ICS 33.040
M 11

YD

中华人民共和国通信行业标准

YD/T 1289.6-2009

同步数字体系（SDH）传送网网络管理技术要求 第6部分：基于IDL/IIOP技术的网元管理系统（EMS）—网络管理系统（NMS）接口信息模型

Technical specification for SDH transport network management system part VI：EMS-NMS interface information model based on IDL/IIOP

2009-12-11 发布　　2010-01-01 实施

中华人民共和国工业和信息化部 发布

目　次

前　言……II
1　范围……1
2　术语、定义和缩略语……1
3　接口规范……2
3.1　配置管理……2
3.2　故障管理……99
3.3　性能管理……104
3.4　通用管理……121
3.5　公共管理……133
附录A（规范性附录）　性能文件格式定义……148
参考文献……150

前　言

《同步数字体系（SDH）传送网网络管理技术要求》分为6个部分：

—— 第1部分：基本原则

—— 第2部分：网元管理系统（EMS）功能

—— 第3部分：网络管理系统（NMS）功能

—— 第4部分：网元管理系统（EMS）—网络管理系统（NMS）接口功能部分

—— 第5部分：网元管理系统（EMS）—网络管理系统（NMS）接口通用信息模型

—— 第6部分：基于IDL/IIOP技术的网元管理系统（EMS）—网络管理系统（NMS）接口信息模型

本部分为第6部分。

本部分中的附录A为规范性附录。

本部分由中国通信标准化协会提出并归口。

本部分起草单位：中兴通讯股份有限公司、华为技术有限公司。

本部分主要起草人：年庆飞、陈　捷、刘少军。

同步数字体系（SDH）传送网网络管理技术要求 第6部分：基于IDL/IIOP技术的网元管理系统（EMS）—网络管理系统（NMS）接口信息模型

1 范围

本部分规定了SDH网管系统EMS-NMS之间基于CORBA IDL的接口规范。

本部分适用于SDH网元管理系统（EMS）与SDH网络管理系统(NMS)之间的接口。

本部分不适用于对WDM光网和PDH系统的管理。

2 术语、定义和缩略语

2.1 术语和定义

下列术语和定义适用于本部分。

2.1.1

网络管理系统 Network Management System

NMS 指 SDH 传送网网络管理系统，即为了管理 SDH 传送网网络所使用的软硬件系统。网络管理系统提供全网的端到端网络视图，能够管理网络内由不同设备供应商提供的 SDH 网元或 SDH 子网。

2.1.2

网元管理系统 Element Management System

EMS 指 SDH 传送网网元管理系统，即为了管理一个或多个 SDH 网元所使用的软硬件系统。网元管理系统管理由单一设备供应商提供的 SDH 网元或 SDH 子网。

注： 本部分中的网元管理系统是传统意义上的网元管理系统和子网管理系统（SNMS）的统称。

2.2 缩略语

下列缩略语适用于本部分。

CORBA	Common Object Request Broker Architecture	公共对象请求代理结构
CTP	Connection Termination Point	连接终端点
EMS	Element Management System	网元管理系统
FTP	File Transfer Protocol	文件传送[输]协议
FTP	Floating Termination Point	浮动终端点
NMS	Network Management System	网络管理系统
OMG	Object Management Group	对象管理组织
PTP	Physical Termination Point	物理终端点
TP	TerminationPoint	终端点
TCA	Threshold Crossed Alert	越门限告警
UTC	Universal Coordinated Time	协调世界时

3 接口规范

3.1 配置管理

3.1.1 EMS 管理模块（module emsMgr）

EMS_T

定义：

```
struct EMS_T
 {
    globaldefs::NamingAttributes_T name;
    string userLabel;
        string nativeEMSName;
        string owner;
        string emsVersion;
        string type;
     globaldefs::NVSList_T additionalInfo;
};
```

说明：

表示 EMS 信息。

属性描述：

globaldefs::NamingAttributes_T name

——表示 EMS 名称，是一个公司内 EMS 的惟一名称。

string userLabel

——表示 EMS 友好名称，缺省是厂家给出的 EMS 名称。

string nativeEMSName

——表示 EMS 的本地名称。

string owner

——表示 EMS 的所有者。

string emsVersion

——表示 EMS 的软件版本。

string type

——表示 EMS 类型，包括“EMS”、“SNMS”、“EMS/SNMS”。

globaldefs::NVSList_T additionalInfo

——表示附加信息，参考取值见表 1。

表 1 附加信息的参考取值

名 称	取 值	说 明
“AlarmStatus”	“Indeterminate”表示未确定； “Critical”表示严重； “Major”表示主要； “Minor”表示次要； “Warning”表示警告； “Cleared”表示清除	告警状态，取当前的最高告警级别

表1（续）

名　　称	取　　值	说　　明
“OperationStatus”	“Available”表示可用； “UnAvailable”：表示不可用	操作状态
“Location”	EMS所在的地理位置	位置信息（精确到机房）

EMSMgr_I接口

说明：

本接口为EMS管理接口，提供对EMS相关信息的管理，本接口继承公共管理部分的Common_I接口。

- 查询EMS信息（getEMS）

定义：

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

说明：

用于NMS向EMS查询EMS信息。

输入参数：

无。

输出参数：

out EMS_T emsInfo

——表示EMS信息。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询所有首层拓扑连接（getAllTopLevelTopologicalLinks）

定义：

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

说明：

用于NMS从EMS查询所有首层拓扑连接。这里的首层拓扑连接指EMS管理的子网间的拓扑连接。

输入参数：

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out topologicalLink::TopologicalLinkList_T topoList

——表示首次查询返回的拓扑连接列表。

out topologicalLink::TopologicalLinkIterator_I topoIt

——表示迭代查询拓扑连接接口。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询所有首层拓扑连接名称（getAllTopLevelTopologicalLinkNames）

定义：

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

说明：

用于 NMS 从 EMS 查询所有首层拓扑连接的名称。

输入参数：

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示首层拓扑连接的名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询名称接口。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询首层拓扑连接（getTopLevelTopologicalLink）

定义：

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

说明：

用于指定名称查询拓扑连接信息。

输入参数：

in globaldefs::NamingAttributes_T name

——表示拓扑连接名称。

输出参数：

out topologicalLink::TopologicalLink_T topoLink

——表示拓扑连接。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询所有当前告警（getAllEMSAndMEActiveAlarms）

定义：

```
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);
};
```

说明：

用于 NMS 从 EMS 查询所有当前告警（包括越门限告警），包括所有网元以及 EMS 本身产生的告警。查询条件包括告警原因和告警级别查询。

输入参数：

in notifications::PerceivedSeverityList_T excludeSeverityList

——表示被排除的告警级别，如果列表是空，表示不排除。

in notifications::ProbableCauseList_T excludeProbCauseList

——表示被排除的告警原因，如果列表是空，表示不排除。

in unsigned long how_many

——表示迭代器第一次返回的数据数目。

输出参数：

out notifications::EventList_T eventList

——表示符合条件的当前告警列表。

out notifications::EventIterator_I eventIt

——用于获得剩余查询数据的事件迭代器操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询所有首层子网（getAllTopLevelSubnetworks）

定义：

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

说明：

用于查询 EMS 内的所有首层子网信息。首层子网指包含物理层且位于子网包含关系的最顶层的子网。

输入参数：

in unsigned long how_many

——表示迭代器第一次返回的数据数目。

输出参数：

out multiLayerSubnetwork::SubnetworkList_T sList

——表示首层子网列表。

out multiLayerSubnetwork::SubnetworkIterator_I sIt

——表示迭代查询子网接口。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询所有首层子网名称（getAllTopLevelSubnetworkNames）

定义：

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

说明：

用于查询 EMS 内的所有首层子网名称。

输入参数：

in unsigned long how_many

——表示迭代器第一次返回的数据数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示首层子网名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询子网名称接口。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 创建告警级别表（createASAP）（可选）

定义：

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

说明：

用于创建告警级别表。

输入参数：

in aSAP::ASAPCreateModifyData_T newASAPCreateData

——表示告警级别表创建数据。

输出参数：

out aSAP::ASAP_T newASAP

——表示创建成功的告警级别表。

out globaldefs::NVSList_T additionalInfo

——表示附加信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）友好名称已存在（EXCPT_USERLABEL_IN_USE）。

（4）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 删除告警级别表（deleteASAP）（可选）

定义：

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

说明：

用于删除告警级别表。

输入参数：

in globaldefs::NamingAttributes_T aSAPName

——表示告警级别表名称。

输出参数：

inout globaldefs::NVSList_T additionalInfo

——表示附加信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 分配告警级别表（assignASAP）（可选）

定义：

```
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);

说明：

用于分配告警级别表。

输入参数：

in globaldefs::NamingAttributes_T aSAPName

——表示告警级别表名称。

in globaldefs::NamingAttributes_T resourceName

——表示告警级别表分配到的资源名称，这里为终端点名称。

in transmissionParameters::LayerRate_T layerRate

——表示告警级别表分配到的资源层速率。

inout globaldefs::NVSList_T additionalInfo

——表示附加信息。

输出参数：

out globaldefs::NVSList_T additionalInfo

——表示附加信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 解除告警级别表分配（deassignASAP）（可选）

定义：

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

说明：

用于解除告警级别表分配。

输入参数：

in globaldefs::NamingAttributes_T aSAPName

——表示告警级别表名称。

in globaldefs::NamingAttributes_T resourceName

——表示告警级别表分配到的资源名称，这里为终端点名称。

in transmissionParameters::LayerRate_T layerRate

——表示告警级别表分配到的资源层速率。

inout globaldefs::NVSList_T additionalInfo

——表示附加信息。

输出参数：

out globaldefs::NVSList_T additionalInfo

——表示附加信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 获取所有告警级别表（getAllASAPs）（可选）

定义：

```
void getAllASAPs(
        in unsigned long how_many,
        out aSAP::ASAPList_T aSAPList,
        out aSAP::ASAPIterator_I asapIt)
      raises(globaldefs::ProcessingFailureException);
```

说明：

用于获取所有告警级别表。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out aSAP::ASAPList_T aSAPList

——表示告警级别表列表。

out aSAP::ASAPIterator_I asapIt

——表示迭代获取告警级别表接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 获取所有告警级别表名称（getAllASAPNames）（可选）

定义：

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

说明：

用于获取所有告警级别表名称。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList t

——表示告警级别表列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代获取告警级别表名称接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 获取告警级别表（getASAP）（可选）

定义：

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

说明：

用于指定名称获取告警级别表。

输入参数：

in globaldefs::NamingAttributes_T aSAPName

——表示告警级别表名称。

输出参数：

out aSAP::ASAP_T aSAP

——表示告警级别表。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

3.1.2 拓扑连接模块（module topologicalLink）

TopologicalLink_T

定义：

```
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;
 };
```

说明：

表示拓扑连接信息。

属性描述：

globaldefs::NamingAttributes_T name

——表示拓扑连接名称。

string userLabel

——表示友好名称。

string nativeEMSName

——表示拓扑连接在 EMS 的本地名称。

string owner

——表示拓扑连接的所有者。

globaldefs::ConnectionDirection_T direction

——表示拓扑连接的方向。

transmissionParameters::LayerRate_T rate

——表示拓扑连接的层速率。

globaldefs::NamingAttributes_T aEndTP

——表示拓扑连接的 A 端点。

globaldefs::NamingAttributes_T zEndTP

——表示拓扑连接的 Z 端点。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

TopologicalLinkList_T

定义：

typedef sequence<TopologicalLink_T> TopologicalLinkList_T;

说明：

表示拓扑连接列表。

TopologicalLinkIterator_I 接口

说明：

表示迭代查询拓扑连接接口。

- 查询下一批拓扑连接数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out TopologicalLinkList_T     topoLinkList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批拓扑连接数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out TopologicalLinkList_T topoLinkList

——表示查询得到的拓扑连接列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength)

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy)

定义：

```
void destroy ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.1.3 网元模块（module managedElement）

CommunicationState_T

定义：

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

说明：

表示 EMS 与网元之间的通讯状态。

属性描述：

CS_AVAILABLE

——表示通讯正常。

CS_UNAVAILABLE

——表示通讯失效。

ManagedElement_T

定义：

```
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;
 };
```

说明：

表示 EMS 的一个基本管理单位，即一个网元。

属性描述：

globaldefs::NamingAttributes_T name

——表示网元名称。

string userLabel

——表示网元友好名称。

string nativeEMSName

——表示网元在 EMS 本地的名称。

string owner

——表示网元的所有者名称。

string location

——表示网元所在的物理位置。

string version

——表示网元的版本。

string productName

——表示产品名称，厂家对这个产品的命名。

CommunicationState_T communicationState

——表示连接状态，EMS 与网元的连接状态。

boolean emsInSyncState

——表示 EMS 能够保持 EMS 数据与网元数据是同步的，并且能够发送所有适当的的通知。当 EMS 将此属性设置为假时，表示 EMS 需要与网元进行数据同步并且不能产生适当的通知（如对象创建、删除、属性改变通知）；当 EMS 将此属性设置回到真表示数据重新同步结束同时通知可以在适当的时候产生。

transmissionParameters::LayerRateList_T supportedRates

——表示网元支持的交叉连接速率。

globaldefs::NVSList_T additionalInfo

——表示附加信息，包括告警状态、操作状态、位置信息，参见 EMS_T 的定义。

ManagedElementList_T

定义：

typedef sequence <ManagedElement_T> ManagedElementList_T;

说明：

表示网元信息列表。

ManagedElementIterator_I 接口

说明：

表示迭代查询网元接口。

- 查询下一批网元数据（next_n）

定义：

boolean next_n (

```
in unsigned long how_many,
out ManagedElementList_T    meList
)
raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批网元数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out ManagedElementList_T　　meList

——表示查询得到的网元列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength)

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy)

定义：

