connectionLayerRateList,

in unsigned long how_many,

out globaldefs::NamingAttributesList_T nameList,

out globaldefs::NamingAttributesIterator_I nameIt)

raises(globaldefs::ProcessingFailureException);

This interface has been added in version 3 to return FTP names only. It allows more selective retrieval than the modified getAllPTPNames() which now returns PTP and FTP names. This operation has exactly the same behaviour as getAllFTPs(),but instead of returning the entire object structures, this operation returns their names. **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	The name of the Managed Element.
❑ tpLayerRateList	List of FTP layer rates for which the FTPs are to be fetched. A FTP must contain at least one of the layer rates specified to be reported. If the list is empty then FTPs of all rates are returned.
❑connectionLayerRateList	List of connection layer rates for which the FTPs are to be fetched. A FTP must support connections for at least one of the layer rates specified to be reported. If the list is empty then FTPs for all connection rates are returned.
❑how_many	Maximum number of FTPs to return in the first batch.
➢ nameList	First batch of FTPs.
➢ nameIt	Iterator to retrieve the remaining FTPs.
raises	**ExceptionType**

ProcessingFailureException	As for getAllPTPs().

7.7.1.13. getGTP

```
void getGTP(
        in globaldefs::NamingAttributes_T gtpName,
        out terminationPoint::GTP_T gtp)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns the GTP structure for the given GTP name **Supported products:** EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Name of the TP to retrieve.
➢ tp	The retrieved TP.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a termination point object EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a TP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.14. getAllGTPs

```
void getAllGTPs (
    in globaldefs::NamingAttributes_T managedElementName,
        in transmissionParameters::LayerRateList_T tpLayerRateList,
    in unsigned long how_many,
    out terminationPoint::GTPlist_T gtpList,
    out terminationPoint::GTPiterator_I gtpIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation allows an NMS to request all of the GTPs (on the given Managed Element) that contain one or more TPs at the specified layer rates. If there are no GTPs that match the layer constraints, an empty list is returned. **Supported products:** EMLNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	The name of the Managed Element for which to retrieve the GTPs.
❑ tpLayerRateList	List of layer rates for which the GTPs are to be fetched. A GTP must contain at least one TP having one of the specified layer rates. If the list is empty then all GTPs (of all rates) are returned.
❑ how_many	Maximum number of GTPs to report in the first batch.
➢ gtpList	First batch of GTPs.

➢ **gtpIt**	Iterator to retrieve remaining GTPs.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a termination point object EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a TP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.15. getAllGTPNames

void getAllGTPNames (
in globaldefs::NamingAttributes_T managedElementName,
in transmissionParameters::LayerRateList_T tpLayerRateList,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
The getAllGTPNames operation has the same behaviour as getAllGTPs(), except that only the GTP names are returned. **Supported products:** EMLNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **managedElementName**	The name of the Managed Element for which to retrieve the GTP names.
❑ **tpLayerRateList**	List of layer rates for which the GTP names are to be fetched. A GTP must contain at least one TP having one of the specified layer rates. If the list is empty then all GTP names (of all rates) are returned.
❑ **how_many**	Maximum number of GTP names to report in the first batch.
➢ **nameList**	First batch of GTP names.
➢ **nameIt**	Iterator to retrieve the remaining GTP names.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a termination point object EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a TP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.16. getContainingGTP

void getContainingGTP (
in globaldefs::NamingAttributes_T ctpName,
out terminationPoint::GTP_T gtp)
raises(globaldefs::ProcessingFailureException);

General comment	
The getContainingGTP operation returns the name of the GTP containing a given CTP **Supported products:** EMLNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **ctpName**	The name of the CTP for which the NMS wants to know the containing GTP.
➢ **gtp**	The name of the GTP containing the CTP.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a termination point object EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a TP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.17. getTP

```
void getTP(
    in globaldefs::NamingAttributes_T tpName,
    out terminationPoint::TerminationPoint_T tp)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This service returns the termination point structure for the given TP name (CTP, FTP or PTP). The termination point name must be explicit (a generic endpoint specification may not be used in this case). The termination point structure contains transmission parameters. The transmission parameters returned will be the parameters in place on the actual termination point on the NE. If there are no transmission parameters or the TP does not actually exist on the NE,then transmissionParams will be empty. The field transmissionParams will also be empty for "potential" ATM VP/VC CTPs. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ tpName	Name of the TP to retrieve.
➢ tp	The retrieved TP.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a termination point object EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a TP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.18. getTTIInformation

```
void getTTIInformation(
                    in globaldefs::NamingAttributes_T tpName,
                    out string ttiExpected,
                    out string ttiReceived,
                    out string ttiSend)
                   raises (globaldefs::ProcessingFailureException);
```

General comment	
This service returns the Trail Trace Identifier(TTI) of given TP.For TPs in different Layers, this operation return the TTI information in that Layer. Currently, only RS Layer TTI in WDM NE (i.e. J0 in WDM NE) is supported. **Supported products:** EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ tpName	Name of the TP whose TTI to be retrieved.
➢ ttiExpected	TTI expected, empty if not set.
➢ttiReceived	TTI received actually, empty if not set.
➢ttiSend	TTI to be sent, empty if not set.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a termination point who has TTI character, or the tp is not exist. EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.19. getManagedElement

```
void getManagedElement(
      in globaldefs::NamingAttributes_T managedElementName,
      out managedElement::ManagedElement_T me)
     raises (globaldefs::ProcessingFailureException);
```

General comment	
This service returns the Managed Element for the given managed element name. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ managedElementName	Name of the ME to retrieve.
➢ me	The retrieved ME.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when managedElementName does not reference a managed element object EXCPT_ENTITY_NOT_FOUND - Raised when managedElementName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to the managed element is lost

7.7.1.20. getContainedPotentialTPs

```
void getContainedPotentialTPs(
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRateList_T layerRateList,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the CTPs are contained by the specified termination point. This service returns all potential contained CTPs for a given TP. The TP may be a PTP, an FTP or a CTP. If the layerRateList is empty then contained CTPs at all flexible and/or static LayerRates are returned. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	The name of the PTP, FTP or CTP for which to get contained CTPs. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
❑ layerRateList	The list of the rates of the contained CTPs to report. An empty list indicates to the EMS to report all contained CTPs (of all rates).
❑ how_many	Maximum number of contained CTPs to report in the first batch.
➢tpList	First batch of contained CTPs.
➢tpIt	Iterator to retrieve the remaining contained CTPs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a PTP, FTP or CTP object or layerRateList contains undefined rates EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a PTP, FTP or CTP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.21. getContainedPotentialTPNames

```
void getContainedPotentialTPNames(
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRateList_T layerRateList,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getContainedPotentialTPs(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Name of the PTP, FTP or CTP for which to get contained CTPs. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
❑ layerRateList	The list of the rates of the contained CTPs to be reported. An empty list indicates to the EMS to get all contained CTPs (of all rates).
❑ how_many	Maximum number of contained CTPs to return in the first batch..
➢ nameList	First batch of contained CTPs.
➢ nameIt	Iterator to retrieve the remaining contained CTPs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a PTP, FTP or CTP object or layerRateList contains undefined rates EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a PTP, FTP or CTP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.22. getContainedInUseTPs

```
void getContainedInUseTPs(
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRateList_T layerRateList,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
  raises(globaldefs::ProcessingFailureException);
```

This service is used to retrieve the "in use" CTPs that are contained within a specific PTP, FTP or CTP, at specific layer rates. An "in use" CTP is defined as a CTP that is used by an SNCin any state (including pending), either as a CM end point or as an intermediate connection point, or a CTP that is terminated and mapped.This operation will be used when there are a large number of potential CTPs(e.g., in ATM).
Example of usage with respect to ATM:
To retrieve all actual ATM Network Interfaces associated with a PTP,this operation will be invoked using the PTP name as the tpName and LR_ATM_NI as the only layer rate in the layerRateList. The VPI and VCI ranges that are specified in the returned ATM Network Interface transmissionParams (i.e., Max_VPI_Bits and Max_VCI_Bits) can then be used to determine the potential VPI/VCI range. A subsequent invocation of this operation using an ATM Network Interface CTP as input can be used to determine which VPIs/VCIs are actually in use (with LR_ATM_VP and LR_ATM_VC included in the connectionRateList).
Example of usage with respect to SONET/SDH:
Consider an STM4 PTP with layerRate: LR_Optical_OC12_and_STM4.
Assume that the set of CTPs returned by operation getContainedPotentialTPs() contains one CTP with layerRate LR_STS3c_and_AU4_VC4 that is terminating an SNC (layerRate LR_STS3c_and_AU4_VC4), and another CTP with layerRate LR_STS3c_and_AU4_VC4 that is terminated and mapped (attribute tpMappingMode is set to TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING).The CTP with layerRate LR_STS3c_and_AU4_VC4 that is terminated and mapped contains one CTP with layerRate LR_VT2_and_TU12_VC12 that is involved in an SNC. None of the other contained CTPs have attribute tpMappingMode set to TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING, or are involved in an SNC.
Operation getContainedInUseTPs then returns three CTPs:
The CTP with layerRate LR_STS3c_and_AU4_VC4 that is terminating an SNC.
The CTP with layerRate LR_STS3c_and_AU4_VC4 that has attribute tpMappingMode set to TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING.
The CTP with layerRate LR_VT2_and_TU12_VC12 that is terminating an SNC.
For details on how TPs should be modelled, see layering
In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. See iterator overview for information on how iterators are used in this interface.
Supported products: SDHNBI1.5
SDHNBI1.6 PKTNBI1.6

❑ Input / Output ➢	Comment
❑ tpName	The name of the PTP, FTP or CTP for which to get contained actual CTPs. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
❑ LayerRateList	The list of rates of the contained actual CTPS to be returned. An empty list indicates to the EMS to get all contained actual CTPs (for all rates).
❑ how_many	The maximum number of CTPs to be returned in the first batch.
➢ tpList	First batch of contained in use CTPs.
➢ tpIt	Iterator to retrieve the remaining contained in use CTPs.

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a PTP, FTP or CTP object or layerRateList contains undefined rates EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a PTP, FTP or CTP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.23. getContainedInUseTPNames

void getContainedInUseTPNames(

in globaldefs::NamingAttributes_T tpName,
in transmissionParameters::LayerRateList_T layerRateList,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

This operation has exactly the same behaviour as getContainedInUseTPs(), but instead of returning the entire object structures, this operation returns their names.
In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. See iterator overviewfor information on how iterators are used in this interface.
Supported products: SDHNBI1.5
SDHNBI1.6 PKTNBI1.6

❑ Input / Output ➢	Comment
❑ tpName	The name of the TP for which to get contained in use TPs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See Object Namingfor further detail on FTP naming.
❑ LayerRateList	The list of rates of the contained in use CTPs to be returned. An empty list indicates to the EMS to get all contained in use CTPs (of all rates).
❑ how_many	Maximum number of CTP names to be returned in the first batch.
➢ nameList	First batch of CTP names.
➢ nameIt	Iterator to retrieve the remaining CTP names.

raises	ExceptionType
ProcessingFailureException	As for getContainedInUseTPs().

7.7.1.24. getContainedCurrentTPs

void getContainedCurrentTPs(
in globaldefs::NamingAttributes_T tpName,
in transmissionParameters::LayerRateList_T layerRateList,
in unsigned long how_many,
out terminationPoint::TerminationPointList_T tpList,
out terminationPoint::TerminationPointIterator_I tpIt)
raises(globaldefs::ProcessingFailureException);

This service is used to retrieve the "current" CTPs that are contained within a specific PTP, FTP or CTP, at specific layer rates. A "current" CTP is defined as a CTP that is either cross-connectable or cross-connected,in the current mapping configuration.
In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. See iterator overview for information on how iterators are used in this interface.
Supported products: EMLNBI1.5
PKTNBI1.6

❑ Input / Output ➢	Comment
❑ tpName	The name of the PTP, FTP or CTP for which to get current contained CTPs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See Object Naming for further detail on FTP naming.
❑ LayerRateList	The list of rates of the current contained CTPS to be returned. An empty list indicates to the EMS to get all current contained CTPs (of all rates).
❑ how_many	The maximum number of CTPs to be returned in the first batch.
➢ tpList	First batch of contained current CTPs.
➢ tpIt	Iterator to retrieve the remaining contained current CTPs.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a PTP, FTP or CTP object or layerRateList contains undefined rates * EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a PTP, FTP or CTP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.25. getContainedCurrentTPNames

void getContainedCurrentTPNames(
in globaldefs::NamingAttributes_T tpName,
in transmissionParameters::LayerRateList_T layerRateList,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

This operation has exactly the same behaviour as getContainedCurrentTPs(), but instead of returning the entire object structures, this operation returns their names.
In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. See iterator overviewfor information on how iterators are used in this interface.
Supported products: EMLNBI1.5
PKTNBI1.6

❑ Input / Output ➢	Comment
❑ tpName	The name of the PTP, FTP or CTP for which to get current contained CTPs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See Object Naming for further detail on FTP naming.
❑ LayerRateList	The list of rates of the current contained CTPS to be returned. An empty list indicates to the EMS to get all current contained CTPs (of all rates).
❑ how_many	Maximum number of CTP names to be returned in the first batch.
➢ nameList	First batch of CTP names.
➢ nameIt	Iterator to retrieve the remaining CTP names.

raises	ExceptionType
ProcessingFailureException	As for getContainedCurrentTPs().

7.7.1.26. getContainingTPs

void getContainingTPs(
in globaldefs::NamingAttributes_T tpName,
out terminationPoint::TerminationPointList_T tpList)
raises (globaldefs::ProcessingFailureException);

This service returns a list of the containing TPs given a CTP. This will return an PTP or FTP where there is only one level of containment. In a case of deeper containment this will return a list of CTPs and a PTP or FTP. Using the UPSR OC3 example used in getContainedPotentialTPs, getContainingTPs on the working T1 CTP will return a working STS1 CTP and a working OC3PTP. If the OC3 was in APS, then getContainingTPs on the T1 CTP would returnan STS1 CTP, an OC3 working PTP and an OC3 protect PTP. For details on how TPs should be modelled, see layering **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 EMLNBI1.6 SDHNBI1.6 PTKNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Name of the CTP for which containing CTPs and PTP/FTP are to be reported. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See Object Naming for further detail on FTP naming.
➢ tpList	List of the containing CTPs and PTPs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a CTP object EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a CTP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.7.1.27. getContainingTPNames

```
void getContainingTPNames(
    in globaldefs::NamingAttributes_T tpName,
    out globaldefs::NamingAttributesList_T tpNameList)
  raises (globaldefs::ProcessingFailureException);
```

This operation has exactly the same behaviour as getContainingTPs(), butinstead of returning the entire object structures, this operation returns their names. **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 EMLNBI1.6 SDHNBI1.6 PTKNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Name of the CTP for which to get the names of the containing TPs.
➢ tpNameList	List of the names of the containing TPs.
raises	**ExceptionType**
ProcessingFailureException	As for getContainingTPs().

7.7.1.28. getAllUnacknowledgedActiveAlarms

```
void getAllUnacknowledgedActiveAlarms(
    in globaldefs::NamingAttributes_T meName,
    in notifications::ProbableCauseList_T excludeProbCauseList,
    in notifications::PerceivedSeverityList_T excludeSeverityList,
    in unsigned long how_many,
    out notifications::EventList_T eventList,
```

out notifications::EventIterator_I eventIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This allows an NMS to request all of the active alarms and TCAs for the specified managed element that have not been acknowledged. Alarms that are not reported by the ME to the EMS should not be reported by this operation. Some alarms may be filtered out (excluded) by specifying their probable causes or severities. **Supported products:** EMLNBI1.6 PTKNBI1.6	
❑ Input / Output ➢	**Comment**
❑ meName	The name of the Managed Element for which to retrieve alarms and TCAs.
❑ excludeProbCauseList	List of probable causes to exclude from the output event list.
❑ excludeSeverityList	List of severities to exclude from the output event list.
❑ how_many	Maximum number of events to report in the first batch.
➢ eventList	First batch of events.
➢ eventIt	Iterator to retrieve the remaining events.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managed element object or excludeProbCauseList or excludeSeverityList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when meName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.29. getAllActiveAlarms

void getAllActiveAlarms(
`in globaldefs::NamingAttributes_T meName,
in notifications::ProbableCauseList_T excludeProbCauseList,
in notifications::PerceivedSeverityList_T excludeSeverityList,
in unsigned long how_many,
out notifications::EventList_T eventList,
out notifications::EventIterator_I eventIt)
raises(globaldefs::ProcessingFailureException);

General comment

This allows an NMS to request all of the active alarms and TCAs for the specified managed element.Alarms that are not reported by the ME to the EMS should not be reported by this operation. Some alarms may be filtered out (excluded) by specifying their probable causes or severities.
For Alcatel RM, this operation always return empty list because in RM system, there only report path, trail, physical connection, link connection alarms.
The result of this operation is independent of the filtering set up by the NMS for the notification service. Alarms which ASAP assigned severity is "NOTALARMED" should not be reported by this operation.

Supported products: EMLNBI1.0 SDHNBI1.0
EMLNBI1.1 SDHNBI1.1
EMLNBI1.2 SDHNBI1.2
EMLNBI1.3 SDHNBI1.3
EMLNBI1.4 SDHNBI1.4
EMLNBI1.5 SDHNBI1.5
EMLNBI1.6 SDHNBI1.6 PTKNBI1.6

❑ **Input / Output ➢**	**Comment**
❑ meName	The name of the Managed Element for which to retrieve alarms and TCAs.
❑ excludeProbCauseList	List of probable causes to exclude from the output event list.
❑ excludeSeverityList	List of severities to exclude from the output event list.
❑how_many	Maximum number of events to report in the first batch.
➢ eventList	First batch of events.
➢ eventIt	Iterator to retrieve the remaining events.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managed element object or excludeProbCauseList or excludeSeverityList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when meName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.30. setTPData

```
void setTPData(
        in subnetworkConnection::TPData_T tpInfo,
        out terminationPoint::TerminationPoint_T modifiedTP)
        raises(globaldefs::ProcessingFailureException);
```

General comment

This service allows the NMS to set parameters on a specified Termination Point (CTP, PTP or FTP).
Currently only support one parameter set per setTPData invoke.
This operation is best effort (except where specified otherwise for a particular parameter). The results of the operation are returned so that the NMS is aware of what modifications succeeded.
If the source TP of a broadcast system is used as input, then the entire multipoint system will be affected based on the new parameter values for the source TP.
The tpMappingMode may be set with this operation. When the mode is set to TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING, the EMS will terminate the specified CTP. In this case the EMS must create the specified CTP on the NE if it does not actually exist.
Setting the tpMappingMode of an ATM VP or VC CTP can only be done if the CTP has been created. Note that the tpMappingMode can be set only on the ingress and egress CTPs of an ATM SNC since they are the only ones which may not be cross-connected.
No change to tpMappingMode or trafficDescriptors will take place if there is any active cross connect (NE cross connect) using the CTP passed in parameter.
The transmissionParams is a "delta" list that needs to be applied to the specified TP, i.e. only a subset of the parameters may be specified in the list, in which case only those should be applied in the NE. Transmission parameters are used to associate a TCA profile with a TP using this service. In this case there are potentially additional failure modes (see exceptions).

Supported products: EMLNBI1.2
EMLNBI1.3
EMLNBI1.4
EMLNBI1.5
EMLNBI1.6

❑ **Input / Output ➢**	**Comment**
❑ tpInfo	Details of modifications required. From EMLNBI1.2, following modifications can beperformed: 1. WDM J0 expected value modification. In transmissionParams, two modes of TTI expected value can be set: TTIExpected_mode1 : <15 bytes string>; TTIExpected_oneRepeatedByte : <1 byte char> Examples: the pair (TTIExpected_mode1, "123456789ABCDEF") indicates the J0 expected value will be set to "123456789ABCDEF" the pair (TTIExpected_oneRepeatedByte, 'j') indicates the J0 expected value will be set to 16 repeated 'j', i.e. "jjjjjjjjjjjjjjjj". 2. WDM ALS(APSD) control status value(i.e. ALS status enable/disable) In transmissionParams, "ALSStatus" can be set: For single laser: ALSStatus: <LASERSTATUS> For multi laser: ALSStatus: <LaserId=LASERSTATUS> The defination of LaserId and LASERSTATUS refer to TerminationPoint_T. From EMLNBI1.3, follow modifications can be perform: Ethernet TP attribute: flowControl Ethernet TP attribute: vlanTag Ethernet TP attribute: defaultVlanID From EMLNBI1.5, following SDH TTI modifications can be performed: In transmissionParams, two type of tti is supported, as TrailTraceExpectedRx and TrailTraceActualTx. There are TWO modes(mode1 and oneRepeatedByte) in J0, and THREE(mode1,oneRepeatedByte and pathTrace) in j1 and j2. For mode1, the value must contain at least 15 bytes. For oneRepeatedByte, the value should have only one byte. For pathTrace, no limit.

	AS : (TrailTraceExpectedRx, mode1:<15 bytes string>) (TrailTraceExpectedRx, oneRepeatedByte:<1 byte char>). (TrailTraceExpectedRx, disabled) --for j1,j2 (TrailTraceActualTx, mode1:<15 bytes string>) (TrailTraceActualTx, oneRepeatedByte:<1 byte char>). Examples: pair (TrailTraceExpectedRx, mode1:123456789ABCDEF) indicates the expected value r> will be set to "123456789ABCDEF" pair (TrailTraceExpectedRx, oneRepeatedByte:j) indicates the expected value will be set to 16 repeated 'j', i.e. "jjjjjjjjjjjjjjjj". pair (TrailTraceExpectedRx, disabled) indicates the expected value will be set disabled(means that, the expected value is disabled. but you still can set it as other mode)
➢ modifiedTP	Result of modification.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_ENTITY_NOT_FOUND - Raised if the TP referred to in tpInfo does not exist. EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_INVALID_INPUT - Raised when an input parameter is syntactical incorrect and raised when a parameter is identified as only "settable" EXCPT_UNABLE_TO_COMPLY - Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort"

7.7.1.31. getAllCrossConnections

```
void getAllCrossConnections(
    in globaldefs::NamingAttributes_T managedElementName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out subnetworkConnection::CrossConnectList_T ccList,
    out subnetworkConnection::CCIterator_I cclt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request a list of the cross-connects for the specified managed element at the specified layer rates. This operation returns cross-connects between CTPs/FTPs and between GTPs In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. For 1626/1696 NE in EMLNBI,additionInfo in CrossConnect_T have some special attribute: ttpIndicator[d]--[YES|NO]. It means that if this tp is kind of TTP. isFixed--[true|false]. It means if this cross if fixed. **Supported products:** EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ managedElementName	Name of the Managed Element for which to retrieve CCs.
❑ connectionRateList	List of rates for which to retrieve CCs. This must not be an empty list. In this case an INVALID_INPUT exception is thrown. For current version, it is meanlingless.

