TTC標準
Standard

ＪＴ-Ｍ３１２０

ＣＯＲＢＡ一般的ネットワークと NE レベル情報モデル

CORBA Generic Network and
NE Level Inoformation Model

第 1 版

2003 年 4 月 23 日制定

社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

＜目次＞

＜参考＞

１．本標準について

本標準は、CORBA による通信ネットワークの情報モデルについて定義と説明、及び通信インターフェイスとしての IDL 定義を記述しており、ITU-T 勧告 M.3120 に準拠している。

２．改版の履歴

版数	制定日	改版内容
第１版	２００３年４月２３日	制定

３．工業所有権

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、ＴＴＣホームページで御覧になれます。

４．その他

(1) 参照している勧告

ITU-T 勧告　Q.816，Q.816.1，X.780，X.780.1，M.3100，G.855.1，G.854.1，G.854.3，

G.854.6，G.854.8，G.854.10

(2) その他

本標準(および ITU-T 勧告 M.3200)では、ITU-T 勧告 M.3100 を参照している。しかし参照している勧告、国際標準との内容に差異がある場合は、参照している勧告、標準が優先するものとする。

５．標準作成部門

網管理専門委員会

CORBA 一般的ネットワークと NE レベル情報モデル
(2001 年)

1. 範囲（Scope）

本稿は、CORBA に基づくテレコミュニケーション・ネットワーク管理において使用される一般的なネットワーク情報モデルを定義する。これは、一組の一般的なインタフェースと定数を、インタフェース定義言語(Interface Definition Language)（IDL)で定義する。このドキュメントの意図するものは、ITU 勧告の X.721 と M.3100 に CMISE を用いて定義されるものと同様な一般的な CORBA/IDL モデルを定義することである。これらの一般的な IDL インタフェースは、 ATM や SONET/SDH などの特定のネットワーク技術を管理するために、様々な産業によって、拡張することができる。

2. 参考文献（References）

以下の ITU-T 勧告とその他の参考文献は、本文における参照を通して、この勧告(Recommendation)の規定を構成する規定を含んでいる。公表時点で、示される版は有効なものであった。すべての勧告と他の参考文献は、改正される場合がある；この勧告の全ての利用者は、このため、以下に列挙された勧告とその他の参考文献の最新版を適用する可能性について調査することを奨励する。現在、有効な ITU-T 勧告の表は定期的に公表される。

[1] ITU-T Draft Recommendation, Q.816, "*CORBA-Based TMN Services*".

[2] ITU-T Draft Recommendation, Q.816.1, "*CORBA-Based TMN Services Extensions to Support Coarse-Grained Interfaces*".

[3] ITU-T Draft Recommendation, X.780, "*TMN Guidelines for Defining CORBA Managed Objects*".

[4] ITU-T Draft Recommendation, X.780.1, "*TMN Guidelines for Defining Coarse-Grained CORBA Managed Objects*".

[5] ITU-T Draft Recommendation, M.3100 (1995) and Amendment 1 (1999), "*Generic Network Information Model*".

[6] ITU-T Draft Recommendation, G.855.1, "*GDMO Engineering Viewpoint for the Generic Network Level Model*".

[7] ITU-T Recommendation G.854.1 (11/96), "*Management of the transport network - Computational interfaces for basic transport network model*".

[8] ITU-T Recommendation G.854.3 (03/99), "*Computational viewpoint for topology management*".

[9] ITU-T Recommendation G.854.6 (03/99), "*Computational viewpoint for trail management*".

[10] ITU-T Recommendation G.854.8 (03/99), "*Computational viewpoint for pre-provisioned adaptation management*".

[11] ITU-T Recommendation G.854.10 (03/99*), "Computational viewpoint for pre-provisioned link connection management*".

[12] ITU-T Recommendation G.854.12 (03/99), "*Computational viewpoint for pre-provisioned link management*".

3. 用語と定義（Terms and Definitions）

この勧告は、ITU-T 勧告 Q.816 で定義される以下の略語*を使用する：
– イベントチャネル
（*TTC 注：「略語」ではなく「用語」と考えられる）

4. 略語と頭文字（Abbreviations and acronyms）

本勧告は、ITU-T 勧告 M.3100 で定義される以下の略語を使用する：

ARC	警報通知制御（Alarm Reporting Control ）
CTP	接続端点（Connection Termination Point）
GTP	グループ端点（Group Termination Point）
MOO	多重オブジェクト操作（Multiple Object Operation）
MP	多重点（Multi-Point）
QI	条件付抑制（Qualified Inhibit ）
TP	端点（Termination Point）
TTP	トレール端点（Trail Termination Point）

5. 一般的情報モデルの概要（Overview of the Generic Information Model）

5.1 IDL モデルの背景（Background of the IDL Models）

本稿の６節は、CORBA の一般的情報モデルのために一組の CORBA インタフェースを定義する。これらのインタフェースは、一組の M.3100 GDMO 管理オブジェクトクラスから手動で翻訳されており、これらは、細粒度の CORBA インタフェースのために、Q.816 と X.780 で与えられている TMN CORBA の枠組みとガイドラインに従っている。

6.5 節における細粒度インタフェースに付け加えて、ファサード(Façade)インタフェースに付随するセットが、6.6 節で定義される。これらのファサードインタフェースは、粗粒度(Coarse-Grained)の拡張フレームワークや、粗粒度 CORBA インタフェースをサポートするための、Q.816.1 と X.780.1 で与えられているガイドラインに従って定義されている。これらのファサードインタフェースの名前は、関連する細粒度インタフェースの名前に"_F"（アンダスコアのあとに大文字 F が続く）が追加されているものである。

6.5 節と 6.6 節では、インタフェースの IDL 署名は、アルファベット順の代わりに、継承順に規定される。継承される順番に列挙される理由は、「一パス方式の」ある IDL コンパイラの限界を克服するためである。

ドキュメントを読み易くするために、CORBA インタフェースのアルファベット順リストが、インタフェースが定義されている節と頁へのポインタとともに、本稿の内容表(Table of Content)の直後の初めに提供されている。

CORBA インタフェースが変換されるところのソースとなる GDMO 定義は、M.3100 とその修正版に定義されている。ただし、BasicLayerNetworkDomain と BasicSubnetwork を除かれており、これらは G.855.1 に定義されている。多くの場合、GDMO 管理オブジェクトのクラスの最新のバージョンが使用された。例えばネットワーク(Network)インタフェースは NetworkR1 GDMO 仕様に基づいている。M3100 にあるように、trailR2 は trailR1 の代替でないので、Trail95 は trailR1 から変換されるが、トレール（Trail）インタフェースは trailR2 から変換される。これらは、コントリビューションに基づいて変更されるため、IDL バージョンと M.3100 GDMO の間の今後の整合性が維持されると期待される。

IDL インタフェースの注釈欄の振舞いと記述文は、直接ソース GDMO 定義から取っている。ネットワークレベルリソースインタフェースの操作で起こる例外処理については、G.854 シリーズ GDMO 振舞い宣言の読み物が、例外処理の意味論を理解するのに必要かもしれない。

このドキュメントの IDL は構文誤りなしで首尾よくコンパイルされた。使用されるコンパイラは、値タイプと M4 マクロ能力を含んだ CORBA 2.3 準拠であることが要求される。

追加インタフェースは、必要であるにもかかわらず、上記に列挙されておらず、表中や 5.2 節の図中にも記載されていていないことに、注意すべきである。その例はファクトリーク

ラスである。さらに、ファクトリーファインダ、チャンネルファインダ、ターミネータ、ハートビート、および MOO サービスなどのように、他のクラスは勧告 Q.816.1 に定義されている。

5.2　インタフェース継承と包含関係（Interface Inheritance and Containment Relationship）

本節は、本稿で定義する CORBA インタフェースの継承と包含関係について示している。継承階層は、表 1、表 2、および図 1 に示される。表 1 では、最初のレベルのサブクラスが、アルファベット順に列挙されており、一方、表２では、M.3100 で使用されるそれらと同様のフラグメントに従って、オブジェクトクラスがグループ化されている。ファサードインタフェースは、対応する細粒度インタフェースと同じ継承階層関係に従っていることに注意せよ。

表１/M.3120 – CORBA インタフェース継承階層（アルファベット順）

最上位クラス	第１レベルサブクラス	第２レベルサブクラス	第３レベルサブクラス	第４レベルサブクラス	第５レベルサブクラス
管理オブジェクト	*AbstractLink*	LogicalLink			
		TopLink			
	AbstractLinkEnd	LogicalLinkEnd			
		TopLinkEnd			
	AccessGroup				
	AlarmSeverityAssignmentProfile				
	ArcIntervalProfile				
	CircuitEndPointSubgroup				
	CrossConnection				
	Equipment	CircuitPack			
		EquipmentHolder			
	ExternalPoint	ControlPoint			
		ScanPoint			
	Fabric				
	GTP				
	ManagedElement				
	ManagedElement Complex				
	MPCrossConnection				
	Network	LayerND	BasicLayerND		
	Pipe	LinkConnection			
		SNC			
		Trail			
	Trail95				
	Software				
	Subnetwork	BasicSubnetwork			
	TP	*CTPAbstract*	CTPSink	CTPBid	
			CTPSource		
		TTPAbstract	TTPSink	TTPBid	
			TTPSource		
		NetworkTP	*NetworkCTPAbstract*	NetworkCTPSink	NetworkCTPBid
				NetworkCTPSource	
			NetworkTTPAbstract	NetworkTTPSink	NetworkTTPBid
				NetworkTTPSource	
	TPPool				

注１：*インスタンス化不可能なクラスはイタリック体で示されている。*

注２：全てのインタフェースは、本稿の itut_m3120 モジュールに定義されている。ただし、最上位レベルのインタフェースである *ManagedObject* を除き、これは X.780 の itut_x780 モジュールに定義されている。

表２/M.3120－CORBA インタフェース継承階層（フラグメント順）

フラグメント	管理オブジェクトの第１レベルサブクラス	第２レベルサブクラス	第３レベルサブクラス	第４レベルサブクラス	第５レベルサブクラス
領域エンティティ	Network	LayerND	BasicLayerND		
	Subnetwork	BasicSubnetwork			
２値の伝送エンティティ	*Pipe*	LinkConnection			
		SNC			
		Trail			
	Trail95				
	AbstractLink	LogicalLink			
		TopLink			
単一要素の伝送エンティティ	GTP				
	TPPool				
	AccessGroup				
	CircuitEndPointSubgroup				
	TP	*CTPAbstract*	CTPSink	CTPBid	
			CTPSource		
		TTPAbstract	TTPSink	TTPBid	
			TTPSource		
		NetworkTP	NetworkCTPAbstract	NetworkCTPSink	NetworkCTPBid
				NetworkCTPSource	
			NetworkTTPAbstract	NetworkTTPSink	NetworkTTPBid
				NetworkTTPSource	
	AbstractLinkEnd	LogicalLinkEnd			
		TopLinkEnd			
NE	ManagedElement				
	ManagedElement Complex				
ハードウェア	Equipment	CircuitPack			
		EquipmentHolder			
ソフトウェア	Software				
クロスコネクト	Fabric				
	CrossConnection				
	MPCrossConnection				
障害管理	AlarmSeverityAssignmentProfile				
	ArcIntervalProfile				
テレメトリ	*ExternalPoint*	ControlPoint			
		ScanPoint			

注１：*インスタンス化不可能なクラスはイタリック体で示されている。*

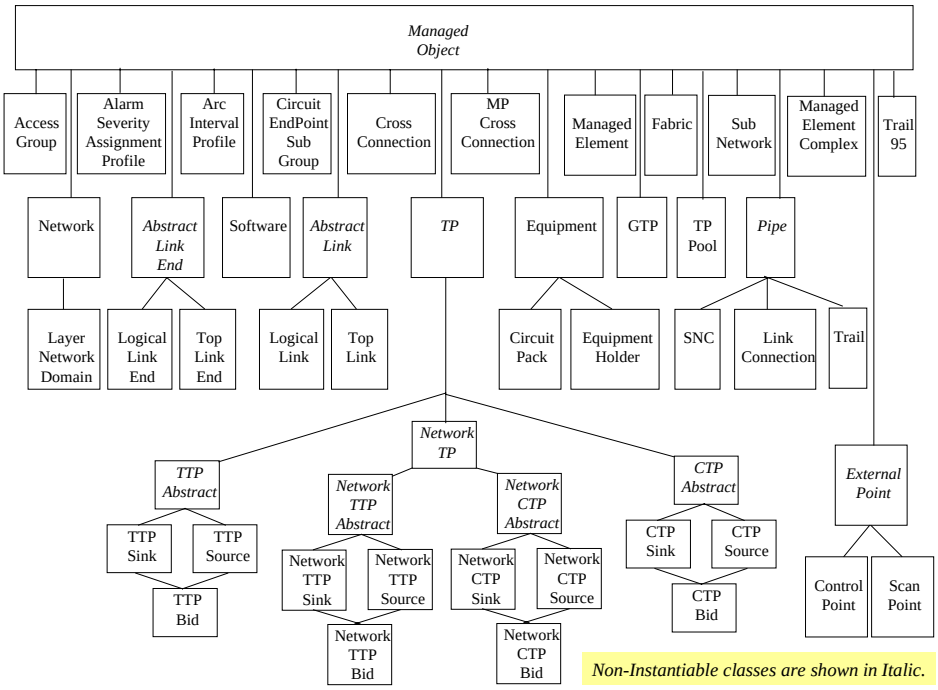

図１/M.3120－インタフェース継承階層

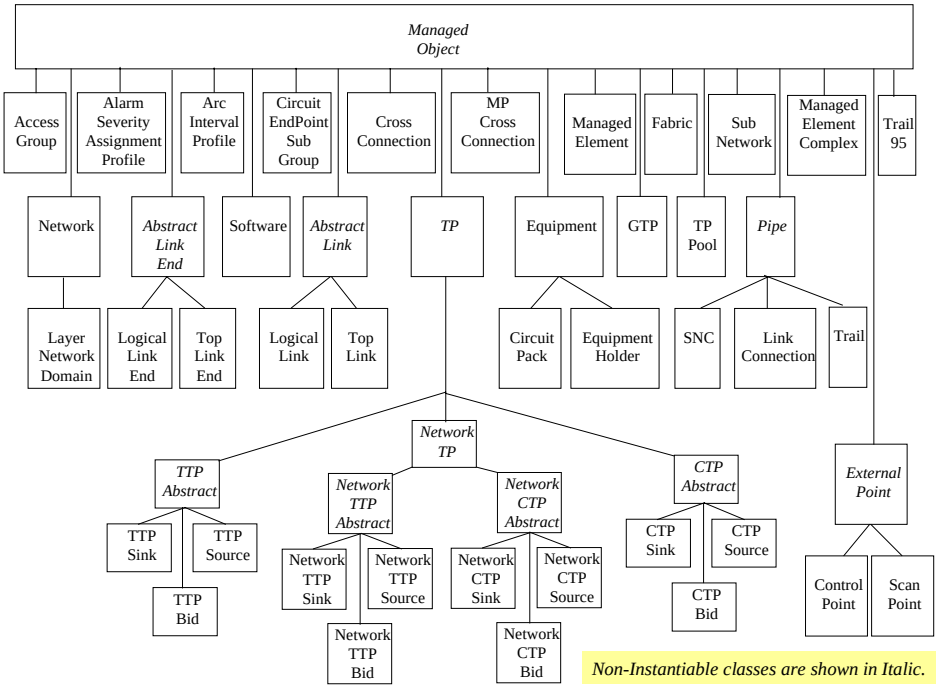

図２/M.3120－ValueType 継承階層

すべてのオブジェクトクラスが、枠組みのドキュメント X.780 の itut_x780 モジュールに定義されている ManagedObject のオブジェクトクラスを除いて、このドキュメントの itut_m3120 モジュールに定義されていることに注意せよ。

以下の表3には、ネームバインディング関係が、下位ー上位の対の形で示されている。それはまた、各関係対について、削除ポリシと生成の制限を示している。明らかに注意書きがない場合、下位のオブジェクトの生成がシステム管理プロトコルの結果である。(即ち、ファクトリインタフェース上の管理システムからの生成操作による)

表3/M.3120－CORBAインターフェース包含関係

下位クラス	上位クラス	生成	削除ポリシ
AccessGroup	LayerND	No auto naming	Only if no contained
AlarmSeverityAssignmentProfile	ManagedElement	--	Only if no contained
ArcIntervalProfile	ManagedElement	Auto created, 0 or 1	Not deletable
	ManagedElementComplex	Auto created, 0 or 1	Not deletable
	Network	Auto created, 0 or 1	Not deletable
CircuitPack	CircuitPack	Auto created (by Insertion)	Delete contained object
	EquipmentHolder	Auto created (by Insertion)	Only if no contained
	EquipmentHolder	--	Only if no contained
	EquipmentHolder	--	Delete contained object
CircuitEndPointSubgroup	Not in this document		
CTPBid	TTPBid	--	Only if no contained
CTPSink	TTPBid	--	Only if no contained
	TTPSink	--	Only if no contained
CTPSource	TTPBid	--	Only if no contained
	TTPSource	--	Only if no contained
ControlPoint	Equipment	--	--
	ManagedElement	--	--
CrossConnection	Fabric	Auto created (by Connect)	Only if no contained
	MPCrossConnection	Auto created (by Connect)	Only if no contained
Equipment	Equipment	--	Only if no contained
	ManagedElement	--	Only if no contained
EquipmentHolder	Equipment	--	Only if no contained
	EquipmentHolder	--	Only if no contained
Fabric	ManagedElement	--	Only if no contained
GTP	Fabric	Auto created (by addTPsToGTP)	Not deletable (By removeTPsFromGTP)
LayerND	Network	No auto-naming	Only if no contained
LinkConnection	LayerND	Auto created	Not deletable
	TopLink	Auto created	Not deletable
LogicalLink	LayerND	Auto created	Not deletable
LogicalLinkEnd	LayerND	Auto created	Not deletable
	Subnetwork	Auto created	Not deletable
ManagedElement	Network	Auto created (at initialization)	Not deletable
ManagedElementComplex	Network	Auto created (at initialization)	Not deletable
MPCrossConnection	Fabric	Connect	Not deletable (By the disconnect operation)
Network	Root	Auto created (at initialization)	Not deletable
NetworkCTPBid	LayerND	Auto created	Not deletable

下位クラス	上位クラス	生成	削除ポリシ
	Subnetwork	Auto created	Not deletable
NetworkCTPSink	LayerND	Auto created	Not deletable
	Subnetwork	Auto created	Not deletable
NetworkCTPSource	LayerND	Auto created	Not deletable
	Subnetwork	Auto created	Not deletable
NetworkTTPBid	LayerND	--	Only if no contained
	Subnetwork	--	Only if no contained
NetworkTTPSink	LayerND	--	Only if no contained
	Subnetwork	--	Only if no contained
NetworkTTPSource	LayerND	--	Only if no contained
	Subnetwork	--	Only if no contained
ScanPoint	Equipment	--	--
	ManagedElement	--	--
Software	Equipment	--	Only if no contained
	ManagedElement	--	Only if no contained
	Software	--	Only if no contained
Subnetwork	LayerND		Only if no contained
SNC	Subnetwork	Auto created	Not deletable
TopLink	LayerND	Auto created	Not deletable
TopLinkEnd	LayerND	Auto created	Not deletable
	Subnetwork	Auto created	Not deletable
TPPool	Fabric	Auto created (by addTPsToTPPool)	Not deletable (By removeTPsFromTPPool)
Trail	LayerND	Auto created	Not deletable
Trail95	LayerND	Auto created	Not deletable
TTPBid	ManagedElement	--	Only if no contained
TTPSink	ManagedElement	--	Only if no contained
TTPSource	ManagedElement	--	Only if no contained

注：

- 生成制限
  - “No auto naming”とは、下位オブジェクト RDN が生成要求で必要とされることを意味する。
  - “Auto created”とは、例えば上位オブジェクトが生成されるか、あるいは、管理システムにおけるイベント発生の結果として、下位オブジェクトが自動的に生成されることを意味している。“Auto created”は、ネームバインディングモジュールの“creation permitted”パラメータをブーリアン値の false に割り付けることによって示される。
  - “0 or 1”とは、下位オブジェクトのせいぜい一つのインスタンスが上位オブジェクトの下に存在することを意味する。
  - “Insertion”とは、下位オブジェクトを代表するサーキットボードの挿入の際に、下位オブジェクトが自動的に生成されることを意味している。
  - “Connect”とは、ファブリックオブジェクトの“connect”操作の結果として下位オブジェクトが生成されることを意味している。
  - ネーミング木の最上位であるクラスについては、ローカルルートのネーミングコンテキストが、ローカル手順によって定義され、名前付けされる。ネーミング木の最上位として利用されるクラスのためのネームバインディングモジュールで、上位の名前パラメータはブランクのままとなる。
- ポリシの削除
  - “Only if no contained”とは、下位オブジェクトが、もし、それ（即ち、下位オブジェクト）が、包含オブジェクトを持っていない場合にのみ削除できることを意味している。

– “Not deletable”とは、下位オブジェクトが管理プロトコルによって、直接的に削除されることはないということを意味している。
– “Delete contained object”とは、下位オブジェクトの包含オブジェクトがもしあれば、下位オブジェクトの削除の際に、自動的にまた削除されることを意味している。

インタフェースの包含関係は、また以下の図 3 と 4 に示されている。NE レベルのオブジェクトクラスが図 3 に示されている、一方、 ネットワークレベルのオブジェクトクラスは図 4 に示されている。図において、矢印は, 下位の(含まれる)オブジェクトから、上位の(含んでいる)オブジェクトまでを指し示すために使用される。逆矢印は、自己包含関係を示しており、即ち、オブジェクトクラスのインスタンスは、同じオブジェクトクラスの他のインスタンスを含むことができる。濃度が包含図に示されていないことに注意せよ。