```
void destroy ()
```

raises (globaldefs::ProcessingFailureException);

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.1.4 网元管理模块（module managedElementManager）

ManagedElementMgr_I 接口

说明：

网元管理接口，用于查询和修改网元信息，本接口继承公共管理部分的 Common_I 接口。

- 查询所有网元信息（getAllManagedElements）

定义：

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

说明：

用于 NMS 从 EMS 查询所有网元。

输入参数：

in unsigned long how_many

——表示首次迭代查询的数目。

输出参数：

out managedElement::ManagedElementList_T meList

——表示网元信息列表。

out managedElement::ManagedElementIterator_I meIt

——表示迭代查询网元接口。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询所有网元名称（getAllManagedElementNames）

定义：

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

说明：

用于 NMS 从 EMS 查询所有网元名称。

输入参数：

in unsigned long how_many

——表示首次迭代查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示网元名称的列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询名称接口。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询网元（getManagedElement）

定义：

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

说明：

用于 NMS 从 EMS 查询指定网元名称的网元信息。

输入参数：

in globaldefs::NamingAttributes_T managedElementName

——表示网元名称。

输出参数：

out managedElement::ManagedElement_T me

——表示网元信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询所有物理终端点（getAllPTPs）

定义：

```
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);
```

说明：

用于 NMS 查询网元内的所有物理终端点或浮动终端点（FTP）。

输入参数：

in globaldefs::NamingAttributes_T managedElementName

——表示网元名称。

in transmissionParameters::LayerRateList_T tpLayerRateList

——表示要求满足的PTP速率等级列表。查询结果中的PTP的层速率至少要符合列表中的一项，如果是空，则没有限制。

in transmissionParameters::LayerRateList_T connectionLayerRateList

——表示要求满足的连接层速率等级列表。查询出的终端点的连接层速率至少要符合列表中的一项，如果是空，则没有限制。

in unsigned long how_many,

——表示首次迭代查询的数目。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示查询获得的终端点列表。

out terminationPoint::TerminationPointIterator_I tpIt

——表示迭代查询终端点接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询所有物理终端点名称（getAllPTPNames）

定义：

```
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);
```


说明：

用于 NMS 查询网元内的所有物理终端点名称。

输入参数：

in globaldefs::NamingAttributes_T managedElementName

——表示网元名称。

in transmissionParameters::LayerRateList_T tpLayerRateList

——表示要求满足的 PTP 层速率等级列表。查询结果中的 PTP 的层速率至少要符合列表中的一项，如果是空，则没有限制。

in transmissionParameters::LayerRateList_T connectionLayerRateList

——表示要求满足的 PTP 连接层速率等级列表。查询结果中的 PTP 的连接层速率至少要符合列表中的一项，如果是空，则没有限制。

in unsigned long how_many

——首次迭代查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——查询获得的终端点名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——迭代查询名称接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询终端点（getTP）

定义：

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

说明：

用于指定名称查询终端点信息。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

输出参数：

out terminationPoint::TerminationPoint_T tp

——表示终端点信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

- 设置终端点（setTPData）

定义：

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

说明：

用于设置终端点信息。可以设置的信息：是否可配置交叉连接以及是否可映射下一层类型；终端点的传送参数。

输入参数：

in subnetworkConnection::TPData_T tpInfo

——表示终端点名称及待修改信息。

输出参数：

out terminationPoint::TerminationPoint_T modifiedTP

——表示修改后的终端点信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

- 查询终端点潜在包含的终端点（getContainedPotentialTPs）

定义：

```
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);
```

说明：

用于查询终端点包含的潜在的（可能的映射方式下）所有的终端点信息，这些终端点包括本终端点直接和间接包含的终端点。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

in transmissionParameters::LayerRateList_T layerRateList

——表示层速率列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示终端点列表。

out terminationPoint::TerminationPointIterator_I tpIt

——表示迭代查询终端点操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（6）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询终端点潜在包含的终端点名称（getContainedPotentialTPNames）

定义：

```
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);
```

说明：

用于查询终端点包含的潜在的（可能的映射方式下）所有的终端点名称信息，这些终端点名称包括本终端点直接和间接包含的终端点的名称。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

in transmissionParameters::LayerRateList_T layerRateList

——表示层速率列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示终端点名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询名称操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（6）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询终端点包含的正在使用的终端点（getContainedInUseTPs）

定义：

```
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);
```


说明：

用于查询终端点包含的正在使用的终端点信息，返回终端点包含正在被 SNC 使用或被终结、被映射的终端点信息。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

in transmissionParameters::LayerRateList_T layerRateList

——表示层速率列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示终端点列表。

out terminationPoint::TerminationPointIterator_I tpIt

——表示迭代查询终端点操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（6）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询终端点包含的正在使用的终端点名称（getContainedInUseTPNames）

定义：

```
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);
```

说明：

用于查询终端点包含的正在使用的终端点名称，返回终端点包含正在被 SNC 使用或被终结、被映射的终端点名称。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

in transmissionParameters::LayerRateList_T layerRateList

——表示层速率列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示终端点名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询名称操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（6）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询终端点包含的可交叉或已交叉的终端点（getContainedCurrentTPs）

定义：

```
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);
```

说明：

用于查询终端点包含的业务可以流通的终端点信息，返回终端点包含可以配置交叉连接或已经配置了交叉连接的终端点信息。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

in transmissionParameters::LayerRateList_T layerRateList

——表示层速率列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示终端点列表。

out terminationPoint::TerminationPointIterator_I tpIt

——表示迭代查询终端点操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（6）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询终端点包含的可交叉或已交叉的终端点名称（getContainedInUseTPNames）

定义：

```
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);
```

说明：

用于查询终端点包含的业务流通的终端点名称，返回终端点包含可以配置交叉连接或已经配置了交叉连接的终端点名称。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

in transmissionParameters::LayerRateList_T layerRateList

——表示层速率列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示终端点名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询名称操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（6）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询包含本终端点的终端点（getContainingTPs）

定义：

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

说明：

用于查询包含本终端点的终端点信息，返回包含本终端点的物理终端点或连接终端点。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示终端点列表。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询包含本终端点的终端点名称（getContainingTPNames）

定义：

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

说明：

用于查询包含本终端点的终端点名称，返回包含本终端点的物理终端点或连接终端点名称。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示终端点列表。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询所有的交叉连接（getAllCrossConnections）

定义：

```
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 ccIt)
      raises(globaldefs::ProcessingFailureException);
```

说明：

用于指定网元名称，获取全部或指定连接速率的交叉连接。

输入参数：

in globaldefs::NamingAttributes_T managedElementName

——表示网元名称。

in transmissionParameters::LayerRateList_T connectionRateList

——表示连接速率列表。如果列表为空，表示全部连接速率（为空表示全部连接速率方式可选）。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out subnetworkConnection::CrossConnectList_T ccList

——表示符合条件的交叉连接列表。

out subnetworkConnection::CCIterator_I ccIt

——表示迭代查询交叉连接接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询所有当前告警（getAllActiveAlarms）

定义：

```
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);
```

说明：

用于 NMS 从 EMS 查询指定网元的所有当前告警，包括越门限告警。可以按照告警原因和告警级别查询。

输入参数：

in globaldefs::NamingAttributes_T meName

——表示网元名称。

in notifications::ProbableCauseList_T excludeProbCauseList

——表示排除的告警原因列表，如果列表是空，表示不排除。

in notifications::PerceivedSeverityList_T excludeSeverityList

——表示排除的告警级别列表，如果列表是空，表示不排除。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out notifications::EventList_T eventList

——表示符合条件的当前告警列表。

out notifications::EventIterator_I eventIt

——表示迭代查询事件接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

3.1.5 设备模块（module equipment）

设备管理功能定义了网元内的设备和相互关系及相关操作。这里的设备指单元盘，设备容器指的是机架、子架（子子架）、槽位（子槽位）。

HolderState_T

定义：

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

说明：

表示设备容器的当前状态。

属性描述：

EMPTY

——表示空状态（网管未配置设备）。

INSTALLED_AND_EXPECTED

——表示安装设备且类型符合。

EXPECTED_AND_NOT_INSTALLED

——表示期待安装设备但未安装（网管已配置了设备）。

INSTALLED_AND_NOT_EXPECTED

——表示安装设备但并未期待（网管未安装）。

MISMATCH_OF_INSTALLED_AND_EXPECTED

——表示安装设备但类型不符合。

UNAVAILABLE

——表示不可用。

UNKNOWN

——表示未知状态。

ServiceState_T

定义：

```
enum ServiceState_T
```

{
IN_SERVICE,
OUT_OF_SERVICE,
OUT_OF_SERVICE_BY_MAINTENANCE,
SERV_NA
};

说明：

表示设备或容器的状态。

属性描述：

IN_SERVICE

——表示单元盘已经插入到槽位中，并已经根据网管系统的配置方式投入运行。

OUT_OF_SERVICE

——表示该单元盘已经不能运行，即不能根据网管系统的配置完成指定的功能，而且不能执行相应的管理动作。

OUT_OF_SERVICE_BY_MAINTENANCE

——表示该单元盘由于维护的目的而处于非正常工作状态。

SERV_NA

——未知，即单元盘的状态不明确。

EquipmentObjectType_T

定义：

typedef string EquipmentObjectType_T;

说明：

表示设备类型，取值由厂家决定。

EquipmentObjectTypeList_T

定义：

typedef sequence <EquipmentObjectType_T> EquipmentObjectTypeList_T;

说明：

表示设备类型列表。

EquipmentHolderType_T

定义：

typedef string EquipmentHolderType_T;

说明：

设备容器类型，其中：

“rack”：表示机架；

“shelf”：表示子架；

“sub_shelf”：表示子子架；

“slot”：表示槽位；

“sub_slot”：表示子槽位。

Equipment_T

定义：

```
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;
};
```

说明：

表示设备，即单元盘。

属性描述：

globaldefs::NamingAttributes_T name

——表示设备的名称。

string userLabel

——表示设备的友好名称。

string nativeEMSName

——表示设备的 EMS 本地名称。

string owner

——表示设备的所有者。

boolean alarmReportingIndicator

——表示设备的告警上报是否被激活，为真表示设备的告警上报处于激活状态，为假表示处于禁止状态。

ServiceState_T serviceState

——表示设备的状态。

EquipmentObjectType_T expectedEquipmentObjectType

——表示期待的设备类型，如果没有可填为空字符串。

EquipmentObjectType_T installedEquipmentObjectType

——表示已安装的设备类型，如果没有可填为空字符串。

string installedPartNumber

——表示安装设备的零件编号，如果没有可填为空字符串。

string installedVersion

——表示安装设备的版本，包括软件版本和硬件版本，如果没有可填为空字符串。

string installedSerialNumber

——表示安装设备的序列号，如果没有可填为空字符串。

ServiceState_T serviceState

——表示设备当前的工作状态。

globaldefs::NVSList_T additionalInfo

——表示附加信息，包括告警状态、操作状态、位置信息，参见 EMS_T 的定义。

EquipmentHolder_T

定义：

```
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;
    equipment::HolderState_T holderState;
    globaldefs::NVSList_T     additionalInfo;
};
```

说明：

表示设备容器，即机架、子架（子子架）、槽位（子槽位）信息。

属性描述：

globaldefs::NamingAttributes_T name

——表示设备容器的名称。

string userLabel

——表示设备容器的友好名称。

string nativeEMSName

——表示设备容器的 EMS 本地名称。

string owner

——表示设备容器的所有者。

boolean alarmReportingIndicator

——表示设备容器的告警上报是否被激活，为真表示设备容器的告警上报处于激活状态，为假表示处于禁止状态。

EquipmentHolderType_T holderType

——表示设备容器类型，包括机架、子架（子子架）、槽位（子槽位）。

globaldefs::NamingAttributes_T expectedOrInstalledEquipment

——表示应安板名称，或者已安板名称。如果是空字符串，表示无应安板。如果非空，当没有安装板，或者安装板不匹配时，上报板不在位或者板不匹配告警。

EquipmentObjectTypeList_T acceptableEquipmentTypeList

——表示可以接受的子设备或者设备容器列表。如果是槽位和子槽位则是必须的，表示槽位可以安装的设备。

equipment::HolderState_T holderState

——表示设备容器当前的状态。

globaldefs::NVSList_T additionalInfo

——表示附加信息，包括告警状态、操作状态、位置信息，参见 EMS_T 的定义。

EquipmentTypeQualifier_T

定义：

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

说明：

设备区分器，在设备与设备容器的联合类型中用于区分设备和设备容器。

属性描述：

EQT

——表示设备。

EQT_HOLDER

——表示设备容器。

EquipmentOrHolder_T

定义：

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

说明：

表示设备容器与设备的联合结构。

属性描述：

case EQT: Equipment_T equip

——表示设备。

case EQT_HOLDER: EquipmentHolder_T holder

——表示设备容器。

EquipmentOrHolderList_T

定义：

typedef sequence <EquipmentOrHolder_T> EquipmentOrHolderList_T;

说明：

表示设备容器与设备的联合结构列表。

EquipmentOrHolderIterator_I 接口

说明：

用于定义查询设备和设备容器的迭代器。

- 迭代查询设备和设备容器（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out EquipmentOrHolderList_T equipmentOrHolderList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批数据。

输入参数：

in unsigned long how_many,

——表示返回数据的最大数目。

输出参数：

out EquipmentOrHolderList_T equipmentOrHolderList

——表示查询得到的列表。

返回值：

boolean

——表示操作结果。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength）

定义：

unsigned long getLength ()

raises (globaldefs::ProcessingFailureException);

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

表示删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

EquipmentInventoryMgr_I 接口

说明：

定义查询设备和设备容器的操作，本接口继承公共管理部分的 Common_I 接口。

- 查询直接包含的设备(getContainedEquipment）

定义：

```
void getContainedEquipment (
    in globaldefs::NamingAttributes_T equipmentHolderName,
    out EquipmentOrHolderList_T equipmentOrHolderList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于 NMS 查询设备容器内的直接包含的设备和设备容器。

输入参数：

in globaldefs::NamingAttributes_T equipmentHolderName

——表示设备容器名称。

输出参数：

out EquipmentOrHolderList_T equipmentOrHolderList

——表示此设备容器直接包含的设备和设备容器列表。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询所有的设备(getAllEquipment）