❑ how_many	Maximum number of CCs to report in the first batch.
➢ ccList	First batch of CCs.
➢ ccIt	Iterator to retrieve remaining CCs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if connectionRateList is empty or contains invalid rates, or if managedElementName does not reference a managed element. EXCPT_ENTITY_NOT_FOUND - Raised when managedElementName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.32. getCrossConnectionById

void getCrossConnectionById(
in globaldefs::NamingAttributes_T managedElementName,
in string id,
out subnetworkConnection::CrossConnect_T cc)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation retrieves the crossConnection by Id in single NE. **Supported products:** EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	The networkElement where the crossConnection staied in.
❑ id	The crossConnection Id, the id is unique in the single NE.
➢ cc	The retrived crossConnection.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INVALID_INPUT - Raised if connectionRateList is empty or contains invalid rates, or if managedElementName does not reference a managed element.

7.7.1.33. getCrossConnection

void getCrossConnection(
in globaldefs::NamingAttributes_T ccName,
out subnetworkConnection::CrossConnect_T cc)
raises(globaldefs::ProcessingFailureException);

General comment

This operation retrieve the CrossConnect_T by given crossconnection name. The CrossConnect_T's name hierarchy is: name="EMS";value="CompanyName/EMSname" name="ManagedElement";value="ManagedElementName" name="CrossConnection";value="CrossConnectionName" **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ ccName	Name of the CC to retrieve.
➢ cc	The retrived crossConnection.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_ENTITY_NOT_FOUND - Raised when ccName references an CC object that does not exist

7.7.1.34. getCrossConnectionsByName

```
void getCrossConnectionsByName(
        in globaldefs::NamingAttributesList_T ccNames,
        in unsigned long how_many,
        out subnetworkConnection::CrossConnectList_T ccList,
        out subnetworkConnection::CCIterator_I ccIt  )
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation retrieves the CrossConnect_Ts by given crossconnection names. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ ccNames	Names of the CCs to retrieve.
❑ how_many	Maximum number of CCs to report in the first batch.
➢ ccList	First batch of CCs.
➢ ccIt	Iterator to retrieve remaining CCs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.7.1.35. getAllCrossConnectionsName

```
void getAllCrossConnectionsName(
        in globaldefs::NamingAttributes_T managedElementName,
        in unsigned long how_many,
        out globaldefs::NamingAttributesList_T ccNames,
        out globaldefs::NamingAttributesIterator_I ccNameIt )
        raises(globaldefs::ProcessingFailureException);
```

General comment

This operation retrieve all crossconnection names in given NE. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	Name of the NE.
❑ how_many	Maximum number of CC names to report in the first batch.
➢ ccNames	First batch of CC names.
➢ ccNameIt	Iterator to retrieve remaining CC names.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_ENTITY_NOT_FOUND - Raised when managedElementName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost.

7.7.1.36. createCrossConnection

void createCrossConnection(
in globaldefs::NamingAttributes_T fromTp,
in globaldefs::NamingAttributesList_T toTpList,
in boolean bidirection,
out globaldefs::NamingAttributes_T crossConnectionName)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation create a crossconnection between TPs. (Only point to point crossconnection creation supported in EMLNBI1.4.) **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ fromTp	The from Tp name.
❑ toTpList	The to Tp name list.(Currently, only one toTp supported.)
❑bidirection	Bidirection or not.
➢ crossConnectionName	If creation successful, contains the created cc's name.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_ENTITY_NOT_FOUND - Raised when from or to tp name references an tp object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.7.1.37. deleteCrossConnection

void deleteCrossConnection(
in globaldefs::NamingAttributesList_T crossConnectionNames,
out globaldefs::NamingAttributesList_T deletedCrossConnectionNames)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation remove some crossconnections.(Only point to point crossconnection deletion supported in EMLNBI1.4.) **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ crossConnectionNames	Names of cc be removing.
➢ deletedCrossConnectionNames	Names of cc removed successfully
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.7.1.38. getPotentialFixedCCs

```
void getPotentialFixedCCs(
    in globaldefs::NamingAttributes_T inputTP,
    out globaldefs::NamingAttributes_T ContainingTP,
    out globaldefs::NamingAttributes_T potentialCCList)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
The operation is used to retrieve fixed connection schemes related to normal and inverse multiplexing. Fixed cross connections are cross connections which cannot be deleted by an NMS. **Supported products:** EMLNBI1.6 PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **inputTP**	any TP of the ME. The operation will return the multiplexing or inverse multiplexing scheme in which this TP is involved. The input TP may be either the containing TP or one of the end TPs of the portentialCCList.
➢ **ContainingTP**,	The TP supporting the attribute clientConnectivity or serverConnectivity
➢ **potentialCCList**	the list of fixed Cross Connect of that multiplexing scheme, i.e. the list of cross connects that will exist when the containing TP clientConnectivity or serverConnectiivty is set to "Connected". The A_end TPs of the cross connects should be the low order TPs that can be flexibly cross connected, and the Z_end TPs the TP client of the high order TP. The list of cross connect does not take any specific order. If there are no potential fixed cross connects and empty list is returned.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if connectionRateList is empty or contains invalid rates, or if managedElementName does not reference a managed element. EXCPT_ENTITY_NOT_FOUND - Raised when managedElementName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.39. getAllFixedCrossConnections

void getAllFixedCrossConnections(
in globaldefs::NamingAttributes_T managedElementName,
in transmissionParameters::LayerRateList_T connectionRateList,
in unsigned long how_many,
out subnetworkConnection::CrossConnectList_T ccList,
out subnetworkConnection::CCIterator_I ccIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllCrossConnections(), but instead returns only fixed Cross Connection object structures. See Subnetwork Connection Types for an explanation of fixed SNCs. Only supported in EMLNBI. A cross connection is identified as fixed using additional information. See Additional Information Usage for detail on additional informationfor SNCs, cross connections and TPs. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. See iterator overview for information on how iterators are used in this interface. **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	Name of the Managed Element for which to retrieve CCs.
❑ connectionRateList	List of rates for which to retrieve CCs. This must not be an empty list. In this case an INVALID_INPUT exception is thrown.
❑ how_many	Maximum number of CCs to report in the first batch.
➢ ccList,	First batch of CCs.
➢ ccIt	Iterator to retrieve remaining CCs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if connectionRateList is empty or contains invalid rates, or if managedElementName does not reference a managed element. EXCPT_ENTITY_NOT_FOUND - Raised when managedElementName references an ME object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.7.1.40. getLcasStatus

void getLcasStatus(
in globaldefs::NamingAttributes_T tpName,
out boolean enableState)
raises(globaldefs::ProcessingFailureException);

General comment	
The NMS uses this operation to request the lcas status of tp. **Supported products:** EMLNBI1.4 EMLNBI1.5	
❑ Input / Output ➢	**Comment**

❑ tpName	Specifies lcas of which tp will be qured. This tp should be XVirutalTTP.
➢ enableState	the result of this operation.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if the EMS does not support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised if the tpName is invalid *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost and this prevents getting the lcas

7.7.1.41. setLcasStatus

void setLcasStatus(
in globaldefs::NamingAttributes_T tpName,
in boolean enableState)
raises(globaldefs::ProcessingFailureException);

General comment	
The NMS uses this operation to set the lcas status of tp. **Supported products:** EMLNBI1.4 EMLNBI1.5	
❑ Input / Output ➢	**Comment**
❑ tpName	Specifies lcas of which tp will be set. This tp should be XVirutalTTP.
❑ enableState	the state of lcas which will be set.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if the EMS does not support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised if the tpName is invalid *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost and this prevents setting lcas

7.7.1.42. getEncapsulateProtocol

void getEncapsulateProtocol(
in globaldefs::NamingAttributes_T tpName,
out string protocolName)
raises(globaldefs::ProcessingFailureException);

General comment	
The NMS uses this operation to get the encapsulate protocol of tp. **Supported products:** EMLNBI1.4 EMLNBI1.5	
❑ Input / Output ➢	**Comment**
❑ tpName	Specifies lcas of which tp will be get. This tp should be XVirutalTTP.
➢ protocolName	the result which defined protocol name will be return.
raises	**ExceptionType**

ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if the EMS does not support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised if the tpName is invalid *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost and this prevents getting protocol name

7.7.1.43. setEncapsulateProtocol

void setEncapsulateProtocol(
in globaldefs::NamingAttributes_T tpName,
in string protocolName)
raises(globaldefs::ProcessingFailureException);

General comment	
The NMS uses this operation to set the encapsulate protocol of tp. **Supported products:** EMLNBI1.4 EMLNBI1.5	
❑ Input / Output ➢	**Comment**
❑ tpName	Specifies lcas of which tp will be set. This tp should be XVirutalTTP.
➢ protocolName	the protocol name which will be set. Protocol name shoud be one of the following list: *gFPnullExtensionNoFCS * gFPnullExtensionWithFCS * gFPlinearExtensionNoFCS * gFPlinearExtensionWithFCS * gFPpacketConcBasic * gFPpacketConcExtended * x86 * hDLC * unknown * gFPnullExtensionNoFCSTransparent * gFPnullExtensionWithFCSTransparent * gFPlinearExtensionNoFCSTransparent * gFPlinearExtensionWithFCSTransparent * Different Ne only support a subset according to ne type.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if the EMS does not support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised if the tpName is invalid *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost and this prevents set protocol name

7.7.1.44. getContainingSubnetworkNames

void getContainingSubnetworkNames(
in globaldefs::NamingAttributes_T managedElementName,
out globaldefs::NamingAttributesList_T subnetNames)
raises (globaldefs::ProcessingFailureException);

General comment

This service returns the list of subnetwork names that the Managed Element supplied as an input parameter belongs to. globaldefs::NamingAttributes managedElementName: The name of the managed element for which to retrieve the containing subnetwork names. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ managedElementName	The name of the managed element for which to retrieve the containing subnetwork names.
➢ subnetNames	The names of the subnetworks this NE belongs to.
raises *ProcessingFailureException*	**ExceptionType** *EXCPT_NOT_IMPLEMENTED* - Raised if the EMS does not support this service *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised if the tpName is invalid *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost and this prevents set protocol name *EXCPT_UNABLE_TO_COMPLY* - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, it may raise this exception. *EXCPT_ENTITY_NOT_FOUND* - Raised when managedElementName references an ME object that does not exist

7.8. MultiLayerSubnetworkMgr_I Interface

The multiLayerSubnetworkMgr_I is used to gain access to subnetworks and their operations.
A handle to an instance of this interface is gained via the emsSession::EmsSession_I::getManager() operation in Manager.

7.8.1. Operations

7.8.1.1. getAllManagedElements

```
void getAllManagedElements(
    in globaldefs::NamingAttributes_T subnetName,
    in unsigned long how_many,
    out managedElement::ManagedElementList_T meList,
    out managedElement::ManagedElementIterator_I melt)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request a list of the ManagedElements that are associated with the specified Subnetwork. This operation also returns the NEs in the all low level subnetworks of the given subnetwork. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6 , PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ how_many	Maximum number of managed elements to return in the first batch.

➢ meList	First batch of managed elements.
➢ meIt	Iterator to retrieve the remaining managed elements.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.2. getAllManagedElementNames

void getAllManagedElementNames(
in globaldefs::NamingAttributes_T subnetName,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllManagedElements(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6 , PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ subnetName	Name of the subnetwork.
❑ how_many	Maximum number of managed element names to return in the first batch.
➢ nameList	First batch of managed element names.
➢ nameIt	Iterator to retrieve the remaining managed element names.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.3. getMultiLayerSubnetwork

void getMultiLayerSubnetwork(
in globaldefs::NamingAttributes_T subnetName,

out MultiLayerSubnetwork_T subnetwork)
raises(globaldefs::ProcessingFailureException);

General comment	
This service returns a Subnetwork given a subnetwork name. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork to retrieve.
➢ subnetwork	Subnetwork structure returned.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references a multiLayerSubnetwork object that does not exist

7.8.1.4. getNextLevelMultiLayerSubnetworks

void getNextLevelMultiLayerSubnetworks(
in globaldefs::NamingAttributes_T subnetName,
in unsigned long how_many,
out SubnetworkList_T sList,
out SubnetworkIterator_I sIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation returns the next level subnetworks of given subnetwork. If the subnetName is empty, this operation will return the top level subnetworks in the EMS. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ how_many	Maximum number of managed elements to return in the first batch.
➢sList	First batch of subnetworks.
➢sIt	Iterator to retrieve the remaining subnetworks.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references an object that does not exist EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.5. getNextLevelMultiLayerSubnetworkNames

void getNextLevelMultiLayerSubnetworkNames(
in globaldefs::NamingAttributes_T subnetName,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getNextLevelMultiLayerSubnetworks(), but instead of returning the entire object structures, this operation returns their names. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ how_many	Maximum number of subnetwork names to return in the first batch.
➢ nameList	First batch of subnetwork names.
➢ nameIt	Iterator to retrieve the remaining subnetwork names.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references an object that does not exist EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.6. getAllTopologicalLinks

void getAllTopologicalLinks(
in globaldefs::NamingAttributes_T subnetName,
in unsigned long how_many,
out topologicalLink::TopologicalLinkList_T topoList,
out topologicalLink::TopologicalLinkIterator_I topoIt)
raises(globaldefs::ProcessingFailureException);

General comment

This service returns a list of TopologicalLinks which exist inside the Subnetwork whose name is passed as a parameter. This operation also returns the TopologicalLinks in the all low level subnetworks of the given subnetwork.
In order to allow the NMS to deal with a large number of objects, this operation uses an iterator.

Supported products: SDHNBI1.0
SDHNBI1.1
SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ subnetName	The name of the Subnetwork.
❑ how_many	Maximum number of topological links to return in the first batch.
➢ topoList	First batch of topological links.
➢ topoIt	Iterator to retrieve the remaining topological links.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.7. getAllTopologicalLinkNames

```
void getAllTopologicalLinkNames(
    in globaldefs::NamingAttributes_T subnetName,
    in  unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment

This operation has exactly the same behaviour as getAllTopologicalLinks, but instead of returning the entire object structures, this operation returns their names.
In order to allow the NMS to deal with a large number of objects, this operation uses an iterator.

Supported products: SDHNBI1.0
SDHNBI1.1
SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ subnetName	The name of the Subnetwork.
❑ how_many	Maximum number of topological link names to return in the first batch.

➢ nameList	First batch of topological link names.
➢ nameIt	Iterator to retrieve the remaining topological link names.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference a multiLayerSubnetwork object EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.8. getTopologicalLink

void getTopologicalLink(
in globaldefs::NamingAttributes_T topoLinkName,
out topologicalLink::TopologicalLink_T topoLink)
raises(globaldefs::ProcessingFailureException);

General comment	
This service returns a subnetwork topological link given its name. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ topoLinkName	Name of the subnetwork topological link to retrieve.
➢ topoLink	Subnetwork topological link returned.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when topoLinkName does not reference a subnetwork topologicallink object EXCPT_ENTITY_NOT_FOUND - Raised when topoLinkName references a subnetwork topological link object that does not exist

7.8.1.9. getAssociatedTP

void getAssociatedTP(
in globaldefs::NamingAttributes_T tpName,
out terminationPoint::TerminationPointList_T tpList)
raises(globaldefs::ProcessingFailureException);

General comment

This service returns a list of PSR (UPSR or SNCP) associated termination points for the TP whose name is passed as a parameter. To create an SNC between CTPs in a TOPO_OPEN_PSR for instance, the NMS needs to have two zEnd CTPs which are in the same SONET or SDH timeslot. The service allows the NMS to query the associated CTP of a given CTP, associated PTP given a PTP or associated FTP of a given FTP. The timeslot of the TPs will be the same in the case of a TOPO_OPEN_PSR subnetwork, but the names of the two TPs will be different. This operation is symmetric on the associated TPs. Given a working TP, the associated TP will be the protecting TP. Note: The termination point name must be explicit (a generic endpoint specification may not be used in this case). When this service is invoked with a TP that is not an edge point, the returned TPs will be on the same Managed Element. **Supported products:** EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	The name of the TP for which to retrieve associated TPs.
❑ tpList	The PSR associated TPs. If there are no PSR associated TPs, then an empty list is returned.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a CTP, FTP or PTP EXCPT_ENTITY_NOT_FOUND - Raised when tpName references a CTP, FTP or PTP object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.8.1.10. getSNCsByUserLabel

```
void getSNCsByUserLabel(
    in string userLabel,
    out subnetworkConnection::SubnetworkConnectionList_T sncList)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This operation will return the SubnetworkConnection structures for the SNCs whose userLabel is supplied as a parameter. If the SNC represents a Control Plane Connection, the operation is rejected and an exception is raised. This operation will provide a bundled SNC structure.if the name provided is that of a bundled SNC **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ userLabel	The userLabel of the SNCs to retrieve.
➢ sncList	The SNCs retrieved.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference an multiLayerSubnetwork object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.11. getAllSubnetworkConnections

```
void getAllSubnetworkConnections(
    in globaldefs::NamingAttributes_T subnetName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out subnetworkConnection::SubnetworkConnectionList_T sncList,
    out subnetworkConnection::SNCIterator_I sncIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request a list of the SNCs for the specified Subnetwork at the specified connectionRates. This operation also returns the subnetworkConnections in all low level Subnetworks of the given subnetwork. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑connectionRateList	List of rates of the SNCs to be reported. If an empty list is specified, then all SNCs of all rates are to be reported.
❑ how_many	Maximum number of SNCs to be reported in the first batch.
➢ sncList	First batch of SNCs.
➢ sncIt	Iterator to retrieve the remaining SNCs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference an multiLayerSubnetwork object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.12. getAllSubnetworkConnectionNames

```
void getAllSubnetworkConnectionNames(
    in globaldefs::NamingAttributes_T subnetName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment
This operation has exactly the same behaviour as getAllSubnetworkConnections(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ subnetName	Name of the subnetwork.
❑connectionRateList	List of rates of the SNC names to be reported. If an empty list is specified, then all SNC names of all rates are to be reported.
❑ how_many	Maximum number of SNC names to be reported in the first batch.
➢ nameList	First batch of SNC names.
➢ nameIt	Iterator to retrieve the remaining SNC names.

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference an multiLayerSubnetwork object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.13. getAllSubnetworkConnectionsWithTP

```
void getAllSubnetworkConnectionsWithTP (
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out subnetworkConnection::SubnetworkConnectionList_T sncList,
    out subnetworkConnection::SNCIterator_I sncIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment

This allows an NMS to request a list of the SNCs using thespecified termination point at the specified connection rates. A SNC is using the specified TP if any of its routes, intended and/or backup, in any state, use this TP. A TP may be a PTP in which case a full list of SNCs using any of its contained CTPs is required. A TP may be:

a CTP, in which case SNCs using that CTP or any of its contained CTPs are required. This includes SNCs in which any of the referenced CTPs participate that match the specified connection rate filter and bundled SNCs that are associated with any GTPs in which any of the referenced CTPs are grouped.

an FTP, in which case a full list of SNCs using the FTP or any of its contained CTPs is required. This includes SNCs in which the FTP or any of the referenced CTPs participate that match the specified connection rate filter and bundled SNCs that are associated with any GTPs in which any of the referenced CTPs are grouped.

a GTP, in which case the bundled SNCs in which that GTP has a role are required. The layerRate is set to LR_Not_Applicable

Both end CTPs/FTPs/GTPs and intermediate CTPs/FTPs/GTPs are considered. (see also Bundled SNC overview for further details).

All legs of a broadcast system can be retrieved using this operation where the source TP of the broadcast system is used as input to the operation.The output will be the list of individual SNCs that make up the broadcast system.

In order to allow the NMS to deal with a large number of objects,this operation uses an iterator.

Supported products: SDHNBI1.5
SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ tpName	Termination point for which to report SNCs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See Object Naming for further detail on FTP naming.
❑ connectionRateList	The list of rates of the SNCs to be reported.If an empty list is specified, then all SNCs of all rates are to be reported.
❑ how_many	Maximum number of SNCs to report in the first batch.
➢ sncList	First batch of SNCs.An SNC is only reported if it respects both the tpName and connectionRateList filters.
➢ sncIt	Iterator to retrieve the remaining SNCs..