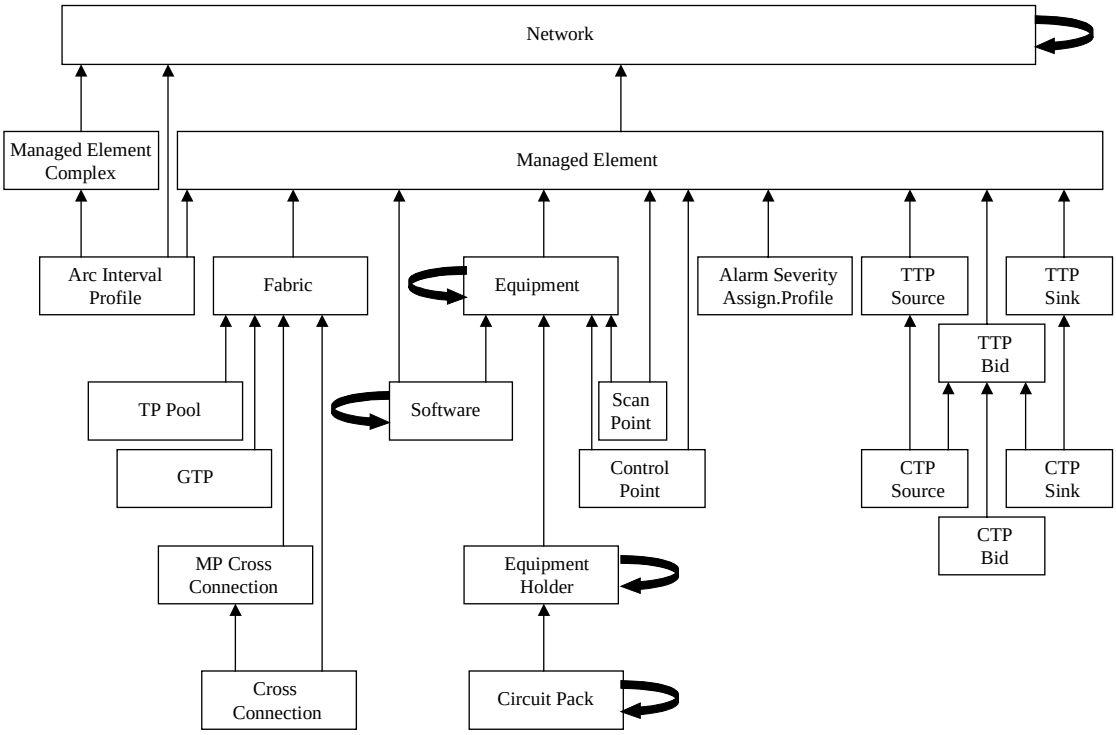

図 3 /M.3120－NE レベルオブジェクトクラスの包含関係

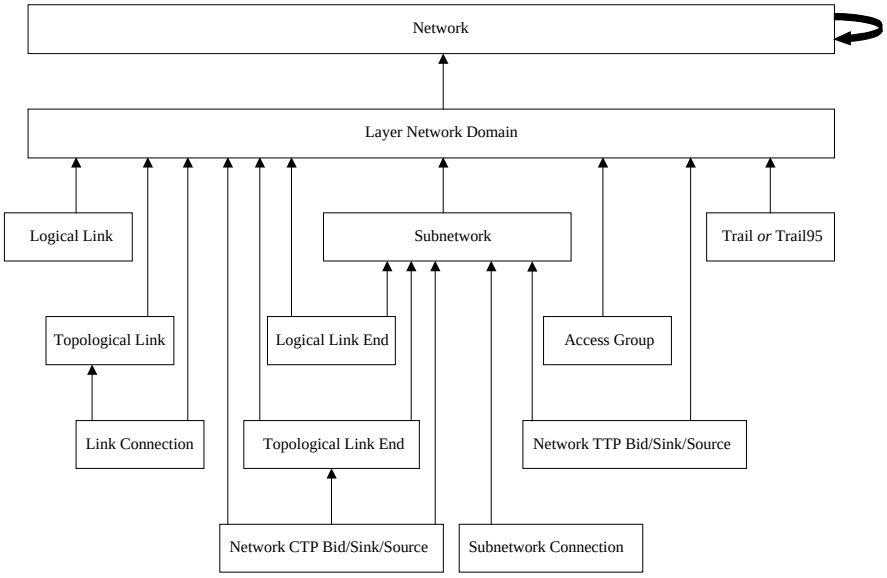

図４/M.3120－ネットワークレベルオブジェクトクラスの包含関係

5.3　IDL ファイルの保存とモデルのドキュメント変換（IDL File Storing and Documentation Convention of the Model）

一般的な情報モデルの IDL 仕様は、IDL モジュール"itut_m3120"に定義されて、６節と７節に記載されている。

6 節はインポートタイプ、フォワード宣言、構造、typedefs、例外、細粒度のインタフェース、およびファサードインタフェースのための IDL コードを含む。6 節の IDL コードは、システム上の、IDL コンパイラによって使用される検索経路に位置する"itut_m3120.idl"という一次ファイルに意図的に格納されている。

6.6 節はファサードインタフェースのための IDL コードを含む。管理システムが、細粒度インタフェースのみをサポートするだけであるならば、6.6 節は、IDL から省略することができる。

7 節は、定数のための IDL コードを含む。7 節の IDL コードが"itut_m3120const.idl"という二次的ファイルに意図的に格納されていて、それは、一次ファイル itut_m3120.idl と同じディレクトリに位置している。この二次ファイルと他の IDL ファイルは、itut_x780.idl などのように、一次的な IDL ファイル itut_m3120.idl によって含まれている。

仕様の(1)読み易さと(2)実現の容易性を改良するために、文書化の慣習がこれらの節に採用されている。読み易さの目的において, 小区分番号と見出しが、IDL コードに使用される。(たとえば、インポート・タイプ、フォワード宣言、構造と typedefs、例外、個々のインタフェース、および定数タイプなど)。実現の容易性の目的のため、小区分番号、レベル２以

下の見出しは、IDL コメントに、閉じ込められている。このように、ドキュメントから機械が実行可能な IDL ファイルを抽出するとき、これらの副節番号と見出しは、取り除く必要はない。

6. 情報モデル IDL(Information Model IDL)

```
#ifndef _itut_m3120_idl_
#define _itut_m3120_idl_

#include <itut_x780.idl>
#include <itut_x780_1.idl>
#include <itut_m3120const.idl>


#pragma prefix "itu.int"


/**
本 IDL コードはシステム上の IDL コンパイラにより使われる検索パス中にある"itut_m3120.idl"と命名されたファイルに保存されることが意図されている。
*/




/**
itut_m3120 であるこのモジュールは M.3100 と G.855.1 で定義されたオブジェクトに基づく IDL の定義を含んでいる。このファイル中の IDL の定義はオブジェクトインタフェースである。定数値は別ファイルに保持されているが、同じ"itut_m3120"モジュールに含まれている。
*/
module itut_m3120
{


/**
```

6.1 流用(Imports)

```
*/

/**
itut_x780 から流用されたタイプ
*/
       typedef itut_x780::AdministrativeStateType AdministrativeStateType;
       typedef itut_x780::AdministrativeStateTypeOpt AdministrativeStateTypeOpt;
       typedef itut_x780::AttributeSetType AttributeSetType;
       typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
       typedef itut_x780::BooleanTypeOpt BooleanTypeOpt;
       typedef itut_x780::ExternalTimeType ExternalTimeType;
       typedef itut_x780::DeletePolicyType DeletePolicyType;
       typedef itut_x780::Istring Istring;
       typedef itut_x780::IstringSetType IstringSetType;
       typedef itut_x780::NameBindingType NameBindingType;
       typedef itut_x780::MONameType MONameType;
       typedef itut_x780::MONameSeqType MONameSeqType;
       typedef itut_x780::MONameSetType MONameSetType;
       typedef itut_x780::ObjectClassType ObjectClassType;
       typedef itut_x780::ObjectClassSetType ObjectClassSetType;
       typedef itut_x780::OperationalStateType OperationalStateType;
       typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
       typedef itut_x780::ProbableCauseType ProbableCauseType;
       typedef itut_x780::StringSetType StringSetType;
       typedef itut_x780::UIDType UIDType;
       typedef itut_x780::UsageStateType UsageStateType;


/**
itut_x780 から流用された例外
```

```
itut_x780::ApplicationError
itut_x780::CreateError
itut_x780::DeleteError

根拠:
IDL コンパイラに支援される完全に範囲指定された valuetype および例外名のための#define の使用は、CORBA コンパイラが十分な C++プリプロセッサを持っていると仮定する。全てのプリプロセッサが許可されることを CORBA 仕様が示していたとしても、全ての CORBA コンパイラがこれをもっているわけではない。

完全に明瞭のために、それらは流用した valuetype および例外の実際に完全範囲指定された名前と置き換えられるべきである。
*/

/**
itut_x780 から流用されたインタフェース

itut_x780::ManagedObject
itut_x780::ManagedObjectFactory

根拠:
typedef は基本タイプの解釈上の改良や要求時に使用されるべきである。

流用されたインタフェース定義から局所範囲指定中でインタフェースを定義するための typedef の使用は
潜在的に厄介な問題である。局所範囲名のための意味に違いは無いので、完全に範囲指定された名前が、明瞭のためには使用可能かつ使用されるべきである。

typedef された局所名のための異なったリポジトリ ID を期待しうることを望んでいない。事実、IR はインタフェース、valuetype、抽象インタフェースのための typedef オブジェクトを生成しないであろう。
*/

/**
itut_x780 から流用された Valuetype

itut_x780::ManagedObjectValueType

*/

/**
```

6.2 先行宣言(Forward Declarations)

```
*/

/**
インタフェース先行宣言
*/
       interface AbstractLink;
       interface AbstractLinkEnd;
       interface AccessGroup;
       interface AlarmSeverityAssignmentProfile;
       interface ArcIntervalProfile;
       interface BasicLayerND;
       interface BasicSubnetwork;
       interface CircuitEndPointSubgroup;
       interface CircuitPack;
       interface ControlPoint;
       interface CTPAbstract;
       interface CTPBid;
       interface CTPSink;
       interface CTPSource;
       interface CrossConnection;
       interface Equipment;
       interface EquipmentHolder;
       interface ExternalPoint;
       interface Fabric;
```

```
        interface GTP;
        interface LayerND;
        interface LinkConnection;
        interface LogicalLink;
        interface LogicalLinkEnd;
        interface ManagedElement;
        interface ManagedElementComplex;
        interface MPCrossConnection;
        interface Network;
        interface NetworkCTPAbstract;
        interface NetworkCTPBid;
        interface NetworkCTPSink;
        interface NetworkCTPSource;
        interface NetworkTP;
        interface NetworkTTPAbstract;
        interface NetworkTTPBid;
        interface NetworkTTPSink;
        interface NetworkTTPSource;
        interface Pipe;
        interface ScanPoint;
        interface Software;
        interface Subnetwork;
        interface SNC;
        interface TopLink;
        interface TopLinkEnd;
        interface TP;
        interface TPPool;
        interface Trail;
        interface Trail95;
        interface TTPAbstract;
        interface TTPBid;
        interface TTPSink;
        interface TTPSource;

/**
Valuetype 先行宣言
*/
        valuetype AbstractLinkValueType;
        valuetype AbstractLinkEndValueType;
        valuetype AccessGroupValueType;
        valuetype AlarmSeverityAssignmentProfileValueType;
        valuetype ArcIntervalProfileValueType;
        valuetype BasicLayerNDValueType;
        valuetype BasicSubnetworkValueType;
        valuetype CircuitEndPointSubgroupValueType;
        valuetype CircuitPackValueType;
        valuetype ControlPointValueType;
        valuetype CTPAbstractValueType;
        valuetype CTPBidValueType;
        valuetype CTPSinkValueType;
        valuetype CTPSourceValueType;
        valuetype CrossConnectionValueType;
        valuetype EquipmentValueType;
        valuetype EquipmentHolderValueType;
        valuetype ExternalPointValueType;
        valuetype FabricValueType;
        valuetype GTPValueType;
        valuetype LayerNDValueType;
        valuetype LinkConnectionValueType;
        valuetype LogicalLinkValueType;
        valuetype LogicalLinkEndValueType;
        valuetype ManagedElementValueType;
        valuetype ManagedElementComplexValueType;
        valuetype MPCrossConnectionValueType;
        valuetype NetworkValueType;
        valuetype NetworkCTPAbstractValueType;
        valuetype NetworkCTPBidValueType;
        valuetype NetworkCTPSinkValueType;
```

```
        valuetype NetworkCTPSourceValueType;
        valuetype NetworkTPValueType;
        valuetype NetworkTTPAbstractValueType;
        valuetype NetworkTTPBidValueType;
        valuetype NetworkTTPSinkValueType;
        valuetype NetworkTTPSourceValueType;
        valuetype PipeValueType;
        valuetype ScanPointValueType;
        valuetype SoftwareValueType;
        valuetype SubnetworkValueType;
        valuetype SNCValueType;
        valuetype TopLinkValueType;
        valuetype TopLinkEndValueType;
        valuetype TPValueType;
        valuetype TPPoolValueType;
        valuetype TrailValueType;
        valuetype TTPAbstractValueType;
        valuetype TTPBidValueType;
        valuetype TTPSinkValueType;
        valuetype TTPSourceValueType;


/**
Typedef 先行宣言
*/
        typedef MONameType AbstractLinkNameType;
        typedef MONameType AbstractLinkEndNameType;
        typedef MONameType AccessGroupNameType;
        typedef MONameType AlarmSeverityAssignmentProfileNameType;
        typedef MONameType ArcIntervalProfileNameType;
        typedef MONameType BasicLayerNDNameType;
        typedef MONameType BasicSubnetworkNameType;
        typedef MONameType CircuitEndPointSubgroupNameType;
        typedef MONameType CircuitPackNameType;
        typedef MONameType ControlPointNameType;
        typedef MONameType CTPAbstractNameType;
        typedef MONameType CTPBidNameType;
        typedef MONameType CTPSinkNameType;
        typedef MONameType CTPSourceNameType;
        typedef MONameType CrossConnectionNameType;
        typedef MONameType EquipmentNameType;
        typedef MONameType EquipmentHolderNameType;
        typedef MONameType ExternalPointNameType;
        typedef MONameType FabricNameType;
        typedef MONameType GTPNameType;
        typedef MONameType LayerNDNameType;
        typedef MONameType LinkConnectionNameType;
        typedef MONameType LogicalLinkNameType;
        typedef MONameType LogicalLinkEndNameType;
        typedef MONameType ManagedElementNameType;
        typedef MONameType ManagedElementComplexNameType;
        typedef MONameType MPCrossConnectionNameType;
        typedef MONameType NetworkNameType;
        typedef MONameType NetworkCTPAbstractNameType;
        typedef MONameType NetworkCTPBidNameType;
        typedef MONameType NetworkCTPSinkNameType;
        typedef MONameType NetworkCTPSourceNameType;
        typedef MONameType NetworkTPNameType;
        typedef MONameType NetworkTTPAbstractNameType;
        typedef MONameType NetworkTTPBidNameType;
        typedef MONameType NetworkTTPSinkNameType;
        typedef MONameType NetworkTTPSourceNameType;
        typedef MONameType PipeNameType;
        typedef MONameType ScanPointNameType;
        typedef MONameType SoftwareNameType;
        typedef MONameType SubnetworkNameType;
        typedef MONameType SNCNameType;
        typedef MONameType TopLinkNameType;
```

```
	typedef MONameType TopLinkEndNameType;
	typedef MONameType TPNameType;
	typedef MONameType TPPoolNameType;
	typedef MONameType TrailNameType;
	typedef MONameType Trail95NameType;
	typedef MONameType TTPAbstractNameType;
	typedef MONameType TTPBidNameType;
	typedef MONameType TTPSinkNameType;
	typedef MONameType TTPSourceNameType;


/**
```

6.3 構造体と Typedef(Structures and Typedefs)

```
*/

/**
```

6.3.1 構造体と Typedef– 1(Structures and Typedefs – 1)

```
下記に定義された構造体や typedef は英数順に並んでいる。
前方での構造体や typedef 宣言が CORBA IDL 中では許可されない問題を解決するために行われた。
*/
	typedef UIDType CharacteristicInfoType;

	typedef sequence<CharacteristicInfoType> CharacteristicInfoSeqType;

	typedef sequence<CharacteristicInfoType> CharacteristicInfoSetType;

/**
ExternalTimeTypeOpt はオプションタイプである。もし、識別子が真の場合は値が与えられ、他の場合は値は空
(nil)である。
*/
	union ExternalTimeTypeOpt switch (boolean)
	{
		case TRUE:
			ExternalTimeType	val;
	};

/**
失敗は操作中に遭遇した論理的もしくは資源の問題を伝える。
*/
	typedef short ProblemCauseType;

/**
下記の値は M.3100 から借用された。
*/

/**
追加された M.3100 にない値である unknown = -1 は M.3100 は未知のために問題原因を整数か(以下に示す様に)空
の間の選択として定義するためここで追加された。空を選択するかわりに、原因不明の問題は整数値の-1 により表され
るであろう。
*/
	const ProblemCauseType
		problemCauseUnknown = -1;
	const ProblemCauseType
		problemCauseNoSuchTpInstance = 0;
	const ProblemCauseType
		problemCauseNoSuchGtpInstance = 1;
	const ProblemCauseType
		problemCauseNoSuchTpPoolInstance = 2;
	const ProblemCauseType
		problemCauseMismatchingTpInstance = 3;
	const ProblemCauseType
```

```
              problemCauseMismatchingGtpInstance = 4;
       const ProblemCauseType
              problemCausePartOfGtp = 5;
       const ProblemCauseType
              problemCauseInvolvedInCrossConnection = 6;
       const ProblemCauseType
              problemCauseMemberOfTpPool = 7;
       const ProblemCauseType
              problemCauseAlreadyMemberOfGtp = 8;
       const ProblemCauseType
              problemCauseNoTpInTpPool = 9;
       const ProblemCauseType
              problemCauseNoMoreThanOneTpIsAllowed = 10;
       const ProblemCauseType
              problemCauseNoMoreThanTwoTpsAreAllowed = 11;
/**
alreadyConnected はクロスコネクトされることを要求した 2 つの端点が既に involvedInCrossConnection であり 、一つ以上の端点がクロスコネクトされているが互いにではないことを示すために使用される、に対してクロスコネクトされていた事を示すために使用される。
*/
       const ProblemCauseType
              problemCauseAlreadyConnected = 12;
       const ProblemCauseType
              problemCauseNotAlreadyConnected = 13;

       struct FailedLogicalProblemType
       {
              MONameSetType       incorrectInstances;
              ProblemCauseType    problemCause;
       };

       enum FailedChoice
       {
              failedChoiceLogicalProblem,
              failedChoiceResourceProblem
       };

       union FailedType switch (FailedChoice)
       {
              case failedChoiceLogicalProblem:
                     FailedLogicalProblemType   logicalProblem;
              case failedChoiceResourceProblem:
                     short                      resourceProblem;
                     // -1 means unknown.
                     // no other constances have been defined for
                     // resouceProblem yet.
       };

       typedef sequence<MONameSetType> MONameSetSetType;

       enum SignalRateChoice
       {
              signalRateChoiceCharacteristicInfo,
              signalRateChoiceObjectClass
       };

       union SignalRateType switch (SignalRateChoice)
       {
              case signalRateChoiceCharacteristicInfo:
                     CharacteristicInfoType     characteristicInfo;
              case signalRateChoiceObjectClass:
                     ObjectClassType            objectClass;
       };

       typedef sequence<SignalRateType> SignalRateSetType;

/**
```

```
集合タイプ順序は重要ではない。
*/
       typedef sequence<TPNameType> TPNameSetType;

/**
シーケンスタイプ順序は重要である。
*/
       typedef sequence<TPNameType> TPNameSeqType;

/**
内部の順序は重要である。外部の順序は重要ではない。
*/
       typedef sequence<TPNameSeqType> TPNameSeqSetType;



/**
```