定义：

```
void getAllEquipment (
    in   globaldefs::NamingAttributes_T   meOrHolderName,
    in   unsigned long   how_many,
    out EquipmentOrHolderList_T eqList,
    out EquipmentOrHolderIterator_I eqIt
    )
```

raises (globaldefs::ProcessingFailureException);

说明：

用于 NMS 查询网元或设备容器内的所有设备或设备容器。

输入参数：

in globaldefs::NamingAttributes_T meOrHolderName

——表示网元或设备容器名称。

in unsigned long how_many

——表示首次迭代查询的数目。

输出参数：

out EquipmentOrHolderList_T eqList

——表示所有设备和设备容器列表。

out EquipmentOrHolderIterator_I eqIt

——表示迭代查询设备容器接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询设备(getEquipment)

定义：

void getEquipment (

in globaldefs::NamingAttributes_T equipmentHolderName,

out EquipmentOrHolder_T equip

)

raises (globaldefs::ProcessingFailureException);

说明：

用于 NMS 查询指定名称的设备或设备容器。

输入参数：

in globaldefs::NamingAttributes_T equipmentHolderName

——表示设备容器名称。

输出参数：

out EquipmentOrHolder_T equip

——表示设备或设备容器信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 开启告警上报(setAlarmReportOn）

定义：

```
void setAlarmReportingOn(
        in globaldefs::NamingAttributes_T equipmentOrHolderName)
    raises(globaldefs::ProcessingFailureException);
```

说明：

用于开启设备或设备容器的告警上报功能。这里设置的告警仅为设备或设备容器自身产生的告警，对终端点的告警不起作用。

输入参数：

in globaldefs::NamingAttributes_T equipmentHolderName

——表示设备容器名称。

输出参数：

无。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

- 关闭告警上报(setAlarmReportOff）

定义：

```
void setAlarmReportingOff(
        in globaldefs::NamingAttributes_T equipmentOrHolderName)
    raises(globaldefs::ProcessingFailureException);
```

说明：

用于关闭设备或设备容器的告警上报功能。这里设置的告警仅为设备或设备容器自身产生的告

警，对终端点的告警不起作用。

输入参数：

in globaldefs::NamingAttributes_T equipmentHolderName

——表示设备容器名称。

输出参数：

无。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

3.1.6 终端点模块（module terminationPoint）

Directionality_T

定义：

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

说明：

表示终端点的方向。

属性描述：

D_NA

——表示未知方向。

D_BIDIRECTIONAL

——表示双向(包括源和宿)。

D_SOURCE

——表示源方向(用于发送信号)。

D_SINK

——表示宿方向(用于接收信号)。

TPConnectionState_T

定义：

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

说明：

表示终端点的连接状态。

属性描述：

TPCS_NA

——表示连接状态未知。

TPCS_SOURCE_CONNECTED

——表示终端点的 SOURCE 部分参与了交叉连接(即信号流出，包括并发的情况)，只能应用于方向为 D_BIDIRECTIONAL 和 D_SOURCE 的终端点。

TPCS_SINK_CONNECTED

——表示终端点的 SINK 部分参与了交叉连接(即信号流入，包括优收的情况)，只能应用于方向为 D_BIDIRECTIONAL 和 D_SINK 的终端点。

TPCS_BI_CONNECTED

——表示终端点的 SINK 和 SOURCE 都参与了交叉连接，只能应用于方向值为 D_BIDIRECTIONAL 的终端点。

TPCS_NOT_CONNECTED

——表示未连接。

TPType_T

定义：

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

说明：

表示终端点的类型。

属性描述：

TPT_PTP

——表示物理终端点。

TPT_CTP

——表示连接终端点。

TPT_TPPool

——表示终端点池。

TerminationMode_T

定义：

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

说明：

表示终端点是否可配置交叉连接以及是否可映射下一层类型。

属性描述：

TM_NA

——表示未知。

TM_NEITHER_TERMINATED_NOR_AVAILABLE_FOR_MAPPING

——表示终端点可以配置交叉连接，但终端点不能进一步展开使用，即终端点已通道化。

TM_TERMINATED_AND_AVAILABLE_FOR_MAPPING

——表示终端点的下层终端点可以使用，终端点本身不能配置交叉连接，即终端点没有通道化。

TPProtectionAssociation_T

定义：

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

说明：

表示终端点与保护的相关类型。

属性描述：

TPPA_NA

——表示未知。

TPPA_PSR_RELATED

——表示与保护相关，即存在与终端点具有保护关系的终端点。

TerminationPoint_T

定义：

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

说明：

表示终端点信息。

属性描述：

globaldefs::NamingAttributes_T name

——表示终端点名称。

string userLabel

——表示终端点的友好名称。

string nativeEMSName

——表示终端点的 EMS 本地名称。

string owner

——表示终端点的所有者。

globaldefs::NamingAttributes_T ingressTrafficDescriptorName

——表示入业务描述符名称。

globaldefs::NamingAttributes_T engressTrafficDescriptorName

——表示出业务描述符名称。

TPType_T type

——表示终端点的类型。

TPConnectionState_T connectionState

——表示终端点的连接状态。

terminationPoint::TerminationMode_T tpMappingMode

——表示终端点是否可配置交叉连接以及是否可映射下一层类型。

Directionality_T direction

——表示终端点的方向。

transmissionParameters::LayeredParameterList_T transmissionParams

——表示终端点的层速率及传送参数列表。

TPProtectionAssociation_T tpProtectionAssociation

——表示终端点与保护相关的类型。

boolean edgePoint

——表示本终端点是否是子网内部拓扑连接的终结点。

globaldefs::NVList_T additionalInfo

——表示附加信息，包括告警状态、操作状态、位置信息，参见 EMS_T 的定义。

TerminationPointList_T

定义：

typedef sequence <TerminationPoint_T> TerminationPointList_T;

说明：

TerminationPointList_T 表示终端点列表。

TerminationPointIterator_I 接口

说明：

TerminationPointIterator_I 表示迭代查询终端点接口。

- 迭代查询终端点信息操作（next_n）

定义：

```
boolean next_n (
        in unsigned long how_many,
        out TerminationPointList_T tpList
        )
        raises (globaldefs::ProcessingFailureException);
```

说明：

迭代查询终端点信息。

输入参数：

in unsigned long how_many

——表示本次查询数据的数目。

输出参数：

out TerminationPointList_T tpList

——表示终端点数据列表。

返回值：

boolean

——表示操作结果，为真表示操作成功；为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目（getLength）

定义：

unsigned long getLength ()

raises (globaldefs::ProcessingFailureException);

说明：

查询迭代器中总的数据数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器（destroy）

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

释放迭代器，释放被迭代器占用的内存。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.1.7 保护模块（module protection）

ProtectionSchemeState_T

定义：

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

说明：

表示保护方案的状态。

属性描述：

PSS_UNKNOWN

——表示当前状态未知。

PSS_ AUTOMATIC

——表示当前处于自动保护倒换状态。

PSS_FORCED_OR_LOCKED_OUT

——表示当前处于强制倒换或保护闭锁状态。

ProtectionCommand_T

定义：

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

说明：

表示执行的保护倒换命令类型。

属性描述：

PC_CLEAR

——表示清除倒换。

PC_LOCKOUT

——表示锁定倒换。

PC_FORCED_SWITCH

——表示强制倒换。

PC_MANUAL_SWITCH

——表示人工倒换。

PC_EXERCISER

——表示练习倒换。

ProtectionGroup_T

定义：

```
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;
};
```

说明：

表示保护组信息。

属性描述：

globaldefs::NamingAttributes_T name

——表示保护组名称，在网元内惟一。

string userLabel

——表示保护组的友好名称。

string nativeEMSName

——表示保护组的 EMS 本地名称。

string owner;

——表示保护组的所有者。

ProtectionGroupType_T protectionGroupType;

——表示保护组类型。

ProtectionSchemeState_T protectionSchemeState;

——表示保护方案状态。

ReversionMode_T reversionMode;

——表示恢复方式。

transmissionParameters::LayerRate_T rate;

——表示层速率。

globaldefs::NamingAttributesList_T pgpTPList;

——表示组成保护组的终端点列表，这些终端点是部分排序的：保护终端点总是排在工作终端点后面；东向的终端点总是连续排列，西向的终端点也同样。

globaldefs::NVSList_T pgpParameters;

——表示保护组参数。

globaldefs::NVSList_T additionalInfo;

——表示附加信息。

ProtectionGroupList_T

定义：

typedef sequence <ProtectionGroup_T> ProtectionGroupList_T;

说明：

表示保护组列表。

EProtectionGroup_T

定义：

```
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;
};

说明：

表示设备保护组信息。

属性描述：

globaldefs::NamingAttributes_T name

—— 表示设备保护组名称，在网元内惟一。

string userLabel

—— 表示设备保护组的友好名称。

string nativeEMSName

—— 表示设备保护组的 EMS 本地名称。

string owner

—— 表示设备保护组的所有者。

EProtectionGroupType_T eProtectionGroupType

—— 表示设备保护组类型。

ProtectionSchemeState_T protectionSchemeState

——表示保护方案状态。

ReversionMode_T reversionMode

——表示恢复方式。

globaldefs::NamingAttributesList_T protectedList

——表示被保护设备名称列表。

globaldefs::NamingAttributesList_T protectingList

——表示保护设备名称列表。

globaldefs::NVSList_T ePgpParameters

——表示设备保护组参数。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

EProtectionGroupList_T

定义：

typedef sequence <EProtectionGroup_T> EProtectionGroupList_T;

说明：

表示设备保护组列表。

ProtectionGroupType_T

定义：

typedef string ProtectionGroupType_T;

说明：

表示保护组的类型。有效的取值是：

“MSP_1_PLUS_1”：表示复用段 1+1 保护。

“MSP_1_FOR_N”：表示复用段 1:*N* 保护。

“2_FIBER_BLSR”：表示二纤共享环保护。

“4_FIBER_BLSR”：表示四纤共享环保护。

EProtectionGroupType_T

定义：

typedef string EProtectionGroupType_T;

说明：

表示设备保护组的类型。

取值有：

"M_FOR_N"：表示 *M* 个设备保护 *N* 个设备，*M* 个设备本身带有业务。

“1_PLUS_1”：表示 1＋1 保护，保护设备只用于保护，本身不带业务。

ProtectionType_T

定义：

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

说明：

用于名称一个保护倒换是非子网连接保护还是子网连接保护倒换。

属性描述：

PT_MSP_APS

——复用段保护。

PT_SNCP

——子网连接保护。

ReversionMode_T

定义：

enum ReversionMode_T

```
    {
      RM_UNKNOWN,
      RM_NON_REVERTIVE,
      RM_REVERTIVE
  };
```

说明：

保护倒换恢复方式。

属性描述：

RM_UNKNOWN

——表示未知。

RM_NON_REVERTIVE

——表示不恢复。

RM_REVERTIVE

——表示恢复。

SwitchData_T

定义：

```
  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;
  };
```

说明：

表示保护倒换数据。

属性描述：

ProtectionType_T protectionType

——表示保护类型。

SwitchReason_T switchReason

——表示倒换原因。

transmissionParameters::LayerRate_T layerRate

——表示层速率。

globaldefs::NamingAttributes_T groupName

——表示保护组名称。

globaldefs::NamingAttributes_T protectedTP

——表示被保护的终端点。

globaldefs::NamingAttributes_T switchToTP

——表示倒换到的终端点。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

SwitchDataList_T

定义：

typedef sequence <SwitchData_T> SwitchDataList_T;

说明：

表示倒换数据列表。

ESwitchData_T

定义：

```
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;
 };
```

说明：

表示设备保护倒换数据。

属性描述：

EProtectionGroupType_T eProtectionGroupType

——表示设备保护组类型。

ESwitchReason_T eSwitchReason

——表示设备倒换原因。

globaldefs::NamingAttributes_T ePGPName

——表示设备保护组名称。

globaldefs::NamingAttributes_T protectedE

——表示被保护的设备。

globaldefs::NamingAttributes_T switchToE

——表示倒换到的设备。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

ESwitchDataList_T

定义：

typedef sequence <ESwitchData_T> ESwitchDataList_T;

说明：

表示设备倒换数据列表。

SwitchReason_T

定义：

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

说明：

表示倒换原因。

属性描述：

SR_NA

——表示未知原因。

SR_RESTORED

——表示倒换恢复。

SR_SIGNAL_FAIL

——表示信号丢失。

SR_SIGNAL_MISMATCH

——表示信号失配。

SR_SIGNAL_DEGRADE

——表示信号劣化。

SR_AUTOMATIC_SWITCH

——表示自动倒换。

SR_MANUAL

——表示人工倒换。

ESwitchReason_T

定义：

typedef string ESwitchReason_T;

说明：

表示设备倒换的原因。取值有：

"SR_NA"：表示未知原因。

"SR_E_FAILURE"：表示设备失效。

"SR_MANUAL"：表示人工倒换。

ProtectionGroupIterator_I 接口

说明：

表示迭代查询保护组接口。

- 查询下一批保护组数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out ProtectionGroupList_T pgpList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批保护组数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out ProtectionGroupList_T pgpList

——表示查询得到的保护组列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength）

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

```
void destroy ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

EProtectionGroupIterator_I 接口

说明：

表示迭代查询设备保护组接口。

- 查询下一批保护组数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out EProtectionGroupList_T ePGPList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批设备保护组数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out EProtectionGroupList_T ePGPList

——表示查询得到的设备保护组列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength)

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy)

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

ProtectionMgr_I 接口

说明：

ProtectionMgr_I 接口用于提供与保护功能有关的操作，本接口继承公共管理部分的 Common_I 接口。

- 查询所有保护组信息（getAllProtectionGroups）

定义：

void getAllProtectionGroups(

in globaldefs::NamingAttributes_T meName,

in unsigned long how_many,

out ProtectionGroupList_T pgList,

out ProtectionGroupIterator_I pgpIt)

raises(globaldefs::ProcessingFailureException);

说明：

查询网元的所有保护组信息。

输入参数：

in globaldefs::NamingAttributes_T meName

——表示网元名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out ProtectionGroupList_T pgList

——表示保护组列表。

out ProtectionGroupIterator_I pgpIt

——表示迭代查询保护组接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询保护组信息（getProtectionGroup）