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if tpName does not reference a terminationPoint object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when tpName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is # lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.14. getAllSubnetworkConnectionNamesWithTP

```
void getAllSubnetworkConnectionNamesWithTP (
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment

This operation has exactly the same behaviour as getAllSubnetworkConnectionsWithTP(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Termination point for whichto report SNCs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See Object Naming for further detail on FTP naming.
❑ connectionRateList	The list of rates of the SNCs to be reported.If an empty list is specified, then all SNCs of all rates are to be reported.
❑ how_many	Maximum number of SNCs to report in the first batch.
➢ nameList	First batch of SNC names.
➢ nameIt	Iterator to retrieve the remaining SNC names.
raises	**ExceptionType**
ProcessingFailureException	As for getAllSubnetworkConnectionsWitTP().

7.8.1.15. getRoute

void getRoute(
in globaldefs::NamingAttributes_T sncName,
in boolean includeHigherOrderCCs,
out subnetworkConnection::Route_T route)
raises (globaldefs::ProcessingFailureException);

General comment	
This service returns the route for the SNC whose name is specified as a parameter. When the SDHNBI parameter RETURN_EMBEDED_ROUTE is true(default) the operation returns the embeded SNC route information of the given SNC, otherwise just returns current level SNC's route information. This mehtod return only the main route information of the SNC even if the current activated route is not the main one. The SNC should be activated before it's route information can be retrieved. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ sncName	The name of the SNC.
❑ includeHigherOrderCCs	Specifies whether the higher order CCs of other SNCs used to carry the queried SNC have to be included in addition to the CCs of the queried SNC.(This parameter is not supported currently.Method always return the same layer CCs.) Not supported currently.
➢ route	The route of the SNC.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when sncName does not reference a subnetworkConnection object EXCPT_ENTITY_NOT_FOUND - Raised when sncName references an SNC object that does not exist

7.8.1.16. getRouteAndTopologicalLinks

void getRouteAndTopologicalLinks(
in globaldefs::NamingAttributes_T sncName,
out subnetworkConnection::Route_T route,
out topologicalLink::TopologicalLinkList_T topologicalLinkList)
raises(globaldefs::ProcessingFailureException);

General comment	
Like getRoute(),this service returns the route, in terms of crossconnects, for the SNC whose name is specified as a parameter. This service also returns the list of topological links for that SNC that are used in the route. This operation return only the main route information of the SNC even if the current activated route is not the main one. This operation only returns the resource in the same layer of the SNC. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ sncName	The name of the SNC.
➢ route	The route of the SNC.
➢topologicalLinkList	The list of the topological links.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when sncName does not reference a subnetworkConnection object EXCPT_ENTITY_NOT_FOUND - Raised when sncName references a subnetworkConnection object that does not exist

7.8.1.17. getSNC

void getSNC(
in globaldefs::NamingAttributes_T sncName,
out subnetworkConnection::SubnetworkConnection_T snc)
raises (globaldefs::ProcessingFailureException);

General comment

This operation will return the SubnetworkConnection structure for the SNC whose name is supplied as a parameter. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ sncName	The name of the SNC to retrieve.
➢ snc	The SNC structure retrieved.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when sncName does not reference a subnetworkConnection object EXCPT_ENTITY_NOT_FOUND - Raised when sncName references an SNC object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.8.1.18. disableLCAS

void disableLCAS(
in globaldefs::NamingAttributes_T sncName,
out string errorReason)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation is to disable LCAS functionality on a SNC (only for RM path).LCAS disabling needs all server trails are not involved in traffic. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ sncName	The name of the SNC.
➢ errorReason	Error string, maybe contains the error reason.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of internal error of EMS. EXCPT_INVALID_INPUT - Raised in case of invalid input from NMS. EXCPT_NOT_IN_VALID_STATE - Raised in case of some server trails are involved in traffic.

7.8.1.19. enableLCAS

void enableLCAS(
in globaldefs::NamingAttributes_T sncName,
out string errorReason)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation is to enable LCAS functionality on a SNC (only for RM path).LCAS enabling needs all server trails are not involved in traffic. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ sncName	The name of the SNC.
➢ errorReason	Error string, maybe contains the error reason.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of internal error of EMS. EXCPT_INVALID_INPUT - Raised in case of invalid input from NMS. EXCPT_NOT_IN_VALID_STATE - Raised in case of some server trails are involved in traffic.

7.8.1.20. activateIdleTrailsOnPath

```
void activateIdleTrailsOnPath(
        in globaldefs::NamingAttributes_T pathName,
        in unsigned long numberOfTrail,
        in string activateMode,
        out string errorReason)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation activates a number of idle server trails on a LCAS enabled path. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pathName	The LCAS enabled path name.
❑numberOfTrail	Indicates how many idle server trails to be activated.
❑activateMode	Available values: "Normal", "Forced" In case of "Normal", if the trail is an alarm trail, this operation will fail. In case of "Forced", the operation will ignore the trail alarm state and go on anyway.
➢ errorReason	Error string maybe contains the error reason.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of internal error of EMS. EXCPT_INVALID_INPUT - Raised in case of invalid input of NMS.

7.8.1.21. deactivateTrailsOnPath

```
void deactivateTrailsOnPath(
        in globaldefs::NamingAttributes_T pathName,
        in unsigned long numberOfTrail,
        out string errorReason)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation deactivates a number of server trails on a LCAS enabled path. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pathName	The LCAS enabled path name.
❑numberOfTrail	Indicates how many server trails to be deactivated.
➢ errorReason	Error string maybe contains the error reason.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of internal error of EMS. EXCPT_INVALID_INPUT - Raised in case of invalid input of NMS.

7.8.1.22. increaseNewTrailsOnPath

```
void increaseNewTrailsOnPath(
        in globaldefs::NamingAttributes_T pathName,
        in unsigned long numberOfTrail,
        in string servState,
        in string memberState,
        in globaldefs::NamingAttributesList_T aEndCtpList,
        in globaldefs::NamingAttributesList_T zEndCtpList,
        out string errorReason)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation increase a number of server trails on a LCAS enabled path.The new server trails will be created automatically by EMS. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pathName	The LCAS enabled path name.
❑numberOfTrail	Indicates how many server trails to be added.
❑servState	Available values: "Defined", "Allocated", "Implemented" This parameter specifies new trails service state.
❑memberState	Available values: "Idle", "Active" This parameter specifies new trails concatenated member state.
❑aEndCtpList, zEndCtpList	In case of not terminated path,aEndCtpList, zEndCtpList indicate the boundary CTPs for new trails; for terminated path, these two parameters are ignored.
➢ errorReason	Error string maybe contains the error reason.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of internal error of EMS. EXCPT_INVALID_INPUT - Raised in case of invalid input of NMS.

7.8.1.23. decreaseIdleTrailsOnPath

```
void decreaseIdleTrailsOnPath(
        in globaldefs::NamingAttributes_T pathName,
        in unsigned long numberOfTrail,
        out string errorReason)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation decrease a number of server trails on a LCAS enabled path. The trails being decreased will be deleted automatically by EMS. **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pathName	The LCAS enabled path name.
❑numberOfTrail	Indicates how many server trails to be decreased.
➢ errorReason	Error string maybe contains the error reason.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of internal error of EMS. EXCPT_INVALID_INPUT - Raised in case of invalid input of NMS.

7.8.1.24. createSNC

```
void createSNC (
    in subnetworkConnection::SNCCreateData_T createData,
    in subnetworkConnection::GradesOfImpact_T tolerableImpact,
    in EMSFreedomLevel_T emsFreedomLevel,
    out subnetworkConnection::SubnetworkConnection_T theSNC,
    out string errorReason)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
The NMS invokes the createSNC service to request the EMS to create an NC given the parameters passed in the method. Failure : This will throw an exception if it fails. No SNC object will be created on the EMS. Success : SNC is created in the EMS and the SNCState is set to the appropriate state in the parameter theSNC. The parameter theSNC will contain the attributes of the created SNC. This operation can create Alcatel RM path and trail. Currently, only point to point SNC's creation are supported. For path: the a/zEnd should be RM NAP. For trail: the a/zEnd should be RM CTP. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ createData	structure describing the subnetwork connection to be created. Currently, only userLabel, direction, staticProtectionLevel(UNPROTECTED), aEnd, zEnd, neTpInclusions,neTpSncExclusions,additionalCreationInfo are supported.neTpInclusions and neTpSncExclusions define snc constrain.

NeTpInclusions should only be sdh port and network element.
NeTpSncExclusions should only be network element.
The value of additionalCreationInfo is listed as the following table.

name	value	[mandatory/optional]
signalRate	lo2Mb,lo34Mb,ho140Mb,lo45 Mb,eth1GbTransp, mandatory eth10or100MbRateAdapteth1 GbRateAdapt, eth100MbTransp,mpls	mandatory
transportRate	au4virtN,tu3virtN,tu12virtN, loTu12,loTu3,hoAu4	Optional [when signal rate is eth1GbRateAdapt, eth10or100MbRateAdapt,m pls]
Concatenation Level	Integer	Optional [when snc is virtual concatenation]
lcasControl	enabled,disabled	Optional [when snc is virtual concatenation]
inclus[1-9]_ section	main,spare,service,common	Optional [when neTpInclusions is valid]
inclus[1-9]_ section	main,spare,service,,common	Optinal[when neTpSncExclusions is valid]
inclus[1-9]_ payload	[1-9][1-9],[1-9][1-9]/[1-3], [1-9][1-9]/[1-3]/[1-7].[1-3]	Optinal [when neTpInclusions is sdhport]
inclus[1-9]_ channel	Integer	Optinal [when neTpInclusions is valid and snc is virtual concatenation path]
exclus[1-9]_ channel	Integer	Optinal [when neTpSncExclusions is valid and snc is virtual concatenation path]

❑ tolerableImpact	the maximum tolerable impact allowed. Not supported currently.
❑ emsFreedomLevel	the maximum level of freedom allowed to the EMS to perform the creation. Not supported currently.
➢theSNC	The newly created SNC. It will have sncState and name set. The EMS selects the SNC names so that they are not reused (within a reasonable time frame) for different SNCs.
➢errorReason	Specifies the creation error(s) if any.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactically incorrect (e.g. field of createData is invalid). EXCPT_ENTITY_NOT_FOUND - Raised when fields of createData refer to an object that does not exist EXCPT_PROTECTION_EFFORT_NOT_MET - Raised if the NMS requests an SNC with a static protection level and protection effort that cannot be met by the EMS EXCPT_UNABLE_TO_COMPLY - Raised if the EMS is unable to find an appropriate route for the SNC EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS - Raised if the EMS does not support the routing constraints specified EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is not met EXCPT_OBJECT_IN_USE - Raised if the intended route is in conflict with an "exclusive" route of another SNC.

7.8.1.25. activateSNC

```
void activateSNC(
    in    globaldefs::NamingAttributes_T sncName,
    in    subnetworkConnection::GradesOfImpact_T tolerableImpact,
    in    EMSFreedomLevel_T emsFreedomLevel,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    out   subnetworkConnection::SubnetworkConnection_T theSNC,
    out   string errorReason)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This service is used to put an SNC into the ACTIVE state. Success: SNCState in the parameter theSNC is set to SNCS_ACTIVE. Failure: No exception thrown, except in the cases listed below. If the SNC or any of its network resources have changed as a result of this operation, then no exception can be thrown so that theSNC can be passed back to the NMS. The SNCState in resulting theSNC will be either SNCS_PARTIAL or SNCS_PENDING. The state will be SNCS_PARTIAL if not all cross-connects on MEs have been successfully set up. The state will be SNCS_PENDING if the EMS mode of operation prevents the activation of the SNC. The errorReason parameter is set accordingly. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ sncName	the name of the subnetwork connection to be deactivated.
❑ tolerableImpact	The maximum tolerable impact allowed. Not supported currently.
❑ emsFreedomLevel	The maximum level of freedom allowed to the EMS to perform the activation. Not supported currently.
❑➢tpsToModify	a list of TPs and parameters to apply,updated to provide the resulting parameters. Not supported currently.
➢ theSNC	The subnetwork connection after the operation.
➢ errorReason	Specifies the activation error(s) if any.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service;this is only allowed if the EMS does not support the PENDING state and if the PARTIAL state is unreachable EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. sncName does not refer to an SNC object or any field in tpsToModify is invalid) EXCPT_OBJECT_IN_USE - Raised if the SNC can not be activated because of CC,or TP conflicts or conflicts between the active route (with equal or higher priority) of this and other SNCs or when CC creation would involve a TP that has an existing fixed CC that does not match that required for the SNC. EXCPT_ENTITY_NOT_FOUND - Raised if sncName or tpsToModify reference an SNC/TP object that does not exist. EXCPT_TIMESLOT_IN_USE - (Obsolete - EXCPT_OBJECT_IN_USE should be used) Raised if the SNC can not be activated because of timeslot conflicts with other SNCs. EXCPT_UNABLE_TO_COMPLY - Raised if the SNC is in PENDING state and is in conflict with another ACTIVE or PARTIAL SNC. Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort".Raised if the SNC cannot be activated because the EMS cannot comply for a reason different from the ones above. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents any change to the SNC EXCPT_NOT_IN_VALID_STATE - Raised where the SNC would involve a CTP that is not connectable due to the state of the server TP or in the case of inverse multiplexing where the SNC would involve a CTP that is not connectable due to the state of the client TP. (Obsolete usage - EXCPT_OBJECT_IN_USE should be used instead for the case where an exception is raised if the SNC is in conflict with another active or partial SNC and can not be created.)

7.8.1.26. createAndActivateSNC

```
void createAndActivateSNC(
    in    subnetworkConnection::SNCCreateData_T       createData,
    in    subnetworkConnection::GradesOfImpact_T      tolerableImpact,
    in    EMSFreedomLevel_T                           emsFreedomLevel,
    inout subnetworkConnection::TPDataList_T          tpsToModify,
    out   subnetworkConnection::SubnetworkConnection_T theSNC,
    out   string                                      errorReason)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This operation provides a way to create and activate a subnetwork connection in one command. As with the createSNC() operation and the activateSNC() operation. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**

❑ createData	Structure describing the SNC to be created and activated. Refer to description of createSNC().
❑ tolerableImpact	The maximum tolerable impact allowed. Not supported currently.
❑ emsFreedomLevel	The maximum level of freedom allowed to the EMS to perform the creation and activation. Not supported currently.
❑➢ tpsToModify	A list of TPs and parameters to apply, updated to provide the resulting parameters. Not supported currently.
➢ theSNC	The resulting SNC. It will have sncState and name set. The EMS selects the SNC names so that they are not reused (within a reasonable time frame) for different SNCs.
➢ errorReason	Specifies the creation and/or activation error(s) if any.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. a field of createData is invalid or any field in tpsToModify is invalid). EXCPT_OBJECT_IN_USE - Raised if the SNC can not be created and activated because of CC or TP conflicts or conflicts between the active route (with equal or higher priority) of this and other SNCs or when CC creation would involve a TP that has an existing fixed CC that does not match that required for the SNC. EXCPT_ENTITY_NOT_FOUND - Raised when fields of createData or tpsToModify reference objects that do not exist. EXCPT_TIMESLOT_IN_USE - (Obsolete - EXCPT_OBJECT_IN_USE should be used) Raised if the SNC can not be created and activated because of timeslot conflicts with other SNCs EXCPT_PROTECTION_EFFORT_NOT_MET - Raised if the NMS requests an SNC with a static protection level and protection effort that cannot be met by the EMS EXCPT_UNABLE_TO_COMPLY - Raised if the EMS is unable to find a route for the SNC. Raised if the EMS can not meet the GradesOfImpact requested by the NMS. Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" .Raised if the SNC cannot be activated because the EMS cannot comply for a reason different from the ones above. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents creation of the SNC EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS - Raised if the EMS does not support the routing constraints specified EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is not met EXCPT_NOT_IN_VALID_STATE - Raised where the SNC would involve a CTP that is not connectable due to the state of the server TP or in the case of inverse multiplexing where the SNC would involve a CTP that is not connectable due to the state of the client TP. (Obsolete usage - EXCPT_OBJECT_IN_USE should be used instead for the case where an exception is raised if the SNC is in conflict with another active or partial SNC and can not be created.)

7.8.1.27. deactivateSNC

```
void deactivateSNC(
    in    globaldefs::NamingAttributes_T                sncName,
    in    subnetworkConnection::GradesOfImpact_T        tolerableImpact,
    in    EMSFreedomLevel_T                             emsFreedomLevel,
    inout subnetworkConnection::TPDataList_T            tpsToModify,
```

```
 out   subnetworkConnection::SubnetworkConnection_T        theSNC,
 out   string                                              errorReason)
raises (globaldefs::ProcessingFailureException );
```

General comment
Deactivate an SNC . Failure - No exception thrown (except for cases described below).If the SNC or any of its network resources have changed as a result of this operation, then no exception can be thrown so that theSNC can be passed back to the NMS.The SNCState will be either SNCS_PARTIAL if the command partially completed or SNCS_ACTIVE if no cross-connects were deleted. The errorReason will be detailed accordingly. Success - SNC is deactivated in the EMS and the SNCState is set to SNCS_PENDING in the out parameter theSNC. The parameter theSNC will contain the attributes of the deactivated SNC. The errorReason parameter may be set to an empty string. **Supported products:** SDHNBI1.0 SDHNBI1.1 SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	The name of the subnetwork connection to be deactivated.
❑ tolerableImpact	The maximum tolerable impact allowed. Indicates the amount of traffic disruption that the NMS user is willing to tolerate as a result of the deactivation request. Not supported currently.
❑ emsFreedomLevel	The maximum level of freedom allowed to the EMS to perform the deactivation. Not supported currently.
❑➢ tpsToModify	A list of TPs and parameters to apply, updated to provide the resulting parameters. Not supported currently.
➢ theSNC	The deactivated subnetwork connection.
➢ errorReason	Specifies the deactivation error(s) if any.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - - Raised when any input parameter is syntactical incorrectbr EXCPT_ENTITY_NOT_FOUND - Raised when sncName or tpsToModify reference an object that does not exist. EXCPT_UNABLE_TO_COMPLY - Raised if the SNC is fixed and can not be deactivated. Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" (such that the TPs will be left in an invalid state after the deactivate operation) EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents the deactivation of the SNC.

7.8.1.28. deleteSNC

```
void deleteSNC(
        in  globaldefs::NamingAttributes_T sncName,
        in  EMSFreedomLevel_T emsFreedomLevel)
        raises (globaldefs::ProcessingFailureException);
```

General comment

This service allows an NMS to request the deletion of a SubnetworkConnection on a specified subnetwork.
The SNC must not be in the active or partial state.
Failure - An exception will be thrown if the operation fails. The SNC object will not be deleted on the EMS. The errorReason field of the exception will contain the reason for the failure.
Success - The SNC object is deleted on the EMS.

Supported products: SDHNBI1.0
SDHNBI1.1
SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	The name of the subnetwork connection to be deleted.
❑ emsFreedomLevel	the maximum level of freedom allowed to the EMS to perform the deletion. Not supported currently.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. sncName does not refer to an SNC object) EXCPT_ENTITY_NOT_FOUND - Raised when sncName references an object that does not exist EXCPT_NOT_IN_VALID_STATE - Raised if the SNC is in the partial or active state

7.8.1.29. deactivateAndDeleteSNC

```
void deactivateAndDeleteSNC(
    in    globaldefs::NamingAttributes_T            sncName,
    in    subnetworkConnection::GradesOfImpact_T    tolerableImpact,
    in    EMSFreedomLevel_T                         emsFreedomLevel,
    inout subnetworkConnection::TPDataList_T        tpsToModify,
    out   subnetworkConnection::SubnetworkConnection_T theSNC,
    out   string                                    errorReason)
   raises (globaldefs::ProcessingFailureException);
```

General comment

This operation provides a way to deactivate and then delete a subnetwork connection in one operation. As with the deactivateSNC(),operation and the deleteSNC() operation.
It conceptually behaves like a call to deactivateSNC followed by a call to deleteSNC. All success/failure conditions that apply to the two base operations also apply to the combined operation.
If the SNC or any of its network resources have changed as a result of this operation, then no exception can be thrown so that theSNC can be passed back to the NMS. Therefore, the exceptions that apply to deleteSNC may not apply to the combined operation: if the deactivation changed the SNC but the deletion is rejected, no exception should be thrown and the resulting SNC should be provided in the out parameter theSNC.

Supported products: SDHNBI1.0
SDHNBI1.1
SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	The name of the subnetwork connection to be deactivated and deleted.
❑ tolerableImpact	Indicates the amount of traffic disruption that the NMS user is willing to tolerate as a result of the deactivation and deletion request. Not supported currently.
❑ emsFreedomLevel	The maximum level of freedom allowed to the EMS to perform the deactivation and deletion. Not supported currently.
❑➢ tpsToModify	A list of TPs and parameters to apply,updated to provide the resulting parameters. Not supported currently.
➢theSNC	The deactivated and deleted subnetwork connection.
➢errorReason	Specifies the deactivation and/or deletion error(s) if any.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. sncName does not refer to an SNC object or any field in tpsToModify is invalid) EXCPT_ENTITY_NOT_FOUND - Raised when sncName or tpsToModify reference an object that does not exist. EXCPT_UNABLE_TO_COMPLY - Raised if the SNC is fixed and can not be deactivated. Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" (such that the TPs will be left in an invalid state after the deactivate operation) EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents the deactivation of the SNC.