6.3.2 構造体と Typedef –2(Structures and Typedefs – 2)

```
下記に定義された構造体や typedef は英数順に並んでいる。
*/

       typedef sequence<AbstractLinkNameType> AbstractLinkNameSetType;

       typedef sequence<AbstractLinkEndNameType> AbstractLinkEndNameSetType;

       typedef sequence<AccessGroupNameType> AccessGroupNameSetType;

/**
警報重要度コード(Alarm Severity Code)
*/
       enum AlarmSeverityCodeType
       {
              alarmSeverityCodeNonalarmed,
              alarmSeverityCodeMinor,
              alarmSeverityCodeMajor,
              alarmSeverityCodeCritical,
              alarmSeverityCodeWarning
       };

       union AlarmSeverityCodeTypeOpt switch (boolean)
       {
              case TRUE:
                     AlarmSeverityCodeType val;
       };

/**
警報重要度割当(Alarm Severity Assignment)。それぞれの警報重要度割当構造体は(固有識別子(Unique ID)が
ある)特定の問題を識別し、それから問題がサービスに影響するかサービスに影響しないかサービスに依存しないか割り
当てられた警報重要度コードを提供する。この構造体は通常 AlarmSeverityAssignmentList の一部である。
*/
       struct AlarmSeverityAssignmentType
       {
              ProbableCauseType   problem;
              AlarmSeverityCodeTypeOpt severityAssignedServiceAffecting;
              AlarmSeverityCodeTypeOpt severityAssignedNonServiceAffecting;
              AlarmSeverityCodeTypeOpt severityAssignedServiceIndependent;
       };

/**
警報重要度割当リスト(Alarm Severity Assignment Lists)はオブジェクトクラスのインスタンス中に存在しうる
全ての異常状態リストを提供し、それぞれの状態のために割当られた警報重要度情報(minor、major 等)を表す。
*/
```

```
	typedef sequence<AlarmSeverityAssignmentType>
		AlarmSeverityAssignmentSetType;

/**
警報様態(Alarm Status)はオブジェクトに関係のある異常な状態の発生を示す。そのタイプの属性はまた特定の資源
に関連した警報状態の要約した指標として機能しうる。それは警報、しきい値状況のような未決の警報条件あるいは(要
約した指標として使用された時)活発な警報条件の中で最も高い重要度の存在を示すために使用される。
要約した指標として使われた時、重要度の順序は以下のとおり(高重要度から低い方へ並んでいる)である。
	activeReportable-Critical
	activeReportable-Major
	activeReportable-Minor
	activeReportable-Indeterminate
	activeReportable-Warning
	activePending
	cleared.

警報を発行しない管理オブジェクトは値"Indeterminate"を使用するべきである。
*/
	enum AlarmStatusType
	{
		alarmStatusCleared,
		alarmStatusActiveReportableIndeterminate,
		alarmStatusActiveReportableWarning,
		alarmStatusActiveReportableMinor,
		alarmStatusActiveReportableMajor,
		alarmStatusActiveReportableCritical,
		alarmStatusActivePending
	};

/**
ArcControlRequest: ArcTimeType の後で定義した
*/

/**
リスト中の想定原因はもしオブジェクトの arcState が"alm"状態でなければ警告通報から抑止される。空のリストは
資源がサポートする想定原因が全て抑止されていることを示す。これらの想定原因指示全てが抑止されている時挙げられ
るならば、オブジェクトの currentProblemList は想定原因は normal として更新されるであろう、しかしそれらの
挙げられた想定原因のための警報様態は'activePending'であろう。
*/
	typedef sequence<ProbableCauseType> ArcProbableCauseSetType;

/**
ArcAlarmDetailSetType は警報報告制御(Alarm Report Control)(ARC)のこの追加機能をサポートするインタフ
ェースに対する arcRetrieveAlarmDetail 操作の中で返される。

操作の返りは警報状態の警報詳細のリストである。おのおのの警報状態のために、警報詳細は想定要因、知覚した重要度、
イベントタイム(オプション)、警報様態を含む。もし警報状態の想定原因が ARC 制御下(即ち資源オブジェクトの
ArcProbableCauseSetType 中に含まれている) ならば警報状態の警報様態の値は"Pending"であろう。 それゆえ、
ArcAlarmDetailSetType 中の想定要因のために、もし警報様態の値が知覚した重要度と同じでなかったら、考えられ
る要因は ARC 制御下である。
*/
	struct ArcAlarmDetailType
	{
		ProbableCauseType   probableCause;
		PerceivedSeverityType      perceivedSeverity;
		ExternalTimeTypeOpt eventTime;
		AlarmStatusType            alarmStatus;
	};

	typedef sequence<ArcAlarmDetailType> ArcAlarmDetailSetType;

/**
以下に示す属性はオブジェクトの未警報 - 条件付抑制(QI) 状態(Not Alarmed - Qualified Inhibit (QI)
state)の様態、即ち nalmQualifiedInhibit を定義する。
*/
```

```
	enum ArcQIStatusType
	{
		arcQIStatusNotApplicable,
		arcQIStatusNalmNotReported,
		arcQIStatusNalmCountDown
	};

/**
下記の属性はオブジェクトの警報報告制御 (ARC)状態を定義する。与えられたオブジェクトに対してサポートされること
が要求される初期状態と ARC 状態の集合はオブジェクトクラスの振舞いの中で定義されるべきである。
*/
	enum ArcStateType
	{
		arcStateAlm,
		arcStateNalm,
		arcStateNalmQualifiedInhibit,
		arcStateNalmTimedInhibit
	};

/**
ARC time の値は 0 から 5940 分(99 時間)のレンジを持つ。FALSE の時は調整されていないことを意味する。
*/
	union ArcTimeType switch (boolean)
	{
		case TRUE:
			short  arcInterval;
	};

/**
Arc 制御要求(Arc Control Request)
*/
	struct ArcControlRequestType
	{
		ArcTimeType  arcTime;
		ArcStateType newState;
	};

	struct BandwidthComponentType
	{
		unsigned long ingress;
		unsigned long egress;
	};

	typedef sequence<BandwidthComponentType> BandwidthSeqType;

/**
下記の属性は利用可能なリンク接続の数、あるいは帯域幅のいずれかとして表現されるリンクの利用可能な容量を示す。
*/
	enum CapacityChoice
	{
		capacityChoiceNumberOfLinkConnections,
		capacityChoiceBandwidth
	};

	union CapacityType switch (CapacityChoice)
	{
		case capacityChoiceNumberOfLinkConnections:
			long               numberOfLinkConnections;
		case capacityChoiceBandwidth:
			BandwidthSeqType   bandwidth;
	};

	struct CapacitiesType
	{
		CapacityType       availableLinkCapacity;
		CapacityType       maxProvisionableCapacity;
		CapacityType       potentialLinkCapacity;
```

```
            CapacityType        provisionedLinkCapacity;
      };

      typedef sequence<long> ChannelSeqType;

/**
CharacteristicInfoType, CharacteristicInfoSeqType, CharacteristicInfoSetType は
6.3.1 節で定義される。
*/

/**
サーキット方向性(Circuit Directionality)はサーキット終点サブグループ内のサーキット方向を示す。
*/
      enum CircuitDirectionalityType
      {
            circuitDirectionalityOnewayOut,
            circuitDirectionalityOnewayIn,
            circuitDirectionalityTwoway
      };

      typedef sequence<CircuitPackNameType> CircuitPackNameSetType;

/**
下記の属性は、ネットワーク端点(Network Termination Point)管理オブジェクト(あるいはサブクラス)の形成された接続性を示す。
この属性の取りうる値は sourceConnect、sinkConnect、bidirectionalConnect、noConnect である。

シンクと等しい pointDirectionality を備えたネットワーク端点管理オブジェクトについては、この属性に対する許可された値は noConnect と sinkConnect である。

ソースと等しい pointDirectionality を備えたネットワーク端点管理オブジェクトについては、この属性に対する許可された値は noConnect と sourceConnect である。

双方向と等しい pointDirectionality を備えたネットワーク端点管理オブジェクトについては、この属性に対する許可された値は noConnect と bidirectionalConnect である。

いくつかの技術については、sinkConnect と sourceConnect はまた双方向ネットワーク端点管理オブジェクトのために許可され得る。
*/
      enum ConfiguredConnectivityType
      {
            configuratedConnectivitySourceConnect,
            configuratedConnectivitySinkConnect,
            configuratedConnectivityBidirectionalConnect,
            configuratedConnectivityNoConnect
      };

      enum ConnectivityEndPointChoice
      {
            connectivityEndPointChoiceSNCTP,
            connectivityEndPointChoiceLinkEnd,
            connectivityEndPointChoiceAccessGroup
      };

      union ConnectivityEndPointType switch (ConnectivityEndPointChoice)
      {
            case connectivityEndPointChoiceSNCTP:
                  NetworkTPNameType   sncTp;
            case connectivityEndPointChoiceLinkEnd:
                  AbstractLinkEndNameType    le;
            case connectivityEndPointChoiceAccessGroup:
                  AccessGroupNameType ac;
      };

      typedef sequence<ConnectivityEndPointType> ConnectivityEndPointSetType;
```

```
/**
下記の属性はファブリック(Fabric)オブジェクトの接続動作により使用される。
*/
	enum ConnectInfoEndPointLegChoice
	{
		connectInfoEndPointLegChoiceExplicitTP,
		connectInfoEndPointLegChoiceTPPool
	};

	union ConnectInfoEndPointLegType switch (ConnectInfoEndPointLegChoice)
	{
		case connectInfoEndPointLegChoiceExplicitTP:
			MONameSetType toTP;
		case connectInfoEndPointLegChoiceTPPool:
			TPPoolNameType	toTPPool;
	};

	typedef sequence<ConnectInfoEndPointLegType>
		ConnectInfoEndPointLegSetType;

	struct ConnectInfoEndPointAddLegType
	{
		MPCrossConnectionNameType  mpXC;
		ConnectInfoEndPointLegSetType	legs;
	};

	struct ConnectInfoEndPointPtoPType
	{
		MONameSetType		fromTP;
			// one TP or GTP of set
		MONameSetType		toTP;
			// one TP or GTP of set
	};

	struct ConnectInfoEndPointPtoMPType
	{
		MONameSetType		fromTP;
			// one TP or GTP of set
		MONameSetSetType	toTPs;
			// multiple TP or GTP, each TP or GTP from inner set
	};

	struct ConnectInfoEndPointPtoTPPoolType
	{
		MONameSetType		fromTP;
			// one TP or GTP of set
		TPPoolNameType			toTP;
			// one TP or GTP of TPPool
	};

	struct ConnectInfoEndPointMTPPoolType
	{
		TPPoolNameType	toTPPool;
		short		numberOfTPs;
	};

	typedef sequence<ConnectInfoEndPointMTPPoolType>
		ConnectInfoEndPointMTPPoolSetType;

	struct ConnectInfoEndPointPtoMTPPoolType
	{
		MONameSetType			fromTP;
			// one TP or GTP of set
		ConnectInfoEndPointMTPPoolSetType toTPs;
			// multiple TP or GTP, each TP or GTP from TPPool
	};

	enum ConnectInfoEndPointChoice
```

```
{
        connectInfoEndPointChoiceUniPtoP,
        connectInfoEndPointChoiceUniPtoTPPool,
        connectInfoEndPointChoiceUniPtoMP,
        connectInfoEndPointChoiceUniPtoMTPPool,
        connectInfoEndPointChoiceBidPtoP,
        connectInfoEndPointChoiceBidPtoTPPool,
        connectInfoEndPointChoideAddLeg
};

union ConnectInfoEndPointType switch (ConnectInfoEndPointChoice)
{
        case connectInfoEndPointChoiceUniPtoP:
        case connectInfoEndPointChoiceBidPtoP:
                ConnectInfoEndPointPtoPType             pToP;
        case connectInfoEndPointChoiceUniPtoTPPool:
        case connectInfoEndPointChoiceBidPtoTPPool:
                ConnectInfoEndPointPtoTPPoolType  pToTPPool;
        case connectInfoEndPointChoiceUniPtoMP:
                ConnectInfoEndPointPtoMPType            pToMP;
        case connectInfoEndPointChoiceUniPtoMTPPool:
                ConnectInfoEndPointPtoMTPPoolType pToMTPPool;
        case connectInfoEndPointChoideAddLeg:
                ConnectInfoEndPointAddLegType           addLeg;
};

/**
下記の属性はファブリックオブジェクトの接続動作により使用される。

AdministrativeState パラメータは新しいクロスコネクトが生成されるという運用状態値を特定する。
新しく生成されたクロスコネクトオブジェクトの参照は結果の構造体中で返される。
*/
        struct ConnectInfoType
        {
                ConnectInfoEndPointType          endPoint;
                AdministrativeStateTypeOpt adminstrativeState;
                Istring                          crossConnectionName;
                        // may be empty string
                BooleanTypeOpt                   redline;
                Istring                          userLabel;
                        // may be empty string
        };

        typedef sequence<ConnectInfoType> ConnectInfoSeqType;

        struct ConnectResultToPointType
        {
                MONameType                atp;
                        // a "leaf" termination point
                CrossConnectionNameType          xcon;
                        // a pt-pt cross-connection to the TP
        };

        typedef sequence<ConnectResultToPointType> ConnectResultToPointSetType;

        struct ConnectResultComponentType
        {
                MONameType                fromTP;
                        // the "root" of a connection, TP or GTP
                ConnectResultToPointSetType      toTP;
                        // the "leaf" of a connection.
                        // for pt-pt, only one leaf in list
                MPCrossConnectionNameType  mpxcon;
                        // the containing pt-multipt xcon.
                        // for pt-pt, this will be nil
        };
```

```
/**
接続結果は区間追加もしくはコネクション生成の結果を返す。
*/
	union ConnectResultType switch (boolean)
	{
		case TRUE:
			FailedType		failed;
			// TRUE indicate failure.
		case FALSE:
			ConnectResultComponentType connected;
	};

/*
下記のデータ構造中の n 番目のエレメントは上記 InfoSeqType 中の n 番目のエレメントに関連する。
順序は重要である。
*/
	typedef sequence<ConnectResultType> ConnectResultSeqType;

/**
unchangedTP は newTP と接続されるであろう。
*/
	struct ConnectSwitchOverInfoType
	{
		CrossConnectionNameType		xcon;
		MONameType		unchangedTP;
		MONameType		newTP;
	};

	typedef sequence<ConnectSwitchOverInfoType>
		ConnectSwitchOverInfoSeqType;

/**
下記の属性は externalControl 動作のための入力パラメタの可能な値を示す。
*/
	enum ControlActionType
	{
		controlActionCloseContinuously,
		controlActionOpenContinuously,
		controlActionCloseMomentarily,
		controlActionOpenMomentarily
	};

/**
下記の属性は externalControl 動作のための出力パラメータの可能な値を示す。
*/
	enum ControlResultType
	{
		controlResultComplete,
		controlResultAlreadyInCondition,
		controlResultFail_InvalidControlActionType,
		controlResultFail_ReasonUnknown
	};

/**
下記の属性は制御点の状態を示す。
*/
	enum ControlStateType
	{
		controlStateClosed,
		controlStateOpen
	};

	enum CrossConnectionMConChoice
	{
		crossConnectionMConChoiceDownstreamNotConnected,
		crossConnectionMConChoiceDownstreamConnected,
		crossConnectionMConChoiceUpstreamNotConnected,
```

```
		crossConnectionMConChoiceUpstreamConnected
	};

	union CrossConnectionMConType switch (CrossConnectionMConChoice)
	{
		case crossConnectionMConChoiceDownstreamNotConnected:
		  FabricNameType		downfab;
		case crossConnectionMConChoiceDownstreamConnected:
		  CrossConnectionNameType	downxcon;
		case crossConnectionMConChoiceUpstreamNotConnected:
		  FabricNameType		upfab;
		case crossConnectionMConChoiceUpstreamConnected:
		  CrossConnectionNameType	upxcon;
	};

	typedef sequence<CrossConnectionMConType> CrossConnectionMConSetType;

/**
以下に示す属性は端点が関連したクロスコネクトオブジェクトを識別するために使用される。
*/
	enum CrossConnectionPointerChoice
	{
		crossConnectionPointerChoiceNotConnected,
		crossConnectionPointerChoiceConnected,
		crossConnectionPointerChoiceMConnections
	};

	union CrossConnectionPointerType switch (CrossConnectionPointerChoice)
	{
		case crossConnectionPointerChoiceNotConnected:
		  FabricNameType		fab;
		case crossConnectionPointerChoiceConnected:
		  CrossConnectionNameType	xcon;
		case crossConnectionPointerChoiceMConnections:
		  CrossConnectionMConSetType mcon;
	};

/**
現行問題構造体はオブジェクトと共に現在存在する問題を確認する。
それは典型的な現行問題リスト(Current Problem List)の要素である。
*/
	struct CurrentProblemType
	{
		ProbableCauseType   problem;
		AlarmStatusType		alarmStatus;
	};

/**
現行問題リストは管理オブジェクトに関連した現在存在する問題を重要度と共に識別する。
*/
	typedef sequence<CurrentProblemType> CurrentProblemSetType;

/**
以下の属性は関連した管理オブジェクトが片方向か双方向であるかを特定する。
*/
	enum DirectionalityType
	{
		directinalityUnidirectional,
		directinalityBidirectional
	};

	union DirectionalityTypeOpt switch (boolean)
	{
		case TRUE:
		  DirectionalityType val;
	};
```

```
	union DisconnectResultType switch (boolean)
	{
		case TRUE:
			FailedType	failed;
			// TRUE indicates failure
		case FALSE:
			MONameType	tps;
			// TP or GTP
	};

	typedef sequence<DisconnectResultType> DisconnectResultSeqType;

/**
下流接続性ポインタタイプ(Downstream Connectivity Pointer Type)は TP が関係する接続のタイプを確認する。
*/
	enum DownstreamConnectivityPointerChoice
	{
		downstreamConnectivityPointerChoiceNone,
		downstreamConnectivityPointerChoiceSingle,
		downstreamConnectivityPointerChoiceConcatenated,
		downstreamConnectivityPointerChoiceBroadcast,
		downstreamConnectivityPointerChoiceBroadcastConcatenated
	};

	union DownstreamConnectivityPointerType
		switch (DownstreamConnectivityPointerChoice)
	{
		case downstreamConnectivityPointerChoiceSingle:
			TPNameType		single;
		case downstreamConnectivityPointerChoiceConcatenated:
			TPNameSeqType		concatenated;
				// order is significant
		case downstreamConnectivityPointerChoiceBroadcast:
			TPNameSetType		broadcast;
				// order is insignificant
		case downstreamConnectivityPointerChoiceBroadcastConcatenated:
			TPNameSeqSetType	broadcastConcatenated;
				// outer sequence order is insignificant
				// inner sequence order is significant
	};

	typedef sequence<ExternalPointNameType> ExternalPointNameSetType;

/**
FailedType は 6.3.1 節で定義されている。
*/

/**
ホルダ様態(Holder Status)は装置ホルダの様態を関連づける。"Empty"は装置ホルダがサーキットパックを含まないことを意味する。もし様態が"acceptable"なら、ホルダ中の回線パックは受理可能なリスト上にある。
"Unacceptable"はサーキットパックタイプは既知であるが受理可能なリスト上に無いことを意味する。"Unknown"は回線パックが未知であるか装置ホルダの様態が決定できないことを意味する。
*/
	enum HolderStatusChoice
	{
		holderStatusChoiceEmpty,
		holderStatusChoiceAcceptable,
		holderStatusChoiceUnacceptable,
		holderStatusChoiceUnknown
	};

	union HolderStatusType switch (HolderStatusChoice)
	{
		case holderStatusChoiceAcceptable:
		case holderStatusChoiceUnacceptable:
			Istring		circuitPackType;
	};
```

```
/**
次の属性は拡張可能な列挙型で、追加の値は今後加えうる。
*/
       typedef short InformationTransferCapabilityType;

       const InformationTransferCapabilityType
              informationTransferCapabilitySpeech = 0;
       const InformationTransferCapabilityType
              informationTransferCapabilityAudio3Pt1 = 1;
       const InformationTransferCapabilityType
              informationTransferCapabilityAudio7 = 2;
       const InformationTransferCapabilityType
              informationTransferCapabilityAudioComb = 3;
       const InformationTransferCapabilityType
              informationTransferCapabilityDigitalRestricted56 = 4;
       const InformationTransferCapabilityType
              informationTransferCapabilityUnrestricted64 = 5;

       typedef sequence<LayerNDNameType> LayerNDNameSetType;

       typedef sequence<LinkConnectionNameType> LinkConnectionNameSetType;

/**
下記の属性は関連したリンク管理オブジェクトが片方向、双方向、未定義なのかを特定する。
*/
       enum LinkDirectionalityType
       {
              linkDirectionalityUnidirectional,
              linkDirectionalityBidirectional,
              linkDirectionalityUndefined
       };

       union LinkDirectionalityTypeOpt switch (boolean)
       {
              case TRUE:
                     LinkDirectionalityType val;
       };

       enum LinkEndChoice
       {
              linkEndChoiceSubnetwork,
              linkEndChoiceAccessGroup,
              linkEndChoiceLinkEnd,
              linkEndChoiceUndefined
       };

       union LinkEndType switch (LinkEndChoice)
       {
              case linkEndChoiceSubnetwork:
                     SubnetworkNameType  sn;
              case linkEndChoiceAccessGroup:
                     AccessGroupNameType ag;
              case linkEndChoiceLinkEnd:
                     AbstractLinkEndNameType    le;
       };

/**
MONameSetSetTYpe は 6.3.1 節で定義されている。
*/

       typedef sequence<NetworkCTPAbstractNameType>
              NetworkCTPAbstractNameSetType;

       typedef sequence<NetworkTPNameType> NetworkTPNameSetType;

       typedef sequence<NetworkTTPAbstractNameType>
```

```
            NetworkTTPAbstractNameSetType;

      typedef sequence<PipeNameType> PipeNameSetType;

/**
下記の属性は余剰能力あるいは余剰帯域幅の量を有する LinkEnd に関連した Network CTP の数を示す。
*/
      enum PointCapacityChoice
      {
            pointCapacityChoiceNumberOfTPs,
            pointCapacityChoiceBandwidth
      };

      union PointCapacityType switch (PointCapacityChoice)
      {
            case pointCapacityChoiceNumberOfTPs:
                  long                numberOfTPs;
            case pointCapacityChoiceBandwidth:
                  BandwidthSeqType    bandwidth;
      };

/**
下記の属性は関連したリンク端管理オブジェクトがシンク、ソース、双方向なのかどうかを特定する。
*/
      enum PointDirectionalityType
      {
            pointDirectinalitySink,
            pointDirectinalitySource,
            pointDirectinalityBidirectional
      };

/**
ポート関連(Port Association)はサーキットパック上の(ストリングの名前によって識別された)ポートをリンク端を
表わす管理オブジェクトに関連づける。それらの集合は通常ポート関連リストの一部である。
*/
      struct PortAssociationType
      {
            Istring             portId;
            TPNameType    portTrail;
                  // empty means unassigned.
      };

/**
ポート関連リスト(Port Association List)はポート関連構造体のシーケンスである。
*/
      typedef sequence<PortAssociationType> PortAssociationSetType;

/**
この構造体はサーキットパックポート(例 port = "0"、rate = stml)とそのペーロードマッピング(例 au3 もしく
は au4)と関連した共に信号速度を識別する。
*/
      struct PortSignalRateAndMappingType
      {
            Istring                           portId;
            SignalRateType                    signalRate;
            CharacteristicInfoSeqType  mappingList;
      };

      typedef sequence<PortSignalRateAndMappingType>
            PortSignalRateAndMappingSetType;

/**
ProblemCauseType は 6.3.1 節で定義されている。
*/

/**
Replaceable はオブジェクトにより表現される資源が物理的に交換可能かかどうかを示す。
```

```
*/
       enum ReplaceableType
       {
              replaceableYes,
              replaceableNo,
              replaceableNotApplicable
       };

       enum RequestedCapacityChoice
       {
              requestedCapacityChoiceSpecificChannelList,
              requestedCapacityChoiceCapacity
       };

       union RequestedCapacityType switch (RequestedCapacityChoice)
       {
              case requestedCapacityChoiceSpecificChannelList:
                     ChannelSeqType      specificChannelList;
              case requestedCapacityChoiceCapacity:
                     CapacityType  capacity;
       };

       enum RequestedPointCapacityChoice
       {
              requestedPointCapacityChoiceSpecificTPList,
              requestedPointCapacityChoicePointCapacity
       };

       union RequestedPointCapacityType switch (RequestedPointCapacityChoice)
       {
              case requestedPointCapacityChoiceSpecificTPList:
                     NetworkTPNameSetTypespecificTPList;
              case requestedPointCapacityChoicePointCapacity:
                     PointCapacityType   capacity;
       };

       typedef short ResetErrorType;

       const ResetErrorType resetErrorTypeResetFail = 0;
       const ResetErrorType resetErrorTypeEntityInService = 1;

       typedef sequence<ScanPointNameType> ScanPointNameSetType;

       typedef short ServiceAffectingErrorType;

       const ServiceAffectingErrorType
              serviceAffectingErrorAffectingExistingService = 0;

/**
下記の属性は拡張できる列挙型で追加の値は将来追加され得る。
*/
       typedef short SignallingCapabilityType;

       const SignallingCapabilityType signallingCapabilityISUP = 0;
       const SignallingCapabilityType signallingCapabilityISUP92 = 1;
       const SignallingCapabilityType signallingCapabilityCCITTNo5 = 2;
       const SignallingCapabilityType signallingCapabilityR2 = 3;
       const SignallingCapabilityType signallingCapabilityCCITTNo6 = 4;
       const SignallingCapabilityType signallingCapabilityTUP = 5;

/**
 (クロスコネクトおよびグループ TP のような)NE レベルエンティティについては、信号識別子タイプ(Signal ID
Type)がユニークにエンティティの信号タイプを識別する。(サブネットワーク、サブネットワーク接続、パイプ
(Pipe)、抽象リンク(Abstract Link)およびネットワークレベル TP(Network level TP)のような)ネットワー
ク・レベルエンティティについては、信号識別子タイプが、考慮中のエンティティが属するレイヤの(G.805 の意味合い
での)特有の情報を定義する。それはサブネットワーク接続/接続性が可能かどうか決めるために使用される。
```

信号タイプは単一レートおよびフォーマット、集約信号を形成する同じ特有の情報を備えたエンティティの束あるいは異なる束のグルーピングを含んでいる複合タイプになりうる。複合タイプは終点位置間の多重並列接続を含むあるマルチメディアアプリケーションに適用可能である。