定义：

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

说明：

指定名称查询保护组信息。

输入参数：

in　globaldefs::NamingAttributes_T pgName

——表示保护组名称。

输出参数：

out protection::ProtectionGroup_T　protectionGroup

——表示保护信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询保护倒换状态（retrieveSwitchData）

定义：

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

说明：

用于查询保护组的倒换状态。

输入参数：

in globaldefs::NamingAttributes_T reliableSinkCtpOrGroupName

——表示宿端点或保护组名称。当查询子网连接保护的倒换状态时，返回倒换数据中的保护组名称为空。

输出参数：

out protection::SwitchDataList_T switchData

——表示保护倒换数据列表。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询所有设备保护组信息（getAllEProtectionGroups）

定义：

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

说明：

查询网元的所有设备保护组信息。

输入参数：

in globaldefs::NamingAttributes_T meName

——表示网元名称。

in unsigned long how_many,

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out EProtectionGroupList_T epgpList

——表示设备保护组列表。

out EProtectionGroupIterator_I epgpIt

——表示迭代查询设备保护组接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询设备保护组信息（getEProtectionGroup）

定义：

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

说明：

指定名称查询设备保护组信息。

输入参数：

in globaldefs::NamingAttributes_T ePGPname

——表示设备保护组名称。

输出参数：

out protection::EProtectionGroup_T eProtectionGroup

——表示设备保护信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询设备保护倒换状态（retrieveESwitchData）

定义：

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

说明：

用于查询设备保护组的倒换状态。

输入参数：

in globaldefs::NamingAttributes_T ePGPName

——表示设备保护组名称。

输出参数：

out protection::ESwitchDataList_T eSwitchDataList

——表示设备保护倒换数据列表。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 保护倒换操作（performProtectionCommand）

定义：

```
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);
```


说明：

用于执行保护倒换操作，执行的对象是保护类型为 MSP 或 SNCP 的保护组。

输入参数：

in ProtectionCommand_T protectionCommand

——表示保护倒换命令。

in globaldefs::NamingAttributes_T reliableSinkCtpOrGroupName

——表示宿端点或保护组名称。

in globaldefs::NamingAttributes_T fromTp

——表示业务被倒换前所在的终端点名称。

in globaldefs::NamingAttributes_T toTp

——表示业务倒换后所在的终端点名称。

输出参数：

out protection::SwitchData_T switchData

——表示执行保护倒换后的倒换状态信息。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

3.1.8 子网模块（module multiLayerSubnetwork）

Topology_T

定义：

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

说明：

表示子网的类型。

属性描述：

TOPO_SINGLETON

——表示简单节点。

TOPO_CHAIN

——表示线型。

TOPO_PSR

——表示通道倒换环。

TOPO_OPEN_PSR

——开放的通道倒换环

TOPO_SPRING

——共享环。

TOPO_OPEN_SPRING

——开放的共享环。

TOPO_MESH

——网孔型。

EMSFreedomLevel_T

定义：

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

说明：

表示 NMS 指定的 EMS 进行 SNC 操作时的自由度等级。

属性描述：

EMSFL_CC_AT_SNC_LAYER

——表示 EMS 可以创建或删除可被和已被 SNC 使用的交叉连接。

EMSFL_TERMINATE_AND_MAP

——表示在 EMSFL_CC_AT_SNC_LAYER 基础上，EMS 可以映射去映射或通道化去通道化可被和已被 SNC 使用的终端点。

EMSFL_HIGHER_ORDER_SNCS

——表示在 EMSFL_TERMINATE_AND_MAP 基础上，EMS 可以创建或删除可被或已被用作承载 SNC 的高阶交叉连接。

EMSFL_RECONFIGURATION

——表示 EMS 可以进行任何操作，即 NMS 不限制 EMS 进行与 SNC 相关的操作。

MultiLayerSubnetwork_T

定义：

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

说明：

表示子网，子网是 SDH 传送网拓扑中的一个单元。

属性描述：

globaldefs::NamingAttributes_T name

——表示子网名称。

string userLabel

——表示子网的友好名称。

string nativeEMSName

——表示子网的本地名称。

string owner

——表示子网的所有者。

Topology_T subnetworkType

——表示子网类型。

transmissionParameters::LayerRateList_T supportedRates

——表示子网支持的可以创建交叉连接的速率。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

SubnetworkList_T

定义：

typedef sequence<MultiLayerSubnetwork_T> SubnetworkList_T;

说明：

表示子网列表。

SubnetworkIterator_I 接口

说明：

表示迭代查询子网接口。

- 查询下一批子网数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out SubnetworkList_T subnetworkList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批子网数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out SubnetworkList_T subnetworkList

——表示子网列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength）

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

```
void destroy ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

MultiLayerSubnetworkMgr_I 接口

说明：

MultiLayerSubnetworkMgr_I 接口用于提供子网相关的操作。

- 查询子网包含的所有网元（getAllManagedElements）

定义：

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

说明：

用于查询子网包含的所有网元。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out managedElement::ManagedElementList_T meList

——表示网元列表。

out managedElement::ManagedElementIterator_I meIt

——表示迭代查询网元接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询子网包含的所有网元名称（getAllManagedElementNames）

定义：

```
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);
```

说明：

用于查询子网包含的所有网元名称。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示网元名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询网元名称接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询子网（getMultiLayerSubnetwork）

定义：

```
void getMultiLayerSubnetwork(
        in globaldefs::NamingAttributes_T subnetName,
        out MultiLayerSubnetwork_T subnetwork)
      raises(globaldefs::ProcessingFailureException);
```

说明：

用于指定子网名称查询子网。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

输出参数：

out MultiLayerSubnetwork_T subnetwork

——表示子网信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

- 查询子网包含的所有拓扑连接（getAllTopologicalLinks）

定义：

```
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);
```

说明：

用于查询子网包含的所有拓扑连接。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out topologicalLink::TopologicalLinkList_T topoList

——表示拓扑连接列表。

out topologicalLink::TopologicalLinkIterator_I topoIt

——表示迭代查询拓扑连接接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询子网包含的所有拓扑连接名称（getAllTopologicalLinkNames）

定义：

```
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);
```

说明：

用于查询子网包含的所有拓扑连接名称。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示拓扑连接名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询拓扑连接名称接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询拓扑连接（getTopologicalLink）

定义：

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

说明：

用于指定拓扑连接名称查询拓扑连接。

输入参数：

in globaldefs::NamingAttributes_T topoLinkName

——表示拓扑连接名称。

输出参数：

out topologicalLink::TopologicalLink_T topoLink

——表示拓扑连接信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

- 查询子网包含的所有子网连接（getAllSubnetworkConnections）

定义：

```
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);
```

说明：

用于查询子网包含的所有子网连接。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in transmissionParameters::LayerRateList_T connectionRateList

——表示返回的子网连接需要满足的连接速率列表，如果为空，表示不限制。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out subnetworkConnection::SubnetworkConnectionList_T sncList

——表示子网连接列表。

out subnetworkConnection::SNCIterator_I sncIt

——表示迭代查询子网连接接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询子网包含的所有子网连接名称（getAllSubnetworkConnectionNames）

定义：

```
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);
```

说明：

用于查询子网包含的所有拓扑连接名称。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in transmissionParameters::LayerRateList_T connectionRateList

——表示返回名称的子网连接需要满足的连接速率列表，如果为空，表示不限制。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示子网连接名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询子网连接名称接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询子网连接路由（getRoute）

定义：

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

说明：

用于指定子网连接名称查询子网连接路由信息，返回路由的高阶交叉连接在本接口中可选。

输入参数：

in globaldefs::NamingAttributes_T sncName

——表示子网连接名称。

in boolean includeHigherOrderCCs

——表示是否返回承载子网连接的路由的高阶交叉连接。

输出参数：

out subnetworkConnection::Route_T route

——表示子网连接的路由。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 查询子网连接（getSNC）

定义：

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

说明：

用于指定子网连接名称查询子网连接信息。

输入参数：

in globaldefs::NamingAttributes_T sncName

——表示子网连接名称。

输出参数：

out subnetworkConnection::SubnetworkConnection_T snc

——表示子网连接信息。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- 创建子网连接（createSNC）

定义：

```
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);
```


说明：

用于创建子网连接，可以指定子网连接的 A/Z 端点、路由限制信息（子网连接经过或不能经过的网元、拓朴连接、端口、CTP、交叉连接）。

输入参数：

in subnetworkConnection::SNCCreateData_T createData

——表示子网连接创建数据。

in subnetworkConnection::GradesOfImpact_T tolerableImpact

——表示可以允许的对业务的影响程度。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

输出参数：

out subnetworkConnection::SubnetworkConnection_T theSNC

——表示子网连接信息。

out string errorReason

——表示操作失败的原因。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（7）友好名称已存在（EXCPT_USERLABEL_IN_USE）。

（8） 对象已被占用（EXCPT_OBJECT_IN_USE）。

（9）路由限制不支持（EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS）。

- 激活子网连接（activateSNC）

定义：

```
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);
```

说明：

用于激活子网连接，将子网连接使用的交叉连接下发到设备。

输入参数：

in globaldefs::NamingAttributes_T sncName

——表示子网连接名称。

in subnetworkConnection::GradesOfImpact_T tolerableImpact

——表示可以允许的对业务的影响程度。

inout subnetworkConnection::TPDataList_T tpsToModify

——表示修改的终端点及其参数。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

输出参数：

out subnetworkConnection::SubnetworkConnection_T theSNC

——表示子网连接信息。

out string errorReason

——表示操作失败的原因。

inout subnetworkConnection::TPDataList_T tpsToModify

——表示修改后的终端点及其参数。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）对象已被占用（EXCPT_OBJECT_IN_USE）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（6）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 创建并激活子网连接（createAndActivateSNC）

定义：

```
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);
```

说明：

用于创建并激活子网连接，创建成功子网连接后，将子网连接使用的交叉连接下发到设备。

输入参数：

in　　subnetworkConnection::SNCCreateData_T　createData

——表示子网连接创建数据。

in subnetworkConnection::GradesOfImpact_T tolerableImpact

——表示可以允许的对业务的影响程度。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

inout subnetworkConnection::TPDataList_T tpsToModify,

——表示修改的终端点及其参数。

输出参数：

out subnetworkConnection::SubnetworkConnection_T theSNC

——表示子网连接信息。

out string errorReason

——表示操作失败的原因。

inout subnetworkConnection::TPDataList_T tpsToModify,

——表示修改后的终端点及其参数。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（7）友好名称已存在（EXCPT_USERLABEL_IN_USE）。

（8）对象已被占用（EXCPT_OBJECT_IN_USE）。

（9）路由限制不支持（EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS）。

- 去激活子网连接（deactivateSNC）

定义：

```
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 );
```

说明：

用于去激活子网连接，保留子网连接数据，将子网连接使用的交叉连接从设备上删除。

输入参数：

in globaldefs::NamingAttributes_T sncName

——表示子网连接名称。

in subnetworkConnection::GradesOfImpact_T tolerableImpact

——表示可以允许的对业务的影响程度。

inout subnetworkConnection::TPDataList_T tpsToModify

——表示修改的终端点及其参数。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

输出参数：

out subnetworkConnection::SubnetworkConnection_T theSNC

——表示子网连接信息。

out string errorReason

——表示操作失败的原因。

inout subnetworkConnection::TPDataList_T tpsToModify

——表示修改后的终端点及其参数。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）对象已被占用（EXCPT_OBJECT_IN_USE）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（6）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 删除子网连接（deleteSNC）

定义：

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

说明：

用于删除子网连接。被删除的子网连接不能处于激活或部分激活状态。

输入参数：

in　　globaldefs::NamingAttributes_T sncName

——表示子网连接名称。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

输出参数：

无。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）对象已被占用（EXCPT_OBJECT_IN_USE）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（6）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 去激活并删除子网连接（deactivateAndDeleteSNC）

定义：

```
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);
```

说明：

用于去激活并删除子网连接，操作成功后交叉连接将从设备删除，子网连接也将被从 EMS 删除。

输入参数：

in globaldefs::NamingAttributes_T　sncName

——表示子网连接名称。

in subnetworkConnection::GradesOfImpact_T tolerableImpact

——表示可以允许的对业务的影响程度。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

inout subnetworkConnection::TPDataList_T tpsToModify

——表示修改的终端点及其参数。

输出参数：

out subnetworkConnection::SubnetworkConnection_T theSNC

——表示子网连接信息。

out string errorReason

——表示操作失败的原因。

inout subnetworkConnection::TPDataList_T tpsToModify,

——表示修改后的终端点及其参数。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）对象已被占用（EXCPT_OBJECT_IN_USE）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（6）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询子网包含的所有终端点池（getAllTPPools）（可选）

定义：

```
void getAllTPPools(
        in globaldefs::NamingAttributes_T subnetworkName,
        in unsigned long how_many,
        out terminationPoint::TerminationPointList_T tpList,
        out terminationPoint::TerminationPointIterator_I tpIt)
      raises(globaldefs::ProcessingFailureException);
```

说明：

用于查询子网包含的所有终端点池。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out terminationPoint::TerminationPointList_T tpList

——表示终端点列表。

out terminationPoint::TerminationPointIterator_I tpIt

——表示迭代查询终端点接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询子网包含的所有终端点池名称（getAllTPPoolNames）（可选）

定义：

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

说明：

用于查询子网包含的所有终端点池名称。

输入参数：

in globaldefs::NamingAttributes_T subnetName

——表示子网名称。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out globaldefs::NamingAttributesList_T nameList

——表示终端点池名称列表。

out globaldefs::NamingAttributesIterator_I nameIt

——表示迭代查询名称接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（4）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 修改子网连接（modifySNC）

定义：

```
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);