7.8.1.30. checkValidSNC

```
void checkValidSNC(
    in subnetworkConnection::SNCCreateData_T createData,
    in subnetworkConnection::TPDataList_T tpsToModify,
    in boolean considerResources,
    out boolean valid)
  raises (globaldefs::ProcessingFailureException);
```

General comment

The NMS uses this operation to check if it is possible to create and maybe activate an SNC as specified in the input parameters.
The test should check for the existence of hardware that will support the requested SNC. If the considerResources parameter is false, the check must be independent of the current specific resource usage in the subnetwork (as in createSNC). If theconsiderResources parameter is true, the check must consider thecurrent specific resource usage in the subnetwork (as in activateSNC);in that case, the rules of the EMS' mode of operation apply to the check (see SNC Management Modes of Operation).

Supported products: SDHNBI1.5
SDHNBI1.6

❑ Input / Output ➢	Comment
❑ createData	data about the potential SNC.It is the same as parameter createData is operation "createSNC".
❑ tpsToModify	a list of TPs and parameters that would be applied to the potential SNC.
❑ considerResources	indicates whether or not resource allocation must be considered.
➢ valid	indicates if this is a valid SNC.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service; the EMS may not support this operation at all or may not support the specified value for considerResources EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_OBJECT_IN_USE - Raised if CC creation for the SNC would involve a TP that has an existing fixed CC that does not match that required for the SNC EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. a field of createData is invalid or any field in tpsToModify is invalid). EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents checking the validity of the SNC EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS - Raised if the EMS does not support the routing constraints specified EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is not met EXCPT_UNABLE_TO_COMPLY - Raised if the EMS is unable to find a route for the SNC. Raised when the EMS is unable to execute the request because at least one of the parameters although valid could not be set and that parameter is identified as "not best effort" in the Layered Transmission Parameters document or in the AdditionalInformation Usage document. Raised if the SNC could not be activated because the EMS cannot comply for a reason different from the ones above. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY EXCPT_OBJECT_IN_USE - Raised if CC creation for the SNC would involve a TP that has an existing fixed CC that does not match that required for the SNC

7.8.1.31. getAllFixedSubnetworkConnections

```
void getAllFixedSubnetworkConnections(
        in globaldefs::NamingAttributes_T subnetName,
        in transmissionParameters::LayerRateList_T connectionRateList,
        in unsigned long how_many,
        out subnetworkConnection::SubnetworkConnectionList_T sncList,
        out subnetworkConnection::SNCIterator_I sncIt)
      raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllSubnetworkConnections(),but instead returns only fixed SNCs object structures. See Subnetwork Connection Types for an explanation of fixed SNCs. An SNC is identified as fixed using additional information. See Additional Information Usage for detail on additional information for SNCs, cross connections and TPs. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** SDHNBI1.5 SDHNBI1.5 SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ connectionRateList	List of rates of the SNCs to be reported.If an empty list is specified, then all SNCs of all rates are to be reported.
❑ how_many	Maximum number of SNCs to be reported in the first batch.
➢ sncList	First batch of SNCs.
➢ sncIt	Iterator to retrieve the remaining SNCs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when subnetName does not reference an multiLayerSubnetwork object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when subnetName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.32. getAllFixedSubnetworkConnectionNames

```
void getAllFixedSubnetworkConnectionNames(
    in globaldefs::NamingAttributes_T subnetName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllFixedSubnetworkConnections(), butinstead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. **Supported products:** EMLNBI1.5 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ connectionRateList	List of rates of the SNC names to be reported.If an empty list is specified, then all SNC names of all rates are to be reported.
❑ how_many	Maximum number of SNC names to be reported in the first

	batch..
➢ nameList	First batch of SNC names.
➢ nameIt	Iterator to retrieve the remaining SNC names.
raises	**ExceptionType**
ProcessingFailureException	As for getAllSubnetworkConnections().

7.8.1.33. getAllFixedSubnetworkConnectionNamesWithTP

void getAllFixedSubnetworkConnectionsWithTP (
in globaldefs::NamingAttributes_T tpName,
in transmissionParameters::LayerRateList_T connectionRateList,
in unsigned long how_many,
out subnetworkConnection::SubnetworkConnectionList_T sncList,
out subnetworkConnection::SNCIterator_I sncIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllSubnetworkConnectionsWithTP(), but instead returns only fixed SNCs object structures. A fixed SNC is an SNC all of whose cross connections are fixed, i.e., cannot be deleted by an NMS.An SNC is identified as fixed using additional information. See SD1-1 Additional Information Usage for detail on additional information for SNCs, cross connections and TPs.In order to allow the NMS to deal with a large number of objects,this operation uses an iterator. See <a href=../../../supportingDocumentation/SD1-15_iterators.pdf >SD1-15 iterator overview for information on how iterators are used in this interface. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Termination point for whichto report SNCs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See SD1-25 Object Naming for further detail on FTP naming.
❑ connectionRateList	The list of rates of the SNCs to be reported. If an empty list is specified, then all SNCs of all rates are to be reported.
❑ how_many	Maximum number of SNCs to report in the first batch.
➢ sncList	First batch of SNCs. An SNC is only reported if it respects both the tpName and connectionRateList filters.
➢ sncIt	Iterator to retrieve the remaining SNCs.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if tpName does not reference a terminationPoint object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when tpName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request,and cannot determine the reason it could not comply, it may raise this exception.

7.8.1.34. getAllFixedSubnetworkConnectionsWithTP

```
void getAllFixedSubnetworkConnectionNamesWithTP (
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRateList_T connectionRateList,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllFixedSubnetworkConnectionsWithTP(), butinstead of returning the entire object structures, this operation returns their names. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	Termination point for whichto report SNCs. The termination point name must be explicit (a generic endpoint specification may not be used in this case). See SD1-25 Object Naming for further detail on FTP naming.
❑ connectionRateList	The list of rates of the SNCs to be reported. If an empty list is specified, then all SNCs of all rates are to be reported.
❑ how_many	Maximum number of SNCs to report in the first batch.
➢ sncList	First batch of SNCs. An SNC is only reported if it respects both the tpName and connectionRateList filters.
➢ sncIt	Iterator to retrieve the remaining SNCs.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if tpName does not reference a terminationPoint object or connectionRateList contains undefined values EXCPT_ENTITY_NOT_FOUND - Raised when tpName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request,and cannot determine the reason it could not comply, it may raise this exception.

7.8.1.35. modifySNC

```
void modifySNC(
    in globaldefs::NamingAttributes_T sncName,
    in string routeId,
    in subnetworkConnection::SNCModifyData_T SNCModifyData,
    in subnetworkConnection::GradesOfImpact_T tolerableImpact,
    in subnetworkConnection::ProtectionEffort_T tolerableImpactEffort,
    in EMSFreedomLevel_T emsFreedomLevel,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    out subnetworkConnection::SubnetworkConnection_T newSNC,
    out string errorReason)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
The NMS invokes the modifySNC to perform the change of SNC's attributes. Currently, only the "sncRmProtectionType" attribute of RM path/trail change is supported. **Supported products:** SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ sncName	the name of the subnetwork connection to be modified.
❑ routeId	The id of the route to be modifed. Empty string,indicates that the "intended" route is to be modified. Not supported currently.
❑ SNCModifyData	Structure describing the new/modified subnetwork connection. Currently, only parameter "sncRmProtectionType" in additionalCreationInfo is supported. sncRmProtectionType: "None"|"In Rings"|"SNCP"|"D&C SNCP"
❑ tolerableImpact	the maximum tolerable impact allowed. Not supported currently.
❑ tolerableImpactEffort	qualifies the conditions under which an SNC modification may be performed is a qualification of the requirement that the tolerableImpact as specified, is met. Not supported currently.
❑emsFreedomLevel	the maximum level of freedom allowed to the EMS to perform the creation. Not supported currently.
❑➢tpsToModify	a list of TPs and parameters to apply, updated to provide the resulting parameters. Not supported currently.
➢newSNC	The modified SNC. It will have sncState and name set. The EMS selects the SNC names so that they are not reused (within a reasonable time frame) for different SNCs.

➢errorReason	Specifies the modification error(s) if any.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when a field of an input parameter is invalid (e.g. sncName does not refer to an SNC object, or any field in tpsToModify is invalid) EXCPT_OBJECT_IN_USE - Raised if the SNC can not be created or activated because of CC or TP conflicts (e.g. because of timeslot conflicts) or conflicts between the active route (with equal or higher priority) of this and other SNCs or when CC creation would involve a TP that has an existing fixed CC that does not match that required for the SNC EXCPT_ENTITY_NOT_FOUND - Raised when fields of input parameters (e.g. SNCModifyData or tpsToModify) refer to an object that does not exist EXCPT_PROTECTION_EFFORT_NOT_MET - Raised if the NMS requests an SNC with a static protection level and protection effort that cannot be met by the EMS EXCPT_UNABLE_TO_COMPLY - Raised if the EMS is unable to find a route for the SNC. Raised if the EMS can not meet the GradesOfImpact requested by the NMS. Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set Raised if the SNC cannot be activated because the EMS cannot comply for a reason different from the ones above. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS - Raised if the EMS does not support the routing constraints specified EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is not met EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents any change to the SNC EXCPT_NOT_IN_VALID_STATE - Raised where the SNC would involve a CTP that is not connectable due to the state of the server TP or in the case of inverse multiplexing where the SNC would involve a CTP that is not connectable due to the state of the client TP

7.8.1.36. getBackupRoutes

```
void getBackupRoutes(
    in globaldefs::NamingAttributes_T sncName,
    in string routeId,
    in boolean includeHigherOrderCCs,
    inout globaldefs::NVSList_T additionalInfo,
    out subnetworkConnection::RouteList_T routeList)
  raises(globaldefs::ProcessingFailureException);
```

General comment
This service returns the requested route for the SNC whose name is specified as a parameter. If the input route is not specified (empty string), the operation replies all the routes of the SNC, intended and backup ones. The rest of the behavior is same as getRoute(). **Supported products:** SDHNBI1.2 SDHNBI1.3 SDHNBI1.4 SDHNBI1.5 SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	The name of the subnetwork connection.
❑ routeId	The id of the route. Not supported currently.
❑ includeHigherOrderCCs	Specifies whether the higher order CCs of other SNCs used to carry the queried SNC route have to be included in addition to the CCs of the queried SNC route.(This parameter is not supported currently.Method always return the same layer CCs.) Not supported currently.
❑➢additionalInfo	To allow the communication of additional information which is not explicitly modelled
➢routeList	The route(s) of the SNC.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when a field of an input parameter is invalid (e.g. sncName does not reference a subnetworkConnection object) EXCPT_ENTITY_NOT_FOUND - Raised when sncName and/or the routeId reference an object that does not exist

7.8.1.37. addRoute

```
void addRoute(
    in globaldefs::NamingAttributes_T sncName,
    in subnetworkConnection::RouteCreateData_T createRoute,
    in subnetworkConnection::GradesOfImpact_T tolerableImpact,
    in EMSFreedomLevel_T emsFreedomLevel,
    out subnetworkConnection::RouteDescriptor_T theRoute ,
    out string errorReason)
  raises(globaldefs::ProcessingFailureException);
```

General comment
This operation creates a new route (e.g. for restoration purposes) and associates it to the given SNC. The route is created in locked state. A route belongs to only one SNC. However XCs/TPs can be shared by routes of different SNCs. A route applies to XCs at the same layer of the SNC. The route id must be an unique identifier within the SNC name, with format up to EMS. It is possible to specify if the creating route is the INTENDED route or not; if intended, then the former intended route is updated to backup route. Only one intended route can be associated to a given SNC at a time. It is possible to specify if the creating route is EXCLUSIVE or not; if EXCLUSIVE, then the EMS must find a route that does not conflict or share CCs with any other existing SNC route, in any administrative state . Once an EXCLUSIVE route has been created by EMS, any further creation operation which conflicts with the exclusive route shall be refused. A route implicitly inherits from its SNC the following attributes: the endpoints owner direction rerouteAllowed networkRouted rate the sncType the staticProtectionLevel - so protectionEffort is considered EFFORT_SAME, i.e. if EMS can not create a route with same static protection level as the SNC, the operation is refused. **Supported products:** SDHNBI1.5 SDHNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	The name of the subnetwork connection.
❑ createRoute	structure describing the route to be created.
❑ tolerableImpact	the maximum tolerable impact allowed.
❑ emsFreedomLevel	the maximum level of freedom allowed to the EMS to perform the creation.
➢ theRoute	the created route.
➢ errorReason	Specifies the creation error(s) if any.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is syntactical incorrect (e.g. sncName does not refer to an SNC object, or any field in createRoute is invalid EXCPT_ENTITY_NOT_FOUND - Raised when sncName or fields of createRoute refer to an object that does not exist EXCPT_PROTECTION_EFFORT_NOT_MET - Raised if the NMS requests a route with a static protection level (inherited from SNC) that cannot be met by the EMS EXCPT_OBJECT_IN_USE - Raised if the route is in conflict with an "exclusive" (even locked) route of another SNC EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents creation of the route EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS - Raised if the EMS does not support the routing constraints specified EXCPT_UNABLE_TO_COMPLY - Raised if the EMS can not meet the GradesOfImpact requested by the NMS. Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" in the Additional Information Usage document. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY

7.8.1.38. removeRoute

```
void removeRoute(
    in globaldefs::NamingAttributes_T sncName,
    in string routeId,
    in EMSFreedomLevel_T emsFreedomLevel,
    inout globaldefs::NVSList_T additionalInfo)
  raises(globaldefs::ProcessingFailureException);
```

General comment
This service allows an NMS to request the deletion of a route of given SubnetworkConnection on a specified subnetwork. The addressed route must not be in the unlocked state, and must not be the intended route. Of course it is possible to delete a locked backup route which is "in use" by other SNC route, because this operation has no side effect on routes of any other SNCs, even if sharing XCs/TPs. Failure - An exception will be thrown if the operation fails. The route will not be deleted on the EMS. The errorReason field of the exception will contain the reason for the failure.</p> Success - The route is deleted on the EMS. **Supported products:** SDHNBI1.5 SDHNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	the name of the subnetwork connection.
❑ routeId	The id of the route.
❑ emsFreedomLevel	the maximum level of freedom allowed to the EMS to perform the deletion.
❑➢ additionalInfo	to allow the communication of additional information which is not explicitly modelled.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when sncName does not refer to an SNC objec EXCPT_ENTITY_NOT_FOUND - Raised when sncName or routeId reference an object that does not exist EXCPT_NOT_IN_VALID_STATE - Raised if the route is in the unlocked state, or if the route is the intended one EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents the deletion of the route EXCPT_UNABLE_TO_COMPLY - Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" in the Additional Information Usage document. If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY

7.8.1.39. getIntendedRoute

```
void getIntendedRoute(
    in globaldefs::NamingAttributes_T sncName,
    in boolean includeHigherOrderCCs,
    inout globaldefs::NVSList_T additionalInfo,
    out subnetworkConnection::Route_T route)
  raises(globaldefs::ProcessingFailureException);
```

General comment

This service returns the intended route for the SNC whose name is specified as a parameter. The behavior is essentially the same as getRoute().

Supported products: SDHNBI1.2
SDHNBI1.3
SDHNBI1.4
SDHNBI1.5
SDHNBI1.6, PKTNBI1.6

❑ Input / Output ➢	Comment
❑ sncName	The name of the subnetwork connection.
❑ includeHigherOrderCCs	Specifies whether the higher order CCs of other SNCs used to carry the queried SNC route have to be included in addition to the CCs of the queried SNC route.(This parameter is not supported currently.Method always return the same layer CCs.)
❑ ➢additionalInfo	To allow the communication of additional information which is not explicitly modelled
➢ route	The route of the SNC.

raises	ExceptionType

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when sncName does not reference a subnetworkConnection object EXCPT_ENTITY_NOT_FOUND - Raised when sncName and/or the routeId reference an object that does not exist EXCPT_UNABLE_TO_COMPLY - Raised when the EMS is unable to execute the request because at least one of the parameters although valid can not be set and that parameter is identified as "not best effort" If the EMS cannot determine the reason it could not comply, it is also allowed to throw EXCPT_UNABLE_TO_COMPLY

7.8.1.40. swapSNC

```
void swapSNC(
    in globaldefs::NamingAttributes_T nameOfSNCtoBeDeactivated,
    in globaldefs::NamingAttributes_T nameOfSNCtoBeActivated,
    in EMSFreedomLevel_T emsFreedomLevel,
    in subnetworkConnection::GradesOfImpact_T tolerableImpact,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    out subnetworkConnection::SNCState_T stateOfActivatedSNC,
    out string errorReason)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
The swapSNC method will deactivate an identified active SNC (state changes to pending) and activate an identified pending SNC. The pending SNC may have been created by using a createSNC() or a createModifiedSNC(). **Supported products:** SDHNBI1.5 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ nameOfSNCtoBeDeactivated	the name of the subnetwork connection to be deactivated.
❑ nameOfSNCtoBeActivated	the name of the subnetwork connection to be activated.
❑ emsFreedomLevel	the maximum level of freedom allowed to the EMS to perform the activation.
❑ tolerableImpact	the maximum tolerable impact allowed.
❑ ➢ tpsToModify	a list of TPs and parameters to apply, updated to provide the resulting parameters.
➢ stateOfActivatedSNC	The state of the activated subnetwork connection after the operation.
➢ errorReason	Specifies the modification error(s) if any
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managedElement object EXCPT_ENTITY_NOT_FOUND - Raised when meName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.41. getAllEdgePoints

```
void getAllEdgePoints(
    in globaldefs::NamingAttributes_T subnetName,
    in transmissionParameters::LayerRateList_T tpLayerRateList,
    in transmissionParameters::LayerRateList_T connectionLayerRateList,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request a list of the edge termination points (PTPs/FTPs) for the specified subnetwork, at one or more of the NMS-specified layers, and that are capable of containing CTPs that can be connected at one or more of the NMS-specifiedconnection layer rates. This operation considers the capability/flexibility of the TPs, not their current states. If the MLSN represents a MLRA, the operation is rejected and an exception raised. **Supported products:** SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ tpLayerRateList	List of TP layer rates for which the edge points are to be fetched. An edge point must contain at least one of the layer rates specified to be reported. If the list is empty then edge points of all rates are returned.
❑ connectionLayerRateList	List of connection layer rates for which the edge points are to be fetched. An edge point must support connections for at least one of the layer rates specified to be reported. If the list is empty then edge points for all connection rates are returned.
❑ how_many	Maximum number of edge points to return in the first batch.
➢tpList	First batch of edge points.
➢ tpIt	Iterator to retrieve the remaining edge points.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managedElement object EXCPT_ENTITY_NOT_FOUND - Raised when meName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.8.1.42. getAllEdgePointNames

```
void getAllEdgePointNames(
    in globaldefs::NamingAttributes_T subnetName,
    in transmissionParameters::LayerRateList_T layerRateList,
    in transmissionParameters::LayerRateList_T connectionLayerRateList,
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
```

raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllEdgePoints(), but instead of returning the entire object structures,this operation returns their names.. **Supported products:** SDHNBI1.6 PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ subnetName	Name of the subnetwork.
❑ layerRateList	List of TP layer rates for which the edge points are to be fetched. An edge point must contain at least one of the layer rates specified to be reported. If the list is empty then edge points of all rates are returned.
❑ connectionLayerRateList	List of connection layer rates for which the edge points are to be fetched. An edge point must support connections for at least one of the layer rates specified to be reported. If the list is empty then edge points for all connection rates are returned.
❑ how_many	Maximum number of edge points to return in the first batch.
➢nameList	First batch of edge points.
➢nameIt	Iterator to retrieve the remaining edge points.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managedElement object EXCPT_ENTITY_NOT_FOUND - Raised when meName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.9. ProtectionMgr_I Interface

The ProtectionMgr_I is used to gain access to protection groups and their operations.

7.9.1. Operations

7.9.1.1. getAllProtectionGroups

```
void getAllProtectionGroups(
        in globaldefs::NamingAttributes_T meName,
        in unsigned long how_many,
        out ProtectionGroupList_T pgList,
        out ProtectionGroupIterator_I pgpIt)
        raises(globaldefs::ProcessingFailureException);
```

General comment

This operation is used by the client to discover all the protection groups currently in operation for the managed element. For current version, protection group returned is sdhMSProtectionGroupR1, sdhMSSPRingProtectionGroup,linearOMSProtectionGroup,connectionProtectionGroupR1. ConnectionProtectionGroupR1 won't be returned unless the value of "MAP_SNCP_TO_OSNCP_1_PLUS_1" defined in param.cfg is True. **Supported products:** EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ meName	the name of the managed element for which the request is made.
❑ how_many	Maximum number of protection groups to report in the first batch.
➢ pgList	First batch of protection groups.
➢ pgIt	Iterator used to access the remaining PGs, if any.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managedElement object EXCPT_ENTITY_NOT_FOUND - Raised when meName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.9.1.2. getAllEProtectionGroups

```
void getAllEProtectionGroups(
    in globaldefs::NamingAttributes_T meName,
    in unsigned long how_many,
    out EProtectionGroupList_T epgpList,
    out EProtectionGroupIterator_I epgpIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation is used by the client to discover all the equipment protection groups currently in operation for the managed element. **Supported products:** EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ meName	The name of the managed element for which the request is made.
❑ how_many	Maximum number of equipment protection groups to report in the first batch.
➢ epgpList	First batch of equipment protection groups.
➢ epgpIt	Iterator used to access the remaining EPGs, if any.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when an EMS is unable to support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managedElement object EXCPT_ENTITY_NOT_FOUND - Raised when meName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.9.1.3. getProtectionGroup

```
void getProtectionGroup(
    in  globaldefs::NamingAttributes_T pgName,
    out protection::ProtectionGroup_T  protectionGroup)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This operation is used to get the current status of a protectionGroup. This service is needed so that even if a filter is established, the NMS can query the status of a protection group. **Supported products:** EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pgName	the name of the protection Group that the client is interested in.
➢ protectionGroup	the returned protection group.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pgName does not reference a protection group EXCPT_ENTITY_NOT_FOUND - Raised when pgName references a PG object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.9.1.4. getEProtectionGroup

```
void getEProtectionGroup(
    in  globaldefs::NamingAttributes_T ePGPname,
    out protection::EProtectionGroup_T eProtectionGroup)
  raises (globaldefs::ProcessingFailureException);
```

General comment

This operation is used to get the current status of a Equipment Protection Group.