信号識別子には単一(simple), バンドル(bundle), 複合(complex)の 3 タイプがある。 単一とバンドルの場合、SignalIDType はシーケンス中で一つの SignalIdComponentType のみを持つ。単一信号のための束ね因子は 1 である。バンドル信号のための要因集合は 1 よりも大きい。複合信号のために SignalIDType は並びのなかで 1 より大きな SignalIdComponentTypes をもつ。それぞれのエレメントは単一もしくはバンドル信号になることができる。
*/
```
	struct SignalIdComponentType
	{
		CharacteristicInfoType	characteristicInfo;
		long			bundlingFactor;
	};

/**
```
SignalIdType は信号により運ばれた情報の型を示す。それは一つ以上の SignalIdComponentType 構造体から成り立っている。
*/
```
	typedef sequence<SignalIdComponentType> SignalIdType;

/**
```
SignalRateType と SignalRateSetType は 6.3.1 節で定義されている。
*/

```
	typedef sequence<SoftwareNameType> SoftwareNameSetType;

	typedef sequence<SNCNameType> SNCNameSetType;

	typedef sequence<SubnetworkNameType> SubnetworkNameSetType;

/**
```
タイミングソースタイプ(Timing Source Type)はタイミングソースのタイプを指し示している。
*/
```
	enum TimingSourceType
	{
		timingSourceInternal,
		timingSourceRemote,
		timingSourceSlavedTimingTerminationSignal
	};

/**
```
SystemTimingSource は同期のための資源のプライマリとセカンダリのタイミングソースを特定するために使用される。
*/

```
	struct SystemTimingSourceComponentType
	{
		TimingSourceType	timingSource;
		MONameType		timingSourceID;
	};

	union SystemTimingSourceSecondaryType switch (boolean)
	{
		case TRUE:
			SystemTimingSourceComponentType  val;
	};

	struct SystemTimingSourceType
	{
		SystemTimingSourceComponentType  primaryTiming;
		SystemTimingSourceSecondaryType  secondaryTiming;
	};

/**
```
上端方向性(Top End Directionality)属性型は関連したリンク終端管理オブジェクトがシンクなのかソースなのか双方向なのか未定義なのかを特定する。

```
*/
       enum TopEndDirectionalityType
       {
              topEndDirectionalityUndefined,
              topEndDirectionalitySink,
              topEndDirectionalitySource,
              topEndDirectionalityBidirectional
       };

       typedef sequence<TopLinkEndNameType> TopLinkEndNameSetType;

       typedef sequence<TopLinkNameType> TopLinkNameSetType;

       struct TPsAddToGTPInfoType
       {
              GTPNameType  agtp;
                     // may be empty only when used as an in parameter
                     // of an operation.
              MONameSeqType tps;
                     // For GTP, order is significant.
       };

       typedef sequence<TPsAddToGTPInfoType> TPsAddToGTPInfoSeqType;

       union TPsAddToGTPResultType switch (boolean)
       {
              case TRUE:
                     FailedType          failed;
                     // TRUE indicate failure.
              case FALSE:
                     TPsAddToGTPInfoType addedTPs;
       };

/*
下記のデータ構造中の n 番目のエレメントは上記 InfoSeqType 中の n 番目のエレメントに関連する。
順序は重要である。
*/
       typedef sequence<TPsAddToGTPResultType> TPsAddToGTPResultSeqType;

       struct TPsAddToTPPoolInfoType
       {
              TPPoolNameType      atppool;
                     // may be nil only when used as an in parameter
                     // of an operation.
              MONameSetType tps;
                     // For TPPool, order is insignificant.
       };

       typedef sequence<TPsAddToTPPoolInfoType> TPsAddToTPPoolInfoSeqType;

       union TPsAddToTPPoolResultType switch (boolean)
       {
              case TRUE:
                     FailedType          failed;
                     // TRUE indicate failure.
              case FALSE:
                     TPsAddToTPPoolInfoType     addedTPs;
       };

/*
下記のデータ構造中の n 番目のエレメントは上記 InfoSeqType 中の n 番目のエレメントに関連する。
順序は重要である。
*/
       typedef sequence<TPsAddToTPPoolResultType> TPsAddToTPPoolResultSeqType;

/**
TPNameSeqType, TPNameSetType, TPNameSeqSetType は 6.3.1 節で定義されている。
```

```
*/

        struct TPsRemoveInfoType
        {
                MONameType    gtpOrTPPool;
                MONameSetType tps;
                        // For remove, order is insignificant.
        };

        typedef sequence<TPsRemoveInfoType> TPsRemoveInfoSeqType;

        struct TPsRemoveResultComponentType
        {
                MONameType    gtpOrTPPool;
                        // For remove operation, if the TPPool or GTP is
                        // deleted, the deleted TPPool or GTP should be
                        // provided in the RemoveTPsResultInformation.
                MONameSetType tps;
                        // For remove operation, order is insignificant at all.
        };

        union TPsRemoveResultType switch (boolean)
        {
                case TRUE:
                        FailedType                 failed;
                        // TRUE indicate failure.
                case FALSE:
                        TPsRemoveResultComponentType     removed;
        };

/*
下記のデータ構造中の n 番目のエレメントは上記 InfoSeqType 中の n 番目のエレメントに関連する。
順序は重要である。
*/
        typedef sequence<TPsRemoveResultType> TPsRemoveResultSeqType;

        typedef sequence<TrailNameType> TrailNameSetType;

/**
伝送特性(Transmission Characteristic)はサーキット終点サブグループによりサポートされた送信のタイプの 1
つを指し示す。
*/
        enum TransmissionCharacteristicType
        {
                transmissionCharacteristicSatellite,
                transmissionCharacteristicDCME,
                transmissionCharacteristicEchoControl
        };

/**
伝送特性は衛星(サーキットサブグループにサポートされたエコー制御)のような異なる特性を指定する。M.3100 中でそ
れらはビットストリングとして指定される。
しかし、IDL では、それらが列挙のシーケンスによって表わされる。特別の特性がリストに見当たらない場合、
それは、サポートされないことを意味する。
*/
        typedef sequence<TransmissionCharacteristicType>
                TransmissionCharacteristicSetType;

/**
UsageCost は範囲 0..255 の整数型である。
*/
        typedef short UsageCostType;

        union UsageCostTypeOpt switch (boolean)
        {
                case TRUE:
                        UsageCostType       val;
```

```
	};

/**
以下の属性は制御ポイントのための制御信号の適正な型を示している。
*/
	enum ValidControlType
	{
		validControlMomentaryOnly,
		validControlContinuousOnly,
		validControlBoth
	};




/**
```

6.4 例外(Exceptions)

```
*/

/**
```

6.4.1 M.3100 特有のエラーパラメータの例外(Exceptions for M.3100 Specific Error Parameter)

以下の例外は削除操作中のみで使われるパラメータを除いた M.3100 中で定義された特定のエラーパラメータのために生成される。

```
*/
	exception ResetError
		{ ResetErrorType error; };

	exception CircuitPackResetError  { };

	exception ServiceAffectedError
		{ ServiceAffectingErrorType error; };

	exception ChannelsAlreadyProvisioned
		{ ChannelSeqType channelList; };

	exception FailureToAddLinkConnection { };

	exception FailureToAddNetworkCTP { };

	exception FailureToAssociateLC { };

	exception FailureToAssociateNetworkTTP { };

	exception FailureToDeassignLinkConnection { };

	exception FailureToDeassignNetworkCTP { };

	exception FailureToDecreaseCapacity
		{ CapacitiesType capacities; };

	exception FailureToIncreaseCapacity
		{ CapacitiesType capacities; };

	exception FailureToRemoveLC { };

	exception FailureToBindLink { };

	exception FailureToBindLinkEnd { };

	exception FailureToBindTopologicalLink { };
```

```
exception FailureToCreateAccessGroup { };

exception FailureToCreateLink { };

exception FailureToCreateLC { };

exception FailureToCreateLinkEnd { };

exception FailureToCreateNetworkTTP { };

exception FailureToCreateSubnetwork { };

exception FailureToDisassociateNetworkTTP { };

exception FailureToSetDirectionality { };

exception FailureToSetLinkConnectionCallerId { };

exception FailureToSetNetworkCTPCallerId { };

exception FailureToSetUserIdentifier { };

exception FailureToSupportLC { };

exception InconsistentDirectionality { };

exception InconsistentSignalIdentification { };

exception InsufficientCapacity
       { CapacitiesType capacities; };

exception InvalidChannelsNumber
       { ChannelSeqType channelList; };

exception InvalidLinkConnection
       { MONameType linkConnection; };

exception InvalidNetworkCTP
       { MONameType networkCTP; };

exception InvalidServiceCharacteristicsRequested { };

exception InvalidTPType { };

exception InvalidTrafficDescriptorRequested { };

exception LinkConnectionAlreadyAssigned
       { MONameSetType linkList; };

exception LinkEndAndNetworkCTPNotCompatible
       { MONameSetType linkList; };

exception LinkAndLinkConnectionNotCompatible
       { MONameSetType linkList; };

exception NetworkCTPAlreadyAssigned
       { MONameType networkCTP; };

exception NetworkTTPAndAccessGroupNotCompatible { };

exception NetworkTTPAndSubnetworkNotCompatible { };

exception NewServiceCharacteristicsExistsAlready
       { SignalIdType signalId; };

exception NewTrafficDescriptorExistsAlready
       { SignalIdType signalId; };
```

```
        exception NoLinkCapacity { };

        exception NoLinkEndCapacity { };

        exception NoSuchLink
                { MONameType link; };

        exception NoSuchLinkEnd
                { MONameType linkEnd; };

        exception NotAssignedToCaller
                { MONameType caller; };

        exception NotEnoughLinkConnection
                { long num; };

        exception NotEnoughNetworkCTP
                { long num; };

/**
オブジェクトはそれらの想定原因の対応する欠陥を宣言しないという事実によりオブジェクトによるサポートを受けられ
ないという想定原因値。
*/
        exception NotSupportedProbableCause
                { ArcProbableCauseSetType probableCauseList; };

/**
```

6.4.2 G.855.1 特有のエラーパラメータの例外(Exceptions for G.855.1 Specific Error Parameter)

以下の例外は削除操作中のみで使われるパラメータを除いた G.855.1 中で定義された特定のエラーパラメータのために生成される。

```
*/
        exception AEndNetworkTPConnected
                { NetworkTPNameSetType tpList; };

        exception CapacityProvisionned
                { CapacitiesType capacities; };

        exception ConsistencyFailure { };

        exception FailureToAssociate { };

        exception FailureToConnect
                { FailedType failed; };

        exception FailureToCreateTopologicalLink { };

        exception FailureToDeleteLink { };

        exception FailureToDeleteTopologicalLinkEnd { };

        exception FailureToDisassociate { };

        exception FailureToRelease
                { FailedType failed; };

        exception FinalCapacitiesFailure
                { CapacitiesType capacities; };

        exception IncorrectLink
                { MONameType link; };

        exception IncorrectLinkEnds
                { MONameType linkEnd; };
```

```
exception IncorrectSubnetwork
       { MONameType subnetwork; };

exception IncorrectSubnetworkTerminationPoints
       { MONameSetType tpList; };

exception InitialCapacitiesFailure
       { CapacitiesType capacities; };

exception InvalidTrafficDescriptor { };

exception InvalidTrail { };

exception InvalidTransportServiceCharacteristics { };

exception LinkAndTrailsNotCompatible { };

exception LinkEndAndNetworkTTPsNotCompatible { };

exception LinkConnectionsExisting
       { MONameSetType lcList; };

exception NetworkCTPsExisting
       { MONameSetType tpList; };

exception NetworkTTPAlreadyAssociated { };

exception NetworkTTPNotAssociated { };

exception NetworkTTPsInAEndAccessGroupConnected
       { MONameSetType networkTTPList; };

exception NetworkTTPsInZEndAccessGroupConnected
       { MONameSetType networkTTPList; };

exception NetworkTTPsNotPartOfLayerND
       { MONameSetType networkTTPList; };

exception NoSuchNetworkTTP
       { MONameType networkTTP; };

exception NoSuchSnc
       { MONameType snc; };

exception NoSuchTrail
       { MONameType trail; };

exception TrailAlreadyAssociated { };

exception SncConnected
       { MONameSetType tpList; };

exception TrailConnected
       { MONameType trail; };

exception TrailNotAssociated { };

exception UnknownSnc
       { long count; };

exception UnknownTrail { };

exception UserIdentifierNotUnique
       { Istring userId; };

exception WrongAEndDirectionality
       { DirectionalityType direction; };
```

```
        exception WrongZEndDirectionality
                { DirectionalityType direction; };

        exception ZEndNetworkTPConnected
                { MONameSetType tpList; };

/**
```

6.4.3 条件付パッケージの例外と定数(Exceptions and Constants for Conditional Package)

```
*/
        exception NOadministrativeOperationalStatesPackage {};

        exception NOadministrativeStatePackage {};

        exception NOaffectedObjectListPackage {};

        exception NOalarmSeverityAssignmentPointerPackage {};

        exception NOarcPackage {};

        exception NOarcRetrieveAlarmDetailPackage {};

        exception NOaudibleVisualLocalAlarmPackage {};

        exception NOavailabilityStatusPackage {};

        exception NObasicConnectionPerformerPackage {};

        exception NOchannelNumberPackage {};

        exception NOcharacteristicInformationPackage {};

        exception NOcircuitPackConfigurationPackage {};

        exception NOcircuitPackResetPackage {};

        exception NOclientCTPListPackage {};

        exception NOclientConnectionListPackage {};

        exception NOclientLinkConnectionPointerListPackage {};

        exception NOclientLinkEndPointerPackage {};

        exception NOclientLinkPointerPackage {};

        exception NOclientTrailPackage {};

        exception NOcomponentPointerPackage {};

        exception NOcompositePointerPackage {};

        exception NOconfiguredConnectivityPackage {};

        exception NOconnectivityPointerPackage {};

        exception NOcontainedAccessGroupListPackage {};

        exception NOcontainedBoardPackage {};

        exception NOcontainedInSubnetworkListPackage {};

        exception NOcontainedLinkEndListPackage {};

        exception NOcontainedLinkListPackage {};
```

```
exception NOcontainedNetworkTPListPackage {};
exception NOcontainedSubnetworkListPackage {};
exception NOcrossConnectionPointerPackage {};
exception NOcurrentProblemListPackage {};
exception NOexternalTimePackage {};
exception NOlayerConnectionListPackage {};
exception NOlinkConnectionPointerListPackage {};
exception NOlinkPointerListPackage {};
exception NOlocationNamePackage {};
exception NOlogicalLinkHandlerPackage {};
exception NOlogicalLinkEndHandlerPackage {};
exception NOmaximumLinkConnectionCountPackage {};
exception NOmaximumNetworkCTPCountPackage {};
exception NOnamedCrossConnectionPackage {};
exception NOneAssignmentPointerPackage {};
exception NOnetworkCTPPackage {};
exception NOnetworkCTPsInLinkEndListPackage {};
exception NOnetworkLevelPackage {};
exception NOnetworkTPPointerPackage {};
exception NOnumberOfPortPackage {};
exception NOnormalControlStatePackage {};
exception NOoperationalStatePackage {};
exception NOportAssociationsPackage {};
exception NOpotentialLinkCapacityPackage {};
exception NOpotentialLinkEndCapacityPackage {};
exception NOprotectedPackage {};
exception NOprovisionedLinkCapacityPackage {};
exception NOprovisionedLinkConnectionCountPackage {};
exception NOprovisionedLinkEndCapacityPackage {};
exception NOprovisionedNetworkCTPCountPackage {};
exception NOquailityOfConnectivityServicePackage {};
exception NOredlinePackage {};
exception NOrelatedRoutingProfilePackage {};
exception NOserverConnectionListPackage {};
```

```
exception NOserverTrailListPackage {};

exception NOserverTTPPointerPackage {};

exception NOsncPointerPackage {};

exception NOsoftwareProcessingErrorAlarmR2Package {};

exception NOsubordinateCircuitPackPackage {};

exception NOsupportableClientListPackage {};

exception NOsupportedByPackage {};

exception NOsystemTimingSourcePackage {};

exception NOtmnCommunicationsAlarmInformationR1Package {};

exception NOtopologicalLinkHandlerPackage {};

exception NOtopologicalLinkEndHandlerPackage {};

exception NOtotalLinkCapacityPackage {};

exception NOtotalLinkEndCapacityPackage {};

exception NOtrafficDescriptorPackage {};

exception NOusageCostPackage {};

exception NOusageStatePackage {};

exception NOuserLabelPackage {};

exception NOvendorNamePackage {};

exception NOversionPackage {};

/**
この定数リストは本勧告で定義された条件付のパッケージを説明する。条件付のパッケージが特定の
条件付きのパッケージが特定の管理オブジェクトインスタンス中でサポートされている時、
これらの文字列の１つはパッケージ属性に包含されているであろう。
*/


const string administrativeOperationalStatesPackage =
       "itut_m3120::administrativeOperationalStatesPackage";
const string administrativeStatePackage =
       "itut_m3120::administrativeStatePackage";
const string affectedObjectListPackage =
       "itut_m3120::affectedObjectListPackage";
const string alarmSeverityAssignmentPointerPackage =
       "itut_m3120::alarmSeverityAssignmentPointerPackage";
const string arcPackage =
       "itut_m3120::arcPackage";
const string arcRetrieveAlarmDetailPackage =
       "itut_m3120::arcRetrieveAlarmDetailPackage";
const string attributeValueChangeNotificationPackage =
       "itut_m3120::attributeValueChangeNotificationPackage";
const string audibleVisualLocalAlarmPackage =
       "itut_m3120::audibleVisualLocalAlarmPackage";
const string availabilityStatusPackage =
       "itut_m3120::availabilityStatusPackage";
const string basicConnectionPerformerPackage =
       "itut_m3120::basicConnectionPerformerPackage";
const string channelNumberPackage =
       "itut_m3120::channelNumberPackage";
```

```
const string characteristicInformationPackage =
       "itut_m3120::characteristicInformationPackage";
const string circuitPackConfigurationPackage =
       "itut_m3120::circuitPackConfigurationPackage";
const string circuitPackResetPackage =
       "itut_m3120::circuitPackResetPackage";
const string clientConnectionListPackage =
       "itut_m3120::clientConnectionListPackage";
const string clientCTPListPackage =
       "itut_m3120::clientCTPListPackage";
const string clientLinkConnectionPointerListPackage =
       "itut_m3120::clientLinkConnectionPointerListPackage";
const string clientLinkEndPointerPackage =
       "itut_m3120::clientLinkEndPointerPackage";
const string clientLinkPointerPackage =
       "itut_m3120::clientLinkPointerPackage";
const string clientTrailPackage =
       "itut_m3120::clientTrailPackage";
const string componentPointerPackage =
       "itut_m3120::componentPointerPackage";
const string compositePointerPackage =
       "itut_m3120::compositePointerPackage";
const string configuredConnectivityPackage =
       "itut_m3120::configuredConnectivityPackage";
const string connectivityPointerPackage =
       "itut_m3120::connectivityPointerPackage";
const string containedAccessGroupListPackage =
       "itut_m3120::containedAccessGroupListPackage";
const string containedBoardPackage =
       "itut_m3120::containedBoardPackage";
const string containedInSubnetworkListPackage =
       "itut_m3120::containedInSubnetworkListPackage";
const string containedLinkEndListPackage =
       "itut_m3120::containedLinkEndListPackage";
const string containedLinkListPackage =
       "itut_m3120::containedLinkListPackage";
const string containedNetworkTPListPackage =
       "itut_m3120::containedNetworkTPListPackage";
const string containedSubnetworkListPackage =
       "itut_m3120::containedSubnetworkListPackage";
const string createDeleteNotificationsPackage =
       "itut_m3120::createDeleteNotificationsPackage";
const string crossConnectionPointerPackage =
       "itut_m3120::crossConnectionPointerPackage";
const string currentProblemListPackage =
       "itut_m3120::currentProblemListPackage";
const string environmentalAlarmR2Package =
       "itut_m3120::environmentalAlarmR2Package";
const string equipmentsEquipmentAlarmR2Package =
       "itut_m3120::equipmentsEquipmentAlarmR2Package";
const string externalTimePackage =
       "itut_m3120::externalTimePackage";
const string layerConnectionListPackage =
       "itut_m3120::layerConnectionListPackage";
const string linkConnectionPointerListPackage =
       "itut_m3120::linkConnectionPointerListPackage";
const string linkPointerListPackage =
       "itut_m3120::linkPointerListPackage";
const string locationNamePackage =
       "itut_m3120::locationNamePackage";
const string logicalLinkEndHandlerPackage =
       "itut_m3120::logicalLinkEndHandlerPackage";
const string logicalLinkHandlerPackage =
       "itut_m3120::logicalLinkHandlerPackage";
const string maximumLinkConnectionCountPackage =
       "itut_m3120::maximumLinkConnectionCountPackage";
const string maximumNetworkCTPCountPackage =
       "itut_m3120::maximumNetworkCTPCountPackage";
```

```
const string namedCrossConnectionPackage =
       "itut_m3120::namedCrossConnectionPackage";
const string neAssignmentPointerPackage =
       "itut_m3120::neAssignmentPointerPackage";
const string networkCTPPackage =
       "itut_m3120::networkCTPPackage";
const string networkCTPsInLinkEndListPackage =
       "itut_m3120::networkCTPsInLinkEndListPackage";
const string networkLevelPackage =
       "itut_m3120::networkLevelPackage";
const string networkTPPointerPackage =
       "itut_m3120::networkTPPointerPackage";
const string normalControlStatePackage =
       "itut_m3120::normalControlStatePackage";
const string numberOfPortPackage =
       "itut_m3120::numberOfPortPackage";
const string objectManagementNotificatoinsPackage =
       "itut_m3120::objectManagementNotificatoinsPackage";
const string operationalStatePackage =
       "itut_m3120::operationalStatePackage";
const string portAssociationsPackage =
       "itut_m3120::portAssociationsPackage";
const string potentialLinkCapacityPackage =
       "itut_m3120::potentialLinkCapacityPackage";
const string potentialLinkEndCapacityPackage =
       "itut_m3120::potentialLinkEndCapacityPackage";
const string processingErrorAlarmR2Package =
       "itut_m3120::processingErrorAlarmR2Package";
const string protectedPackage =
       "itut_m3120::protectedPackage";
const string provisionedLinkCapacityPackage =
       "itut_m3120::provisionedLinkCapacityPackage";
const string provisionedLinkConnectionCountPackage =
       "itut_m3120::provisionedLinkConnectionCountPackage";
const string provisionedLinkEndCapacityPackage =
       "itut_m3120::provisionedLinkEndCapacityPackage";
const string provisionedNetworkCTPCountPackage =
       "itut_m3120::provisionedNetworkCTPCountPackage";
const string quailityOfConnectivityServicePackage =
       "itut_m3120::quailityOfConnectivityServicePackage";
const string redlinePackage =
       "itut_m3120::redlinePackage";
const string relatedRoutingProfilePackage =
       "itut_m3120::relatedRoutingProfilePackage";
const string serverConnectionListPackage =
       "itut_m3120::serverConnectionListPackage";
const string serverTrailListPackage =
       "itut_m3120::serverTrailListPackage";
const string serverTTPPointerPackage =
       "itut_m3120::serverTTPPointerPackage";
const string sncPointerPackage =
       "itut_m3120::sncPointerPackage";
const string softwareProcessingErrorAlarmR2Package =
       "itut_m3120::softwareProcessingErrorAlarmR2Package";
const string stateChangeNotificationPackage =
       "itut_m3120::stateChangeNotificationPackage";
const string subordinateCircuitPackPackage =
       "itut_m3120::subordinateCircuitPackPackage";
const string supportableClientListPackage =
       "itut_m3120::supportableClientListPackage";
const string supportedByPackage =
       "itut_m3120::supportedByPackage";
const string systemTimingSourcePackage =
       "itut_m3120::systemTimingSourcePackage";
const string tmnCommunicationsAlarmInformationR1Package =
       "itut_m3120::tmnCommunicationsAlarmInformationR1Package";
const string topologicalLinkEndHandlerPackage =
       "itut_m3120::topologicalLinkEndHandlerPackage";
```

```
        const string topologicalLinkHandlerPackage =
               "itut_m3120::topologicalLinkHandlerPackage";
        const string totalLinkCapacityPackage =
               "itut_m3120::totalLinkCapacityPackage";
        const string totalLinkEndCapacityPackage =
               "itut_m3120::totalLinkEndCapacityPackage";
        const string trafficDescriptorPackage =
               "itut_m3120::trafficDescriptorPackage";
        const string usageCostPackage =
               "itut_m3120::usageCostPackage";
        const string usageStatePackage =
               "itut_m3120::usageStatePackage";
        const string userLabelPackage =
               "itut_m3120::userLabelPackage";
        const string vendorNamePackage =
               "itut_m3120::vendorNamePackage";
        const string versionPackage =
               "itut_m3120::versionPackage";

/**
```