说明：

用于修改子网连接。

输入参数：

in globaldefs::NamingAttributes_T sncName

——表示子网连接名称。

in string routeId

——表示子网连接的路由标识，在子网连接具有多条路由时用来标记修改的路由，如果为空，表示修改当前正在使用的路由。

in subnetworkConnection::SNCModifyData_T SNCModifyData

——表示修改子网连接数据。

in subnetworkConnection::GradesOfImpact_T tolerableImpact

——表示可以允许的对业务的影响程度。

in subnetworkConnection::ProtectionEffort_T tolerableImpactEffort,

——表示可以允许的对保护尽力程度的影响。

in EMSFreedomLevel_T emsFreedomLevel

——表示 EMS 对子网连接操作的自由度。

inout subnetworkConnection::TPDataList_T tpsToModify,

——表示修改的终端点及其参数。

输出参数：

inout subnetworkConnection::TPDataList_T tpsToModify,

——表示修改后的终端点及其参数。

out subnetworkConnection::SubnetworkConnection_T newSNC

——表示修改后的子网连接信息。

out string errorReason

——表示操作失败的原因。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（7）友好名称已存在（EXCPT_USERLABEL_IN_USE）。

（8）对象已被占用（EXCPT_OBJECT_IN_USE）。

（9）路由限制不支持（EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS）。

3.1.9 子网连接模块（module subnetworkConnection）

StaticProtectionLevel_T

定义：

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

说明：

表示静态保护等级。

属性描述：

PREEMPTIBLE

——表示先占方式。

UNPROTECTED

——表示无保护。

PARTIALLY_PROTECTED

——表示部分保护。

FULLY_PROTECTED

——表示完全保护。

HIGHLY_PROTECTED

——表示高级保护。

ProtectionEffort_T

定义：

```
enum ProtectionEffort_T
    {
    EFFORT_WHATEVER,
        EFFORT_SAME_OR_BETTER,
    EFFORT_SAME_OR_WORSE,
```

EFFORT_SAME

};

说明：

表示保护尽力程度，在创建子网连接时与静态保护等级配合使用。

属性描述：

EFFORT_WHATEVER

——表示其他保护级别也可以。

EFFORT_SAME_OR_BETTER

——表示可选择相同或更好的保护级别。

EFFORT_SAME_OR_WORSE

——表示可选择相同或差的保护级别。

EFFORT_SAME

——表示选择相同的保护级别。

SNCState_T

定义：

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

说明：

表示子网连接的状态。

属性描述：

SNCS_NONEXISTENT

——表示子网连接不存在。

SNCS_PENDING,

——表示悬置状态。

SNCS_ACTIVE

——表示激活状态。

SNCS_PARTIAL

——表示子网连接为部分激活状态。

GradesOfImpact_T

定义：

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

说明：

表示业务影响程度。

属性描述：

GOI_HITLESS

——表示无影响。

GOI_MINOR_IMPACT,

——表示对业务的影响小于 50ms。

GOI_MAJOR_IMPACT

——表示对业务的影响大于 50ms。

TPData_T

定义：

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

说明：

表示终端点及业务参数相关数据。

属性描述：

globaldefs::NamingAttributes_T tpName

——表示终端点名称。

terminationPoint::TerminationMode_T tpMappingMode

——表示终端点是否可配置交叉连接以及是否可映射下一层类型。

transmissionParameters::LayeredParameterList_T transmissionParams

——表示终端点的传送参数。

globaldefs::NamingAttributes_T ingressTrafficDescriptorName

——表示入口业务描述符名称，如果没有可以为空字符串。

globaldefs::NamingAttributes_T egressTrafficDescriptorName

——表示出口业务描述符名称，如果没有可以为空字符串。

TPDataList_T

定义：

typedef sequence<TPData_T> TPDataList_T;

说明：

表示终端点及业务参数相关数据列表。

SNCType_T

定义：

```
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
    };
```

说明：

表示子网连接的类型，对子网连接类型的具体定义请参见引用文件 TMF 814（2003）“Multi-Technology Network Management Solution Set Document NML-EML Interface Version 3.0”中的“SNCTypes.pdf”。。

属性描述：

ST_SIMPLE

——表示简单型。

ST_ADD_DROP_A

——表示 A 点 AddDrop 型。

ST_ADD_DROP_Z

——表示 Z 点 AddDrop 型。

ST_INTERCONNECT

——表示互连型。

ST_DOUBLE_INTERCONNECT

——双互连型。

ST_DOUBLE_ADD_DROP

——表示双向 AddDrop 型。

OPEN_ADD_DROP

——表示开放的 AddDrop 型。

ST_EXPLICIT

——表示其他型。

Reroute_T

定义：

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

说明：

表示是否允许重路由。

属性描述：

RR_NA

——表示不关心。

RR_NO

——表示不允许。

RR_YES

——表示允许。

NetworkRouted_T

定义：

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

说明：

表示路由是否必须在网络级计算。

属性描述：

RR_NA

——表示不关心。

RR_NO

——表示不必。

RR_YES

——表示必须。

SubnetworkConnection_T

定义：

```
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;
};
```

说明：

表示子网连接。

属性描述：

globaldefs::NamingAttributes_T name

——表示子网连接名称。

string userLabel

——表示子网连接的友好名称。

string nativeEMSName

——表示子网连接的本地名称。

string owner

——表示子网连接的所有者。

SNCState_T sncState

——表示子网连接状态。

globaldefs::ConnectionDirection_T direction

——表示子网连接方向。

transmissionParameters::LayerRate_T rate

——表示子网连接速率。

StaticProtectionLevel_T staticProtectionLevel

——表示子网连接静态保护等级。

SNCType_T sncType

——表示子网连接类型。

TPDataList_T aEnd

——表示子网连接的 A 端点。

TPDataList_T zEnd

——表示子网连接的 Z 端点。

Reroute_T rerouteAllowed

——表示子网连接是否允许重路由。

NetworkRouted_T networkRouted

——表示子网连接的路由是否必须在网络层计算。如果重路由被允许，本属性用来标识重路由是由网络层还是 EMS 来进行。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

SubnetworkConnectionList_T

定义：

```
typedef sequence<SubnetworkConnection_T> SubnetworkConnectionList_T;
```

说明：

表示子网连接列表。

CrossConnect_T

定义：

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

};

说明：

表示交叉连接。

属性描述：

boolean active

——表示是否激活。

globaldefs::ConnectionDirection_T direction

——表示方向。

SNCType_T ccType

——表示交叉连接类型。

globaldefs::NamingAttributesList_T aEndNameList

——表示交叉连接 A 端点列表。

globaldefs::NamingAttributesList_T zEndNameList

——表示交叉连接 Z 端点列表。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

CrossConnectList_T

定义：

typedef sequence<CrossConnect_T> CrossConnectList_T;

说明：

表示交叉连接列表。

Resource_T

定义：

typedef globaldefs::NamingAttributes_T Resource_T;

说明：

表示资源信息。

ResourceList_T

定义：

typedef sequence<Resource_T> ResourceList_T;

说明：

表示资源信息列表，包含网元、拓朴连接、物理终端点、连接终端点及子网连接。

Route_T

定义：

```
struct Route_T
  {
     string id;
     CrossConnectList_T routeXCs;
     globaldefs::NVSList_T additionalInfo;
   }
```

说明：

表示子网连接路由描述符。

属性描述：

string id

——表示路由描述符标识，在子网连接的路由中惟一。

CrossConnect_T routeXCs

——表示路由包含的交叉连接列表。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

RouteList_T

定义：

```
typedef sequence<Route_T> RouteList_T;
```

说明：

表示子网连接路由列表。

SNCCreateData_T

定义：

```
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;
    };
```

说明：

表示子网连接创建数据。

属性描述：

string userLabel

——表示子网连接的友好名称。

boolean forceUniqueness

——表示是否要求子网连接的友好名称惟一。

string owner

——表示子网连接的所有者。

globaldefs::ConnectionDirection_T direction

——表示子网连接方向。

StaticProtectionLevel_T staticProtectionLevel

——表示子网连接静态保护等级。

ProtectionEffort_T protectionEffort

——表示保护尽力程度。

Reroute_T rerouteAllowed;

——表示是否允许重路由。

NetworkRouted_T networkRouted

——表示是否要求路由在网络级计算。

SNCType_T sncType

——表示子网连接类型。

transmissionParameters::LayerRate_T layerRate

——表示子网连接速率。

CrossConnectList_T ccInclusions

——表示子网连接必须包含的交叉连接。

ResourceList_T neTpInclusions

——表示子网连接必须包含的资源，可以包括网元或终端点。

boolean fullRoute

——表示是否提供了全路由。

ResourceList_T neTpSncExclusions

——表示子网连接不能包含的资源，可以包括网元、终端点或子网连接。

TPDataList_T aEnd

——表示子网连接的 A 端点。

TPDataList_T zEnd

——表示子网连接的 Z 端点。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

SNCModifyData_T

定义：

```
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;
};
```

说明：

表示子网连接修改数据。

属性描述：

string userLabel

——表示子网连接的友好名称。

boolean forceUniqueness

——表示是否要求子网连接的友好名称惟一。

string owner

——表示子网连接的所有者。

globaldefs::ConnectionDirection_T direction

——表示子网连接方向。

string modifyType

——表示修改的类型，包括“rerouting”（重路由）“add_protection”（增加保护）"remove_protection"（删除保护）。

boolean retainOldSNC

——表示是否保留原来的 SNC。

boolean modifyServers_allowed

——表示是否可以修改服务层来完成保护约束。

StaticProtectionLevel_T staticProtectionLevel

——表示子网连接静态保护等级。

ProtectionEffort_T protectionEffort

——表示保护尽力程度。

Reroute_T rerouteAllowed

——表示是否允许重路由。

NetworkRouted_T networkRouted

——表示是否要求路由在网络级计算。

SNCType_T sncType

——表示子网连接类型。

transmissionParameters::LayerRate_T layerRate

——表示子网连接速率。

RouteList_T addedOrNewRoute

——表示增加的路由列表。

RouteList_T removedRoute

——表示删除的路由。

CrossConnectList_T ccInclusions

——表示子网连接必须包含的交叉连接。

ResourceList_T neTpInclusions

——表示子网连接必须包含的资源，包括网元或终端点。

boolean fullRoute

——表示是否提供了全路由。

ResourceList_T neTpSncExclusions

——表示子网连接不能包含的资源，包括网元或终端点或子网连接。

TPDataList_T aEnd

——表示子网连接的 A 端点。

TPDataList_T zEnd

——表示子网连接的 Z 端点。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

SNCIterator_I 接口

说明：

表示迭代查询子网连接接口。

- 查询下一批子网连接数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out SubnetworkConnectionList_T sncList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批子网连接数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out SubnetworkConnectionList_T sncList

——表示查询得到的子网连接列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength)

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

```
void destroy ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

CCIterator_I 接口

说明：

表示迭代查询交叉连接接口。

- 查询下一批交叉连接数据（next_n）

定义：

```
boolean next_n (
  in unsigned long how_many,
  out CrossConnectList_T ccList
  )
```

raises (globaldefs::ProcessingFailureException);

说明：

通过迭代器查询下一批交叉连接数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out CrossConnectList_T ccList

——表示查询得到的交叉连接列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength）

定义：

unsigned long getLength ()

raises (globaldefs::ProcessingFailureException);

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.2 故障管理

故障管理功能包括告警上报功能、告警同步功能、告警级别表管理功能。

3.2.1 告警上报（module notifications）

告警上报数据类型请参见公共管理部分的通知管理模块。

3.2.2 告警级别模块（module alarmServerity）

PerceivedSeverity_T

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

说明：

表示告警级别。

属性描述：

PS_INDETERMINATE

——表示未确认。

PS_CRITICAL

——表示严重。

PS_MAJOR

——表示主要。

PS_MINOR

——表示次要。

PS_WARNING

——表示警告。

PS_CLEARED

——表示清除。

PerceivedSeverityList_T

定义：

typedef sequence<PerceivedSeverityList_T> PerceivedSeverityList_T;

说明：

表示告警级别列表。

AlarmSeverityAssignment_T

定义：

```
struct AlarmSeverityAssignment_T
  {
   ProbableCause_T probableCause;
   string probableCauseQualifier;
   string nativeProbableCause;
   PerceivedSeverity_T currentSeverity;
   PerceivedSeverity_T defaultSeverity;
};
```

说明：

表示告警级别设置。

属性描述：

ProbableCause_T probableCause

——表示告警原因。

string probableCauseQualifier

——表示告警原因惟一名称。当告警原因不能保证惟一时，用来惟一标识告警级别信息。

string nativeProbableCause

——表示告警在 EMS 本地的原因，当告警原因不能保证惟一时，也可以用来惟一标识名称告警级别信息。

PerceivedSeverity_T currentSeverity

——表示当前设置的告警级别。

PerceivedSeverity_T defaultSeverity

——表示缺省的告警级别。

AlarmSeverityAssignmentList_T

定义：

typedef sequence <AlarmSeverityAssignment_T> AlarmSeverityAssignmentList_T;

说明：

表示告警级别设置信息列表。

ASAP_T

定义：

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

说明：

表示告警级别配置表。

属性描述：

globaldefs::NamingAttributes_T name

——表示告警级别配置表名称。

string userLabel

——表示友好名称。

string nativeEMSName

——表示 EMS 本地名称。

string owner

——表示所有者。

AlarmSeverityAssignmentList_T alarmSeverityAssignmentList

——表示告警级别设置列表。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

ASAPList_T

定义：

```
typedef sequence<ASAP_T> ASAPList_T;
```

说明：

表示告警级别配置表列表。

ASAPCreateModifyData_T

定义：

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

说明：

表示告警级别配置表创建数据。

属性描述：

string userLabel

——表示友好名称。

boolean forceUniqueness

——表示是否要求友好名称惟一。

string owner

——表示所有者。

AlarmSeverityAssignmentList_T alarmSeverityAssignmentList;

——表示告警级别设置列表。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

ASAPIterator_I 接口（可选）

说明：

表示迭代查询告警级别表接口。

- 查询下一批告警级别配置数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out ASAPList_T aSAPList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批告警级别表数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out ASAPList_T aSAPList

——表示查询得到的告警级别表列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength）

定义：

unsigned long getLength ()

raises (globaldefs::ProcessingFailureException);

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.3 性能管理

3.3.1 性能管理模块（module performance）

Granularity_T

定义：