Supported products: EMLNBI1.0
EMLNBI1.1
EMLNBI1.2
EMLNBI1.3
EMLNBI1.4
EMLNBI1.5
EMLNBI1.6

❑ Input / Output ➢	Comment
❑ ePGPname	The name of the equipment protection group that the client is interested in.
➢ eProtectionGroup	The returned equipment protection group.

raises	ExceptionType
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_NOT_IMPLEMENTED - Raised when an EMS is unable to support this service EXCPT_INVALID_INPUT - Raised when ePGPName does not reference an equipment protection group EXCPT_ENTITY_NOT_FOUND - Raised when ePGPName references an object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.9.1.5. retrieveSwitchData

```
void retrieveSwitchData(
    in globaldefs::NamingAttributes_T reliableSinkCtpOrGroupName,
    out protection::SwitchDataList_T switchData)
  raises(globaldefs::ProcessingFailureException);
```

General comment

This service is used by the NMS to get the latest switch status on a SNC or a MSP group.
It should be noted that although the term MSP was chosen as the original specific protection scheme to which the related behaviour applied was Multiplex Section Protection, the label is now more generally applied to any 1+1 or 1:N Trail protection scheme.
When used on the reliable CTP/FTP of an SNC, a single struct is returned and the group name is NULL. The switchToTP indicates the presently active source of the traffic to the protectedTP.
When used on a 1+1 MSP, a single struct is provided with the relevant data.
When used on a 1:N MSP, a struct per worker TP is presented with protectedTP being the worker TP Name and the switchToTP identifying the present source of the traffic.
When used on a 2F BLSR, two structs are returned, one per TP.
In a stable state, the protectedTP and the switchToTP are the same TP. In a switched state, the switchToTP is the same for both protectedTPs.
When used on a 4F BLSR, two structs are returned, each one identifying a span with the protectedTP being the worker TP and the switchToTP identifying the present source of the ring traffic for that span.

Supported products: EMLNBI1.2
EMLNBI1.3
EMLNBI1.4
EMLNBI1.5
EMLNBI1.6

❑ Input / Output ➢	Comment
❑ reliableSinkCtpOrGroupName	This is the CTP/FTP that is the output of a service selector in case of the SNC or the group name for which the switch data is being requested. The termination point name must be explicit (a generic endpoint specification may not be used in this case).

➢ switchData	The current protection switch status of the CTP, FTP or PG provided.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised when an EMS is unable to support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when reliableSinkCtpOrGroupName does not reference a reliable CTP nor FTP nor PG object EXCPT_ENTITY_NOT_FOUND - Raised when reliableSinkCtpOrGroupName references a CTP or FTP or PG or object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.9.1.6. retrieveESwitchData

void retrieveESwitchData(
in globaldefs::NamingAttributes_T ePGPName,
out protection::ESwitchDataList_T eSwitchDataList)
raises(globaldefs::ProcessingFailureException);

General comment	
This service is used by the NMS to get the latest switch status on an equipment protection group. For a retrieval of a revertive M: N group, N ESwitchData_T are returned as a result of retrieveESwitchData (one for each worker equipment instance). For a retrieval of a non-revertive M: N group, N ESwitchData_T are returned as a result of retrieveESwitchData (one for each active equipment instance). **Supported products:** EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ ePGPName	This is the equipment protection group name for which the switch data is being requested.
➢ eSwitchDataList	The current protection switch status of the equipment protection group provided.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised when an EMS is unable to support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when ePGPName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when ePGPName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.9.1.7. performProtectionCommand

void performProtectionCommand(
in ProtectionCommand_T protectionCommand,
in globaldefs::NamingAttributes_T reliableSinkCtpOrGroupName,
in globaldefs::NamingAttributes_T fromTp,

```
 in  globaldefs::NamingAttributes_T toTp,
 out protection::SwitchData_T switchData)
raises(globaldefs::ProcessingFailureException);
```

General comment	
This service is used to execute a protection switch. The protection switch may be performed via a protection switch command, on a protection group or on a CTP/FTP involved in an SNCP. The NMS requests the EMS to move the traffic received from the fromTP to the toTP. The same command is used to clear all existing commands. For 4-fiber SONET/SDH SPRINGS, the span switch is performed on the PGT_MSP_1_FOR_N protection group, and the ring switch on the PGT_4_FIBER_BLSR group. It should be noted that although the term MSP was chosen as the original specific protection scheme to which the related behaviour applied was Multiplex Section Protection, the label is now more generally applied to any 1+1 or 1:N Trail protection scheme. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ protectionCommand	The command to be performed.
❑reliableSinkCtpOrGroupName	This is the CTP/FTP that is the output of a service selector in case of the SNC or the group name for which the switch data is being requested. The termination point name must be explicit
❑ fromTp	The present source of the traffic. The termination point name must be explicit (a generic endpoint specification may not be used in this case)
❑ toTp	The requested source of the traffic after the command. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
➢ switchData	The protection switch status of the toTp provided after the execution of the command.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised when an EMS is unable to support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when reliableSinkCtpOrGroupName, fromTp, or toTp do not reference objects of the correct type. Raised when the reliableSinkCtpOrGroupName, fromTp, or toTp reference objects of the correct type, but that are invalid in the context of this operation EXCPT_ENTITY_NOT_FOUND - Raised when reliableSinkCtpOrGroupName references a CTP, FTP or PG object that does not exist, or when fromTp or toTp references a CTP/FTP object that does not exist EXCPT_UNABLE_TO_COMPLY - Raised if the EMS is unable to perform the operation EXCPT_NE_COMM_LOSS - Raised when communications to the managed element is lost For current version,EMLNBI only support sncp operation.Parameter protectionCommand may be PC_CLEAR, PC_LOCKOUT,PC_FORCED_SWITCH or PC_MANUAL_SWITCH.Different type of ne support different protectionCommand.

7.9.1.8. getContainingPGNames

```
void getContainingPGNames(
    in globaldefs::NamingAttributes_T pTPName,
```

out globaldefs::NamingAttributesList_T pgNameList)
raises (globaldefs::ProcessingFailureException);

General comment	
This service returns the names of the Protection Groups containing the given PhysicalTerminationPoint. **Supported products:** EMLNBI1.3 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pTPName	The name of the PhysicalTerminationPoint.
➢ pgNameList	The names of the containing ProtectionGroups.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when reliableSinkCtpOrGroupName is syntactically incorrect EXCPT_ENTITY_NOT_FOUND - Raised when reliableSinkCtpOrGroupName references a object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to some ME is lost

7.9.1.9. setProtectionGroupParameter

void setProtectionGroupParameter(
in globaldefs::NamingAttributes_T reliableSinkCtpOrGroupName,
in globaldefs::NVSList_T parameters)
raises (globaldefs::ProcessingFailureException);

General comment	
This service provides changing the parameter of protection Group. **Supported products:** EMLNBI1.3 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ reliableSinkCtpOrGroupName	The name of reliable sink ctp or protection group.
❑ parameters	The parameters of protection group which should be modified. Now parameters only support the attribute list as the following: "revertive" -- "enable|disable" "wtr"--"[d]+"
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when reliableSinkCtpOrGroupName is syntactically incorrect EXCPT_ENTITY_NOT_FOUND - Raised when reliableSinkCtpOrGroupName references a object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to some ME is lost

7.10. Session_I Interface

The Session_I interface provides capabilities to manage the client-server connection.
Its main purpose is to enable either a client or server to detect the loss of communication with the associated party.
For a single communication session between an NMS and an EMS, there are two Session_I objects. One is maintained on the NMS; the other one is maintained on the EMS. The Session_I object maintained on the EMS is actually an EmsSession_I, while the Session_I object maintained on the NMS is actually an NmsSession_I (both inherit from Session_I).
Each Session_I object is responsible to "ping" the other Session_I object periodically to detect communication failures. Exactly when this is done is up to the implementation.
When a Session_I object detects a communication failure, or when the endSession operation is called on it, all resources allocated with that communication session must be freed and the Session_I object must be deleted.

7.10.1. Operations

7.10.1.1. ping

void ping();

Allows for the detection of loss of communication. It is implementation specific to differenciate intermittent problems from loss of connection.

Supported products: EMLNBI1.0, SDHNBI1.0
EMLNBI1.1, SDHNBI1.1
EMLNBI1.2, SDHNBI1.2
EMLNBI1.3, SDHNBI1.3
EMLNBI1.4, SDHNBI1.4
EMLNBI1.5, SDHNBI1.5
EMLNBI1.6, SDHNBI1.6, PKTNBI1.6

7.10.1.2. endSession

oneway void endSession();

Allows for a controlled disconnect between parties. All resources allocated for the session are deleted by operation.

Supported products: EMLNBI1.0, SDHNBI1.0
EMLNBI1.1, SDHNBI1.1
EMLNBI1.2, SDHNBI1.2
EMLNBI1.3, SDHNBI1.3
EMLNBI1.4, SDHNBI1.4
EMLNBI1.5, SDHNBI1.5
EMLNBI1.6, SDHNBI1.6

7.11. PerformanceManagementMgr_I Interface

The PerformanceManagementMgr_I is used to gain access to operations which deal with performance Monitoring.
A handle to an instance of this interface is gained via the emsSession::EmsSession_I::getManager() operation in managerInterface when the managerName "PerformanceManagement" is used.

7.11.1. Operations

7.11.1.1. disablePMData

```
void disablePMData(
        in PMTPSelectList_T pmTPSelectList,
        out PMTPSelectList_T failedTPSelectList)
```

raises(globaldefs::ProcessingFailureException);

General comment	
This operation instructs the EMS to turn off PM data collection for a list of measurement points. Within the request for each measurement point, one may specify the granularity (15min, 24h, NA, etc.) and location (nearEnd and/or farEnd and/or bidirectional) for the PM parameters that are to be deactivated. Disabling is done on a best-effort basis. If PM could not be disabled for a subset of cases in the PMTPSelectList a list identifying this subset is returned. PM collection stops immediately, i.e., before the completion of the current 15 minute or 24 hour monitoring period. This may lead to incomplete collection periods. If PM collection was never started for this TP, the operation is considered successful. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pmTPSelectList	This struct contains the relevant data for the disablePMData request. Only the pmTPSelectList[0].name is required. globaldefs::NamingAttributes_T pmTPSelectList[0].name: name of NetworkElement/TerminationPoint/PerfomancePoint. NetworkElement: The selection applies to all perfomance points contained within the ManagedElement. TerminationPoint: The selection applies to all perfomance points contained within the TerminationPoint. PerfomancePoint: perfomance points for disable.
➢ failedTPSelectList	List of points which were not completely disabled. An empty list indicates that the total request was successful.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if pmTPSelectList is empty or contains invalid data EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.11.1.2. enablePMData

void enablePMData(
in PMTPSelectList_T pmTPSelectList,
out PMTPSelectList_T failedTPSelectList)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation instructs the EMS to enable (turn on) PM data collection for a list of measurement points. Within the request for each measurement point, one may specify the granularity (15min, 24h, NA, etc.) and location (nearEnd and/or farEnd and/or bidirectional) for the PM parameters that are to be activated. Enabling is done on best-effort basis. If PM could not be enabled for a subset of cases in the pmTPSelectList, a list identifying this subset is returned. PM data collection starts immediately, i.e., before the completion of the current 15 minute or 24 hour monitoring period. This may lead to incomplete collection periods. If PM collection was already on for a TP, the operation is considered successful. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pmTPSelectList	This struct contains the relevant data for the enablePMData request. This must not be empty. globaldefs::NamingAttributes_T pmTPSelectList[0].name:

	name of TerminationPoint. Supported tp type is: rsCTPBidirectional rsCTPSink gdcCTPBidirectional gdcCTPSink ochgdcCTPBidirectional ochgdcCTPSink rsTTPSink rsTTPBidirectional ochCTPSink ochCTPBidirectional otuCTPSink otuCTPBidirectional gMAUCTPBidirectional gMAUCTPSink rsCTPBidirectional rsCTPSink gdcCTPBidirectional gdcCTPSink ochgdcCTPBidirectional ochgdcCTPSink odu2CTPSource odu2CTPSink odu1CTPSource odu1CTPSink odu1CTPBidirectional odu2CTPBidirectional ochCTPSink odu2TTPSink odu1TTPSink PMTPSelectList_T[].PMLocationList_T[]: Only "near" is supported, the list must contains one of the parameters: PML_NEAR_END_Rx PML_NEAR_END_Tx PMTPSelectList_T[].LayerRateList_T[]: Null. PMTPSelectList_T[].GranularityList_T[]:"15min"/"1h"/"24h"/"NA"
➢ failedTPSelectList	List of points which were not completely enabled. An empty list indicates that the total request was successful.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if pmTPSelectList is empty or contains invalid data EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_CAPACITY_EXCEEDED - Raised when the maximum number of simultaneously enabled monitoring points is exceeded

7.11.1.3. disableUnsolicitedPMDataReporting

void disableUnsolicitedPMDataReporting(
)
raises(globaldefs::ProcessingFailureException);

General comment

This operation instructs the EMS to disable (turn off) unsolicited PM data reporting. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.11.1.4. enableUnsolicitedPMDataReporting

void enableUnsolicitedPMDataReporting(
)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation instructs the EMS to enable (turn on) unsolicited PM data reporting (every 2 hours in default).. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.11.1.5. clearPMData

void clearPMData(
in PMTPSelectList_T pmTPSelectList,
out PMTPSelectList_T failedTPSelectList)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation instructs the EMS to clear (reset) the PM registers for a list of measurement points. Within the request for each measurement point, one may specify the granularity (15min, 24h, NA, etc.) and location (nearEnd and/or farEnd and/or bidirectional) for the PM registers that are to be reset. Clearing PM gauge minimum, maximum, and average registers means to reset them to the current measurement; for the actual gauge measurement, it has no effect. This is not considered a failure case. Clearing PM registers is done on best-effort basis. If registers could not be completely cleared for a subset of cases in the pmTPSelectList, a list identifying this subset is returned. **Supported products:** EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pmTPSelectList	This struct contains the relevant data for the clearPMData

	request. It must not be empty.
➢ failedTPSelectList	List of points for which PM registers were not completely cleared. An empty list indicates that the total request was successful.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS is unable to support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if pmTPSelectList is empty or contains invalid data EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.11.1.6. getTCATPParameter

void getTCATPParameter (
in globaldefs::NamingAttributes_T tpName,
in transmissionParameters::LayerRate_T layerRate,
in Granularity_T granularity,
out TCAParameters_T tcaParameter)
raises (globaldefs::ProcessingFailureException);

General comment	
The purpose of this operation is to retrieve the values of PM thresholds on a TP/layerRate measurement point. The operation is best-effort. Results are returned in the out parameter of this operation. The operation can be applied to a PTP, an FTP or CTP. The NMS requests the TCA parameters for the particular TP and granularity specified. **Supported products:** EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ tpName	Identification of the TP for which the values of the TCA parameters are to be retrieved. The termination point name must be explicit (a generic endpoint specification may not be used in this case).
❑layerRate	LayerRate for which the values of the TCA parameters are to be retrieved.
❑granularity	Granularity for which the TCA parameters are to be retrieved.
➢ tcaParameter	Result of the operation.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when tpName does not reference a terminationPoint object or layerRate contains an undefined rate or Granularity contains an undefined value EXCPT_ENTITY_NOT_FOUND - Raised when tpName references an object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.11.1.7. getHistoryPMData

```
void getHistoryPMData (
                    in Destination_T destination,
                    in string userName,
                    in string password,
                    in PMTPSelectList_T pmTPSelectList,
                    in PMParameterNameList_T pmParameters,
                    in globaldefs::Time_T startTime,
                    in globaldefs::Time_T endTime,
                    in boolean forceUpload)
                    raises(globaldefs::ProcessingFailureException);
```

General comment

This operation instructs the EMS to store history PM data into a file, and to send the file path, ftp user/pass to EOS. So EOS can get the file via FTP (File Transfer Protocol). Measurement intervals and the given time frame are considered as half open intervals to the right, i.e. startTime <= t < endTime.
A particular historic measurement interval (of duration 15 min resp. 24 h) is defined to be covered by the specified time frame if there is a non empty intersection between the measurement interval and the given time frame, i.e. endTime[PM record] > startTime[request parameter] AND startTime[PM record] < endTime[request parameter]. PM data are returned for all covered measurement intervals.
Performance Monitoring Data transfer normally involves huge amounts of data. The capability to send PM data to a controlled destination other than the NMS allows for flexibility in the choice of the PM data file format, the particular file transfer
protocol (including the possibility to apply data compression techniques) and the destination machine so as to make optimal use of the available data network capacity. The file transfer protocol to transfer PM data is the FTP protocol.
This operation is asynchrounous and is not required to wait until the PM data is extracted or until the FTP transfer is over before it returns. Notifications can be generated to inform the NMS of file ready.
Performance monitoring data on multiple TPs of multiple MEs is transferred in one data file.

Supported products: EMLNBI1.0 SDHNBI1.0
EMLNBI1.1 SDHNBI1.1
EMLNBI1.2 SDHNBI1.2
EMLNBI1.3 SDHNBI1.3
EMLNBI1.4 SDHNBI1.4
EMLNBI1.5 SDHNBI1.5
EMLNBI1.6 SDHNBI1.6

❑ Input / Output ➢	Comment
❑ destination	The destination to which the Performance Monitoring Data file is to be send. Not supported in this version.
❑ userName	necessary for file transfer with FTP. Not supported in this version.
❑ password	necessary for file transfer with FTP. Not supported in this version.
❑ pmTPSelectList	This parameter specifies which history PM data to return. If pmTPSelectList is empty, PM data for all monitored TPs of all MEs managed by the EMS (all layer rates, all granularities) are stored in the file. ME, TP and PMP can be included in this list.
❑pmParameters	This parameter specifies which PM parameters within the scope of the pmTPSelectList shall be contained in the * file. An empty list means to store all supported parameters. The returned parameters are best effort, i.e. among the parameters specified only the subset of supported parameters will be stored.
❑startTime	Specifies the start of the time window for collection (included).