6.5　インタフェイス－細粒度（Interface -　Fine-Grained）

6.5.1 AbstractLink

AbstractLink interface をサポートする管理オブジェクトは２つの隣接するサブネットワーク（Sub-netowk）間、又は２つのリンク・エンド（Link　End）間、又はネットワークトレール端点（Network　Trail　End）が最も大きいサブネットワークの境界外にある場合のサブネットワークとアクセスグループ（Access　Group）の間の容量のトポロジカルな記述を示す。

個々の属性や通知の説明に使われる用語については下記に記述する。

-a end:リンクの片方を終端するリンクエンド、サブネットワーク、又はアクセスグループ
-利用可能リンク容量: 空きのリンク接続数（Link　Connection）、または空き帯域幅
-z end:リンクの片方を終端するリンクエンド、サブネットワーク、又はアクセスグループ
-信号識別子: リンクの容量を提供するリンク接続の信号識別子

リンクは同一の信号識別子のリンク接続（Link　Connections）により容量が提供されなければならない。

属性変化通知（Attribute value change notification）: availableLinkCapacity,又は totalLinkCapacity の属性値が変化した場合に発信される。
AbstractLink interface はインスタンス可能でない。
本クラスは、以下の条件付パッケージを含んでいる。
- usageCostPackage: リンクが割り当てられた使用コストを保有する時、存在する。
- userLabelPackage: ユーザラベルがサポートされる時、存在する。

```
*/
       valuetype AbstractLinkValueType: itut_x780::ManagedObjectValueType
       {
              public MONameType   aEnd;
                     // GET, SET-BY-CREATE
              public CapacityType availableLinkCapacity;
                     // GET
              public SignalIdType signalId;
                     // GET
              public itut_x780::ManagedObject  zEnd;
                     // GET, SET-BY-CREATE
              public UsageCostTypeusageCost;
                     // conditional
                     // usageCostPackage
                     // GET-REPLACE
              public Istring             userLabel;
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
       }; // valuetype AbstractLinkValueType

       interface AbstractLink: itut_x780::ManagedObject
       {
/**
以下の属性は同一ネットワークレイヤドメインのサブネットワーク、リンクエンド、又はアクセスグループへのポインタである。
*/
              MONameType aEndGet ()
                     raises (itut_x780::ApplicationError);
/**
以下の属性は、利用可能なリンク接続数または本リンクで利用可能な帯域幅として表現される利用可能リンク容量を示す。
*/
              CapacityType availableLinkCapacityGet ()
                     raises (itut_x780::ApplicationError);
/**
以下の属性は、考慮対象のエンティティが属する（G.805 の意味合いの）レイヤ特性情報を定義する。
本属性は、サブネットワークの接続／接続性が可能かどうかを決定するために使用される。
*/
              SignalIdType signalIdGet ()
                     raises (itut_x780::ApplicationError);
/**
```

```
以下の属性は同一ネットワークレイヤドメインのサブネットワーク、リンクエンド、又はアクセスグループへのポインタ
である。
*/
              MONameType zEndGet ()
                     raises (itut_x780::ApplicationError);
/**
本リンクを使用するコストを返す。これは選択／ルーチングの基準に使用される。
*/
              UsageCostType usageCostGet ()
                     raises (itut_x780::ApplicationError,
                            NOusageCostPackage);

/**
本リンクを使用するコストを設定する。これは選択／ルーチングの基準に使用される。
*/
              void usageCostSet
                     (in UsageCostType usageCost)
                     raises (itut_x780::ApplicationError,
                            NOusageCostPackage);
/**
以下のメソッドは関連するオブジェクトに使い勝手のよい名前を返す。
*/
              Istring userLabelGet ()
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);
/**
以下のメソッドは関連するオブジェクトに使い勝手のよい名前を割当てる。
*/
              void userLabelSet
                     (in Istring label)
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion)
/**
もし標準 JT-x721 にて定義された属性値変化通知が本管理オブジェクトクラスのインスタンスによりサポートされる場
合にサポートされる。
*/
              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, attributeValueChange,
                     attributeValueChangeNotificationPackage)
       };  // interface AbstractLink
/**
```

6.5.1.1 LogicalLink

LogicalLink interface をサポートする管理オブジェクトは、１つ以上のトポロジカルなリンクやその他の論理リンクにより提供されるリンク接続や帯域幅から管理上構成され得るリンクを表現している。
本インタフェースは以下の条件付パッケージを含んでいる。

```
- LinkConnectionPointerListPackage: 伝送技術により予めリンク接続された場合に存在する。
*/
       valuetype LogicalLinkValueType: AbstractLinkValueType
       {
                     // logicalLinkCapacityPackage, has actions only
              public LinkDirectionalityType    linkDirectionality;
                     // GET
              public LinkConnectionNameSetType
                     linkConnectionPointerList;
                     // conditional
                     // linkConnectionPointerListPackage
                     // GET-REPLACE, ADD-REMOVE
       }; // valuetype LogicalLinkValueType

       interface LogicalLink : AbstractLink
       {
/**
リンク方向性属性は連携したリンク管理オブジェクトが片方向、双方向又は未定義のいずれであるかを規定する。
*/
              LinkDirectionalityType linkDirectionalityGet ()
```

```
                    raises (itut_x780::ApplicationError);
/**
以下の属性は同一のレイヤにおける論理リンクを構成し得る特定レイヤのリンクコネクションリストを定義する。
*/
              LinkConnectionNameSetType linkConnectionPointerListGet ()
                    raises (itut_x780::ApplicationError,
                           NOlinkConnectionPointerListPackage);

              void linkConnectionPointerListSet
                    (in LinkConnectionNameSetType connectionList)
                    raises (itut_x780::ApplicationError,
                           NOlinkConnectionPointerListPackage);

              void linkConnectionPointerListAdd
                    (in LinkConnectionNameSetType connectionList)
                    raises (itut_x780::ApplicationError,
                           NOlinkConnectionPointerListPackage);

              void linkConnectionPointerListRemove
                    (in LinkConnectionNameSetType connectionList)
                    raises (itut_x780::ApplicationError,
                           NOlinkConnectionPointerListPackage);
/**
以下の２つのアクションは論理リンクの容量の管理に対するサポートを提供する。本アクションはリンク接続かつ／またはリンクの帯域幅の割当てと解放のアクションを規定する。
*/

/**
以下のアクションは論理リンクにリンク接続を割当てる。割当てられたリンクコネクションへのポインタは
logicalLink 管理オブジェクトの linkConnectionPointerList 属性に追加される。
*/
              void assignLinkConnectionOnLogicalLink
                    (in LayerNDNameType layer,
                    in LinkConnectionNameSetType requestedConnectionList,
                    out LinkConnectionNameSetType resultingConnectionList)
                    raises (itut_x780::ApplicationError,
                           LinkAndLinkConnectionNotCompatible,
                           InvalidLinkConnection,
                           NotEnoughLinkConnection,
                           LinkConnectionAlreadyAssigned,
                           InconsistentSignalIdentification,
                           InconsistentDirectionality,
                           FailureToSetLinkConnectionCallerId,
                           FailureToDecreaseCapacity);

/**
以下のアクションは、レイヤドメイン中の論理リンクに対して、同一レイヤドメイン中のリンクコネクションの割当て解除を行う。
*/
              void deassignLinkConnectionFromLogicalLink
                    (in LinkConnectionNameSetType requestedConnectionList)
                    raises (itut_x780::ApplicationError,
                           LinkAndLinkConnectionNotCompatible,
                           InvalidLinkConnection,
                           NotAssignedToCaller,
                           FailureToDeassignLinkConnection,
                           FailureToIncreaseCapacity);

       }; // interface LogicalLink

       interface LogicalLinkFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                    (in NameBindingType nameBinding,
                    in MONameType superior,
                    in string reqID,    // auto naming if empty string
                    out MONameType name,
                    in StringSetType packageNameList,
                    in MONameType aEnd,
                           // abstractLinkPackage
                           // GET, SET-BY-CREATE
                    in MONameType zEnd,
                           // abstractLinkPackage
                           // GET, SET-BY-CREATE
                    in Istring userLabel,
```

```
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                    in LinkConnectionNameSetType linkConnectionPointerList)
                        // conditional
                        // linkConnectionPointerListPackage
                        // GET-REPLACE, ADD-REMOVE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

    }; // interface LogicalLinkFactory
/**
```

6.5.1.2 TopLink (Topological Link)

TopLink (Topological Link) インタフェースをサポートする管理オブジェクトは、
唯一のサーバトレールより与えられるクライアントレイヤリンクを表現する。
ServerTrail 属性は、本トポロジカルリンクをサポートするサーバレイヤネットワークドメイン内トレールへのポインタである。本トポロジカルリンクをサポートするサーバレイヤネットワークドメイン内トレールが割当てられていない場合には serverTrail 属性は空であり得る。
個々の属性と通知の説明に関する用語説明は以下のとおりである。

-総リンク容量: 利用可能なリンク接続総数、または帯域幅

-最大リンク接続数: 柔軟な帯域管理による接続において利用できるリンク接続最大数

-潜在的リンク容量: 提供可能な潜在的リンク接続数又は潜在的帯域幅

-提供リンク容量: 提供されたリンク接続総数または帯域幅

-提供リンク接続数: 柔軟な帯域管理により割当てられたリンク接続数

属性値変化通知は totalLinkCapacity、maximumLinkConnectionCount、 potentialLinkCapacity、 provisionedLinkCapacity、 provisionedLinkConnectionCount のいずれかの属性値が変化した場合に発信される。

本インタフェースは、以下の条件付パッケージを含んでいる。

- totalLinkCapacityPackage: 予め提供されたアダプテーション、リンク接続またはリンク管理が伝送技術によりサポートされている場合、存在する。
- maximumLinkConnectionCountPackage: 柔軟な帯域割当てがサポートされている場合、存在する。

- potentialLinkCapacityPackage: 予め提供されたアダプテーション、リンク接続またはリンク管理が伝送技術によりサポートされている場合、存在する。
- ProvisionedLinkCapacityPackage: 予め提供されたアダプテーション、リンク接続またはリンク管理が伝送技術によりサポートされている場合、存在する。
- ProvisionedLinkConnectionCountPackage: 柔軟な帯域割当てがサポートされている場合、存在する。

```
*/
    valuetype TopLinkValueType: AbstractLinkValueType
    {
        public DirectionalityType  directionality;
                // GET
        public TrailNameType       serverTrail;
                // GET
        public CapacityType        totalLinkCapacity;
                // conditional
                // totalLinkCapacityPackage
                // GET
        public long                maximumLinkConnectionCount;
                // conditional
                // maximumLinkConnectionCountPackage
                // GET
        public CapacityType        potentialLinkCapacity;
                // conditional
                // potentialLinkCapacityPackage
                // GET
        public CapacityType        provisionedLinkCapacity;
                // conditional
                // provisionedLinkCapacityPackage
                // GET
```

```
		public long		provisionedLinkConnectionCount;
			// conditional
			// provisionedLinkConnectionCountPackage
			// GET
	}; // valuetype TopLinkValueType

	interface TopLink : AbstractLink
	{
		DirectionalityType directionalityGet ()
			raises (itut_x780::ApplicationError);
/**
以下の属性はクライアント内のリンクをサポートするサーバレイヤのトレールをポインタ参照する。
*/
		TrailNameType serverTrailGet ()
			raises (itut_x780::ApplicationError);
/**
以下の属性は、リンクの総容量を示す。本容量は、リンクに含まれるリンク接続数、又はリンクで利用可能な総帯域幅になり得る。
*/
		CapacityType totalLinkCapacityGet ()
			raises (itut_x780::ApplicationError,
				NOtotalLinkCapacityPackage);
/**
以下の属性は柔軟な帯域割当てがサポートされた場合、リンクに割当てられたリンク接続数の最大値を示す。
*/
		long maximumLinkConnectionCountGet ()
			raises (itut_x780::ApplicationError,
				NOmaximumLinkConnectionCountPackage);
/**
以下の属性は、リンク接続数、またはリンクにまだ割当てされていないがサーバトレールからリンクに割当て可能な帯域幅を示す。
*/
		CapacityType potentialLinkCapacityPackageGet ()
			raises (itut_x780::ApplicationError,
				NOpotentialLinkCapacityPackage);
/**
以下の属性はリンクに割当てられたリンク接続数、または帯域量を示す。
*/
		CapacityType provisionedLinkCapacityGet ()
			raises (itut_x780::ApplicationError,
				NOprovisionedLinkCapacityPackage);

/**
以下の属性は柔軟な帯域割当てがサポートされた場合、リンクに割当てられたリンク接続数を示す。
*/
		long provisionedLinkConnectionCountGet ()
			raises (itut_x780::ApplicationError,
				NOprovisionedLinkConnectionCountPackage);
/**
以下の２つのアクションはトポロジカルリンクの容量の管理に対するサポートを提供する。本アクションはリンク接続かつ／またはトポロジカルリンクの帯域の割当て／解放を規定する。
*/

/**
以下のアクションは、リンク接続又は利用可能帯域幅を増加させることにより、トポロジカルリンクへの容量を追加する。本アクションは resultingLinkConnections フィールド付き AddCapacityToTopLinkResult を返す。resultingLinkConnections フィールドは動的帯域が割当て中の場合、空を持つ。
*/
		void addCapacityToTopLink
			(in RequestedCapacityType requestedCapacity,
			out CapacityType resultingCapacity,
			out LinkConnectionNameSetType resultingLinkConnection)
			raises (itut_x780::ApplicationError,
				NoSuchLink,
				InsufficientCapacity,
				InvalidChannelsNumber,
				ChannelsAlreadyProvisioned,
				FailureToCreateLC,
				FailureToAssociateLC,
				FailureToSupportLC,
				FailureToIncreaseCapacity);
/**
```

以下のアクションはリンクから、リンクコネクションかつ／または帯域を削除する事により、トポロジカルリンクから容量を削除する。
*/

```
              void removeCapacityFromTopLink
                     (in RequestedCapacityType requestedCapacity,
                     out CapacityType resultingCapacity)
                     raises (itut_x780::ApplicationError,
                            NoSuchLink,
                            InsufficientCapacity,
                            InvalidChannelsNumber,
                            FailureToRemoveLC,
                            FailureToDecreaseCapacity);

       }; // interface TopLink

       interface TopLinkFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID,    // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList,
                     in MONameType aEnd,
                            // abstractLinkPackage
                            // GET, SET-BY-CREATE
                     in MONameType zEnd,
                            // abstractLinkPackage
                            // GET, SET-BY-CREATE
                     in Istring userLabel)
                            // conditional
                            // userLabelPackage
                            // GET-REPLACE
                     raises (itut_x780::ApplicationError,
                            itut_x780::CreateError);

       }; // interface TopLinkFactory
/**
```

6.5.2 AbstractLinkEnd

AbstractLinkEnd インタフェースをサポートする管理オブジェクトはトポロジを表現する目的でネットワーク接続端点を含む。
個々の属性と通知説明に使用される用語の説明は以下のとおりである。

-利用リンクエンド容量:リンクエンドの余剰容量を表す。
-リンクポインタ: 関連するリンク管理オブジェクトインスタンスの識別名

-サブネットワークリスト内包含: 論理リンクの親サブネットワークを表す識別名

状態変化通知は、availableLinkEndCapacity 又は containedInSubNetworkList の値が変化した場合に発信される。

AbstractLinkEnd インタフェースは、インスタンス化が可能でない。
本インタフェースは、以下の条件付パッケージを含んでいる。

-containedInSubnetworkListPackage: 本リンクエンド　オブジェクトインスタンスがサブネットワーク管理オブジェクトの名前を引き継いでいない場合に存在する。

本パッケージは分割化を通してコンポーネントサブネットワークを含む集約サブネットワークを識別するために使われる。コンポーネントサブネットワークは（互換性のある信号識別子を持つ別の networkR1 管理ドメインと関連する）異なる layerNetworkDomain 名がつけてよい。
LayerNetworkDomain はポリシにより許可されていれば集約サブネットワークでもよい。

-　userLabelPackage: ユーザラベルがサポートされていれば存在する。

```
*/
       valuetype AbstractLinkEndValueType: itut_x780::ManagedObjectValueType
       {
              public PointCapacityType   availableLinkEndCapacity;
                     // GET
```

```
              public AbstractLinkNameType       linkPointer;
                     // GET
              public SubnetworkNameSetType      containedInSubnetworkList;
                     // conditional
                     // containedInSubnetworkListPackage,
                     // GET-REPLACE, ADD-REMOVE
              public Istring                    userLabel;
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
       }; // valuetype AbstractLinkEndValueType

       interface AbstractLinkEnd: itut_x780::ManagedObject
       {
/**
以下の属性は余剰容量を保有するリンクエンドに関連するネットワーク CTP 数又はリンクエンドに関連する余剰帯域幅
を示す。
*/
              PointCapacityType availableLinkEndCapacityGet ()
                     raises (itut_x780::ApplicationError);
/**
以下の属性は、リンクエンドからリンクをポイントする。
*/
              AbstractLinkNameType linkPointerGet ()
                     raises (itut_x780::ApplicationError);
/**
以下の属性は、与えられたレイヤのリンクエンドを含む親サブネットワークのリストを定義する。

*/
              SubnetworkNameSetType containedInSubnetworkListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListSet
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListAdd
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListRemove
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

/**
以下のメソッドは関連するオブジェクトに使い勝手のよい名前を返す。
*/
              Istring userLabelGet ()
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

/**
以下のメソッドは関連するオブジェクトに使い勝手のよい名前を割当てる。
*/
              void userLabelSet
                     (in Istring label)
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, attributeValueChange)

       }; // interface AbstractLinkEnd

/**
```

6.5.2.1 LogicalLinkEnd

LogicalLinkEnd インタフェースをサポートする管理オブジェクトは論理リンクの終端を表す。
リンクエンドリストパッケージ中の Network CTP が存在する場合、論理リンクエンドに存在するネットワーク CTP を示す。論理リンクエンドと（論理リンクに関連する）ネットワーク CTP の間にはネームバインディングはつけられていない。

本インタフェースは、以下の条件付パッケージを含む。

- networkCTPsInLinkEndListPackage: 伝送技術により予め提供されたネットワーク CTP がサポートされていれば存在する。

```
*/
       valuetype LogicalLinkEndValueType: AbstractLinkEndValueType
       {
                     // linkEndCapacityPackage, has actions only
              public PointDirectionalityType    logicalEndDirectionality;
                     // GET
              public NetworkCTPAbstractNameSetType networkCTPInLinkEndList;
                     // conditional
                     // networkCTPsInLinkEndListPackage
                     // GET
       }; // valuetype LogicalLinkEndValueType


       interface LogicalLinkEnd: AbstractLinkEnd
       {
/**
以下の属性は、関連するリンクエンド管理オブジェクトがシンク、ソース、又は双方向であるかどうかを規定する。
*/
              PointDirectionalityType logicalEndDirectionalityGet ()
                            raises (itut_x780::ApplicationError);

/**
以下の属性は論理リンクエンド管理オブジェクト中に存在するネットワーク CTP をリストとして表す。

*/
              NetworkCTPAbstractNameSetType networkCTPInLinkEndListGet ()
                     raises (itut_x780::ApplicationError,
                            NOnetworkCTPsInLinkEndListPackage);

/**

以下の２つのアクションはリンクエンドの容量管理のサポートを提供する。本アクションは、リンクエンドに対するネットワーク CTP かつ／または帯域幅の割当てと解放のアクションを規定する。
*/

/**
以下のアクションはネットワーク CTP を論理リンクエンドへ割当てる。
*/
              void assignNetworkCTPOnLogicalLinkEnd
                     (in NetworkCTPAbstractNameSetType
                            requestedNetworkCTPList,
                     out NetworkCTPAbstractNameSetType
                            resultingNetworkCTPList)
                     raises (itut_x780::ApplicationError,
                            LinkEndAndNetworkCTPNotCompatible,
                            InvalidNetworkCTP,
                            NotEnoughNetworkCTP,
                            NetworkCTPAlreadyAssigned,
                            InconsistentSignalIdentification,
                            InconsistentDirectionality,
                            FailureToSetNetworkCTPCallerId,
                            FailureToDecreaseCapacity);

/**
以下のアクションは、レイヤドメイン内のリンクコネクションから、同一ドメイン内の論理リンクコネクションへの割当てを解除する。
*/
              void deassignNetworkCTPFromLogicalLinkEnd
                     (in NetworkCTPAbstractNameSetType
                            requestedNetworkCTPList)
```

```
                raises (itut_x780::ApplicationError,
                        LinkEndAndNetworkCTPNotCompatible,
                        InvalidNetworkCTP,
                        NotAssignedToCaller,
                        FailureToDeassignNetworkCTP,
                        FailureToIncreaseCapacity);
        }; // interface LogicalLinkEnd

    interface LogicalLinkEndFactory: itut_x780::ManagedObjectFactory
    {
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in SubnetworkNameSetType containedInSubnetworkList,
                        // conditional
                        // containedInSubnetworkListPackage,
                        // GET-REPLACE, ADD-REMOVE
                in Istring userLabel)
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);
    }; // interface LogicalLinkEndFactory

/**
```

6.5.2.2 TopLinkEnd (Topological Link End)

TopLinkEnd (Topological Link End) インタフェースをサポートする管理オブジェクトは、点の観点から見えるトポロジカルリンクエンドを表現する。

Top Link End オブジェクトはサーバレイヤの唯一の TTP に関連づけられている。

個々の属性や通知の用語の説明の詳細は以下のとおりである。

-総リンクエンド容量: 利用可能なネットワーク CTP 総数または帯域幅
-最大ネットワーク CTP 数: 柔軟な帯域管理使用によりリンクエンドで利用可能なネットワーク CTP 数の最大値

-潜在リンクエンド容量:提供可能な潜在的ネットワーク CTP 数または潜在的帯域幅

-提供リンクエンド容量: 提供されたネットワーク CTP または帯域幅

-提供ネットワーク CTP 数: 柔軟な帯域管理によりリンクエンドに割当てられたネットワーク CTP 数

属性値変化通知は、totalLinkEndCapacity, maximumNetworkCTPCount, potentialLinkEndCapacity, provisionedLinkEndCapacity 又は provisionedNetworkCTPCount が変化した場合に発信される。

本インタフェースは、以下の条件付きパッケージを含んでいる。

- totalLinkEndCapacityPackage: 伝送技術により予め提供されるアダプテーション、リンクコネクション、又はリンク管理がサポートされる場合、存在する。

- maximumNetworkCTPCountPackage: 柔軟な帯域幅割当てがサポートされる場合、存在する。

- potentialLinkEndCapacityPackage: 伝送技術により予め提供されるアダプテーション、リンクコネクション、又はリンク管理がサポートされる場合、存在する。

- provisonedLinkEndCapacityPackage: 伝送技術によりサポートされる予め提供されるアダプテーション、リンクコネクション、又はリンク管理がサポートされる場合、存在する。