```
typedef string Granularity_T;
```

说明：

表示性能数据采集的粒度周期。

有效值是：

"15min"：表示粒度周期为 15 分钟。

"24h"：表示粒度周期为 24 小时。

"NA" ：表示当前的即时值。

GranularityList_T

定义：

```
typedef sequence <Granularity_T> GranularityList_T;
```

说明：

表示粒度周期列表。

PMParameter_T

定义：

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

说明：

表示性能监测参数。

属性描述：

PMParameterName_T pmParameterName

——表示性能监测参数名称。

PMLocation_T pmLocation

——表示性能监测位置。

PMMeasurement_T

定义：

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

说明：

表示性能监测值。

属性描述：

PMParameterName_T pmParameterName

——表示性能监测参数名称。

PMLocation_T pmLocation

——表示性能监测的位置。

float value

——表示性能值。

string unit

——表示门限单位（可以用自由格式字符串描述）。

string intervalStatus

——表示描述性能值的有效性,允许的值有：

"Valid" ：表示数据有效。

"Incomplete" ：表示整个采集间隔(15min、24h)内，数据不可用。

"Invalid" ：表示指定的时间间隔内，数据可用但标记为无效

"Unavailable" ：表示指定的时间间隔内，数据完全不可用。

"Zero-suppressed" ：表示零抑制时间段。

PMMeasurementList_T

定义：

typedef sequence <PMMeasurement_T> PMMeasurementList_T;

说明：

表示性能监测值列表。

PMData_T

定义：

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

说明：

表示性能数据。

属性描述：

globaldefs::NamingAttributes_T tpName

——表示终端点名称。

transmissionParameters::LayerRate_T layerRate

——表示层速率。

Granularity_T granularity

——表示粒度周期。

globaldefs::Time_T retrievalTime

——表示性能数据采集的时间。

PMMeasurementList_T pmMeasurementList

——表示性能监测值列表。

PMDataList_T

定义：

```
typedef sequence <PMData_T> PMDataList_T;
```

说明：

表示性能数据列表。

PMTPSelect_T

定义：

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

说明：

表示性能操作选择对象的范围。

属性描述：

globaldefs::NamingAttributes_T name

——表示性能操作对象，为如下对象之一：

网元：操作该网元上的所有终端点所对应的性能。

终端点：可以是连接终端点或物理终端点，但不包括它所包含的任何终端点。

transmissionParameters::LayerRateList_T layerRate

——操作对象的层速率集合。如果为空，表示设备支持的所有层速率。

PMLocationList_T pMLocationList

——表示性能监测位置列表，如果为空，表示设备支持的所有性能监测位置。

GranularityList_T granularityList

——表示粒度周期列表：15min 和或者 24h。

如果为空，表示选择设备所支持的所有粒度周期列表。

PMTPSelectList_T

定义：

typedef sequence <PMTPSelect_T> PMTPSelectList_T;

说明：

表示性能操作对象的选择列表。每次操作可选择网元、终端点之一，但不能在列表中同时包含。

PMLocation_T

定义：

typedef string PMLocation_T;

说明：

表示性能监测的位置，可参考引用文件 TMF 814（2003） “Multi-Technology Network Management Solution Set Document NML-EML Interface Version 3.0”中的“LocationIdentification.pdf”。

取值为：

"PML_NEAR_END_Rx"：表示近端接收；

"PML_FAR_END_Rx"：表示远端接收；

"PML_NEAR_END_Tx"：表示近端发送；

"PML_FAR_END_Tx"：表示远端发送；

"PML_BIDIRECTIONAL"：表示双向；

"PML_CONTRA_NEAR_END_Rx"：表示反向的近端接收；

"PML_CONTRA_FAR_END_Rx"：表示反向的远端接收。

PMLocationList_T

定义：

typedef sequence < PMLocation_T > PMLocationList_T;

说明：

表示性能监测的位置列表。

Destination_T

定义：

typedef string Destination_T;

说明：

在查询历史性能数据并使用 FTP 传输方式传输性能数据时（方法 getHistoryPMData()），NMS 需要指定性能数据文件的目的地址（包含目录名和文件名）。在目的地址中，需要指明目标机器的机器名和目标文件在目标机器上的完整路径。机器名和完整路径之间用冒号（:）分割。路径名使用的分隔符“/”或“\”依赖于目的地址所在的机器，由客户端指定。

PMThresholdType_T

定义：

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

说明：

描述性能门限类型。

当前测量的模拟量性能超过 TWM_HIGHEST 和 TWM_HIGH 时会产生 TCA 告警，低于 TWM_LOW 和 TWM_LOWEST 时，也要产生 TCA 告警。

如果使用双门限，则使用门限类型 TWM_LOW 和 TWM_HIGH。

如果使用四门限，则使用门限类型 TWM_LOWEST、TWM_LOW、TWM_HIGH 和 TWM_HIGHEST。

属性描述：

TWM_HIGHEST：

——表示最高门限。

TWM_HIGH：

——表示高门限。

TWM_LOW：

——表示低门限。

TWM_LOWEST：

——表示最低门限。

PMThresholdValue_T

定义：

```
struct PMThresholdValue_T
  {
    PMParameter_T pmParameter;
    PMThresholdType_T thresholdType;
    boolean triggerFlag;
    float value;
    string unit;
  };
```

说明：

表示性能门限值。

属性描述：

PMParameter_T　pmParameter

——表示性能监测参数。

PMThresholdType_T　thresholdType

——表示门限值类型。

float　thresholdValue

——表示门限值。

boolean triggerFlag

——表示触发门限或清除门限。

string unit:

——表示门限单位。

PMThresholdValueList_T

定义：

```
typedef sequence <PMThresholdValue_T> PMThresholdValueList_T;
```

说明：

表示性能门限值列表。

TCAParameters_T

定义：

```
struct TCAParameters_T
{
```

transmissionParameters::LayerRate_T layerRate;

Granularity_T granularity;

PMThresholdValueList_T tcaTypeValues;

};

说明：

用于查询或设置性能门限到指定的终端点/层/粒度。

属性描述：

transmissionParameters::LayerRate_T layerRate

——表示层速率。

Granularity_T granularity

——表示粒度周期。

PMThresholdValueList_T tcaTypeValues

——表示性能门限值列表。

PMDataIterator_I 接口

说明：

用于提供对性能数据的迭代操作接口。

- 迭代查询性能数据（next_n）

定义：

```
boolean next_n (
        in unsigned long how_many,
        out PMDataList_T pmDataList
        )
        raises (globaldefs::ProcessingFailureException);
```

说明：

返回指定数目的查询结果。

输入参数：

in unsigned long how_many

——查询的数目。

输出参数：

out PMDataList_T pmDataList

——表示性能数据列表。

返回值：

boolean

——表示操作结果。为真表示还有未返回的数据，为假表示无未返回的数据。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目（getLength）

定义：

unsigned long getLength ()

raises (globaldefs::ProcessingFailureException);

说明：

查询迭代器中的总的数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——数据数目。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器（destroy）

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

释放迭代器，释放被迭代器占用的内存。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

PerformanceManagementMgr_I 接口

说明：

表示性能管理接口，本接口继承公共管理部分的 Common_I 接口。

其中性能采集计划类（创建、挂起、恢复、设置、删除、查询）和性能采集使能类（开启、关

闭、清除）操作必须选择其中一类。

- 查询当前性能数据(getAllCurrentPMData)

定义：

```
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);
```

说明：

查询当前性能数据。

输入参数：

in PMTPSelectList_T pmTPSelectList

——表示性能操作的选择对象的范围列表。

in PMParameterNameList_T pmParameters

——表示性能监测参数列表。

in unsigned long how_many

——表示迭代查询数据方式下首次查询的数目。

输出参数：

out PMDataList_T pmDataList

——表示当前性能数据列表。

out PMDataIterator_I pmIt

——表示迭代查询接口。

返回值：

无。

异常：

（1）操作没有完成或不支持操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

- 查询历史性能数据(getHistoryPMData)

定义：

```
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);
```

说明：

采用 FTP 方式查询历史性能数据，操作成功后历史性能数据将以文件形式被传送到指定的目的地。

输入参数：

in Destination_T destination

——表示 FTP 文件传送的目的地。

in string userName

——表示调用此接口方提供的 FTP 登录用户名。

in string password

——表示调用此接口方提供的 FTP 登录用户口令。

in PMTPSelectList_T pmTPSelectList

——表示性能操作的选择对象的范围列表。

in PMParameterNameList_T pmParameters

——表示性能监测参数列表。

in globaldefs::Time_T startTime

——表示起始时间。

in globaldefs::Time_T endTime

——表示终止时间。

in boolean forceUpload

——表示是否 EMS 必须从网元上采集性能数据。

输出参数：

无。

返回值：

无。

异常：

（1）操作没有完成或不支持操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

- 查询性能门限(getTCATPParameter)

定义：

```
void getTCATPParameter (
    in globaldefs::NamingAttributes_T tpName,
    in transmissionParameters::LayerRate_T layerRate,
    in Granularity_T granularity,
    out TCAParameters_T tcaParameter
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

查询终端点的 15min/24h 的性能门限值。

输入参数：

in globaldefs::NamingAttributes_T tpName:

——表示终端点名称。

in transmissionParameters::LayerRate_T layerRate:

——表示层速率。

in Granularity_T granularity:

——表示粒度周期。

输出参数：

out TCAParameters_T tcaParameter

——表示查询的性能门限结果。

返回值：

无。

异常：

内部处理错误（INTERNAL_ERROR）。

- 设置性能门限(setTCATPParameter)

定义：

```
void setTCATPParameter (
    in globaldefs::NamingAttributes_T tpName,
    inout TCAParameters_T tcaParameters
    )
    raises (globaldefs::ProcessingFailureException);
};
```

说明：

设置性能门限，操作对象为终端点。

输入参数：

in globaldefs::NamingAttributes_T tpName

——表示终端点名称。

输入/输出参数：

inout TCAParameters_T tcaParameters

——表示设置以及设置成功后的返回的新的性能越门限告警参数。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 创建性能采集计划(createPMCollectionPlan)（可选）

定义：

```
void createPMCollectionPlan(
        in PMCollectionPlan_T pmPlan,
        out string pmPlanId)
      raises (globaldefs::ProcessingFailureException);
```

说明：

创建性能采集上报任务。

输入参数：

in PMCollectionPlan_T pmPlan

——表示性能采集计划。

输出参数：

out string pmPlanId

——表示性能采集计划名称。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

- 挂起性能采集计划(suspendPMCollectionPlan)（可选）

定义：

```
void suspendPMCollectionPlan(
        in string pmPlanId,
```

out PMCollectionPlan_T pmPlan)
raises (globaldefs::ProcessingFailureException);

说明：

挂起性能采集上报任务。

输入参数：

in string pmPlanId

——表示性能采集计划名称。

输出参数：

out PMCollectionPlan_T pmPlan

——表示性能采集计划。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

- 恢复性能采集计划(resumePMCollectionPlan)（可选）

定义：

void resumePMCollectionPlan(
in string pmPlanId,
out PMCollectionPlan_T pmPlan)
raises (globaldefs::ProcessingFailureException);

说明：

恢复性能采集上报任务。

输入参数：

in string pmPlanId

——表示性能采集计划名称。

输出参数：

out PMCollectionPlan_T pmPlan

——表示性能采集计划。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

- 删除性能采集计划(deletePMCollectionPlan)（可选）

定义：

```
void deletePMCollectionPlan(
        in string pmPlanId)
        raises (globaldefs::ProcessingFailureException);
```

说明：

删除性能采集计划。

输入参数：

in string pmPlanId

——表示性能采集计划名称。

输出参数：

无。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

- 设置性能采集计划(setPMCollectionPlan)（可选）

定义：

```
void setPMCollectionPlan(
        inout PMCollectionPlan_T pmPlan)
      raises (globaldefs::ProcessingFailureException);
```

说明：

设置性能采集上报任务。

输入参数：

inout PMCollectionPlan_T pmPlan

——表示性能采集计划。

输出参数：

inout PMCollectionPlan_T pmPlan

——表示性能采集计划。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

- 查询性能采集计划（getPMCollectionPlan）（可选）

定义：

```
void getPMCollectionPlan(
        in string pmPlanId,
        out PMCollectionPlan_T pmPlan)
        raises (globaldefs::ProcessingFailureException);
```

说明：

查询性能采集上报计划任务。

输入参数：

in string pmPlanId

——表示性能采集计划名称。

输出参数：

out PMCollectionPlan_T pmPlan

——表示性能采集计划。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

- 查询所有性能采集计划（getPMCollectionPlan）（可选）

定义：

```
void getAllPMCollectionPlans(
        out PMCollectionPlanList_T pmPlanList)
        raises (globaldefs::ProcessingFailureException);
};
```

说明：

查询所有性能采集上报计划任务。

输入参数：

无。

输出参数：

out PMCollectionPlanList_T pmPlanList

——表示性能采集计划列表。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

- 开启性能采集(enablePMData)（可选）

定义：

```
void enablePMData(
        in PMTPSelectList_T pmTPSelectList,
        out PMTPSelectList_T failedTPSelectList)
      raises(globaldefs::ProcessingFailureException);
```

说明：

开启指定性能测量点上的性能采集。本操作为尽力操作，对不成功的操作在输出参数中体现。

输入参数：

in PMTPSelectList_T pmTPSelectList:

——表示性能操作的范围列表，包括终端点、层速率、位置信息和粒度周期。

输出参数：

out PMTPSelectList_T failedTPSelectList

——表示不成功的操作范围。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

- 关闭性能采集(disablePMData)（可选）

定义：

```
void disablePMData(
        in PMTPSelectList_T pmTPSelectList,
        out PMTPSelectList_T failedTPSelectList)
```

raises(globaldefs::ProcessingFailureException);

说明：

关闭指定性能测量点上的性能采集，本操作为尽力操作，对不成功的操作在输出参数中体现。

输入参数：

in PMTPSelectList_T pmTPSelectList:

——表示性能操作的范围列表，包括终端点、层速率、位置信息和粒度周期。

输出参数：

out PMTPSelectList_T failedTPSelectList

——表示不成功的操作范围。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

- 清空性能采集寄存器(clearPMData)（可选）

定义：

```
void clearPMData(
        in PMTPSelectList_T pmTPSelectList,
        out PMTPSelectList_T failedTPSelectList)
      raises(globaldefs::ProcessingFailureException);
```

说明：

清空指定性能测量点上的性能采集寄存器，将保存的最大、最小、平均测量值恢复到当前测量值，对当前测量值不起作用。本操作为尽力操作，对不成功的操作在输出参数中体现。

输入参数：

in PMTPSelectList_T pmTPSelectList:

——表示性能操作的范围列表，包括终端点、层速率、位置信息和粒度周期。

输出参数：

out PMTPSelectList_T failedTPSelectList

——表示不成功的操作范围。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（5）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

3.4 通用管理

3.4.1 通信链路监视（module heartbeatService）

HeartbeatServiceMgr_I 接口

说明：

心跳服务是指提供对通道及通信网的连接情况的检测服务。通过周期性的向与之相关联的通道发送一个通知，在通知中传送系统名称信息和发送时间信息，在通道的对端的系统通过检测此通知可以判断通道的运行状况。心跳服务的心跳通知的发送周期可以设置。

本接口继承公共管理部分的 Common_I 接口。

- 查询心跳服务参数（getHeartbeatParameter）（可选）

定义：

```
void getHeartbeatParameter (
        out long reportInterval
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

表示查询通信探测包的发送时间间隔。

输入参数：

无。

输出参数：

out long reportInterval

——表示探测包发送的时间间隔，单位为秒。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 设置心跳服务参数（setHeartbeatParameter）（可选）

定义：

```
void setHeartbeatParameter (
    in long reportInterval
```

)

raises (globaldefs::ProcessingFailureException);

说明：

设置通信探测包的发送时间间隔。EMS 收到此命令后要调整通信探测包的发送时间间隔，按设定的时间重新发送。

输入参数：

in long reportInterval

——表示探测包发送的时间间隔，单位为秒。如果设置为 0，就停止发送心跳通知。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.4.2 时间同步管理（module timeMgr）

EMSTimeMgr_I 接口

说明：

时间管理接口。主要提供对 EMS 时间的操作，如果 EMS 支持 NTP 协议，设置时间操作可选。

本接口继承公共管理部分的 Common_I 接口。

- 查询 EMS 时间（getEMSTime）(可选)

定义：

void getEMSTime (

out globaldefs::Time_T emsTime

)

raises (globaldefs::ProcessingFailureException);

说明：

用于NMS查询EMS时间。

输入参数：

无。

输出参数：

out globaldefs::Time_T emsTime

——表示 EMS 的当前时间。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 设置 EMS 时间（setEMSTime）（可选）

定义：

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

说明：

用于NMS设置EMS时间，在EMS不支持NTP协议时使用。

输入参数：

in globaldefs::Time_T setTime

——表示设置时间。

输出参数：

无。

返回值：

无。

异常：1.内部处理错误（EXCPT_INTERNAL_ERROR）。

3.4.3 维护管理（module maintenanceOps）

MaintenanceOperation_T

定义：

```
typedef string MaintenanceOperation_T;
```

说明：

表示维护操作类型，具体定义如下：

"FACILITY_LOOPBACK"：表示向设备侧环回。

"TERMINAL_LOOPBACK"：表示向终端侧环回。

"FACILITY_FORCED_AIS"：表示向设备侧强制插入告警指示信号。

"TERMINAL_FORCED_AIS"：表示向终端侧强制插入告警指示信号。

"FORCE_RDI"：表示强插入远端缺陷指示。

MaintenanceOperationMode_T

定义：

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

说明：

表示维护操作模式。

属性描述：

MOM_OPERATE：表示执行维护操作。

MOM_RELEASE：表示解除维护操作。

CurrentMaintenanceOperation_T

定义：

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

说明：

表示当前的维护操作。

属性描述：

globaldefs::NamingAttributes_T tpName

——表示终端点名称。

MaintenanceOperation_T maintenanceOperation

——表示维护操作类型。

transmissionParameters::LayerRate_T layerRate

——表示层速率。

globaldefs::NVSList_T additionalInfo

——表示附加信息。

CurrentMaintenanceOperationList_T

定义：

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

说明：

表示当前的维护操作列表。

CurrentMaintenanceOperationIterator_I 接口

说明：

用于提供对当前维护操作数据的迭代操作接口。

- 迭代查询当前维护操作数据（next_n）

定义：

```
boolean next_n (
        in unsigned long how_many,
        out CurrentMaintenanceOperationList_T cMOList
        )
        raises (globaldefs::ProcessingFailureException);
```

说明：

返回指定数目的查询结果。

输入参数：

in unsigned long how_many

——查询的数目。

输出参数：

out CurrentMaintenanceOperationList_T cMOList

——表示当前维护操作数据列表。

返回值：

boolean

——表示操作结果，为真表示还有未返回的数据，为假表示无未返回的数据。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目（getLength）

定义：

```
unsigned long getLength ()
        raises (globaldefs::ProcessingFailureException);
```

说明：

查询迭代器中的总的数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器（destroy）

定义：

void destroy ()

raises (globaldefs::ProcessingFailureException);

说明：

释放迭代器，释放被迭代器占用的内存。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

MaintenanceMgr_I 接口

说明：

维护操作管理接口。主要提供执行维护操作和获取当前的维护操作数据等操作。

本接口继承公共管理部分的 Common_I 接口。

- 执行或解除维护操作（performMaintenanceOperation）

定义：

void performMaintenanceOperation(

in CurrentMaintenanceOperation_T maintenanceOperation,

in MaintenanceOperationMode_T maintenanceOperationMode)

raises (globaldefs::ProcessingFailureException);

说明：

用于NMS向EMS下发维护操作命令，可以是执行维护操作或解除已经执行的维护操作。

输入参数：

in CurrentMaintenanceOperation_T maintenanceOperation

——表示当前的维护操作信息。

in MaintenanceOperationMode_T maintenanceOperationMode

——表示维护操作的方式，包括执行或解除两种。

输出参数：

无。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（7）终端点对象状态不正确，不能实施操作（EXCPT_NOT_IN_VALID_STATE）。

- 获取当前的维护操作（getActiveMaintenanceOperations）

定义：

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

说明：

用于NMS获取网元或终端点上当前的维护操作信息。

输入参数：

in globaldefs::NamingAttributes_T tpOrMeName

——表示终端点或网元名称。

in unsigned long how_many

——表示迭代获取数据时首次获取的数目。

输出参数：

out CurrentMaintenanceOperationList_T currentMaintenanceOpeationList

——表示当前的维护操作列表。

out CurrentMaintenanceOperationIterator_I cmoIt

——表示迭代获取当前的维护操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（7）打开的迭代器数目过多（EXCPT_TOO_MANY_OPEN_ITERATORS）。

3.4.4 会话管理模块（module session）

ession_I 接口

说明：

会话管理接口。接口提供对通信的检测。

会话部分具体实现有两个会话接口，一个实现在 EMS，即 EMSSession_I，另一个实现在 NMS，即 NMSSession_I，它们都是本接口的派生接口。

- 通信检测操作（ping）

定义：

void ping ();

说明：

探测通信的丢失。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

无。

- 结束会话（endSession）

定义：

oneway boolean endSession ( )

说明：

用于结束会话。NMS 和 EMS 都可以发起此操作。

输入参数：

名称无。

输出参数：

无。

返回值：

无。

异常：

无。

3.4.5 EMS 会话厂管理模块（module emsSessionFactory）

EmsSessionFactory_I 接口

说明：

EMS 会话厂接口。提供了访问 EMS 的入口。通过 EMS 会话厂可以进一步查询 EMS 会话接口。

- 查询 EMS 会话接口（getEmsSession）

定义：

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

说明：

操作允许 NMS 查询 EmsSession_I 接口对象，其包含了 EMS 能够包含地管理者对象。

输入参数：

in string user

——表示用户名。

in string password

——表示用户口令。

in nmsSession::NmsSession_I client

——表示 NMS 的会话操作接口，用于 EMS 操作使用，如上报事件等。

输出参数：

out emsSession::EmsSession_I emsSessionInterface

——表示 EMS 会话操作接口。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）无效的输入（EXCPT_INVALID_INPUT）。

（3）访问被拒绝（EXCPT_EXCPT_ACCESS_DENIED）。

3.4.6 EMS 会话管理模块（module emsSession）

ManagerNames_T

定义：

typedef sequence <string> ManagerNames_T;

说明：

管理者名称信息。在 NMS 查询 EMS 支持的管理者信息时使用。

EmsSession_I 接口

说明：

EMS 会话管理接口，用于提供对 EMS 访问操作的各管理者。

- 查询 EMS 支持的管理者信息（getSupportedManagers）

定义：

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

说明：

查询 EMS 支持的管理者信息。根据查询到的管理者信息，可以进一步查询管理者的操作接口。

输入参数：

无。

输出参数：

out managerNames_T supportedManagerList

——表示管理者名称列表。要求必须一致的管理者名称有：

"EMS"：EMS 管理者。

"ManagedElement"：网元管理者。

"MultiLayerSubnetwork" ：子网管理者。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询管理者操作接口操作（getManager）

定义：

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

说明：

查询 EMS 支持的管理者信息。根据查询到的管理者信息，可以进一步查询管理者的操作接口。

输入参数：

in string managerName

——表示管理者名称。

输出参数：

out common::Common_I managerInterface

——表示管理者操作接口，管理者的操作接口都是从 common 接口继承的。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）访问被拒绝（EXCPT_EXCPT_ACCESS_DENIED）。

（3）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

- NMS 查询事件通道（getEventChannel）

定义：

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


说明：

查询 EMS 提供的事件通道。NMS 可以利用此事件通道接收 EMS 发送过来的事件。

输入参数：

无。

输出参数：

out CosNotifyChannelAdmin::EventChannel eventChannel

——EMS 提供的事件通道。

返回值：

无。

异常：

（1）内部处理错误（EXCPT_INTERNAL_ERROR）。

（2）访问被拒绝（EXCPT_EXCPT_ACCESS_DENIED）。

3.4.7 NMS 会话管理模块（module nmsSession）

NmsSession_I 接口

说明：

NMS 会话管理接口。本接口提供给 EMS 操作使用，在查询 EMS 会话接口时提供。

- 通知事件丢失（eventLossOccurred）

定义：

```
void eventLossOccurred (
        in globaldefs::Time_T startTime,
        in string notificationId
        );
```

说明：

事件丢失通知。当 EMS 向 NMS 推事件失败时，它会使用此操作向 NMS 发出事件丢失通知。

输入参数：

in globaldefs::Time_T startTime

——表示事件丢失的起始时间。

in string notificationId

——表示通知名称。

输出参数：

无。

返回值：

无。

异常：

无。

- 通知事件丢失清除（eventLossCleared）

定义：

```
void eventLossCleared (
        in globaldefs::Time_T endTime
        );
```

说明：

事件丢失清除通知。当 EMS 向 NMS 推事件由失败恢复正常时，它会使用此操作向 NMS 发出事件丢失清除通知。

输入参数：

in globaldefs::Time_T endTime

——表示事件丢失的结束时间。

输出参数：

无。

返回值：

无。

异常：

无。

3.4.8 传送参数管理模块（module transmissionParameters）

LayerRate_T

定义：

typedef short LayerRate_T;

说明：

表示层速率。

LayerRateList_T

定义：

typedef sequence<LayerRate_T> LayerRateList_T;

说明：

表示层速率列表。

LayeredParameters_T

定义：

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

说明：

表示表示层速率及其相关的传送参数列表。

属性描述：

LayerRate_T layer

——表示层速率。

globaldefs::NVSList_T transmissionParams

——表示层参数列表。

LayeredParameterList_T

定义：

typedef sequence<LayeredParameters_T> LayeredParameterList_T;

说明：

表示层速率及参数列表的列表。

3.5 公共管理

3.5.1 全局定义（module globaldefs）

全局定义包括一些公用的类型，这些类型可以被其他模块使用。

ConnectionDirection_T

定义：

```
enum ConnectionDirection_T
  {
      CD_UNI,
   CD_BI
};
```

说明：

表示连接方向。可表示子网连接、交叉连接、拓扑连接的方向。

属性描述：

CD_UNI

——表示单向。

CD_BI

——表示双向。

ExceptionType_T

定义：

```
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
```

};

说明：

在处理出错时抛出的异常类型。

属性描述：

EXCPT_NOT_IMPLEMENTED

——操作没有完成或不支持操作。

EXCPT_INTERNAL_ERROR

——表示 EMS 内部处理错误。

EXCPT_INVALID_INPUT

——无效的输入。

EXCPT_OBJECT_IN_USE

——对象正被使用中。如配置的终端点已被用作交叉连接或被终结或已被映射，就不能再被用来做有冲突的配置。

EXCPT_TP_INVALID_ENDPOINT

——表明指定的终端点不存在或不能被创建。

EXCPT_ENTITY_NOT_FOUND

——实体没有找到。如使用 EMS 支持的某个对象名称作为参数来操作某个对象，但又没有找到此对象，就可抛出此异常。

EXCPT_TIMESLOT_IN_USE

——表示创建和激活子网连接时涉及的时隙正在被 EMS 所占用。

EXCPT_PROTECTION_EFFORT_NOT_MET

——NMS 要求 SNC 的保护尽力程度 EMS 不能满足。

EXCPT_NOT_IN_VALID_STATE

——表示当前的状态不能进行某一操作。如不能删除处于激活状态的 SNC。

EXCPT_UNABLE_TO_COMPLY

——在 EMS 不能响应操作请求时抛出。

EXCPT_NE_COMM_LOSS

——与网元的通信中断。

EXCPT_CAPACITY_EXCEEDED

——当某一操作请求超出了 EMS 或网元的处理能力时抛出。

EXCPT_ACCESS_DENIED

——当操作的安全认证失败时抛出。

EXCPT_TOO_MANY_OPEN_ITERATORS

——当请求迭代操作时总的迭代器数目超出了 EMS 的迭代器数目限制时抛出。

EXCPT_UNSUPPORTED_ROUTING_CONSTRAINTS

——EMS 不支持设置的路由限制。

EXCPT_USERLABEL_IN_USE

——用户友好名称正在使用中，即友好名称已存在。

NameAndStringValue_T

定义：

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

说明：

NameAndStringValue_T 结构用于替代 OMG 定义的名值对列表（NVList）。从性能和开销的角度考虑，将 value 定义为 string 比将其定义为 any 类型更有效。本接口使用元组序列来表示对象的命名。

属性描述：

string name

——表示对象的名称。

string value

——表示对象的值。

NamingAttributes_T

定义：