	For 15m, default=first period of current day; for 24h, default= earliest period of all available, it means all data: from the beginning.
❑endTime	Specifies the end of the time window for collection (excluded). default=current time
❑forceUpload	Specifies if the EMS must upload all available PM data requested from the MEs. Not supported in this version.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is not well formed

7.11.1.8. getAllCurrentPMData

```
void getAllCurrentPMData(
        in PMTPSelectList_T pmTPSelectList,
        in PMParameterNameList_T pmParameters,
        in unsigned long how_many,
        out PMDataList_T pmDataList,
        out PMDataIterator_I pmIt)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request a filtered set (scoped by the input parameter pmParameters) of the current PM data for a list of TP measurement points on Alcatel Q3 NetworkElement. Within the request for each measurement point, granularity (15min, 24h, NA, etc.) and location (nearEnd and/or farEnd and/or bidirectional) may be specified. The operation mode is best effort. Current data will generally be marked as incomplete, as the current bin is not completed. If no PM data are available as specified, an empty list is returned. In order to allow the NMS to deal with a large number of objects this operation uses an iterator. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pmTPSelectList	list of measurement points for which to get the PM data.
❑ pmParameters	specifies which PM parameters within the scope of the pmTPSelectList shall be returned. An empty list means to return all supported parameters. The returned parameters are best effort, i.e. among the parameters specified only the subset of supported parameters will be returned.
❑ how_many	maximum number of PMData to return in the first batch.
➢ pmDataList	first batch of PMData returned.
➢pmIt	iterator returned to access the remaining PMData.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_NE_COMM_LOSS - Raised when communications to the Managed Element is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.11.1.9. getProfileAssociatedTPs

void getProfileAssociatedTPs(
in globaldefs::NamingAttributes_T profileName,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T tpNames,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation gets the set of PMPs that are associated with a TCA Parameter Profile. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ profileName	gives the name of the profile.
❑ how_many	maximum number of PMP names to return in the first batch.
➢ tpNames	For current version,the value of tpNames returned by EMLNBI is always PMP. It provides set of PMP names associated with the profile provided.
➢ nameIt	Iterator to retrieve the remaining PMP names.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - Raised when profileName references an object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.11.1.10. createTCAParameterProfile

void createTCAParameterProfile(
in globaldefs::NamingAttributes_T managedElementName,
in transmissionParameters::LayerRate_T layerRate,
in string userLabel,
in boolean forceUniqueness,
in string owner,
in globaldefs::NVSList_T additionalInfo,
in TCAParameterList_T listOfTCAParameter,
out TCAParameterProfile_T tcaParameterProfile)

raises(globaldefs::ProcessingFailureException);

General comment
This operation creates a new TCA Parameter Profile. For current version,TCAParameterProfile is supported on Q3 ne. For most ne,layerRate is meanlingless except 1626 and 1696. Ne 1696 only support 4 default profile,dose not allow creation,deletion. The four default profile are "RS Counter 24h","RS Counter 15min","OCH Counter 24h","OCH Counter 15m". So to 1696,create operation is equivalent to modify operation.Which profile is modified depends on layerate. layerrate --------- TCAProfile [22|23|90] ---------- "RS Counter" [107|108|109] ---------- "OCH Counter" For 1626,ne support 10 default TCAProfiles whose name are "RS 24h","RS 15m","8B/10B 24h","8B/10B 15m", "64B/66B 24h","64B/66B 15m","OCH 24h","OCH 15m","ODU 24h","ODU 15m". All create operation is equivalent to clone operation plus modify operation which are automatically done by EMLNBI.Which profile is cloned depends on layerrate. layerrate --------- TCAProfile [22|23|90] ---------- "RS Counter" [107|108|109] ---------- "OCH Counter" [97|98] ---------- "8B/10B" [113] ---------- "64B/66B" [104|105|106] ----------"ODU" **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6

❑ **Input / Output ➢**	**Comment**
❑ managedElementName	ME under which the profile is to be created.
❑ layerRate	defines the layer rate of the profile to be created.
❑userLabel	contains the NMS defined name of the profile to be created.
❑ forceUniqueness	if set to TRUE the EMS has to check if the provided user label is unique in the network element domain.
❑owner	A label of the owner of the Profile. This is an optional parameter.
❑additionalInfo	Some additional information may be specified by the NMS.
❑listOfTCAParameter	contains sets of threshold values.
➢tcaParameterProfile	EMS returns the new profile to the NMS.

raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - Raised when specified managedElement does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost

7.11.1.11.deleteTCAParameterProfile

void deleteTCAParameterProfile(
in globaldefs::NamingAttributes_T tcaParameterProfileName)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation deletes a TCA Parameter Profile. Ne 1696 does not support this operation. Ne 1626 only support non-default profile deletion. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ tcaParameterProfileName	name of the profile to be deleted.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - Raised when specified Profile does not exist EXCPT_OBJECT_IN_USE - Raised when Profile is still assigned to at least one TP EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost

7.11.1.12. getTCAParameterProfile

```
void getTCAParameterProfile(
        in globaldefs::NamingAttributes_T tcaParameterProfileName,
        out TCAParameterProfile_T tcaParameterProfile)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation gets all threshold values of a TCA Parameter Profile. **Supported products:** EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ tcaParameterProfileName	name of the profile to be retrieved.
➢tcaParameterProfile	contains the sets of threshold values.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - Raised when specified Profile does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost

7.11.1.13. setTCAParameterProfile

```
void setTCAParameterProfile(
        in globaldefs::NamingAttributes_T tcaParameterProfileName,
        in TCAParameterList_T listOfTCAParameter,
        in unsigned long how_many,
        out TCAParameterProfile_T tcaParameterProfile,
```

```
out globaldefs::NamingAttributesList_T failedTPList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation configures all threshold values of a TCA Parameter Profile and overwrites all the existing threshold values of the profile with the new provided threshold values. All threshold values of all TPs associated to this Profile will be changedaccording to the new values. Note: This includes also deletion of thresholds if the threshold is no longer contained in the provided list of TCA Parameters. The EMS has to return all TPs that could not be changed to the new threshold values due to some error reasons. **Supported products:** EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tcaParameterProfileName	name of the profile to be configured.
❑listOfTCAParameter	contains sets of threshold values to change the Profile completely.
❑ how_many	maximum number of failedTPs to return in the first batch.
➢tcaParameterProfile	returns the changed TCA Parameter Profile.
➢failedTPList	set of TPs that could not be changed to the new threshold values.
➢nameIt	Iterator to retrieve the remaining failedTP names.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - Raised when TCA Parameter Profile to be changed does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost

7.11.1.14. getAllTCAParameterProfileNames

```
void getAllTCAParameterProfileNames(
        in globaldefs::NamingAttributes_T meName,
        in unsigned long how_many,
        out globaldefs::NamingAttributesList_T tcaParameterProfileNames,
        out globaldefs::NamingAttributesIterator_I nameIt)
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation gets all TCA Parameter Profile names regardless of whether the profile is assigned to a TP or not. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**

❑ meName	name of the Network Element containing the Profiles.
❑ how_many	maximum number of tcaParameterProfile names to return in the first batch.
➢ tcaParameterProfileNames	contains the names of the existing TCA Parameter Profiles.
➢nameIt	Iterator to retrieve the remaining tcaParameterProfile names.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - Raised when specified managed element does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.11.1.15. getAllPMPs

```
void getAllPMPs(
        in globaldefs::NamingAttributes_T tpOrMeName,
        in unsigned long how_many,
        in Granularity_T granularity,
        out PMPList_T pmpList,
        out PMPIterator_I pmpIt )
        raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation allows an NMS to retrieve all PMPs contained in a ME specified. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpOrMeName	The name of the object to which this selection applies. This may be: ManagedElement: The selection applies to all termination points contained within the ManagedElement.
❑ how_many	Maximum number of PMPs to return in the first batch.
❑ granularity	only the PMPs that support the granularity will be returnned. Default is 15min.
➢ pmpList	First batch of PMPs.
➢pmpIt	Iterator to retrieve remaining PMPs.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this operation EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when references an object of type other than ME or TP EXCPT_ENTITY_NOT_FOUND - Raised when tpOrMeName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.11.1.16. getAllPMPNames

void getAllPMPNames(
in globaldefs::NamingAttributes_T tpOrMeName,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllPMPs(), but instead of returning the entire object structures, this operation returns their names. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.0 SDHNBI1.0 EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpOrMeName	The name of the object to which this selection applies. This may be: ManagedElement: The selection applies to all termination points contained within the ManagedElement. TerminationPoint: The selection applies only to the named termination point which will be a PTP, FTP or a CTP. Unlike the case for managedElement, the operation will not apply to any contained TPs when a PTP, FTP or CTP is specified.
❑ how_many	Maximum number of PMP names to return in the first batch.
➢ nameList	First batch of PMP names.
➢nameIt	Iterator to retrieve the remaining PMP names.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if an EMS does not implement this operation EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when references an object of type other than ME or TP EXCPT_ENTITY_NOT_FOUND - Raised when tpOrMeName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.11.1.17. setTCAParameterProfilePointer

void setTCAParameterProfilePointer(
in globaldefs::NamingAttributes_T tpName,
in globaldefs::NamingAttributes_T addTCAParameterProfile,
in globaldefs::NamingAttributes_T removeTCAParameterProfile)
raises(globaldefs::ProcessingFailureException);

General comment	
No further automatic updating based on changes in the profile will be done. For current version, the value of parameter tpName is always PMP. One PMP only have point one or zero profile. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tpName	The name of the TP to which to assign the pointer to the TCA parameter profile.
❑ addTCAParameterProfile	The name of the TCAParameterProfile to be added to PMP.
❑removeTCAParameterProfile	the name of the TCAParameterProfile to be removed from PMP.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when input parameter are syntactical incorrect EXCPT_ENTITY_NOT_FOUND - - Raised when tpName references an object that does not exist or when TCAParameterProfile to be assigned does not exist EXCPT_OBJECT_IN_USE - Raised when TCAParameterProfile of same Layer is already assigned to the TP EXCPT_UNABLE_TO_COMPLY - Raised when threshold values in the TP could not be configured EXCPT_NE_COMM_LOSS - Raised when communications to managed element is lost

7.11.1.18. getAllTCAParameterProfiles

void getAllTCAParameterProfiles(
in globaldefs::NamingAttributes_T meName,
in unsigned long how_many,
out TCAParameterProfileList_T tcaParameterProfileList,
out TCAParameterProfileIterator_I tcaParameterProfileIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation allows an NMS to request all the TCA parameter profiles associated with the specified managed element. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ meName	Name of the managed element containing the TCA parameter profiles.
❑ how_many	Maximum number of TCA parameter profiles to return in the first batch.
➢ tcaParameterProfileList	The first batch of TCA parameter profiles.
➢tcaParameterProfileIt	Iterator to retrieve the remaining TCA parameter profiles.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised when the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when meName does not reference a managed element object EXCPT_ENTITY_NOT_FOUND - Raised when meName references an object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.11.1.19. createPMCollectionPlan

```
void createPMCollectionPlan(
                    in PMCollectionPlan_T pmPlan,
                    out string pmPlanId)
                    raises (globaldefs::ProcessingFailureException);
```

General comment	
This operation is to create a PM data collection plan. A PM data collection plan is used to collect PM data every interval time in future. So, the startTime and endTime should be a future time, ohterwise the plan can't collect any data(but in the case that startTime is early than current time and endTime is later than current time, the plan will be executed from point near now time.). Furthermore, the whole plan time (endTime - startTime) should be multiple of interval time. When a period pm data file is ready, a notification named "NT_PM_PLAN_DATA_FILE_READY" will be sent to users. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pmPlan	Specifies the plan. PMCollectionPlan_T::Id: ignored, assigned by EMS automatically. PMCollectionPlan_T::tpSelected: specifies which PM data to

	collect (Ne or Tp name). PMCollectionPlan_T::pMlocationList: ignored currently. PMCollectionPlan_T::beginTime: specifies the start of the time window for collection (included). If empty current time assumed. PMCollectionPlan_T::endTime: specifies the end of the time window for collection (excluded). If empty, the pm data will be collected for ever. PMCollectionPlan_T::reprotInterval: specifies the report interval, must be multiple of 15 minutes. PMCollectionPlan_T::pmMonitorStatus: specifies the plan is to resume or suspend. PMCollectionPlan_T::pmReportStatus: specifies whether the plan data ready notification is to report to NMS automaticlly or not.
➢ pmPlanId	The unique plan id generated by EMS.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is illegal.

7.11.1.20. suspendPMCollectionPlan

void suspendPMCollectionPlan(
in string pmPlanId,
out PMCollectionPlan_T pmPlan)
raises (globaldefs::ProcessingFailureException);

General comment	
This operation instructs the EMS to suspend a existing PM plan. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pmPlanId	Specifies the plan Id.
➢ pmPlan	NMS return the plan which has been suspended to EMS.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is not well formed.

7.11.1.21. resumePMCollectionPlan

void resumePMCollectionPlan(
in string pmPlanId,
out PMCollectionPlan_T pmPlan)
raises (globaldefs::ProcessingFailureException);

General comment	
This operation instructs the EMS to resume a existing PM plan. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pmPlanId	Specifies the plan Id.
➢ pmPlan	NMS return the plan which has been resumed to EMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is not well formed.

7.11.1.22. deletePMCollectionPlan

```
void deletePMCollectionPlan(
                    in string pmPlanId)
                    raises (globaldefs::ProcessingFailureException);
```

General comment	
This operation instructs the EMS to delete a existing PM plan. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ pmPlanId	Specifies the plan Id.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is not well formed.

7.11.1.23. setPMCollectionPlan

```
void setPMCollectionPlan(
                    inout PMCollectionPlan_T pmPlan)
                    raises (globaldefs::ProcessingFailureException);
```

General comment

This operation instructs the EMS to modify a existing PM plan. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑➢ pmPlan	Specifies the plan Id and modify data to modify the plan.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is not well formed

7.11.1.24. getPMCollectionPlan

```
void getPMCollectionPlan(
                    in string pmPlanId,
                    out PMCollectionPlan_T pmPlan)
                    raises (globaldefs::ProcessingFailureException);
```

General comment	
This operation instructs the EMS to return a existing PM plan. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
❑ pmPlanId	Specifies the plan to return.
➢pmPlan	The plan return to NMS.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when any input parameter is not well formed.

7.11.1.25. getAllPMCollectionPlans

```
void getAllPMCollectionPlans(
                    out PMCollectionPlanList_T pmPlanList)
                    raises (globaldefs::ProcessingFailureException);
```

General comment

This operation instructs the EMS to return all existing PM plans. **Supported products:** EMLNBI1.1 SDHNBI1.1 EMLNBI1.2 SDHNBI1.2 EMLNBI1.3 SDHNBI1.3 EMLNBI1.4 SDHNBI1.4 EMLNBI1.5 SDHNBI1.5 EMLNBI1.6 SDHNBI1.6	
❑ Input / Output ➢	**Comment**
➢ pmPlanList	The plan list return to NMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure.

7.12. NotifyPublish Interface

An interface used by event publishers.

7.12.1. Operations

7.12.1.1. offer_change

```
void offer_change (
        in CosNotification::EventTypeSeq added,
        in CosNotification::EventTypeSeq removed )
        raises ( InvalidEventType );
};
```

General comment	
This operation indicates that a supplier is changing the names of the types of events it is publishing.	
❑ Input / Output ➢	**Comment**
❑ added	The event types added.
❑ removed	The event types removed.
raises	**ExceptionType**
InvalidEventType	*Exception used to indicate an invalid event type. Raises InvalidEventType if any of the event type names in either of the input sequences are invalid.*

7.13. NotifySubscribe Interface

An interface used by event subscribers.

7.13.1. Operations

7.13.1.1. subscription_change

```
void subscription_change(
        in CosNotification::EventTypeSeq added,
        in CosNotification::EventTypeSeq removed )
        raises ( InvalidEventType );
```

```
};
```

General comment	
This operation indicates that a consumer is changing the names of the types of events it is subscribed to.	
❑ Input / Output ➢	Comment
❑ added	The event types added.
❑ removed	The event types removed.
raises	**ExceptionType**
InvalidEventType	*Raises InvalidEventType If any of the event type names in either of the input sequences are invalid.*

7.14. PushConsumer Interface

An interface used by event subscribers.

7.14.1. Operations

7.14.1.1. subscription_change

```
void subscription_change(
        in CosNotification::EventTypeSeq added,
        in CosNotification::EventTypeSeq removed )
        raises ( InvalidEventType );
};
```

General comment	
This operation indicates that a consumer is changing the names of the types of events it is subscribed to.	
❑ Input / Output ➢	Comment
❑ added	The event types added.
❑ removed	The event types removed.
raises	**ExceptionType**
InvalidEventType	*Raises InvalidEventType If any of the event type names in either the of the input sequences are invalid.*

7.15. EquipmentInventoryMgr_I Interface

The EquipmentInventoryMgr_I is used to gain access to operations, which deal with equipment.

7.15.1 Operations

7.15.1.1. getContainedEquipment

```
void getContainedEquipment (
        in globaldefs::NamingAttributes_T equipmentHolderName,
        out EquipmentOrHolderList_T       equipmentOrHolderList)
       raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns the equipments and equipment holders directly contained by a specific equipment holder. This method differs from getAllEquipment in that it only looks at the next level of the containment hierarchy. **Supported products**: EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 NMBIN1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ equipmentHolderName	Name of the equipment holder for which to retrieve the directly contained equipment and equipment holders.
➢ equipmentOrHolderList	The directly contained equipments and equipment holders.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure* *EXCPT_INVALID_INPUT - Raised when equipmentHolderName does not reference an equipment holder object* *EXCPT_ENTITY_NOT_FOUND - Raised when equipmentHolderName references an equipment holder that does not exist* *EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost*

7.15.1.2. getEquipment

```
void getEquipment(
    in globaldefs::NamingAttributes_T equipmentOrHolderName,
    out EquipmentOrHolder_T          equip)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns the equipment or equipment holder for the given equipment or equipment holder name. **Supported products**: EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 NMBIN1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ equipmentOrHolderName	Name of the equipment or equipment holder to retrieve.
➢equip	The returned equipment or equipment holder.

raises	ExceptionType
ProcessingFailureException	*EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure* *EXCPT_INVALID_INPUT - Raised when equipmentOrHolderName does not reference an equipment nor an equipment holder* *EXCPT_ENTITY_NOT_FOUND - Raised when equipmentOrHolderName references an equipment or equipment holder that does not exist* *EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost*

7.15.1.3. getAllEquipment

```
void getAllEquipment(
    in  globaldefs::NamingAttributes_T      meOrHolderName,
    in  unsigned long                       how_many,
    out EquipmentOrHolderList_T             eqList,
    out EquipmentOrHolderIterator_I         eqIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the equipments and equipment holders that are contained in a managed element or equipment holder. **Supported products**: EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 NMBIN1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ meOrHolderName	The name of the ME or equipment holder for which to retrieve contained equipments and equipment holders. Equipment holder is not supported in this version.
❑ how_many	Maximum number of EquipmentOrHolder_Ts to return in the first batch.
➢ eqList	The first batch of EquipmentOrHolder_T s
➢ eqIt	The iterator used to retrieve the remaining EquipmentOrHolder_T s.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when managedElementOrEquipmentName does not reference a managed element or an equipment holder EXCPT_ENTITY_NOT_FOUND - Raised when managedElementOrEquipmentName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.15.1.4. getAllEquipmentNames

```
void getAllEquipmentNames(
    in globaldefs::NamingAttributes_T          meOrHolderName,
    in unsigned long                           how_many,
    out globaldefs::NamingAttributesList_T     nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllEquipment(), but returns the object names instead of returning the entire objects. **Supported products**: EMLNBI1.0 EMLNBI1.1 EMLNBI1.2 EMLNBI1.3 EMLNBI1.4 NMBIN1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ meOrHolderName	The name of the ME or equipment holder for which to retrieve contained equipment and equipment holder names.
❑ how_many	Maximum number of names to return in the first batch.
➢ nameList	The first batch of names.
➢ nameIt	The iterator to retrieve the remaining names.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when managedElementOrEquipmentName does not reference a managed element or an equipment holder EXCPT_ENTITY_NOT_FOUND - Raised when managedElementOrEquipmentName references object which does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.15.1.5. getAllSupportedPTPs

```
void getAllSupportedPTPs(
    in  globaldefs::NamingAttributes_T              equipmentName,
    in  unsigned long                               how_many,
    out terminationPoint::TerminationPointList_T    tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This service allows an NMS to request the PTPs/FTPs that aredirectly supported by a given equipment. The PTPs that are returned are those that share their physical layer with the primary equipment (i.e. that represent a port on the equipment or are connected by a fibre, wire, etc.). The FTPs that are returned are those which are implemented by the physical circuitry supported by the equipment and will include FTPs that are named from the specified equipment (the FTPs will depend upon the presence of that equipment for their most basic operation). When there is equipment protection, this operation reports PTPs/FTPs for the primary equipment only. That is, when called on a protecting equipment (as opposed to the primary equipment), this operation returns an empty list, irrespective of the current switch status. **Supported products:** EMLNBI1.3 EMLNBI1.4 NMBIN1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ equipmentName	The name of the equipment for which to retrieve supported PTPs/FTPs.
❑ how_many	Maximum number of PTPs/FTPs to be reported in the first batch.
➢ tpList	The first batch of PTPs/FTPs.
➢ tpIt	The iterator to retrieve the remaining PTPs/FTPs.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.15.1.6. getAllSupportedPTPNames

void getAllSupportedPTPNames(
in globaldefs::NamingAttributes_T equipmentName,
in unsigned long how_many,
out globaldefs::NamingAttributesList_T nameList,
out globaldefs::NamingAttributesIterator_I nameIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllSupportedPTPs(), but returns the object names instead of returning the entire objects. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ equipmentName	The name of the equipment for which to retrieve supported PTPs/FTPs.
❑ how_many	Maximum number of PTPs/FTPs to be reported in the first batch.
➢ nameList	The first batch of PTP/FTP names.
➢ nameIt	The iterator to retrieve the remaining PTP/FTP names.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached.