- provisionedNetworkCTPCountPackage: 柔軟な帯域割当てがサポートされる場合、存在する。

```
*/
    valuetype TopLinkEndValueType: AbstractLinkEndValueType
    {
```

```
            public PointDirectionalityType   pointDirectionality;
                   // GET
            public NetworkTTPAbstractNameSetType serverTTPPointer;
                   // GET
            public PointCapacityType   totalLinkEndCapacity;
                   // conditional
                   // totalLinkEndCapacityPackage
                   // GET
            public long                maximumNetworkCTPCount;
                   // conditional
                   // maximumNetworkCTPCountPackage
                   // GET
            public PointCapacityType   potentialLinkEndCapacity;
                   // conditional
                   // potentialLinkEndCapacityPackage
                   // GET
            public PointCapacityType   provisionedLinkEndCapacity;
                   // conditional
                   // provisonedLinkEndCapacityPackage
                   // GET
            public long                provisionedNeworkCTPCount;
                   // conditional
                   // provisionedNetworkCTPCountPackage
                   // GET
      }; // valuetype TopLinkEndValueType


      interface TopLinkEnd : AbstractLinkEnd
      {

            PointDirectionalityType pointDirectionalityGet ()
                  raises (itut_x780::ApplicationError);

/**
以下の属性は他レイヤで CTP かつ／またはリンクエンドにサービスを提供し得る TTP を定義する。通常、高次レイヤの
TTP が低次レイヤの CTP にサービスを提供する。
*/
            NetworkTTPAbstractNameSetType serverTTPPointerGet ()
                  raises (itut_x780::ApplicationError);

/**
以下の属性は、リンクエンドに関連するネットワーク CTP 総数又はリンクエンドの総帯域幅のいずれかであるリンクエ
ンドの総容量を示す。

*/
            PointCapacityType totalLinkEndCapacityGet ()
                  raises (itut_x780::ApplicationError,
                        NOtotalLinkEndCapacityPackage);

/**
以下の属性は、柔軟な帯域割当てがサポートされている場合にリンクエンドに関連するネットワーク CTP の最大値を示
す。
*/
            long maximumNetworkCTPCountGet ()
                  raises (itut_x780::ApplicationError,
                        NOmaximumNetworkCTPCountPackage);

/**
以下の属性はまだリンクエンドに割当てされていないがサーバトレール端点から今後リンクエンドに割当てされ得るネッ
トワーク CTP 数又は帯域幅を示す。
*/
            PointCapacityType potentialLinkEndCapacityPackageGet ()
                  raises (itut_x780::ApplicationError,
                        NOpotentialLinkEndCapacityPackage);

/**
以下の属性はリンクエンドに割当てされたネットワーク CTP 数又はリンクエンドに割当てされた帯域幅を示す。
*/
            PointCapacityType provisionedLinkEndCapacityGet ()
                  raises (itut_x780::ApplicationError,
                        NOprovisionedLinkEndCapacityPackage);

/**
```

以下の属性は柔軟な帯域割当てがサポートされる場合にリンクエンドに関連して割り当てられたネットワークCTPの数を示す。
*/
```
              long provisionedNetworkCTPCountGet ()
                     raises (itut_x780::ApplicationError,
                            NOprovisionedNetworkCTPCountPackage);

/**
```
以下の２つのアクションはトポロジカルリンクエンドの容量管理機能をサポートする。本アクションは、トポロジカルリンクエンドに対するネットワークCTPかつ／または帯域の割当てと解放を規定する。
```
*/

/**
```
以下のアクションはネットワークCTPを追加する事により、又は利用可能な帯域幅を増加させる事により、トポロジカルリンクに対する容量を追加する。
```
*/
              void addCapacityToTopLinkEnd
                     (in RequestedPointCapacityType requestedCapacity,
                     out PointCapacityType resultingCapacity,
                     out NetworkCTPAbstractNameSetType
                            resultingNetworkCTPList,
                     out PointCapacityType resultingProvisionedCapacity)
                     raises (itut_x780::ApplicationError,
                            NoSuchLinkEnd,
                            InsufficientCapacity,
                            InvalidChannelsNumber,
                            ChannelsAlreadyProvisioned,
                            FailureToCreateLC,
                            FailureToAssociateLC,
                            FailureToSupportLC,
                            FailureToIncreaseCapacity);

/**
```
以下のアクションは、トポロジカルリンクエンドからのネットワークCTPの削除かつ／または帯域の削除によりトポロジカルリンクから容量を削除する。
本アクションは、動的帯域が割当てられていない場合、resultingLinkConnectionList に空を返す。
```
*/
              void removeCapacityFromTopLinkEnd
                     (in RequestedPointCapacityType requestedCapacity,
                     out PointCapacityType resultingCapacity,
                     out LinkConnectionNameSetType
                            resultingLinkConnectionList)
                     raises (itut_x780::ApplicationError,
                            NoSuchLinkEnd,
                            InsufficientCapacity,
                            InvalidChannelsNumber,
                            FailureToRemoveLC,
                            FailureToDecreaseCapacity);

       };  // interface TopLinkEnd

       interface TopLinkEndFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID,    // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList,
                     in SubnetworkNameSetType containedInSubnetworkList,
                            // conditional
                            // containedInSubnetworkListPackage,
                            // GET-REPLACE, ADD-REMOVE
                     in Istring userLabel)
                            // conditional
                            // userLabelPackage
                            // GET-REPLACE
                     raises (itut_x780::ApplicationError,
                            itut_x780::CreateError);

       }; // interface TopLinkEndFactory
/**
```

6.5.3 AccessGroup

AccessGroup インタフェースをサポートする管理オブジェクトは管理する目的でネットワークトレール端点をグループとしてまとめる。
本インタフェースは以下の条件付パッケージを含む。

- containedInSubnetworkListPackage: オブジェクトがサブネットワークに含まれる場合、存在する。
本パッケージはコンポーネントサブネットワークが分割化を通して含まれる集約サブネットワークを表す。
コンポーネントサブネットワークは（互換性のある信号識別子を持つ別の networkR1 管理ドメインと関連する）異なる layerNetworkDomain の名前がつけてもよい。本 layerNetworkDomain は、ポリシにより許可されていれば、集約サブネットワークとなり得る。

- LinkPointerListPackage: トポロジ管理がサポートされていれば、存在する。

- userLabelPackage: ユーザラベルがサポートする場合に存在する

```
*/
       valuetype AccessGroupValueType: itut_x780::ManagedObjectValueType
       {
              public NetworkTPNameSetType      accessPointList;
                     // GET-REPLACE, ADD-REMOVE
              public TopEndDirectionalityType  topEndDirectionality;
                     // GET
              public SignalIdType        signalId;
                     // GET
              public SubnetworkNameSetType     containedInSubnetworkList;
                     // conditional
                     // containedInSubnetworkListPackage,
                     // GET-REPLACE, ADD-REMOVE
              public AbstractLinkNameSetType   linkPointerList;
                     // conditional
                     // linkPointerListPackage
                     // GET
              public Istring                   userLabel;
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
       }; // valuetype AccessGroupValueType

       interface AccessGroup: itut_x780::ManagedObject
       {
/**
```

アクセスポイントリスト属性は管理オブジェクトクラス AccessGroup のインスタンス中の全てのネットワークトレール端点リストを示している。

```
*/
              NetworkTPNameSetType accessPointListGet ()
                     raises (itut_x780::ApplicationError);

              void accessPointListSet
                     (in NetworkTPNameSetType accessPointList)
                     raises (itut_x780::ApplicationError,
                            NetworkTTPAndAccessGroupNotCompatible,
                            FailureToAssociateNetworkTTP,
                            FailureToDisassociateNetworkTTP);
              void accessPointListAdd
                     (in NetworkTPNameSetType accessPointList)
                     raises (itut_x780::ApplicationError,
                            NetworkTTPAndAccessGroupNotCompatible,
                            FailureToAssociateNetworkTTP);

              void accessPointListRemove
                     (in NetworkTPNameSetType accessPointList)
                     raises (itut_x780::ApplicationError,
                            NetworkTTPAndAccessGroupNotCompatible,
                            FailureToDisassociateNetworkTTP);

/**
```

TopEndDirectionality 属性タイプは、関係するリンクエンド管理オブジェクトがシンク、ソース、双方向、又は未定義のいずれであるかを指定する。

```
*/
              TopEndDirectionalityType topEndDirectionalityGet ()
                     raises (itut_x780::ApplicationError);

/**
```

```
以下の属性は考慮対象のエンティティが属する（G.805 の意味合いの）レイヤの特性情報を定義する。
本属性はサブネットワークの接続／接続性が可能であるかどうかを決定するために使用される。
*/

              SignalIdType signalIdGet ()
                     raises (itut_x780::ApplicationError);

/**
以下の属性は、与えられたレイヤのアクセスグループを含むサブネットワークのリストを定義する。
*/
              SubnetworkNameSetType containedInSubnetworkListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListSet
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListAdd
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListRemove
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);
/**
以下の属性は、アクセスグループにより終端されたリンクをポイントしている。トポロジカル管理がサポートされている
場合に、サポートされる。
*/
              AbstractLinkNameSetType linkPointerListGet ()
                     raises (itut_x780::ApplicationError,
                            NOlinkPointerListPackage);
/**
以下のメソッドは、関連するオブジェクトに対して使い勝手のよい名前を返す。
*/
              Istring userLabelGet ()
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

/**
以下のメソッドは、関連するオブジェクトに使い勝手のよい名前を割り当てる。
*/
              void userLabelSet
                     (in Istring label)
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

       }; // interface AccessGroup

       interface AccessGroupFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID, // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList,
                     in NetworkTPNameSetType accessPointList,
                            // accessGroupPackage
                            // GET-REPLACE, ADD-REMOVE
                     in SubnetworkNameSetType containedInSubnetworkList,
                            // conditional
                            // containedInSubnetworkListPackage
                            // GET-REPLACE, ADD-REMOVE
                     in Istring userLabel)
                            // conditional
                            // userLabelPackage
                            // GET-REPLACE
                     raises (itut_x780::ApplicationError,
                            itut_x780::CreateError,
                            FailureToSetDirectionality,
                            FailureToCreateAccessGroup);
```

```
            // DELETE_ERROR:
            //     NetworkTTPsExisting
            //     FailureToRemoveAccessGroup

      }; // interface AccessGroupFactory
/**
```

6.5.4 AlarmSeverityAssignmentProfile

AlarmSeverityAssignmentProfile インタフェースをサポートする管理オブジェクトは他の管理オブジェクトに対して警報重要度割当を規定する。本インタフェースのインスタンスは、管理オブジェクトの alarmSeverityAssignmentProfilePointer 属性により参照される。

```
*/
      valuetype AlarmSeverityAssignmentProfileValueType:
            itut_x780::ManagedObjectValueType
      {
            public AlarmSeverityAssignmentSetType
                  alarmSeverityAssignmentList;
                  // GET-REPLACE, ADD-REMOVE
      }; // valuetype AlarmSeverityAssignmentProfileValueType

      interface AlarmSeverityAssignmentProfile: itut_x780::ManagedObject
      {
/**
以下のメソッドは、オブジェクトの警報重要度割り当てリストを取得するために使われる。
*/
            AlarmSeverityAssignmentSetType alarmSeverityAssignmentListGet()
                  raises (itut_x780::ApplicationError);

/**
以下のメソッドは、オブジェクトの警報重要度割当リストに警報を追加する場合に使われる。
属性値変化通知は、本オブジェクトがその通知をサポートしている場合、送信される。例外が発生した場合にはオブジェクトは変化しない。

*/
            void alarmSeverityAssignmentsAdd
                  (in AlarmSeverityAssignmentSetType
                        alarmSeverityAssignmentList)
                  raises (itut_x780::ApplicationError);
/**
以下のメソッドはエントリをオブジェクトの警報重要度割当リストから削除するために使われる。
本オブジェクトがその通知をサポートしている場合、属性値変化通知は送信される。例外が発生した場合にはオブジェクトは変化しない。
*/
            void alarmSeverityAssignmentsRemove
                  (in AlarmSeverityAssignmentSetType
                        alarmSeverityAssignmentList)
                  raises (itut_x780::ApplicationError);
/**
以下のメソッドは送られたリストに従ってオブジェクトの警報重要度割当てリストの全てのエントリを置き換えるために使われる。本オブジェクトがその通知をサポートしている場合、属性値変化通知は送信される。例外が発生した場合にはオブジェクトは変化しない。

*/
            void alarmSeverityAssignmentListSet
                  (in AlarmSeverityAssignmentSetType
                        alarmSeverityAssignmentList)
                  raises (itut_x780::ApplicationError);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                        objectManagementNotificatoinsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                        objectManagementNotificatoinsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                        objectManagementNotificatoinsPackage)
```

```
	}; // interface AlarmSeverityAssignmentProfile

	interface AlarmSeverityAssignmentProfileFactory:
		itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID,    // auto naming if empty string
			out MONameType name,
			in StringSetType packageNameList,
			in AlarmSeverityAssignmentSetType list)
				// alarmSeverityAssignmentProfilePackage
				// GET-REPLACE, ADD-REMOVE
			raises (itut_x780::ApplicationError,
				itut_x780::CreateError);

	}; // interface AlarmSeverityAssignmentProfileFactory
/**
```

6.5.5 ArcIntervalProfile (Alarm Report Control Interval Profile)

本インタフェースは、時間ベースのトランザクションをサポートするための ARC（警報報告制御）状態用デフォルト ARC 時間間隔を定義する ArcIntervalProfile (警報報告制御間隔プロファイル（Alarm Report Control Interval Profile）)機能をサポートする。警報報告抑止モードから警報報告許可モードに遷移する基準の要素（但し、必ずしも、唯一の要素である必要はないが）である時間に従ってエージェント内にて自動的に遷移する ARC 状態にのみ、時間間隔プロファイルは、適用できる。
警報報告許可モードから警報報告抑止モードに遷移する自動的エージェント状態遷移は許可されていない。関連する ARC 状態に設定可能な時間間隔が必要とされる場合にのみ、本オブジェクトに対するサポートが必要である。本インタフェースは以下の条件付きパッケージが含まれる。

- userLabelPackage: ユーザラベルがサポートされる場合に存在する。

```
*/
	valuetype ArcIntervalProfileValueType:
		itut_x780::ManagedObjectValueType
	{
		public ArcTimeType  arcDefaultTimedInterval;
			// GET-REPLACE
		public ArcTimeType  arcDefaultPersistenceInterval;
			// GET-REPLACE
		public Istring             userLabel;
			// conditional
			// userLabelPackage
			// GET-REPLACE
	}; // valuetype ArcIntervalProfileValueType

	interface ArcIntervalProfile: itut_x780::ManagedObject
	{
/**
以下のメソッドは、ARC  NALM-TI 状態の ARC 時間間隔のデフォルト値／初期値を返す。
*/
		ArcTimeType arcDefaultTimedIntervalGet ()
			raises (itut_x780::ApplicationError);
/**
以下のメソッドは、ARC  NALM-TI 状態の ARC 時間間隔のデフォルト値／初期値を変化させる。


*/
		void arcDefaultTimedIntervalSet
			(in ArcTimeType arcDefaultTimedInterval)
			raises (itut_x780::ApplicationError);

/**
以下のメソッドは、ARC  NALM-QI 状態の ARC 時間間隔のデフォルト値／初期値を返す。

*/
		ArcTimeType arcDefaultPersistenceIntervalGet ()
			raises (itut_x780::ApplicationError);

/**
以下のメソッドは、ARC  NALM-QI 状態の ARC 時間間隔のデフォルト値／初期値を変化させる。
*/
```

```
            void arcDefaultPersistenceIntervalSet
                  (in ArcTimeType arcDefaultPersistenceInterval)
                  raises (itut_x780::ApplicationError);

/**
以下のメソッドは、本オブジェクトを識別するために管理システムにより利用され得るラベルを返す。
*/
            Istring userLabelGet ()
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

/**
以下のメソッドは、インスタンスへのラベル割当てに使用される。その値はオブジェクトに対してではなくクライアント
に対しての意味を持っている。もし本属性がサポートされていれば、この値は本オブジェクトから発信される通知に含ま
れるべきである。
*/
            void userLabelSet
                  (in Istring userLabel)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)

      }; // interface ArcIntervalProfile

      interface ArcIntervalProfileFactory: itut_x780::ManagedObjectFactory
      {
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID,    // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList,
                  in ArcTimeType arcDefaultTimedInterval,
                        // arcIntervalProfilePackage
                        // GET-REPLACE
                  in ArcTimeType arcDefaultPersistenceInterval,
                        // arcIntervalProfilePackage
                        // GET-REPLACE
                  in Istring userLabel)
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                  raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);
      }; // interface ArcIntervalProfileFactory

/**
```

6.5.6 CircuitEndPointSubgroup

CircuitEndPointSubgroup インタフェースは、一つの交換機を他のものと直接接続する、一組のサーキット・エンドポイントを表し、このパッケージの中に記載された属性として共通の値を持っている。交換という用語は、PBX が適当な場合はそれを含んでいるのに注意せよ。
*/

```
	valuetype CircuitEndPointSubgroupValueType:
		itut_x780::ManagedObjectValueType
	{
		public long		numberOfCircuits;
			// GET
		public Istring			labelOfFarEndExchange;
			// GET
		public SignallingCapabilityType  signallingCapability;
			// GET
		public InformationTransferCapabilityType
			informationTransferCapability;
			// GET
		public CircuitDirectionalityType circuitDirectionality;
			// GET
		public TransmissionCharacteristicSetType
			transmissionCharacteristics;
			// GET
		public Istring			userLabel;
			// GET-REPLACE
	}; // valuetype CircuitEndPointSubgroup


	interface CircuitEndPointSubgroup: itut_x780::ManagedObject
	{
/**
サーキット・サブグループにおける、サーキットの数を返す。
*/
		long numberOfCircuitsGet ()
			raises (itut_x780::ApplicationError);

/**
このサーキット・サブグループを終端する遠端の交換機(Far End Exchange)のために使い勝手のよい名前を返す。
*/
		Istring labelOfFarEndExchangeGet ()
			raises (itut_x780::ApplicationError);

/**
サーキット・サブグループでサポートされる、シグナリングタイプを返す。
*/
		SignallingCapabilityType signallingCapabilityGet ()
			raises (itut_x780::ApplicationError);

/**
異なったサービスタイプ、即ち、サーキット・サブグループでサポートされる、音声や 64kbits の無制限なデータなどの値を返す。
*/
		InformationTransferCapabilityType
			informationTransferCapabilityGet ()
			raises (itut_x780::ApplicationError);

/**
サーキット・サブグループにおける、サーキットの方向性を返す。
*/
		CircuitDirectionalityType circuitDirectionalityGet ()
			raises (itut_x780::ApplicationError);

/**
```

```
サーキット・サブグループによってサポートされるか、あるいはサポートされない衛星やエコー制御などの異なった伝送
特性を返す。リストに列挙型を含むのは、特定の特性がサポートされるかどうかを示す。
*/
              TransmissionCharacteristicSetType
                     transmissionCharacteristicsGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドはオブジェクトのために使い勝手のよい名前を返す。
*/
              Istring userLabelGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは使い勝手のよい名前をオブジェクトに割当てる。
*/
              void userLabelSet (in Istring userLabel)
                     raises (itut_x780::ApplicationError);

              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, attributeValueChange)

       }; // interface CircuitEndPointSubgroup


       interface CircuitEndPointSubgroupFactory:
              itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID,    // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList,
                     in Istring userLabel)
                            // circuitEndPointSubgroupPackage
                            // GET-REPLACE
                     raises (itut_x780::ApplicationError,
                            itut_x780::CreateError);

       }; // interface CircuitEndPointSubgroupFactory


/**
```

6.5.7 CrossConnection

CrossConnection クラスの管理オブジェクトは、端点か、From Termination 属性に記載されている GTP オブジェクトと、端点、またはこの管理オブジェクトの To Termination 属性に列挙されている GTP オブジェクト間の割付け関係を表す。

To Termination 属性はいつも非空(non-nil)となる。 From termination 属性は、ポイントツーマルチポイント構成の場合に限り、空となる。もし From Termination 属性が空値であるならば、包含する多重点クロスコネクション（Multipoint Cross-Connection）管理オブジェクトの From Termination 属性に記載されている端点オブジェクトか GTP オブジェクトと、この管理オブジェクトの To Termination 属性に記載されている端点オブジェクトか GTP オブジェクトの間には、割付け関係がある。

以下の間でポイントツーポイントのクロスコネクションを確立することができる: CTP シンク(TPType がこれに等しい), CTP 双方向, TTP ソース, TTP 双方向, あるいは GTP、のうちのひとつ; そして、CTP ソースのひとつ, CTP 双方向, TTP シンク, TTP 双方向, あるいは GTP、のうちのひとつ。

片方向のクロスコネクションでは、To Termination 属性(このオブジェクトか、包含する mpCrossConnection 中に)によって示される、端点か GTP オブジェクトと、From Termination によって示された端点か GTP オブジェクトが、トラヒックがこれらの管理オブジェクトによって表される端点の間を流れることができるような方法で関係づけられる。双方向のクロスコネクションでは、情報は両方の方向に流れる。

もし From Termination と To Termination 属性に列挙されているオブジェクトが GTP であるならば、From Termination GTP の n 番目の要素は To Termination GTP(あらゆる n について)の n 番目の要素に関連する。

もし fromTermination 属性が空値であるならば、方向性属性に、値' unidirectional'がなければならない。

From Terminations の全レートは To Termination の全レートと等しくなければならない。

属性、信号タイプ（Signal Type）はクロスコネクトされた信号について説明する。クロスコネクトされた端点か GTP には、コンパチブルな信号タイプがなければならない。

このオブジェクトクラスのインスタンスが、多重点クロスコネクションに含まれ、多重点クロスコネクションの操作状態が'disabled'である場合、このオブジェクトの操作状態もまた'disabled'となる。

以下は運用状態と操作状態の属性の定義である:

運用状態（Administrative State）:

- Unlocked: クロスコネクションオブジェクトは、管理上アンロックされている。トラヒックは、コネクションを通過することを許されている。
- Locked: どんなトラヒックもクロスコネクションを通過することができない。クロスコネクトされた端点の、接続ポインタは、空である。

操作状態（Operational State）:

- Enabled: クロスコネクトが通常機能を実行している。
- Disabled: クロスコネクトには通常のクロスコネクト機能を実行することができない。

このインタフェースは、"redline"、"userLable"、および"crossConnectionName"といった属性を含んでおり、それは接続（Connection）を、M.3100 における NamedConnection に拡張するものである。