```
typedef NVSList_T NamingAttributes_T;
```

说明：

表示对象的名称和取值。

本部分对象命名格式定义请参见引用文件 TMF 814（2003） “Multi-Technology Network Management Solution Set Document NML-EML Interface Version 3.0”中的“objectNaming.pdf”。

NamingAttributesList_T

定义：

```
typedef sequence <NamingAttributes_T> NamingAttributesList_T;
```

说明：

表示对象的名称和取值列表。

属性描述：

无。

ProcessingFailureException

定义：

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

说明：

表示处理失败抛出的异常。

属性描述：

ExceptionType_T exceptionType

——表示异常类型。

string errorReason

——表示产生异常的原因。是对前面的 ExceptionType 进行补充说明，对 errorReason 的具体格式没有定义，只要是一个说明出错原因的可读 string 即可。

Time_T

定义：

```
typedef string Time_T;
```

说明：

表示时间的类型定义，定义时间类型为一字符串。

其表示方式遵循 ITU-T X.208 中关于* 时间表示的定义，其格式为"yyyyMMddhhmmss.s[Z|{+|-}HHMm]"，

其中：

yyyy	"0000".."9999"	年
MM	"01".."12"	月
dd	"01".."31"	日
hh	"00".."23"	时
mm	"00".."59"	分
ss	"00".."59"	秒
Z	"Z"	为 UTC，而不是本地时间
{+|-}	"+" or "－"	与 UTC 时间的时差
HH	"00".."23"	时差中的小时部分
Mm	"00".."59"	时差中的分钟部分

NamingAttributesIterator_I 接口

说明：

表示迭代查询名称接口。

- 查询下一批名称数据（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out NamingAttributesList_T    nameList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批名称数据。

输入参数：

in unsigned long how_many

——表示返回数据的最大数目。

输出参数：

out NamingAttributesList_T　nameList

——表示查询得到的名称列表。

返回值：

boolean

——表示操作结果，为真表示操作成功，为假表示操作失败。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目(getLength）

定义：

```
unsigned long getLength ()
    raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——表示数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器(destroy）

定义：

```
void destroy ()
 raises (globaldefs::ProcessingFailureException);
```

说明：

用于删除迭代器。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

3.5.2 公用接口模块（module common）

本接口是管理对象的公用操作接口，其他管理对象类接口模块都要从本接口派生。

Common_I 接口

本接口用于实现一些公用的操作，可被其他管理对象继承。

- 设置友好名称（setUserLabel）

定义：

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

说明：

友好名称为 NMS 所有，是由 NMS 设置到对象上的，当 NMS 创建对象时，NMS 通过本接口设置对象的友好名称，一旦对象被创建，只有 NMS 可以通过本接口设置其友好名称。

输入参数：

in globaldefs::NamingAttributes_T objectName

——要设置友好名称的对象。

in string userLabel

——对象的友好名称。

in boolean enforceUniqueness

——表明是否要在 EMS 检查友好名称的惟一性。如果要检查惟一性且在 EMS 的同类对象中不惟一，本命令将失败。

输出参数：

无。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

（7）友好名称已存在（EXCPT_USERLABEL_IN_USE）。

- 设置 EMS 本地名称（setNativeEMSName）

定义：

```
void setNativeEMSName(
        in globaldefs::NamingAttributes_T objectName,
        in string nativeEMSName)
      raises(globaldefs::ProcessingFailureException);
```

说明：

设置对象的 EMS 本地名称。

输入参数：

in globaldefs::NamingAttributes_T objectName

——对象名称。

in string nativeEMSName

——EMS 本地名称。

输出参数：

无。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

- 设置所有者（setOwner）

定义：

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

说明：

设置对象的 EMS 本地名称。

输入参数：

in globaldefs::NamingAttributes_T objectName

——对象名称。

in string owner

——对象所有者名称。

输出参数：

无。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

- 设置附加信息（setAdditionalInfo）

定义：

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


};
```

说明：

设置对象的附加信息。

输入参数：

in globaldefs::NamingAttributes_T objectName

——对象名称。

inout globaldefs::NVSList_T additionalInfo

——对象附加信息。

输出参数：

inout globaldefs::NVSList_T additionalInfo

——对象附加信息。

返回值：

无。

异常：

（1）未实现此操作（EXCPT_NOT_IMPLEMENTED）。

（2）内部处理错误（EXCPT_INTERNAL_ERROR）。

（3）无效的输入（EXCPT_INVALID_INPUT）。

（4）实体未找到（EXCPT_ENTITY_NOT_FOUND）。

（5）与网元的通信失败（EXCPT_NE_COMM_LOSS）。

（6）操作无法完成（EXCPT_UNABLE_TO_COMPLY）。

3.5.3 通知管理模块（module notifications）

说明：

通知管理功能包括通知定购功能，通知上报功能和事件同步功能。本部分通知格式定义（心跳通知除外）请参见引用文件 TMF 814（2003）“Multi-Technology Network Management Solution Set Document NML-EML Interface Version 3.0”中的“OMGServicesUsage.pdf”；CosNotification::StructuredEvent的定义请参见“CosNotification.idl”。

心跳通知的事件头、事件体格式均符合TMF 814（2003）的定义，其中可过滤体部分定义见下表。

名　称	类　型	取值描述
“objectName”	string	发送心跳通知的系统名称
"emsTime"	globaldefs::Time_T	发送心跳通知的时间

ProbableCause_T

定义：

typedef string ProbableCause_T;

说明：

告警/事件原因。

ProbableCauseList_T

定义：

typedef sequence < ProbableCause_T > ProbableCauseList_T;

说明：

告警/事件原因列表。

EventList_T

定义：

typedef sequence <CosNotification::StructuredEvent> EventList_T;

说明：

表示事件信息列表。

属性描述：

无。

FileTransferStatus_T

定义：

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

说明：

表示文件的传输状态。当 EMS 与 NMS 通过 FTP 方式传输数据时，EMS 要向 NMS 上报文件的传输状态。

属性描述：

FT_IN_PROGRESS

——表示文件传送中。

FT_FAILED

——表示文件传送失败。

FT_COMPLETED

——表示文件传送完成。

NameAndAnyValue_T

定义：

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

};

说明：

表示对象的名称和取值，这里的对象取值是 any 类型，本结构主要用于事件通知的通知结构中。

属性描述：

string name

——表示对象名称。

any value

——表示对象取值。

EventList_T

定义：

typedef sequence <CosNotification::StructuredEvent> EventList_T;

说明：

表示事件信息列表。

属性描述：

无。

NVList_T

定义：

typedef sequence <NameAndAnyValue_T> NVList_T;

说明：

名值对列表，表示一个或多个对象的名称和取值对。

属性描述：

无。

ObjectType_T

定义：

```
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
};
```

说明：

对象类型。

属性描述：

OT_EMS

——表示 EMS。

OT_MANAGED_ELEMENT

——表示网元。

OT_MULTILAYER_SUBNETWORK

——表示子网。

OT_TOPOLOGICAL_LINK

——表示拓扑连接。

OT_SUBNETWORK_CONNECTION

——表示子网连接。

OT_PHYSICAL_TERMINATION_POINT

——表示物理终端点。

OT_CONNECTION_TERMINATION_POINT

——表示连接终端点。

OT_TERMINATION_POINT_POOL

——表示终端点池。

OT_EQUIPMENT_HOLDER

——表示设备容器。

OT_EQUIPMENT

——表示设备。

OT_PROTECTION_GROUP

——表示保护组。

OT_TRAFFIC_DESCRIPTOR

——表示通信描述符。

OT_AID

——表示未定义对象。

EventIterator_I 接口

说明：

用于提供对事件数据迭代操作接口。

- 迭代查询事件信息操作（next_n）

定义：

```
boolean next_n (
    in unsigned long how_many,
    out EventList_T eventList
    )
    raises (globaldefs::ProcessingFailureException);
```

说明：

通过迭代器查询下一批事件数据。

输入参数：

in unsigned long how_many

——查询的数目。

输出参数：

out EventList_T eventList

——表示事件数据列表。

返回值：

boolean

——表示操作结果。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 查询迭代器数据数目（getLength）

定义：

```
unsigned long getLength ()
        raises (globaldefs::ProcessingFailureException);
```

说明：

用于查询迭代器中剩余数据的数目。

输入参数：

无。

输出参数：

无。

返回值：

unsigned long

——数据数目。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

- 删除迭代器（destroy）

定义：

```
void destroy ()
        raises (globaldefs::ProcessingFailureException);
```

说明：

释放迭代器，释放被迭代器占用的内存。

输入参数：

无。

输出参数：

无。

返回值：

无。

异常：

内部处理错误（EXCPT_INTERNAL_ERROR）。

附 录 A
（规范性附录）
性能文件格式定义

本附录定义的性能文件格式用于性能管理中的PerformanceManagementMgr_I接口的查询历史性能数据(getAllHistoryPMData)操作，按本附录定义的文件格式EMS组织历史性能文件，NMS解析该文件，获得历史性能数据。

表A.1描述了性能文件的具体结构和文件中各字段的含义。

表 A.1 性能文件格式定义

类 型	组成定义	说 明
性能文件	表结构记录（1条）＋性能数据记录（*n*条）	每个表结构字段名称两边使用引号包含，不同的表结构字段名使用逗号分隔
表结构记录	"User Label","EMS Name","EMS Native Name","ME Name","ME Native Name","PTP Name","PTP Native Name","CTP Name","CTP Native Name","Layer Rate", "Granularity","Period Start Time","Monitored Time", "PM Parameter","Location","Value","Unit","Status","Nr Of Periods"	位于文件的第1行，用来描述字段的名称
性能数据记录	描述符记录（1条）＋性能数据记录（*n*条）	位于文件的第1至第*n*行，是具体对象的性能数据。每1条描述符记录对应*n*条性能值记录，每个文件里可以有*n*条描述符记录（*n*=[1..*n*]）
描述符记录	<User Label>,<EMS Name>,<EMS Native Name>,<ME Name>,<ME Native Name>,<PTP Name>,<PTP Native Name>,<CTP Name>,<CTP Native Name>,<Layer Rate>, <Granularity>	描述具体的对象信息和层速率及粒度周期
性能值记录	11个分隔符（英文逗号）＋ <Period Start Time>, <Monitored Time>,<PM Parameter>,<Location>, <Value>, <Unit>,<Status>,<Nr Of Periods>	性能记录
<User Label>	String	终端点的友好名称
<EMS Name>	String	EMS名称
<EMS Native Name>	String	EMS本地名称
<ME Name>	String	网元名称
<ME Native Name>	String	网元本地名称
<PTP Name>	String	物理终端点名称。
<PTP Native Name>	String	物理终端点本地名称
<CTP Name>	String	连接终端点名称
<CTP Native Name>	String	连接终端点本地名称
<Layer Rate>	String	层速率
<Granularity>	String	粒度周期
<Period Start Time>	String	格式定义与Time_T相同
<Monitored Time>	Integer	可选，表示监测性能的秒数，为空可表示整个周期都监测或不支持

表A.1（续）

类　型	组成定义	说　明
<PM Parameter>	String	性能参数名称
<Location>	String	性能监测位置
<Value>	Float	性能值
<Unit>	String	性能值单位
<Status>	"Valid" | "Incomplete" | "Invalid" | "Unavailable" | "Zero-suppressed"	表示性能数据的有效性。 Valid表示性能值在整个周期合法。 Incomplete表示性能值是一部分性能监测周期的。 Invalid表示数据有效但标记为非法。 Unavailable表示未监测到可用数据。 Zero-suppressed表示为零性能抑制数据
<Nr Of Periods>	Integer	一系列的零性能抑制数据可以由一个数据表示，零值间隔数由此字段表示。可选，空表示不支持
注：“<>”中的信息是具体的字段信息而不是名称		

参 考 文 献

【1】YD/T 1289.1-2003 同步数字体系（SDH）传送网网络管理技术要求 第1部分：基本原则

【2】YD/T 1289.4-2006 同步数字体系（SDH）传送网网络管理技术要求 第4部分：网元管理系统（EMS）与网络管理系统（NMS）接口功能

【3】YD/T 1289.5-2007 同步数字体系（SDH）传送网网络管理技术要求 第5部分：网元管理系统（EMS）-网络管理系统（NMS）接口通用信息模型

【4】TMF 513（2003） 多技术网络管理事务协定 版本3.0（Multi-Technology Network Management Business Agreement NML-EML Interface Version 3.0）

【5】TMF 608（2003） 多技术网络管理信息协定 版本3.0（Multi-Technology Network Management Information Agreement NML-EML Interface Version 3.0）

【6】TMF 814（2003） 多技术网络管理解决方案 版本3.0 （Multi-Technology Network Management Solution Set Document NML-EML Interface Version 3.0）

【7】TMF 814A（2003） 多技术网络管理实现声明模板和指南 版本3.0 （TM FORUM MTNM Implementation Statement (IS) Template and Guidelines NML-EML Interface Version 3.0）

中 华 人 民 共 和 国

通 信 行 业 标 准

同步数字体系（SDH）传送网网络管理技术要求 第6部分：基于IDL/IIOP技术的网元管理系统（EMS）—网络管理系统（NMS）接口信息模型

YD/T 1289.6-2009

*

人民邮电出版社出版发行

北京市崇文区夕照寺街14号A座

邮政编码：100061

北京新瑞铭印刷有限公司印刷

*

开本：880×1230 1/16 2010年1月第1版

印张：9.75 2010年1月北京第1次印刷

字数：274千字

ISBN 978-7-115-1992/10-54

定价：80元