7.15.1.7. getAllSupportingEquipment

void getAllSupportingEquipment(

in globaldefs::NamingAttributes_T ptpOrMfdName,
out EquipmentOrHolderList_T eqList)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows an NMS to request all of the equipment which implement a PTP/FTP or a Matrix Flow Domain. For a PTP, the equipments that are returned are those which support the physical layer of the PTP (i.e. have the port on them or are connected by a fibre, wire, etc.). For an FTP, the equipments returned are those which support the physical circuitry implementing the FTP and will include the equipment from which the FTP is named (the FTPs will depend upon the presence of that equipment for their most basic operation). For a particular PTP/FTP the Tx port/function and Rx port/function may be on different cards and in this case both should be returned. Equipment that are used by the PTPs/FTPs, but that do not support them directly (such as a shared DEMUX card) are not reported. When there is equipment protection, this operation reports the primary equipment only. The protecting equipment will not be returned by this operation. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ *ptpOrMfdName*	The name of the PTP/FTP or MFD for which to retrieve the supporting equipments.
➢ *eqList*	The list of equipments (not equipment holders) directly implementing the PTP/FTP or matrix flow domain.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.15.1.8. getAllSupportingEquipmentNames

void getAllSupportingEquipmentNames(
in globaldefs::NamingAttributes_T ptpOrMfdName,
out globaldefs::NamingAttributesList_T nameList)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getAllSupportingEquipment(), but returns the object names instead of returning the entire objects. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**

❑ *ptpOrMfdName*	The name of the PTP/FTP or MFD for which to retrieve the supporting equipments.
➢ *nameList*	The list of equipments (not equipment holders) directly implementing the PTP/FTP or matrix flow domain.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.15.1.9. getSupportingEquipment

void getSupportingEquipment(
in globaldefs::NamingAttributes_T equipmentName,
out EquipmentOrHolderList_T eqList)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows an NMS to request all of the equipment that supports a given piece of equipment, e.g., the power pack and synchronization card that support a given piece of equipment. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ *ptpOrMfdName*	the name of the equipment for which to retrieve the supporting equipments.
➢ *eqList*	the list of equipment (not equipment holders) that support the given equipment.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.15.1.10. getSupportingEquipmentNames

void getSupportingEquipmentNames(
in globaldefs::NamingAttributes_T equipmentName,

out globaldefs::NamingAttributesList_T nameList)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getSupportingEquipment(), but returns the object names instead of returning the entire objects. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ *ptpOrMfdName*	the name of the equipment for which to retrieve the supporting equipments.
➢ *eqList*	the list of equipment (not equipment holders) that support the given equipment.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.15.1.11. getSupportedEquipment

void getSupportedEquipment (
in globaldefs::NamingAttributes_T equipmentName,
out EquipmentOrHolderList_T eqList)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows an NMS to request all of the pieces of equipment supported by a given piece of equipment, e.g., a power pack might support many circuit packs. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ *equipmentName*	the name of the equipment for which to retrieve the supported equipment list.
➢ *eqList*	the list of equipment (not equipment holders) supported by the given piece of equipment.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.15.1.12. getSupportedEquipmentNames

void getSupportedEquipmentNames(
in globaldefs::NamingAttributes_T equipmentName,
out globaldefs::NamingAttributesList_T nameList)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation has exactly the same behaviour as getSupportedEquipment(), but returns the object names instead of returning the entire objects. **Supported products:** EMLNBI1.3 EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ *equipmentName*	the name of the equipment for which to retrieve the supported equipment list.
➢ *nameList*	the list of equipment names (not equipment holders) that are support by a given piece of equipment.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if EMS cannot support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when equipmentName does not reference an equipment object EXCPT_ENTITY_NOT_FOUND - Raised when equipmentName references object that does not exist EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost

7.16. MstpManagementMgr_I Interface

7.16.1 Operations

7.16.1.1. getAllVirtualBridges

void getAllVirtualBridges(
in globaldefs::NamingAttributes_T meName,
in unsigned long how_many,

out VirtualBridgeList_T virtualBridgeList,
out VirtualBridgeIterator_I virtualBridgeIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to get all virtual bridges on a specified object.One ne must at least contains one birdge which id is 0. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ meName	Ne Name
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure

7.16.1.2. createVirtualBridge

void createVirtualBridge(
in globaldefs::NamingAttributes_T meName,
in unsigned long stpId,
out VirtualBridge_T virtualBridge)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to create a specified object.Only if bridgeType=bridge8021ad and stpType=mstp, this operation is valid. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ meName	Ne Name
❑stpId	1--15
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.3. removeVirtualBridge

void removeVirtualBridge(
in globaldefs::NamingAttributes_T vBridgeName)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to remove a specified object. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**

❑ vBridgeName	bridge Name
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure

7.16.1.4. setVirtualBridgeParameter

void setVirtualBridgeParameter(
in globaldefs::NamingAttributes_T vBridgeName,
in globaldefs::NVSList_T attributeList,
out globaldefs::NVSList_T failedAttributeList)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to get virtual bridge parameter on a specified object. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ vBridgeName	bridge Name
❑attributeList	one or more attribute's name and value.Those 4 params can be set: vStpPriority: INTEGER (0..65535) vStpBridgeMaxAge: INTEGER(mstp:600-4000, rstp:4000--7700) vStpBridgeHelloTime: INTEGER(100..1000) vStpBridgeForwardDelay: INTEGER(400..3000) If stpType=mstp, those params can be set only in bridge0(CIST Bridge).
➢failedAttributeList	Failed attributes of input.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure

7.16.1.5. getAllVlans

void getAllVlans(
in globaldefs::NamingAttributes_T meName,
in unsigned long how_many,
out VlanList_T vlanList,
out VlanIterator_I vlanIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This operation is used by the client to discover all the vlan currently in operation for the managed element. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**

❑ meName	The name of the managed element for which the request is made.
❑ how_many	Maximum number of vlan to report in the first batch.
➢ vlanList	First batch of vlans.
➢vlanIt	Iterator used to access the remaining vlan, if any.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised when meName does not reference a managedElement object *EXCPT_ENTITY_NOT_FOUND* - Raised when meName references object which does not exist *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost *EXCPT_TOO_MANY_OPEN_ITERATORS* - Raised when maximum number of iterators that the EMS can support has been reached

7.16.1.6. createVlan

void createVlan(
in globaldefs::NamingAttributes_T meName,
in unsigned long vlanId,
in globaldefs::NamingAttributes_T vBridgeName,
in globaldefs::NamingAttributesList_T egressPortList,
in globaldefs::NamingAttributesList_T untaggedPortList,
out Vlan_T vlan)
raises(globaldefs::ProcessingFailureException);

General comment	
The NMS uses this operation to request the creation of a vlan. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ meName	The name of the managed element.
❑ vlanId	The id of vlan which will be created. If neType=ISA-ES16-2.2, untaggedPortList is not supported, and always empty.
❑ vBridgeName	Specifies bridge which has relationship with this vlan. It is meaningless except mstp protocol.
❑ egressPortList	The egress ports collection.
❑ untaggedPortList	The untagged ports collection.
➢ vlan	the success result of create operation.
raises	**ExceptionType**
ProcessingFailureException	*EXCPT_NOT_IMPLEMENTED* - Raised if the EMS does not support vlan *EXCPT_INTERNAL_ERROR* - Raised in case of non-specific EMS internal failure *EXCPT_INVALID_INPUT* - Raised if the meName,vBirdgeName is invalid *EXCPT_NE_COMM_LOSS* - Raised when communications to managedElement is lost and this prevents creating the vlan

7.16.1.7. removeVlan

void removeVlan(
in globaldefs::NamingAttributes_T vlanName)
raises(globaldefs::ProcessingFailureException);

General comment	
The NMS uses this operation to request the removing of a vlan. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ vlanName	Specifies vlan which will be removed.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support vlan EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised if the vlanName is invalid EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is lost and this prevents removing the vlan

7.16.1.8. getDynamicalForwardingTable

void getDynamicalForwardingTable(
in globaldefs::NamingAttributes_T neOrVBridgeName,
in unsigned long how_many,
out ForwardingRecordList_T forwardingRecords,
out ForwardingRecordIterator_I forwardingRecordsIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to get dynamical forwarding records on a specified object. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ neOrVBridgeName	ne or bridge Name. if no vlan, must input ne name,otherwise must vlan name.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.9. setStaticForwardingTable

void setStaticForwardingTable(
in globaldefs::NamingAttributes_T managedElementNameOrvlanName,
in MacAddress macAddressValue,
in globaldefs::NamingAttributesList_T forwardingPortsName)
raises(globaldefs::ProcessingFailureException);

General comment

This service allows the NMS to set static forwarding records on a specified object. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ managedElementNameOrvlanName	ne or vlan Name.if no vlan, must input ne name,otherwise must vlan name.
❑macAddressValue	Hex String, the length should be 12. e.g."01005EFFFFFF", if the last bit of the second byte is '1' means multicast record. other is for unicast record.
❑forwardingPortsName	one or more ptp name. if set unicast record, this is unicast allowed to go to ports.if set multicast record, this is egress ports.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.10. getStaticForwardingTable

void getStaticForwardingTable(
in globaldefs::NamingAttributes_T neOrVBridgeName,
in unsigned long how_many,
out ForwardingRecordList_T forwardingRecords,
out ForwardingRecordIterator_I forwardingRecordsIt)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to get static forwarding records on a specified object. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ neOrVBridgeName	ne or virtual bridge Name.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.11. delStaticForwardingTable

void delStaticForwardingTable(
in globaldefs::NamingAttributes_T forwardingName)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to delete on a specified forwarding record. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ **Input / Output ➢**	**Comment**

❑ forwardingName	forwarding record Name.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.12. setvStpMode

void setvStpMode(
in globaldefs::NamingAttributes_T neOrVBridgeName,
in StpMode_T stpStauts)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to set mode on a specified ne or vitrual bridge port. If some port isn't down, this operation wouldn't be success. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ neOrVBridgeName	ne or virtual bridge name.
❑stpStauts	the specified mode.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.13. getvStpPortParameter

void getvStpPortParameter(
in globaldefs::NamingAttributes_T vBridgeName,
in unsigned long portId,
out StpPortParam_T stpPortParams)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to get parameters on a specified vitrual bridge port. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ vBridgeName	virtual bridge name.
❑ portId	the specified ptp id of the input vBridgeName.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.16.1.14. setvStpPortParameter

void setvStpPortParameter(
in globaldefs::NamingAttributes_T vBridgeName,
in unsigned long portId,
in globaldefs::NVSList_T stpPortParams,

out globaldefs::NVSList_T failedStpPortParams)
raises(globaldefs::ProcessingFailureException);

General comment	
This service allows the NMS to set parameters on a specified vitrual bridge port. **Supported products**: EMLNBI1.4 EMLNBI1.5 EMLNBI1.6	
❑ Input / Output ➢	**Comment**
❑ vBridgeName	virtual bridge name.
❑ portId	the specified ptp id of the input vBridgeName.
❑stpPortParams	one or more parameter's name and value, 5 parameters can be set: vStpPortPriority:INTEGER (0--16*15) vStpPortPathCost:INTEGER (0..65535) vStpPortAdminConnectionType: INTEGER, can be modified when stpmode is p-Vlan. nopointtopoint(1), pointtopoint(2), autopointtopoint(3), edgeport(4) vStpPortManualMode: INTEGER no(1),blocking(2), forwarding(3) vStpPortBridgeHelloTime: use setVirtualBridgeParameter() to set vStpBridgeHelloTimen, it will be change, only when p-vlan
➢failedStpPortParams	Iterator to retrieve the failed params.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure

7.17. Version_I Interface

7.15.1 Operations

7.17.1.1. getVersion

string getVersion();

General comment
This service returns the version of the IDL that the corresponding EMS object supports. It's 3.0 in this version. The format of the return string is as follows: Release.Major.Minor where Release,Major and Minor are strings that contain only digits. For example, 2.1 indicates release 2 and major release 1, 1.3 indicates release 1 and major release 3, and so on. Note that x.y has the same meaning as x.y.0 The minor digit is used for bug fixing the major release. e.g 1.2.1 is a minor release on 1.2. Any changes to any IDL files must be approved by the Specification Authority. For TMF 814 version 2.1, the version is "2.1". For TMF 814 version 3.0, the version is "3_0" or "3.0". Returns string: The version of the interface. **Supported products**: EMLNBI1.0, SDHNBI1.0 EMLNBI1.1, SDHNBI1.1 EMLNBI1.2, SDHNBI1.2 EMLNBI1.3, SDHNBI1.3 EMLNBI1.4, SDHNBI1.4 EMLNBI1.5, SDHNBI1.5 EMLNBI1.6, SDHNBI1.6, PKTNBI1.6

7.18. FlowDomainMgr_I Interface

Operations

7.18.1.1. getAllFlowDomains

```
void getAllFlowDomains(
    in unsigned long how_many,
    out FDList_T flowDomains,
    out FDIterator_I fdIt)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This method allows an NMS to request a list of the flow domains that exist in the EMS. In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ how_many	maximum number of PMData to return in the first batch.
➢ flowDomains	First batch of flow domains.
➢ fdIt	Iterator to retrieve the remaining flow domains.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_NE_COMM_LOSS - Raised when communications to the Managed Element is lost EXCPT_TOO_MANY_OPEN_ITERATORS - Raised when maximum number of iterators that the EMS can support has been reached

7.18.1.2. getFlowDomainsByUserLabel

```
void getFlowDomainsByUserLabel(
    in string  userLabel,
    out FDList_T  flowDomains)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation returns the flow domain structures for the flow domains whose userLabel is supplied as a parameter. The operation does not use an iterator, since the number of flow domains returned is usually expected to be 1. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ userLabel	The user label of the flow domains to retrieve.
➢ flowDomains	The list of identified flow domains.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.3. getFlowDomain

```
void getFlowDomain(
    in globaldefs::NamingAttributes_T fdName,
    out FlowDomain_T flowDomain)
    raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns a flow domain given a flow domain name. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ fdName	Name of the flow domain to retrieve.
➢ flowDomain	Flow domain structure returned.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.4. getAssociatingFD

```
void getAssociatingFD(
    in globaldefs::NamingAttributes_T  mfdName,
    out FlowDomain_T  flowDomain )
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation returns the flow domain structure for the (single) flow domain that the Matrix Flow Domain that is supplied as a parameter is associated to. ● **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **mfdName**	The name of MFD.
➢ **flowDomain**	The flow domain that the Matrix Flow Domain is associated to.

raises	ExceptionType
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.5. getTransmissionParams

```
void getTransmissionParams(
    in globaldefs::NamingAttributes_T name,
    in ParameterGroupsList_T filter,
    out transmissionParameters::LayeredParameterList_T transmissionParams)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns the transmission parameters of a flow domain, matrix flow domain, flow domain fragment or transmission descriptor, given the name of the object. A set of groups of transmission parameters to be returned may be specified. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **name**	Name of the flow domain, matrix flow domain, flow domain fragment or transmission descriptor for which the transmission parameters shall be retrieved.
❑ **filter**	This filter allows to specify a set of parameter groups. Only transmission parameters that belong to one of the groups specified are returned. See the supporting document SD1-16_LayeredParameters.pdf for a set of available transmission parameters and their groupings. If an empty list is specified, all transmission parameters are returned.
➢ **transmissionParams**	A list or requested transmission parameters. For a flow domain fragment only one layer is contained by the list.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.6. createFlowDomain

```
void createFlowDomain(
    in FDCreateData_T createData,
    inout globaldefs::NamingAttributesList_T assignedCPTPs,
```

```
inout subnetworkConnection::TPDataList_T tpsToModify,
out FlowDomain_T theFD,
out string errorReason)
raises (globaldefs::ProcessingFailureException);
```

General comment	
This service allows the NMS to request from the EMS the creation of a flow domain given the parameters passed in the method. The NMS may specify matrix flow domains or flow domain edge CPTPs to be associated with the created flow domain. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **createData**	Structure describing the flow domain structure to be created.
❑ **assignedCPTPs**	Identifies the list of assigned CPTPs to be associated as flow domain edge CPTPs to the flow domain. This may be an empty list. Associating CPTPs to the flow domain is done on a best effort basis.
➢❑ **tpsToModify**	in: The list of TPs with associated parameters to be applied. out: The list of TPs with associated applied parameters.
➢ **theFD**	The new created FD. The EMS is responsible for guaranteeing uniqueness of the name of the FD. The name may be specified by the NMS in the createData.
➢ **errorReason**	Iterator to retrieve the remaining CPTPs.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.7. associateCPTPsWithFlowDomain

```
void associateCPTPsWithFlowDomain(
    in globaldefs::NamingAttributes_T  fdName,
    in globaldefs::NamingAttributesList_T  cptpNames,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    out string errorReason)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This service allows an NMS to request from the EMS the association of one or more CPTPs with a flow domain as flow domain edge CPTPs. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **fdName**	The name of the flow domain to be modified.
❑ **cptpNames**	The names of the CPTPs to be associated with the flow domain. If the list is empty nothing is done on the EMS and the method returns success.
➢❑ **tpsToModify**	in: The list of TPs with associated parameters to be applied.

	out: The list of TPs with associated applied parameters.
➢ **errorReason**	In case a "best effort" parameter could not be set or a CPTP could not be associated with the flow domain an indication of the fault reason is provided by the EMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.8. deAssociateCPTPsWithFlowDomain

```
void deAssociateCPTPsFromFlowDomain (
    in globaldefs::NamingAttributes_T  fdName,
    in globaldefs::NamingAttributesList_T  tpNames,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    out string errorReason)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This service allows an NMS to request from the EMS the de-association of one or more CPTPs from a flow domain. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **fdName**	The name of the flow domain to be modified.
❑ **tpNames**	The names of the CPTPs to be de-associated from the flow domain. If the list is empty nothing is done on the EMS and the method returns success.
➢❑ **tpsToModify**	in: The list of TPs with associated parameters to be applied. out: The list of TPs with associated applied parameters.
➢ **errorReason**	In case a "best effort" parameter could not be set or a CPTP could not be associated with the flow domain an indication of the fault reason is provided by the EMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.9. getAllCPTPs

```
void getAllCPTPs(
    in globaldefs::NamingAttributes_T  fdName,
```

```
    in CPTP_Role_T                cptpRole,
    in unsigned long  how_many,
    out terminationPoint::TerminationPointList_T  cptpList,
    out terminationPoint::TerminationPointIterator_I  cptpIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns the structures of all CPTPs, or all flow domain edge CPTPs, or all flow domain internal CPTPs, associated with a flow domain. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ fdName	Name of the flow domain whose associated CPTPs shall be returned.
❑ cptpRole	Which CPTPs to return : flow domain edge, flow domain internal, or all CPTPs.
❑ how_many	Maximum number of CPTPs to return in the first batch.
➢ cptpList	First batch of CPTPs.
➢ cptpIt	Iterator to retrieve the remaining CPTPs.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.10. getAllFDFrs

```
void getAllFDFrs(
    in globaldefs::NamingAttributes_T fdName,
    in unsigned long how_many,
    in transmissionParameters::LayerRateList_T connectivityRateList,
    out flowDomainFragment::FDFrList_T fdfrList,
    out flowDomainFragment::FDFrIterator_I fdfrIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request a list of flow domain fragments for the specified flow domain at the specified connectivityRates. ● **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ fdName	Name of the flow domain
❑ how_many	List of rates of the flow domain fragments to be reported. If an empty list is specified, then all flow domain fragments of all rates are to be reported.
❑ connectivityRateList	Maximum number of flow domai fragments to be reported in the first batch.

➢ **fdfrList**	First batch of flow domain fragments.
➢ **fdfrIt**	Iterator to retrieve the remaining flow domain fragment.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.11. getFDFrsWithTP

```
void getFDFrsWithTP(
    in globaldefs::NamingAttributes_T  cptpName,
    in unsigned long  how_many,
    out flowDomainFragment::FDFrList_T  fdfrList,
    out flowDomainFragment::FDFrIterator_I  fdfrIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This service allows the NMS to request from the EMS the flow domain fragments that are connected to a specified flow point or CPTP. In case of a flow point as input parameter not more than one flow domain fragment will be returned. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **cptpName**	The name of the flow point or CPTP for which the connected flow domain fragments shall be returned.
❑ **how_many**	Maximum number of flow domain fragments to be reported in the first batch.
➢ **fdfrList**	First batch of flow domain fragments.
➢ **fdfrIt**	Iterator to retrieve the remaining flow domain fragments.<
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.12. getFDFrsByUserLabel

```
void getFDFrsByUserLabel(
    in string  userLabel,
    out flowDomainFragment::FDFrList_T  fdfrs)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation returns the flow domain fragment structures for the flow domain fragments whose userLabel is supplied as a parameter.The operation does not use an iterator, since the number of flow domain fragments returned is usually expected to be 1. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ userLabel	The user label of the flow domain fragments to retrieve.
➢ flowDomains	The list of identified flow domain fragments.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.13. getFDFr

void getFDFr(
in globaldefs::NamingAttributes_T fdfrName,
out flowDomainFragment::FlowDomainFragment_T fdfr)
raises (globaldefs::ProcessingFailureException);

General comment	
This operation returns the flow domain fragment structure for the flow domain fragment whose name is supplied as a parameter. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ fdfrName	The name of the flow domain fragment to retrieve.
➢ fdfr	The flow domain fragment structure retrieved.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.14. modifyFDFr

void modifyFDFr(
in globaldefs::NamingAttributes_T fdfrName,
in flowDomainFragment::FDFrModifyData_T fdfrModifyData,
in ConnectivityRequirement_T connectivityRequirement,

```
inout subnetworkConnection::TPDataList_T tpsToModify,
out globaldefs::NamingAttributesList_T failedTPList,
out globaldefs::NamingAttributesList_T parameterProblemsTPList,
out flowDomainFragment::FlowDomainFragment_T newFDFr,
out string errorReason)
raises (globaldefs::ProcessingFailureException);
```

General comment	
The NMS invokes the modifyFDFr service to request the EMS to modify an existing flow domain fragment as specified by the parameters passed in the method. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **fdfrName**	The name of the flow domain fragment to modify.
❑ **fdfrModifyData**	Structure describing how the flow domain fragment should be modified. Modification of connectionless parameters is best effort (except where specified otherwise for a particular parameter).
❑ **connectivityRequirement**	For a "connectivity-aware" EMS, this parameter shall identify the requested operation mode in case not all FPs have potential connectivity to one another.
❑➢ **tpsToModify**	A list of TPs and parameters to apply. On method return the list is updated to provide the resulting parameters. The list may refer to flow points or to the containing CPTPs.
➢ **failedTPList**	The list of Connectionless Port Termination Points or Flow Points (requested in the fdfrModifyData parameter) that could not be added or could not be removed, whichever was requested.
➢ **parameterProblemsTPList**	The list of Connectionless Port Termination Points and Flow Points FPs for which only best-effort transmission parameters could not be set.
➢ **newFDFr**	The modified flow domain fragment.
➢ **errorReason**	In case a "best effort" parameter could not be set an indication of the fault reason is provided by the EMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.15. createAndActivateFDFr

```
void createAndActivateFDFr(
    in flowDomainFragment::FDFrCreateData_T createData,
    in ConnectivityRequirement_T connectivityRequirement,
    in globaldefs::NamingAttributesList_T aEnd,
    in globaldefs::NamingAttributesList_T zEnd,
    inout globaldefs::NamingAttributesList_T internalTPs,
    inout flowDomainFragment::MatrixFlowDomainFragmentList_T mfdfrs,
    inout subnetworkConnection::TPDataList_T tpsToModify,
```