このインタフェースは、次の条件付きパッケージを含んでいる。

- redlinePackage: インスタンスがそれをサポートすると存在する。

- userLabelPackage: インスタンスがそれをサポートすると存在する。

- namedCrossConnectionPackage: インスタンスがそれをサポートすると存在する。

- redlinePackage: インスタンスがそれをサポートすると存在する。

- userLabelPackage: インスタンスがそれをサポートすると存在する。

- namedCrossConnectionPackage: インスタンスがそれをサポートすると存在する。

```
*/
       valuetype CrossConnectionValueType: itut_x780::ManagedObjectValueType
       {
              public AdministrativeStateType    adminstrativeState;
                     // GET-REPLACE
              public OperationalStateType       operationalState;
                     // GET
              public SignalIdType        signalId;
                     // GET
              public TPNameType          fromTermination;
                     // GET
              public TPNameType          toTermination;
                     // GET
              public DirectionalityType  directionality;
```

```
                // GET
        public boolean                  redline;
                // conditional
                // redlinePackage
                // GET-REPLACE
        public Istring                  userLabel;
                // conditional
                // userLabelPackage
                // GET-REPLACE
        public Istring                  crossConnectionName;
                // conditional
                // namedCrossConnectionPackage
                // GET-REPLACE
};  // valuetype CrossConnectionValueType


interface CrossConnection : itut_x780::ManagedObject
{
        AdministrativeStateType administrativeStateGet ()
                raises (itut_x780::ApplicationError);

        void administrativeStateSet
                (in AdministrativeStateType adminstrativeState)
                raises (itut_x780::ApplicationError);

        OperationalStateType operationalStateGet ()
                raises (itut_x780::ApplicationError);

/**
以下の属性はクロスコネクション、TP プールまたは GTP の信号タイプを独自に特定する。信号タイプは、単一（simple）、バンドル（bundle）、または複合（complex）という場合がある。信号タイプが単一であるならば、それは単一タイプの特性情報から成る。信号タイプがバンドルであるならば、それはすべて同じ特性情報をもつ、多くの信号タイプで作られる。信号タイプが複合であるならば、それは束ねられた信号タイプの系列から成る。複雑な信号タイプの順序は、信号の実際の構成を表わす。
*/
        SignalIdType signalIdGet ()
                raises (itut_x780::ApplicationError);

/**
以下の属性は、TTP(ソースの、または、双方向の)、CTP(シンクの、または、双方向の)、またはこれらのカテゴリの一つのメンバで構成される GTP を特定する。
*/
        MONameType fromTerminationGet ()
                raises (itut_x780::ApplicationError);

/**
以下の属性は CTP(ソースの、または双方向の)、TTP(シンクの、または双方向の)、またはこれらのカテゴリの一つのメンバで構成される GTP を特定する。
*/
        MONameType toTerminationGet ()
                raises (itut_x780::ApplicationError);

/**
方向性（Directionality）属性タイプは、関連する管理オブジェクトが、片方向かそれとも双方向であるかを指定する。それが uni-directional であるならば、情報の流れは、A end から Z end である。
*/
        DirectionalityType directionalityGet ()
                raises (itut_x780::ApplicationError);

/**
以下の属性は、関連する管理オブジェクトが、運用限界を越えたかどうか特定する。例えば、敏感なサーキットの一部であるとして特定される場合である。
*/
        boolean redlineGet ()
                raises (itut_x780::ApplicationError,
                        NOredlinePackage);
```

```
                void redlineSet
                        (in boolean redline)
                        raises (itut_x780::ApplicationError,
                                NOredlinePackage);

                Istring userLabelGet ()
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                void userLabelSet
                        (in Istring userLable)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

/**
以下の属性はクロスコネクトのための、記述的な名前である。
*/
                Istring crossConnectionNameGet ()
                        raises (itut_x780::ApplicationError,
                                NOnamedCrossConnectionPackage);

                void crossConnectionNameSet
                        (in Istring crossConnectionName)
                        raises (itut_x780::ApplicationError,
                                NOnamedCrossConnectionPackage);

                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectCreation,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, attributValueChange,
                        attributeValueChangeNotificationPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, stateChange,
                        stateChangeNotificationPackage)

        }; // interface CrossConnection


        interface CrossConnectionFactory: itut_x780::ManagedObjectFactory
        {
                itut_x780::ManagedObject create
                        (in NameBindingType nameBinding,
                        in MONameType superior,
                        in string reqID,    // auto naming if empty string
                        out MONameType name,
                        in StringSetType packageNameList,
                        in AdministrativeStateType adminstrativeState,
                                // crossConnectionPackage
                                // GET-REPLACE
                        in boolean redline,
                                // conditional
                                // redlinePackage
                                // GET-REPLACE
                        in Istring userLabel,
                                // conditional
                                // userLabelPackage
                                // GET-REPLACE
                        in Istring crossConnectionName)
                                // conditional
                                // namedCrossConnectionPackage
                                // GET-REPLACE
                        raises (itut_x780::ApplicationError,
                                itut_x780::CreateError);
```

```
};  // CrossConnectionFactory

/**
```

6.5.8 Equipment

装置(Equipment)インタフェースは 1995 年の M.3100 Equipment R1 オブジェクト以降にモデル化された。それは、以前の M.3100 Equipment オブジェクトのサブクラスである。

装置オブジェクトのクラスは、管理された要素の物理的な部品を表す管理オブジェクトのクラスであり、取り替え可能な部品を含む。このオブジェクトクラスのインスタンスは単一の地理的な位置に存在している。装置は別の装置の中で入れ子にされて、その結果、包含関係を生成することができる。装置タイプは、このオブジェクトクラスをサブクラス化して特定されなければならない。サブクラスの名前か属性のどちらかが、装置タイプを特定するのに使用することができる。

オブジェクトが属性値の変化通知をサポートするとき、以下の属性の一つの値が変化するとき、attributeValueChange 通知が発出されるものとする:警報状態、影響を受けるオブジェクトのリスト、ユーザラベル、バージョン、位置の名称、および現在の問題リスト。上記の属性のサポートが条件付きであるので、対応する属性が管理オブジェクトでサポートされるときだけ、属性値の変化通知を発出するための振る舞いが適用される。オブジェクトが状態の変化通知をサポートするとき、運用状態か操作状態の値が変化するならば(これらの属性がサポートされるならば)stateChangeNotification が発出されるべきである。

この値のタイプは、一回の操作で装置属性の全てを検索するために利用される。それらがサポートされないと、多くのサポートされない属性が空の文字列かリストとして返されるであろう。でも、空の文字列値を受領すると、属性がサポートされないという意味ではない。

本インタフェースは次の条件付パッケージを含んでいる。

- administrativeOperationalStatesPackage: インスタンスがそれをサポートするときに存在する。

- affectedObjectListPackage: インスタンスがそれをサポートするときに存在する。

- alarmSeverityAssignmentPointerPackage: インスタンスが警報重要度の構成をサポートするときに存在する。

- userLabelPackage:ユーザラベルがサポートされるとき存在する。

- vendorNamePackage: ユーザラベルがサポートされるとき存在する。

- versionPackage: ユーザラベルがサポートされるとき存在する。

- locationNamePackage: インスタンスがそれをサポートするときに存在する。

- arcPackage:インスタンスがアラームレポート制御をサポートするときに存在する。

- arcRetrieveAlarmDetailPackage:
インスタンスが、それをサポートし、arcPackage もまたサポートされるときに存在する。

```
*/
       valuetype EquipmentValueType: itut_x780::ManagedObjectValueType
       {
              public ReplaceableType           replaceable;
                     // GET, SET-BY-CREATE
              public Istring                   serialNumber;
                     // GET
              public Istring                   type;
                     // GET, SET-BY-CREATE
              public MONameSetType             supportedByObjectList;
                     // GET-REPLACE, ADD-REMOVE
              public AlarmStatusType           alarmStatus;
                     // GET
              public CurrentProblemSetType     currentProblemList;
                     // GET
              public OperationalStateType      operationalState;
```

```
                // conditional
                // administrativeOperationalStatesPackage
                // GET
        public AdministrativeStateType   administrativeState;
                // conditional
                // administrativeOperationalStatesPackage
                // GET-REPLACE
        public MONameSetType             affectedObjects;
                // conditional
                // affectedObjectListPackage
                // GET
        public AlarmSeverityAssignmentProfileNameType
                alarmSeverityAssignmentProfilePointer;
                // conditional
                // alarmSeverityAssignmentPointerPackage
                // GET-REPLACE
        public Istring                   userLabel;
                // conditional
                // userLabelPackage
                // GET-REPLACE
        public Istring                   vendorName;
                // conditional
                // vendorNamePackage
                // GET-REPLACE
        public Istring                   version;
                // conditional
                // versionPackage
                // GET-REPLACE
        public Istring                   locationName;
                // conditional
                // locationNamePackage
                // GET-REPLACE
        public ArcStateType        arcState;
                // conditional
                // arcPackage
                // GET
        public ArcQIStatusType           arcQIStatus;
                // conditional
                // arcPackage
                // GET
        public ArcProbableCauseSetType   arcProbableCauseList;
                // conditional
                // arcPackage
                // GET-REPLACE, ADD-REMOVE
        public ArcIntervalProfileNameType arcIntervalProfilePointer;
                // conditional
                // arcPackage
                // GET-REPLACE
        public ArcTimeType         arcManagementRequestedInterval;
                // conditional
                // arcPackage
                // GET-REPLACE
        public ArcTimeType         arcTimeRemaining;
                // conditional
                // arcPackage
                // GET
    }; // valuetype EquipmentValueType


    interface Equipment: itut_x780::ManagedObject
    {
/**
置換可能（Replaceable）属性タイプは、関連する資源が取替え可能であるか、または取替え不可能であるかを示す。
*/
        ReplaceableType replaceableGet ()
                raises (itut_x780::ApplicationError);

/**
```

```
以下のメソッドは、物理資源の通し番号を返す。
*/
              Istring serialNumberGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、M.3100 corrigendum の Equipment R2 オブジェクトに定義されているような資源のタイプの原
文の記述を含む文字列を返す。
*/
              Istring typeGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、特定の管理オブジェクトに直接的に影響することができるオブジェクトのインスタンスのリストを返
す。オブジェクトのインスタンスは物理的なものと論理的なオブジェクトを含んでいる。この属性は指定されるべきであ
る内部の詳細ではなく、管理に要求される必要なレベルの詳細だけを実行する。もしそのオブジェクトに対して、管理オ
ブジェクトをサポートするオブジェクトのインスタンスが未知であるならば、このメソッドは空のリストを返す。
*/
              MONameSetType supportedByObjectListGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、このオブジェクトをサポートしているオブジェクトのインスタンスのリストを置き換えるために使用
される。
*/
              void supportedByObjectListSet
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、このオブジェクトをサポートしているオブジェクトのインスタンスのリストに加えるのに使用される。
*/
              void supportedByObjectListAdd
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、このオブジェクトをサポートしているオブジェクトのインスタンスのリストからオブジェクトを取り
除くために使用される。
*/
              void supportedByObjectListRemove
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、オブジェクトの現在の警報状態を返す。
*/
              AlarmStatusType alarmStatusGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは、重要度によって、管理オブジェクトに関連づけられる、現在の既存の問題を返す。
*/
              CurrentProblemSetType currentProblemListGet ()
                     raises (itut_x780::ApplicationError);

              OperationalStateType operationalStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOadministrativeOperationalStatesPackage);

              AdministrativeStateType administrativeStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOadministrativeOperationalStatesPackage);

              void administrativeStateSet
                     (in AdministrativeStateType administrativeState)
```

```
                raises (itut_x780::ApplicationError,
                       NOadministrativeOperationalStatesPackage);

/**
影響するオブジェクトリスト（Affected Object List）属性は、状態の変化か、この管理オブジェクトの削除によって直接影響するオブジェクトのインスタンスを指定する。属性は指定されるべきである内部の詳細ではなく、管理に要求される必要なレベルの詳細だけを強制する。
*/
             MONameSetType affectedObjectsGet ()
                raises (itut_x780::ApplicationError,
                       NOaffectedObjectListPackage);

/**
以下のメソッドは、警報重要度割付プロフィール・ポインタを検索するために使用される。もし警報重要度割付プロフィール・ポインタが空であるならば、以下の 2 つの選択のうち一つが、警報を報告するときに、適用される:
a) 管理されたシステムが重要度を割り付けるか、b) '不定'の値が使用されている。
*/
             AlarmSeverityAssignmentProfileNameType
                alarmSeverityAssignmentProfilePointerGet ()
                raises (itut_x780::ApplicationError,
                       NOalarmSeverityAssignmentPointerPackage);

/**
以下のメソッドは、警報重要度割付プロフィール・ポインタを設定するのに使用される。
*/
             void alarmSeverityAssignmentProfilePointerSet
                (in AlarmSeverityAssignmentProfileNameType profile)
                raises (itut_x780::ApplicationError,
                       NOalarmSeverityAssignmentPointerPackage);

/**
以下のメソッドは、オブジェクトを特定するために、管理システムによって使用されることができるラベルを返す。
*/
             Istring userLabelGet ()
                raises (itut_x780::ApplicationError,
                       NOuserLabelPackage);

/**
以下のメソッドは、ラベルをインスタンスに割当てるために使用される。その値はオブジェクトにではなく、クライアントに対して意味を持つことができる。もしこの属性がサポートされるならば、それはこのオブジェクトによって発出される通知に含まれるべきである。
*/
             void userLabelSet
                (in Istring userLabel)
                raises (itut_x780::ApplicationError,
                       NOuserLabelPackage);

/**
以下のメソッドは、関連資源の提供者の名前を返す。
*/
             Istring vendorNameGet ()
                raises (itut_x780::ApplicationError,
                       NOvendorNamePackage);

/**
以下のメソッドは、関連資源の提供者の名前を設定する。
*/
             void vendorNameSet
                (in Istring vendorName)
                raises (itut_x780::ApplicationError,
                       NOvendorNamePackage);

/**
以下のメソッドは、関連資源の版数を返す。
*/
             Istring versionGet ()
```

```
                raises (itut_x780::ApplicationError,
                       NOversionPackage);

/**
以下のメソッドは、関連資源の版数を設定する。
*/
              void versionSet
                       (in Istring version)
                       raises (itut_x780::ApplicationError,
                              NOversionPackage);

/**
以下のメソッドは、関連資源の物理的位置を返す。
*/
              Istring locationNameGet ()
                       raises (itut_x780::ApplicationError,
                              NOlocationNamePackage);

/**
以下のメソッドは、関連資源の物理的位置を設定する。
*/
              void locationNameSet
                       (in Istring locationName)
                       raises (itut_x780::ApplicationError,
                              NOlocationNamePackage);

/**
状態変化通知は、この属性の値変更を示すために利用されなければならないであろう。
*/
              ArcStateType arcStateGet ()
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

/**
状態変化通知も属性値の変化通知も、この属性のための値の変化を示すために使用されないだろう。
*/
              ArcQIStatusType     arcQIStatusGet ()
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

              ArcProbableCauseSetType    arcProbableCauseListGet ()
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

              void arcProbableCauseListSet
                       (in ArcProbableCauseSetType arcProbableCauseList)
                       raises (itut_x780::ApplicationError,
                              NotSupportedProbableCause,
                              NOarcPackage);

              void arcProbableCauseListAdd
                       (in ArcProbableCauseSetType arcProbableCauseList)
                       raises (itut_x780::ApplicationError,
                              NotSupportedProbableCause,
                              NOarcPackage);

              void arcProbableCauseListRemove
                       (in ArcProbableCauseSetType arcProbableCauseList)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

/**
以下のメソッドは、関連する ARC 間隔プロファイル（Interval Profile）オブジェクトを特定する ARC 間隔プロファイルポインタ（Interval Profile Pointer）の値を検索するのに使用される。もし連続性と時間間隔が使用されないならば、この属性の値は空である。(即ち、nalm-qi と nalm-ti 状態のために使用されない)
*/
              ArcIntervalProfileNameType arcIntervalProfilePointerGet ()
```

```
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下のメソッドは、関連する ARC 間隔プロファイルオブジェクトを特定する ARC 間隔プロファイルポインタの値を変更するのに使用される。もし連続性と時間間隔が使用されないならば、この属性の値は空である。(即ち、nalm-qi と nalm-ti 状態のために使用されない)
*/
            void arcIntervalProfilePointerSet
                (in ArcIntervalProfileNameType profile)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下のメソッドは ARC 管理要求間隔（Management Requested Interval）の値を検索するのに使用され、これは ARC 間隔のための、管理要求時間を特定するものである。この属性は、管理要求が発生したときか、あるいは資源が自動的に alm 状態に推移するときにのみ、値を変更する。この属性の値を変えるという管理要求は、それが無効であるときに、否定される。例えば管理された資源が alm か nalm 状態にあるときである。特定の瞬間での管理要求によって ARC 間隔を調整することができるか否かに関係なく、この属性の値は反映する。
*/
            ArcTimeType arcManagementRequestedIntervalGet ()
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下のメソッドは ARC 管理要求間隔の値を変えるのに使用され、それは、ARC 間隔のための管理要求時間を特定するものである。
*/
            void arcManagementRequestedIntervalSet
                (in ArcTimeType time)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下のメソッドは、ARC 間隔(即ち、nalm-qi 状態の持続間隔や、nalm-ti 状態の 時間間隔)の残りの時間を検索するために利用される。それは必ずしもその状態の残り時間を示すというわけではないことに注意せよ。例えば、arcTimeRemaining が nalm-qi 状態において 30 分であるとして、ARC 間隔 タイマが期間が切れる前に条件付の問題が管理された資源のために上げられるな ら、それは、再度、条件付の問題が解消し、タイマを再起動して、再び残りの時間 を減少させ始めるまで、タイマを中止して無期限に待つことになる。nalm-ti、nalm、または nalm-qi 状態に資源の遷移があるとき、この属性値は管 理要求間隔に初期化される。動作中のタイマが全く無いとき、その値はタイマが動作していないことを示す (即ち、時間調整が全く行われない)。何れの属性値の変化通知も、この属性値における変化のために送られないことに注意せよ。
*/
            ArcTimeType arcTimeRemainingGet ()
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下のメソッドは、アラーム報告をオンにするか、またはアラーム報告をオフにするためどちらかに使用される。これは、必要な ARC 状態を特定することによって、達成される。いくつかの場合、状態が限定された資源タイプのためにサポートされないので、動作は否定される。状態を指定することに加えて、マネージャは一回の使用のためのデフォルト以外に arcInterval を要求することができる。そのようなオーバーライドは nalm-qi と naml-ti 状態への変遷にのみ適用できる。
*/
            boolean arcControl
                (in ArcControlRequestType request)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下の操作は宣言している警報状態のための警報の詳細を返す。また、この機能をサポートするために、インタフェースは arcPackage をサポートするものとする。
*/
            ArcAlarmDetailSetType arcRetrieveAlarmDetail ()
```

```
                raises (itut_x780::ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);


        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, equipmentAlarm,
                equipmentsEquipmentAlarmR2Package)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, environmentalAlarm,
                environmentalAlarmR2Package)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, processingErrorAlarm,
                processingErrorAlarmR2Package)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, communicationsAlarm,
                tmnCommunicationsAlarmInformationR1Package)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)

};  // interface Equipment


interface EquipmentFactory: itut_x780::ManagedObjectFactory
{
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in ReplaceableType replaceable,
                        // equipmentPackage
                        // GET, SET-BY-CREATE
                in Istring type,
                        // equipmentR2Package
                        // GET, SET-BY-CREATE
                in MONameSetType supportedByObjectList,
                        // may be nil
                        // equipmentR1Package
                        // GET-REPLACE, ADD-REMOVE
                in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeOperationalStatesPackage
                        // GET-REPLACE
                in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                in Istring userLabel,
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                in Istring vendorName,
                        // conditional
                        // vendorNamePackage
                        // GET-REPLACE
                in Istring version,
                        // conditional
```

```
                        // versionPackage
                        // GET-REPLACE
                    in Istring locationName,
                        // conditional
                        // locationNamePackage
                        // GET-REPLACE
                    in ArcProbableCauseSetType arcProbableCauseList,
                        // conditional
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval)
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

    }; // interface EquipmentFactory



/**
```

6.5.8.1 CircuitPack

CircuitPack インタフェースをサポートする管理オブジェクトは Network Element の装置ホルダに挿入するか取り除くことができるプラグインの取替え可能なユニットを表す。プラグ・イン・カードに関する例は、回線カード、プロセッサ、および電源ユニットを含んでいる。継承する属性 textType(構文 GraphicString の)は、サーキットパックのタイプを示すのに使用される。この属性の値は包含される equipmentHolder オブジェクトの(構文 PrintableString の)acceptableCircuitPackTypeList 属性の値の一つに適合すべきである。サーキットパックのタイプが、PrintableString 文字セットの外部の GraphicString 文字セットであると、それは acceptableCircuitPackList 属性のどんな値にも適合しないであろう。この場合、circuitPackR1 のどんなインスタンスも、インスタンス化されるべきではない、そして、equipmentHolder オブジェクトの holderStatus 属性に、値の'unknownType'があるはずである。属性 availabilityStatus は、サーキットパックの可用性を示すのに使用される。availabilityStatus 属性はセット値の属性である。以下の値が使用されることができる:

-fail: サーキットパックは失敗している、

-inTest: サーキットパックは試験中である、

-notInstall: 物理的サーキットパックが挿入されない、あるいは、挿入されているが、しかし、そのタイプは circuitPackR1 インスタンスの textType 属性で指定されるタイプに合致していない。(たとえ、物理的サーキットパックが、包含される装置ホルダの許容できるサーキットパックタイプのひとつであったとしても)。

-degraded: サーキットパックのポートの部分集合には、欠陥がある、

-dependency: サーキットパックが依存している資源が利用可能でないため、サーキットパックが無効状態である、そして、

-offLine: サーキットパックが初期値設定中である(即ち、リセット中)。

circuitPackR1 は付加的な circuitPackR1 オブジェクトを包含することができる。

この構造は一回の操作で CircuitPack 属性の全てを、検索するために使用されえる。それらがサポートされないと、多くののサポートされない属性が空の文字列かリストとして返されるであろう。でも、空の文字列値の受信は、属性がサポートされないということを意味しない。