```
out flowDomainFragment::FlowDomainFragment_T theFDFr,
out globaldefs::NamingAttributesList_T notConnectableCPTPList,
out globaldefs::NamingAttributesList_T parameterProblemsTPList,
out string errorReason)
raises (globaldefs::ProcessingFailureException);
```

General comment	
The NMS invokes the createAndActivateFDFr service to request the EMS to create and activate a flow domain fragment given the parameters passed in the method. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **createData**	Structure describing the FDFr structure to be created. Setting of connectionless parameters is best effort (except where specified otherwise for a particular parameter).
❑ **connectivityRequirement**	For a "connectivity-aware" EMS, this parameter shall identify the requested operation mode in case not all FPs have potential connectivity to one another. If the EMS is not connectivity-aware, this parameter is ignored.
❑ **aEnd**	A list of of CPTP names that delimit the FDFr and characterize its edges (entrance and/or exit points). As a result of creating the FDFr, FPs are created as clients of the FD Edge CPTPs. In case of a unidirectional FDFr this attribute contains the list of source FD Edge CPTPs. In case of a bidirectional FDFr this attribute may be combined with the zEnd TPs attribute to obtain all the FD Edge CPTPs that are associated to the FDFr. (N.B. For a bidirectional point to point FDFr it is suggested, but not mandatory, to put one TP in the aEnd and one in the zEnd, as with SNCs and TLs. For a multipoint FDFr, or a point-to-point FDFr that may be expanded to multipoint, it is suggested to put all the TPs in the aEnd.)
❑ **zEnd**	In case of a unidirectional FDFr this attribute contains the list of sink FD Edge CPTPs that delimit the FDFr and characterize its edges (exit points). As a result of creating the FDFr, FPs are created as clients of the FD Edge CPTPs. In case of a bidirectional FDFr this attribute may be combined with the aEnd TPs attribute to obtain all the FD Edge CPTPs that are associated to the FDFr.
❑➢ **internalTPs**	A (possibly empty) list of internal CPTP names that must be included in the route of the FDFr. As a result of creating the FDFr, FPs are created as clients of the internal CPTPs.
❑➢ **mfdfrs**	An optional (possibly empty) list of MFDFrs that make up the route of the FDFr. This attribute may be omitted if the FDFr is routed by the network. As a result of creating the FDFr, MFDFrs are created in the various MFDs.
❑➢ **tpsToModify**	A list of TPs and parameters to apply. On method return the list is updated to provide the resulting parameters. The list may refer to flow points that are being created during the createAndActivateFDFr request or to the containing CPTPs.
➢ **theFDFr**	The new created and activated flow domain fragment. The EMS is responsible for guaranteeing uniqueness of the name of the flow domain fragment.
➢ **notConnectableCPTPList**	The list of Connectionless Port Termination Points that could not be connected.
➢ **parameterProblemsTPList**	The list of Connectionless Port Termination Points and Flow Points FPs for which only best-effort transmission parameters could not be set.
➢ **errorReason**	In case a "best effort" parameter could not be set or a flow point could not be connected an indication of the fault reason is

	provided by the EMS.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_ENTITY_NOT_FOUND - Raised when one of the provided CPTPs does not exist. EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is not met; the specified user label is currently being used. EXCPT_UNABLE_TO_COMPLY - Raised when the EMS is unable to execute the request because of one of the following conditions: at least one of the parameters although valid cannot be set and that parameter is identified as "not best effort"; the name specified by the NMS exists already in the EMS; The FP total connectivity constraint is not met; Unrecognized mapping criteria; Frames map to more than one FDFr; The FDFr would have less than two FPs. EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.16. deactivateAndDeleteFDFr

void deactivateAndDeleteFDFr(
in globaldefs::NamingAttributes_T fdfrName,
inout subnetworkConnection::TPDataList_T tpsToModify,
out flowDomainFragment::FlowDomainFragment_T theFDFr,
out string errorReason)

General comment	
This service allows an NMS to request the deactivation and deletion of a flow domain fragment from a flow domain. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ fdfrName	The user label of the flow domains to retrieve.
❑➢ tpsToModify	A list of TPs and parameters to apply. On method return the list is updated to provide the resulting parameters. The list may refer only to TPs that take part in the flow domain fragment deletion process.
➢theFDFr	The deactivated and deleted flow domain fragment
➢errorReason	Specifies the fault reason if any.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.17. getAllTopologicalLinksOfFD

```
void getAllTopologicalLinksOfFD(
    in globaldefs::NamingAttributes_T flowDomainName,
    in unsigned long how_many,
    out topologicalLink::TopologicalLinkList_T topoList,
    out topologicalLink::TopologicalLinkIterator_I topoIt)
   raises(globaldefs::ProcessingFailureException);
```

General comment	
This service returns a list of TopologicalLinks which are terminated at the Flow Domain whose name is passed as a parameter. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ flowDomainName	The name of the Flow Domain.
❑ how_many	Maximum number of topological links to return in the first batch.
➢ topoList	First batch of topological links.
➢ topoIt	Iterator to retrieve the remaining topological links.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.18.1.18. getFDFrRoute

```
void getFDFrRoute(
    in globaldefs::NamingAttributes_T fdfrName,
    out flowDomainFragment::FDFrRoute_T route)
   raises (globaldefs::ProcessingFailureException);
```

General comment

This service returns the route for the FDFr whose name is specified as a parameter. It is only used for systems where the route is provisioned by the NMS (e.g., it is not useful for Ethernet networks using GVRP). **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ fdfrName	The name of the FDFr.
➢ route	The route of the FDFr.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.19. TCProfileMgr_I

7.19.1. Operations

7.19.1.1. getAllTCProfiles

```
void getAllTCProfiles(
    in unsigned long how_many,
    out TCProfileList_T tcProfileList,
    out TCProfileIterator_I tcProfileIt)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the TC Profiles that are under the control of the TCProfileMgr_I. >In order to allow the NMS to deal with a large number of objects, this operation uses an iterator. See SD1-15 iterator overview for information on how iterators are used in this interface. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ how_many	The user label of the flow domains to retrieve.
➢ tcProfileList	the first batch of iterators.
➢ tcProfileIt	the iterator used to access the remaining TC Profiles.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.19.1.2. getTCProfile

```
void getTCProfile(
    in globaldefs::NamingAttributes_T tcProfileName,
    out TCProfile_T tcProfile)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This service returns the Traffic Conditioning Profile struct for the given tcProfileName. The Traffic Conditioning Profile structure contains an NVSList of traffic parameters. The traffic parameters returned will be the parameters in place on the actual Traffic Conditioning Profile. If there are notraffic parameters, then the NVSList will be empty. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tcProfileName	name of the TC Profile.
➢ tcProfile	the returned Traffic Conditioning Profile.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.19.1.3. getTCProfileAssociatedTPs

```
void getTCProfileAssociatedTPs(
    in globaldefs::NamingAttributes_T tcProfileName,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the TPs associated with the specified TC Profile. If there are no TPs which are associated with the specified TC Profile, then an empty list is returned. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**

❑ **tcProfileName**	the name of the TC Profile.
❑ **how_many**	maximum number of TPs to return in the first batch.
➢ **tpList**	first batch of TPs.
➢ **tpIt**	iterator to access the remaining TPs.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.19.1.4. createTCProfile

```
void createTCProfile(
    in  TCProfileCreateData_T      newTCProfileCreateData,
    out TCProfile_T newTCProfile)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
The createTCProfile operation is used to create a new Traffic Conditioning Profile on the server. A Traffic Conditioning Profile Create Data structure, representing the new Traffic Conditioning Profile, is passed as input. The resulting Traffic Conditioning Profile is returned as a result. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **newTCProfileCreateData**	information about the TC Profile to be created.
➢ **newTCProfile**	result of the creation.
raises *ProcessingFailureException*	**ExceptionType** * EXCPT_NOT_IMPLEMENTED - Raised if EMS does not support creation of TC Profiles via * this interface * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure * EXCPT_INVALID_INPUT - Raised if newTCProfileCreateData contains invalid data * EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is * not met * EXCPT_CAPACITY_EXCEEDED - Raised when maximum number of TC * Profiles has been reached * EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element * involved in this operation * EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, * and cannot determine the reason it could not comply, * it may raise this exception.

7.19.1.5. deleteTCProfile

```
void deleteTCProfile(
    in globaldefs::NamingAttributes_T tcProfileName)
    raises (globaldefs::ProcessingFailureException);
```

General comment	
The delete Traffic Conditioning Profile operation is used to delete a Traffic Conditioning Profile on the server. This operation is idempotent. If the service is called with the name of a non-existent Traffic Conditioning Profile, it will succeed. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tcProfileName	The name of the Traffic Conditioning profile to be deleted.
raises *ProcessingFailureException*	**ExceptionType** * The name of the Traffic Conditioning profile to be deleted. * Raises globaldefs::ProcessingFailureException * EXCPT_NOT_IMPLEMENTED - Raised if EMS does not support deletion of TC Profiles via * this interface * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure * EXCPT_OBJECT_IN_USE - Raised if there are TPs that are using * the Traffic Conditioning Profile * EXCPT_INVALID_INPUT - Raised when input parameter is syntactical incorrect. * EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element * involved in this operation * EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, * and cannot determine the reason it could not comply, * it may raise this exception.

7.19.1.6. modifyTCProfile

```
void modifyTCProfile(
    in globaldefs::NamingAttributes_T tcProfileName,
    in  TCProfileCreateData_T      tcProfileModifyData,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    out TCProfile_T modifiedTCProfile,
    out string errorReason)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
The NMS invokes the modifyTCProfile service to request from the EMS to modify an existing Traffic Conditioning Profile as specified by the parameters passed in the method.A Traffic Conditioning Profile Modify Data structure, representing the data to be changed, is passed as input. The resulting Traffic Conditioning Profile is returned as a result. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**

❑ **tcProfileName**	The name of the Traffic Conditioning profile to be modified.
❑ **tcProfileModifyData**	TC Profile data to be changed.
❑➢ **tpsToModify**	The TPs to be modified. The modified TPs are returned.
➢ **modifiedTCProfile**	the modified TC Profile.
➢**errorReason**	In case a "best effort" parameter could not be set an indication of the fault reason is provided by the EMS
raises *ProcessingFailureException*	**ExceptionType** * EXCPT_NOT_IMPLEMENTED - Raised if EMS does not support modification of TC Profiles via * this interface * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure * EXCPT_INVALID_INPUT - Raised if tcProfileName or tcProfileModifyData contains invalid data * EXCPT_ENTITY_NOT_FOUND - Raised when the Traffic Conditioning Profile to be modified does not exist. * EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is * not met * EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation. * EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, * and cannot determine the reason it could not comply, * it may raise this exception.

7.20. TransmissionDescriptorMgr_I

7.20.1. Operations

7.20.1.1. getAllTransmissionDescriptorNames

```
void getAllTransmissionDescriptorNames(
    in unsigned long how_many,
    out globaldefs::NamingAttributesList_T nameList,
    out globaldefs::NamingAttributesIterator_I nameIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This operation has exactly the same behaviour as getAllTransmissionDescriptors, but instead of returning the entire object structures, this operation returns their names. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **how_many**	the number of iterators to return in nameList.
➢ **nameList**	the first batch of iterator names.
➢ **nameIt**	the iterator used to retrieve the remaining iterator names.
raises	**ExceptionType**

ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.20.1.2. getAllTransmissionDescriptors

```
void getAllTransmissionDescriptors(
    in unsigned long how_many,
    out TransmissionDescriptorList_T transmissionDescList,
    out TransmissionDescriptorIterator_I transmissionDescIt)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the transmissionDescriptors that are under the control of the transmissionDescriptorMgr_I. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ how_many	the number of iterators to return in nameList.
➢ transmissionDescList	the first batch of iterator.
➢ transmissionDescIt	the iterator used to retrieve the remaining iterator.
raises	**ExceptionType**
ProcessingFailureException	EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.20.1.3. getTransmissionDescriptor

```
void getTransmissionDescriptor(
    in globaldefs::NamingAttributes_T tmdName,
    out TransmissionDescriptor_T tmd)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
This service returns the Transmission Descriptor struct for the given tmdName. The Transmission Descriptor structure contains an NVSList of traffic parameters. The traffic parameters returned will be the parameters in place on the actual Transmission Descriptor. If there are no traffic parameters, then the NVSList will be empty. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**

❑ **tmdName**	Name of the Transmission Descriptor.
➢ **tmd**	The returned Transmission Descriptor.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.20.1.4. getAssociatedTPs

```
void getAssociatedTPs(
    in globaldefs::NamingAttributes_T transmissionDescriptorName,
    in unsigned long how_many,
    out terminationPoint::TerminationPointList_T tpList,
    out terminationPoint::TerminationPointIterator_I tpIt)
  raises(globaldefs::ProcessingFailureException);
```

General comment	
This allows an NMS to request all of the TPs associated with the specified Transmission Descriptor. If there are no TPs which are associated with the specified Transmission Descriptor, then an empty list is returned. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **transmissionDescriptorName**	the name of the Transmission Descriptor.
❑ **how_many**	maximum number of TPs to return in the first batch.
➢ **tpList**	first batch of TPs.
➢ **tpIt**	iterator to access the remaining TPs.
raises *ProcessingFailureException*	**ExceptionType** EXCPT_NOT_IMPLEMENTED - Raised if the EMS does not support this service EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal failure EXCPT_INVALID_INPUT - Raised when pmTPSelectList is empty or contains invalid data EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, and cannot determine the reason it could not comply, EXCPT_NE_COMM_LOSS - Raised when communication is lost to a managed element involved in this operation

7.20.1.5. setTMDAssociation

```
void setTMDAssociation (
                    in globaldefs::NamingAttributes_T tmdName,
                    inout TPorMFDorFDFr_T tPorMFDorFDFr)
  raises (globaldefs::ProcessingFailureException);
```

General comment	
The association of a TMD to an an object by this operation amounts to a forced overwriting of the layered transmission parameters of the object by the layered transmission parameters of the TMD and to a forced overwriting of the additional info parameters of the object by the additional TP info parameters of the TMD. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tmdName	the name of the Transmission Descriptor.
❑➢ tPorMFDorFDFr	maximum number of TPs to return in the first batch.
raises	**ExceptionType**
ProcessingFailureException	* Raises globaldefs::ProcessingFailureException * EXCPT_NOT_IMPLEMENTED - Raised when EMS does not support * this operation. * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS * internal failure. * EXCPT_INVALID_INPUT - Raised when any input parameter is syntactically * incorrect. * EXCPT_ENTITY_NOT_FOUND - Raised when the TransmissionDescriptor * to be assigned does not exist. * EXCPT_UNABLE_TO_COMPLY - Raised when transmission parameter values * could not be configured in the MFD. * EXCPT_NE_COMM_LOSS - Raised when communication to managed element * is lost.

7.20.1.6. validateTMDAssignmentToObject

```
void validateTMDAssignmentToObject(
    in globaldefs::NamingAttributes_T objectName,
    out string objectAssignmentState,
    out transmissionParameters::LayeredParameterList_T transmissionParams,
    out globaldefs::NVSList_T additionalTPInfo)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
This service validates the TMD state of the specified TP, MFD or FDFr object identified by ObjectName. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ objectName	the TP, MFD or FDFr object whose TMD assignment shall be verified.
➢ objectAssignmentState	the TMD state (s) of the object as determined by the verification procedure.
➢ transmissionParams	a "delta" list of mismatched TMD transmission parameters, or empty.
➢ additionalTPInfo	a "delta" list of mismatched "additional TP info" parameters, or empty
raises	**ExceptionType**

ProcessingFailureException	* EXCPT_NOT_IMPLEMENTED - Raised if the entire request is not supported * by the EMS or the request with the specified input parameters * is not supported. * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure (see errorReason attribute of ProcessingFailureException()) * EXCPT_INVALID_INPUT - Raised when objectName is incorrectly formed * EXCPT_ENTITY_NOT_FOUND - Raised when objectName references an object * that does not exist * EXCPT_NE_COMM_LOSS - Raised when the communication to the managed * element containing objectName is lost * EXCPT_UNABLE_TO_COMPLY - Raised when the EMS is unable to * execute the request

7.20.1.7. modifyTransmissionDescriptor

```
void modifyTransmissionDescriptor(
    in globaldefs::NamingAttributes_T tmdName,
    in  TMDModifyData_T      tmdModifyData,
    inout subnetworkConnection::TPDataList_T tpsToModify,
    inout TransmissionDescriptor_T modifiedTransmissionDescriptor,
    out globaldefs::NamingAttributesList_T failedMEList,
    out globaldefs::NamingAttributesList_T failedTPsMFDsList,
    out string errorReason)
   raises (globaldefs::ProcessingFailureException);
```

General comment	
The modifyTransmissionDescriptor operation is used to modify a Transmission Descriptor (TMD) in the EMS. The resulting Transmission Descriptor is returned as a result.. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ tmdName	The name of the Transmission Descriptor to be modified.
❑ tmdModifyData	Information about the Transmission Descriptor to be modified.
❑➢ tpsToModify	in: The list of TPs with associated parameters to be applied. out: The list of TPs with associated applied parameters.
❑➢ modifiedTransmissionDescriptor	result of the modification.
➢failedMEList	the names of all TMD associated MEs that could not be reached by the EMS.
➢ failedTPsMFDsList	the names of all TPs and MFDs that could not be changed to the new parameter values due to some error reasons.
➢ errorReason	In case a "best effort" parameter could not be set an indication of the fault reason is provided by the EMS.
raises	**ExceptionType**

ProcessingFailureException	* EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure * EXCPT_INVALID_INPUT - Raised if tmdModifyData contains invalid data * EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is * not met * EXCPT_ENTITY_NOT_FOUND - Raised when tmdName references object which does * not exist * EXCPT_NE_COMM_LOSS - Raised when communications to managedElement is * lost * EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, * and cannot determine the reason it could not comply, * it may raise this exception.

7.20.1.8. deleteTransmissionDescriptor

void deleteTransmissionDescriptor(
in globaldefs::NamingAttributes_T transmissionDescriptorName)
raises (globaldefs::ProcessingFailureException);

General comment	
The delete Transmission Descriptor operation is used to delete a Transmission Descriptor on the server. This operation is idempotent. If the service is called with the name of a non-existent Transmission Descriptor, it will succeed. **Supported products:** PKTNBI1.6	
❑ Input / Output ➢	**Comment**
❑ transmissionDescriptorName	The name of the TransmissionDescriptor to be deleted.
raises	**ExceptionType**
ProcessingFailureException	* EXCPT_NOT_IMPLEMENTED - Raised if EMS does not support deletion of TMDs via * this interface * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure * EXCPT_OBJECT_IN_USE - Raised if there are TPs or Matrix Flow Domains * that are using the Transmission Descriptor * EXCPT_INVALID_INPUT - Raised when input parameter is syntactical incorrect. * EXCPT_ENTITY_NOT_FOUND - Raised when the Transmission Descriptor to be * deleted does not exist. * EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, * and cannot determine the reason it could not comply, * it may raise this exception.

7.20.1.9. createTransmissionDescriptor

void createTransmissionDescriptor(
in TMDCreateData_T newTMDCreateData,
out TransmissionDescriptor_T newTransmissionDescriptor)

raises (globaldefs::ProcessingFailureException);

General comment	
The createTransmissionDescriptor operation is used to create a new Transmission Descriptor on the server. A Transmission Descriptor Create Data structure, representing the new Transmission Descriptor, is passed as input. The resulting Transmission Descriptor is returned as a result. **Supported products:** PKTNBI1.6	
❑ **Input / Output ➢**	**Comment**
❑ **newTMDCreateData**	information about the Transmission Descriptor to be created.
➢ **newTransmissionDescriptor**	result of the creation.
raises	**ExceptionType**
ProcessingFailureException	* EXCPT_NOT_IMPLEMENTED - Raised if EMS does not support creation of TMDs via * this interface * EXCPT_INTERNAL_ERROR - Raised in case of non-specific EMS internal * failure * EXCPT_INVALID_INPUT - Raised if newTMDCreateData contains invalid data * EXCPT_USERLABEL_IN_USE - Raised when the userLabel uniqueness constraint is * not met * EXCPT_CAPACITY_EXCEEDED - Raised when maximum number of Transmission * Descriptors has been reached * EXCPT_UNABLE_TO_COMPLY - If the EMS cannot comply with the request, * and cannot determine the reason it could not comply, * it may raise this exception.

APPENDIX A

A.1 Notification Types

Notification Types

A.2 Operation Samples

Operation Samples

A.3 LayerRate

TMF Entity	Layer Rate	String
SubnetworkConnection	1000	LR_T_MPLS_PATH
SubnetworkConnection	1001	LR_T_MPLS_CHANNEL
ManagedElement	1	LR_Not_Applicable
TopologicalLink	1	LR_Not_Applicable
FDFr	96	LR_Ethernet
FP	96	LR_Ethernet

A.4 Alarm Object Type

TMF Entity	Alarm Object Type
SubnetworkConnection	OT_SUBNETWORK_CONNECTION
ManagedElement	OT_MANAGED_ELEMENT
TopologicalLink	OT_TOPOLOGICAL_LINK
FDFr	OT_FLOW_DOMAIN_FRAGMENT
FP	OT_CONNECTION_TERMINATION_POINT

End of Document