本インタフェースは、次の条件付パッケージ（CONDITIONAL PACKAGES ）を含んでいる。

```
- numberOfPortPackage: インスタンスがそれをサポートするならば存在する。

- portAssociationsPackage: インスタンスがそれをサポートするならば存在する。

- circuitPackConfigurationPackage: インスタンスがそれをサポートするならば存在する。

- containedBoardPackage: インスタンスがそれをサポートするならば存在する。
*/
       valuetype CircuitPackValueType: EquipmentValueType
       {
/**
以下の属性は EquipmentValueType で条件付きであった。
それらは現在、必須である。 そして、それらのそれぞれの操作は、条件付きのパッケージ例外を上げない。

              public CurrentProblemSetType     currentProblemList;
                     // currentProblemListPackage
                     // GET
              public AlarmSeverityAssignmentProfileNameType
                     alarmSeverityAssignmentProfilePointer;
                     // alarmServityAssignmentPointerPackage
                     // GET-REPLACE
              public OperationalStateType      operationalState;
                     // administrativeOperationalStatesPackage
                     // GET
              public AdministrativeStateType   administrativeState;
                     // administrativeOperationalStatesPackage
                     // GET-REPLACE
*/
              public AvailabilityStatusSetType availabilityStatus;
                     // GET
              public short               numOfPorts;
                     // conditional
                     // numberOfPortPackage
                     // GET
              public PortAssociationSetType    portAssociationList;
                     // conditional
                     // portAssociationsPackage
                     // GET
              public SignalRateSetType   availableSignalRateList;
                     // conditional
                     // circuitPackConfigurationPackage
                     // GET
              public PortSignalRateAndMappingSetType
                     portSignalRateAndMappingList;
                     // conditional
                     // circuitPackConfigurationPackage
                     // GET-REPLACE, ADD-REMOVE
              public IstringSetType            acceptableCircuitPackTypeList;
                     // conditional
                     // containedBoardPackage
                     // GET-REPLACE, ADD-REMOVE
       }; // valuetype CircuitPackValueType


       interface CircuitPack: Equipment
       {
/**
属性 availabilityStatus は正しい物理的回線パックが挿入されているか否かを示すために使用されている。この属性は値 notInstalled を含む列挙の系列である。挿入された物理回線パックのタイプが circuitPackType 属性の値に合うならば(circuitPack インスタンスに関連して)、availabilityStatus の値は空集合である。さもなければ、それが許容されるサーキットパックタイプのひとつであっても、availabilityStatus 属性の値は notInstalled である。
*/
              AvailabilityStatusSetType availabilityStatusGet ()
                     raises (itut_x780::ApplicationError);
```

```
/**
次のメソッドはサーキットパックでサポートしているポート数を返す。
*/
              short numOfPortsGet ()
                     raises (itut_x780::ApplicationError,
                            NOnumberOfPortPackage);

/**
次のものは、(それらの文字列名で、特定される)サーキットパックの上の物理的なポートに関連するリストを、ほかの
場所に含まれたそれらのポートを表すオブジェクトとともに返す。それらには物理的な暗示的意味があるので、文字列名
のための値の選択は重要である。
*/
              PortAssociationSetType portAssociationListGet ()
                     raises (itut_x780::ApplicationError,
                            NOportAssociationsPackage);

/**
次の属性は、サーキットパックエンティティでサポートされる信号レートを特定する。
*/
              SignalRateSetType availableSignalRateSetTypeGet ()
                     raises (itut_x780::ApplicationError,
                            NOcircuitPackConfigurationPackage);

/**
以下の属性はサーキットパックポートとそのペイロードマッピングに関連している信号レートを特定する。信号レートと
ペイロードのマッピングはプロビジョン可能である。例えば、信号レート stm4 があるポートには、au4-4 に関するペ
イロードマッピングがあることがある。このレートの別の可能なマッピングは、4 の個々の au4(即ち、au4、au4、au4、
au4)あるいは、系列か混ぜられた au3 と au4(例えば、au3、au3、au3、au4、au4、au4、au3、au3、au3)の系列
である。
*/
              PortSignalRateAndMappingSetType
                     portSignalRateAndMappingListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcircuitPackConfigurationPackage);

/**
portSignalRateAndMappingList 属性の取り替え操作は、端点オブジェクトの削除と生成の原因となることがある。
もしこの場合、objectDeletion と objectCreation 通知は削除されて生成されたオブジェクトから発せられる。し
かしながら、もしそのような削除、そして/又は、生成が既存のユーザ・サービスに影響するならば、置き換え要求は否
定されるべきであり、ServiceAffectedError の例外処理が上げられるべきである。
*/
              void portSignalRateAndMappingListSet
                     (in PortSignalRateAndMappingSetType rateAndMappingList)
                     raises (itut_x780::ApplicationError,
                            ServiceAffectedError,
                            NOcircuitPackConfigurationPackage);

              void portSignalRateAndMappingListAdd
                     (in PortSignalRateAndMappingSetType rateAndMappingList)
                     raises (itut_x780::ApplicationError,
                            ServiceAffectedError,
                            NOcircuitPackConfigurationPackage);

              void portSignalRateAndMappingListRemove
                     (in PortSignalRateAndMappingSetType rateAndMappingList)
                     raises (itut_x780::ApplicationError,
                            ServiceAffectedError,
                            NOcircuitPackConfigurationPackage);

/**
以下の属性はサーキット・パックオブジェクトに含むことができるボードのタイプを示す。
*/
              IstringSetType acceptableCircuitPackTypeListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedBoardPackage);
```

```
            void acceptableCircuitPackTypeListSet
                  (in IstringSetType acceptableCircuitPackTypeList)
                  raises (itut_x780::ApplicationError,
                        NOcontainedBoardPackage);

            void acceptableCircuitPackTypeListAdd
                  (in IstringSetType acceptableCircuitPackTypeList)
                  raises (itut_x780::ApplicationError,
                        NOcontainedBoardPackage);

            void acceptableCircuitPackTypeListRemove
                  (in IstringSetType acceptableCircuitPackTypeList)
                  raises (itut_x780::ApplicationError,
                        NOcontainedBoardPackage);

/**
以下のアクションは、サーキット・パックを初期化する要求のために使用されている。要求は、完全なリセットか部分的
なリセットであることがある。完全なリセット要求は action argument における負の数によって示される。
部分的なリセット要求は非負の整数によって示される。値ゼロは最少のレベルのリセットを意味する。
整数値が高ければ高いほど、より徹底的なリセットを意味する。完全なリセットに同等な最も高い整数値の決定はローカ
ルマターである。サーキット・パックがリセット途中であるとき、availablilityStatus 属性の offLine 値は、示
されるものとする。サーキット・パックがユーザ・サービスに過敏であるならば、サーキット・パックがロックされた
administrativeState にあるときだけ、リセットは実行されるものとする。サーキット・パックがロックされた
administrativeState でないならば、リセット要求は否定されるものとし、そして、CircuitPackResetError 例
外処理が上げられるものとする。

パッケージが実行されないという以外のいかなる理由であれ、リセットアクションが失敗するとき、ResetErrorType
値は CircuitPackResetError 例外処理のパラメタ部分に含まれている。
*/
            boolean reset
                  (in short resetLevel)
                  raises (itut_x780::ApplicationError,
                        CircuitPackResetError,
                        NOcircuitPackResetPackage);

            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, objectCreation)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, staeChange)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, equipmentAlarm)

      }; // interface CircuitPack


      interface CircuitPackFactory: itut_x780::ManagedObjectFactory
      {
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID,    // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList,
                  in ReplaceableType replaceable,
                        // equipmentPackage
                        // GET, SET-BY-CREATE
                  in Istring type,
                        // equipmentR2Package
                        // GET, SET-BY-CREATE
                  in MONameSetType supportedByObjectList,
                        // may be nil
                        // equipmentR1Package
                        // GET-REPLACE, ADD-REMOVE
```

```
                    in AdministrativeStateType adminstrativeState,
                           // conditional
                           // administrativeOperationalStatesPackage
                           // GET-REPLACE
                    in AlarmSeverityAssignmentProfileNameType profile,
                           // conditional
                           // alarmSeverityAssignmentPointerPackage
                           // GET-REPLACE
                    in Istring userLabel,
                           // conditional
                           // userLabelPackage
                           // GET-REPLACE
                    in Istring vendorName,
                           // conditional
                           // vendorNamePackage
                           // GET-REPLACE
                    in Istring version,
                           // conditional
                           // versionPackage
                           // GET-REPLACE
                    in Istring locationName,
                           // conditional
                           // locationNamePackage
                           // GET-REPLACE
                    in ArcProbableCauseSetType arcProbableCauseList,
                           // conditional
                           // arcPackage
                           // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                           // conditional
                           // arcPackage
                           // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval,
                           // conditional
                           // arcPackage
                           // GET-REPLACE
                    in PortSignalRateAndMappingSetType
                           rateAndMappingList,
                           // conditional
                           // circuitPackConfigurationPackage
                           // GET-REPLACE, ADD-REMOVE
                    in IstringSetType
                           acceptableCircuitPackTypeList)
                           // conditional
                           // containedBoardPackage
                           // GET-REPLACE, ADD-REMOVE
                    raises (itut_x780::ApplicationError,
                           itut_x780::CreateError);

      }; // interface CircuitPackFactory


/**
```

6.5.8.2 *EquipmentHolder*

`EquipmentHolder` オブジェクトは他の物理資源を保持することができるネットワークエレメントの物理資源を表す。このオブジェクトのクラスのインスタンスによって表される資源の例は、装置の架と、シェルフとスロットである。

この構造は一回の操作で `EquipmentHolder` 属性の全てを、検索する際に使用される。もし、それらがサポートされないと、多くのサポートされない属性が空の文字列かリストとして返され得る。それでも、空の文字列値の受信は、属性がサポートされなということを意味しない。

本インタフェースは、次の条件付パッケージを含んでいる。

- `subordinateCircuitPackPackage:` もしインスタンスがサーキットパックを含むことを許されるならば存在する。

```
*/
       valuetype EquipmentHolderValueType: EquipmentValueType
       {
              public IstringSetType            equipmentHolderAddress;
                     // GET, SET-BY-CREATE
              public IstringSetType            acceptableCircuitPackTypeList;
                     // conditional
                     // subordinateCircuitPackPackage
                     // GET-REPLACE, ADD-REMOVE
              public HolderStatusType    holderStatus;
                     // conditional
                     // subordinateCircuitPackPackage
                     // GET
              public SoftwareNameSetType subCircuitPackSoftwareLoad;
                     // conditional
                     // subordinateCircuitPackPackage
                     // GET-REPLACE
       }; // valuetype EquipmentHolderValueType


       interface EquipmentHolder: Equipment
       {
/**
以下のメソッドは equipmentHolder インスタンスによって表される資源の物理的な位置を返す。
管理システムの equipmentHolder の包含階層構造によって, この属性の値は異なることがある。
例えば, システムが架, シェルフ, およびスロットで表される 3 つのレベルの装置ホルダを持つならば(即ち, 管理エレメントは複数架の装置ホルダを含んでいて,そして各架の装置ホルダは複数のシェルフの装置ホルダを含んでいて,そして各シェルフの装置ホルダは複数のスロット装置ホルダを含んでいる)、その場合、

-架を示す equipmentHolder のために、フレーム識別（Frame Identification）コードがこの属性の値として使用されることがある。

-シェルフを表す equipmentHolder のために、架のシェルフコードはこの属性の値として使用されることがある;

-スロットを表す equipmentHolder のために、位置のコードはこの属性の値として使用されることがある。

システムが一つのレベルの装置ホルダだけを使用するならば, それはシェルフ（Shelves）を表していて(即ち、managedElement が多重シェルフ装置ホルダを含んでいる、そして、それぞれの各シェルフ装置ホルダはサーキットパックを含んでいる), それから この属性の値はフレーム識別（Frame Identification）コードと架とシェルフのコードの順序を示す。
*/
              IstringSetType equipmentHolderAddressGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドはこのホルダにインストールされていることがある許容可能なサーキットパックタイプのリストを返す。
値は, このセット値の属性に加えられるか, 取り替えられるか, または移されることがある。
*/
              IstringSetType acceptableCircuitPackTypeListGet ()
                     raises (itut_x780::ApplicationError,
                            NOsubordinateCircuitPackPackage);

/**
以下のメソッドは、このホルダにインストールされることがある許容可能なサーキットパックタイプのリストを置き換える。equipmentHolder が現在 circuitPack を含むならば, 対応する (circuitPack の)タイプの値は, この属性から取り替えられないものとし, あるいは取り除かれないものとする。含まれる circuitPack のタイプはこの属性に指定されるタイプのひとつとなるだろう。
*/
              void acceptableCircuitPackTypeListSet
                     (in IstringSetType list)
                     raises (itut_x780::ApplicationError,
                            NOsubordinateCircuitPackPackage);

/**
```

以下のメソッドは，このホルダにインストールされることがある、許容可能なサーキットパックタイプのリストに値を追加する。
*/
```
		void acceptableCircuitPackTypeListAdd
			(in IstringSetType list)
			raises (itut_x780::ApplicationError,
				NOsubordinateCircuitPackPackage);

/**
```
以下のメソッドはこのホルダにインストールされることがある許容可能なサーキットパックタイプのリストから値を取り除く。
*/
```
		void acceptableCircuitPackTypeListRemove
			(in IstringSetType list)
			raises (itut_x780::ApplicationError,
				NOsubordinateCircuitPackPackage);

/**
```
以下のメソッドは装置ホルダの状態を返す。 ホルダの状態は以下の一つであることがある。

-どんな取替え可能なユニットもホルダに存在しないことを示す、空（empty）の状態。

-ホルダは acceptableCircuitPackType リストにタイプのひとつであるユニットを含んでいる。

-ホルダはネットワーク・エレメントで認識可能なユニットを含んでいる; しかし、acceptableCircuitPackTypeList のタイプのひとつでなはない。

-認識されていない取替え可能なユニット。

ホルダが許容できるユニットを含んでおり、そのタイプが circuitPack オブジェクトの circuitPackType 属性の値に合うと、そのとき、circuitPack の availabilityStatus は空集合になるだろう。 他のすべての場合では、availabilityStatus は notInstalled 値を含むだろう。
*/
```
		HolderStatusType holderStatusGet ()
			raises (itut_x780::ApplicationError,
				NOsubordinateCircuitPackPackage);

/**
```
以下のメソッドは、ソフトウェアの自動再ロードが必要であるときはいつでも、含まれるサーキット・パックにロードされるように現在指定されたソフトウェアがあればそれを返す。リストが空ならば、含まれるサーキットパックは、ソフトウェアロード可能ではないか、どんなソフトウェアロードも指定されてないかである。さもなければ，その系列はソフトウェアがロードされる順序を特定するために利用される、ソフトウェアインスタンスのセットの順番を特定する。順序が重要であるならば、ICS に記載することを勧める。M.3100 はまた、文字列の順序や、「ローカル」の実装合意で決められた値を許容する。そのオプションはここではサポートされない。
*/
```
		SoftwareNameSetType subordinateCircuitPackSoftwareLoadGet ()
			raises (itut_x780::ApplicationError,
				NOsubordinateCircuitPackPackage);

/**
```
以下のメソッドは、ソフトウェアの自動再ロードが必要なときはいつでも、含まれるサーキットパックにロードされるソフトウェアを設定する。
*/
```
		void subordinateCircuitPackSoftwareLoadSet
			(in SoftwareNameSetType list)
			raises (itut_x780::ApplicationError,
				NOsubordinateCircuitPackPackage);

	}; // interface EquipmentHolder


	interface EquipmentHolderFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
```

```
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in ReplaceableType replaceable,
                       // equipmentPackage
                       // GET, SET-BY-CREATE
                in Istring type,
                       // equipmentR2Package
                       // GET, SET-BY-CREATE
                in MONameSetType supportedByObjectList,
                       // may be nil
                       // equipmentR1Package
                       // GET-REPLACE, ADD-REMOVE
                in AdministrativeStateType adminstrativeState,
                       // conditional
                       // administrativeOperationalStatesPackage
                       // GET-REPLACE
                in AlarmSeverityAssignmentProfileNameType profile,
                       // conditional
                       // alarmSeverityAssignmentPointerPackage
                       // GET-REPLACE
                in Istring userLabel,
                       // conditional
                       // userLabelPackage
                       // GET-REPLACE
                in Istring vendorName,
                       // conditional
                       // vendorNamePackage
                       // GET-REPLACE
                in Istring version,
                       // conditional
                       // versionPackage
                       // GET-REPLACE
                in Istring locationName,
                       // conditional
                       // locationNamePackage
                       // GET-REPLACE
                in ArcProbableCauseSetType arcProbableCauseList,
                       // conditional
                       // arcPackage
                       // GET-REPLACE, ADD-REMOVE
                in ArcIntervalProfileNameType arcIntervalProfilePointer,
                       // conditional
                       // arcPackage
                       // GET-REPLACE
                in ArcTimeType arcManagementRequestedInterval,
                       // conditional
                       // arcPackage
                       // GET-REPLACE
                in IstringSetType equipmentHolderAddress,
                       // equipmentHolderPackage
                       // GET, SET-BY-CREATE
                in IstringSetType acceptableCircuitPackTypeList,
                       // conditional
                       // subordinateCircuitPackPackage
                       // GET-REPLACE, ADD-REMOVE
                in SoftwareNameSetType subCircuitPackSoftwareLoad)
                       // conditional
                       // subordinateCircuitPackPackage
                       // GET-REPLACE
                raises (itut_x780::ApplicationError,
                       itut_x780::CreateError);

};  //interface EquipmentHolderFactory
```

```
/**
```

6.5.9 ExternalPoint

ExternalPoint インタフェースは、controlPoint と scanPoint オブジェクトのクラスのスーパークラスであり、外部の装置を制御するか、または外部の状態をモニタするのにそれぞれ使用される。このオブジェクトクラスは controlPoint と scanPoint オブジェクトクラスの共通面を含む。操作上の状態と運用状態は、制御とスキャン機能の状態を表している。即ち、外部エンティティの状態ではない。

本インタフェースは次の条件付パッケージを含む。

```
- locationNamePackage: インスタンスがそれをサポートすると存在する。
*/
       valuetype ExternalPointValueType: itut_x780::ManagedObjectValueType
       {
              public OperationalStateType      operationalState;
                     // GET
              public AdministrativeStateType   administrativeState;
                     // GET-REPLACE
              public MONameSetType       supportedByObjectList;
                     // GET
              public long                exteralPointId;
                     // GET, SET-BY-CREATE
              public Istring                   externalPointMessage;
                     // GET-REPLACE
              public Istring                   locationName;
                     // conditional
                     // locationNamePackage
                     // GET-REPLACE
       }; // valuetype ExternalPointValueType


       interface ExternalPoint: itut_x780::ManagedObject
       {

              OperationalStateType operationalStateGet ()
                     raises (itut_x780::ApplicationError);

              AdministrativeStateType administrativeStateGet ()
                     raises (itut_x780::ApplicationError);

              void administrativeStateSet
                     (in AdministrativeStateType adminstrativeState)
                     raises (itut_x780::ApplicationError);

              MONameSetType supportedByObjectListGet ()
                     raises (itut_x780::ApplicationError);

/**
以下の属性は、モニタされたかもしくは制御された外部の装置が付属しているポート番号を特定する。また、それは管理オブジェクトのための命名属性を提供する。したがって、NameComponent の id 文字列は、ポート番号の stringified 整数になるはずである。
*/
              long externalPointIdGet ()
                     raises (itut_x780::ApplicationError);

/**
以下の属性は、外部ポイントの何らかの原文の定義を提供することができる。また、外部ポイントの位置を特定するのにそれを使用することができる。
*/
              Istring externalPointMessageGet ()
                     raises (itut_x780::ApplicationError);

              void externalPointMessageSet
```

```
                (in Istring message)
                raises (itut_x780::ApplicationError);

        Istring locationNameGet ()
                raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

        void locationNameSet
                (in Istring locationName)
                raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, stateChange)

};  // interface ExternalPoint


/**
```

6.5.9.1 ControlPoint

ControlPoint インタフェースをサポートする管理オブジェクトは、管理システムに関連づけられた外部の装置を制御する。それは、ベル、ランプ、ジェネレータ、ヒータ、またはエアコンのためのリレー閉回路である。このクラスの各インスタンスは一つの制御点を表す。

制御点の現在の状態は、閉じているか、(即ち、アクティベート)、または、開いているか (即ち、リリースド)のどちらかである。制御点は、任意的に正常状態であることがある(即ち、閉じられる、オープン、一方または他方)。

制御点によって表される外部装置は、”コントロール”アクションによって、遠隔操作することができる。
制御機能は、一時的なもの(即ち、一時的に閉じているか開いている)、または連続しているもの(連続的に閉じているか開いている)である。

制御点の有効なコントロールタイプは一時的なもののみか、連続的なもののみか、あるいはその両方である。制御コントロール動作タイプ(連続的か一時的)が制御点において有効でないと、コントロール動作は否定されるであろう。

コントロール動作の制御点への効果は Table 1/M.3100 Amendment 1 に与えられている。

現在の状態、有効な制御タイプ、正常な状態(任意の)、テキストメッセージ(使い勝手のよいラベルやテキスト)、および制御点の位置(任意の)が別々の属性によっている。

本インタフェースは次の条件付パッケージ（CONDITIONAL PACKAGES）を含んでいる。

```
- normalControlStatePackage: インスタンスがそれをサポートすると存在する。
*/
        valuetype ControlPointValueType: ExternalPointValueType
        {
                public ControlStateType    currentControlState;
                // GET
                public ValidControlType    validControl;
                // GET-REPLACE, SET-BY-CREATE
                public ControlStateType    normalControlState;
                // conditional
                // normalControlStatePackage
                // GET-REPLACE
        }; // valuetype ControlPointValueType


        interface ControlPoint: ExternalPoint
```

```
	{
/**
つぎの属性は制御点の現在の状態を示す。
*/
		ControlStateType currentControlStateGet ()
			raises (itut_x780::ApplicationError);

/**
つぎの属性は、この制御ポイントのための制御信号の有効なタイプを示す。
*/
		ValidControlType validControlGet ()
			raises (itut_x780::ApplicationError);

		void validControlSet
			(in ValidControlType validControl)
			raises (itut_x780::ApplicationError);

/**
つぎの属性は制御点の正常状態を示す。
*/
		ControlStateType normalControlStateGet ()
			raises (NOnormalControlStatePackage);

		void normalControlStateSet
			(in ControlStateType controlState)
			raises (itut_x780::ApplicationError,
				NOnormalControlStatePackage);

/**
以下の動作は、NE に対して、制御点で表される、（リレー閉回路のような）外部の制御装置を、一時的な操作(クローズまたはオープン)をするか、または連続的な操作(クローズまたはオープン)を指示するものである。コントロール動作タイプパラメタは要求に含まれている。
*/
		ControlResultType externalControl
			(in ControlActionType controlAction)
			raises (itut_x780::ApplicationError);

	}; // interface ControlPoint


	interface ControlPointFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID,
			out MONameType name,
			in StringSetType packageNameList,
			in AdministrativeStateType adminstrativeState,
				// externalPointPackage
				// GET-REPLACE
			in long exteralPointId,
				// externalPointPackage
				// GET, SET-BY-CREATE
			in Istring externalPointMessage,
				// externalPointPackage
				// GET-REPLACE
			in Istring locationName,
				// conditional
				// locationNamePackage
				// GET-REPLACE
			in ValidControlType validControl,
				// controlPointPackage,
				// GET-REPLACE, SET-BY-CREATE
			in ControlStateType normalControlState)
				// conditional
				// normalControlStatePackage,
```

```
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface ControlPointFactory


/**
```

6.5.9.2 ScanPoint

管理エレメントに関する外部状態をモニタするための ScanPoint インタフェースをサポートする管理オブジェクトは、外部の装置のイベント(停電、火災報知機、ドアオープン、湿度などの)のためにモニタされる。このオブジェクトクラスの各インスタンスは一つのスキャンポイントを表す。スキャンポイントが異常な状態を検出すると、環境警報が発出されるだろう。警報がスキャンポイントに発出されるとき、externalPointMessage 属性で指定されるテキストメッセージは、environmentalAlarm 通知の additionalText フィールドに含まれることになっている。任意のパッケージを通して、そのような警報の重要度を設定することができる。

currentProblemList は、モニタされる外部エンティティの現在の問題を表す。即ち、スキャン機能自体の現在の問題でない。currentProblemList の想定原因は、それ自体、サービスに影響する警報(例えば、予備資源による)の正確な指標ではなく、そして、serviceAffected 属性は、サービスが影響する状態の統一指標として使用される。

本インタフェースは次の条件付パッケージ（CONDITIONAL PACKAGES）を含んでいる。

```
- alarmSeverityAssignmentPointerPackage: インスタンスが、警報重要度の構成を含んでいるならば存在する。
*/
        valuetype ScanPointValueType: ExternalPointValueType
        {
                public CurrentProblemSetType     currentProblemList;
                        // GET
                public boolean                   serviceAffected;
                        // GET
                public AlarmSeverityAssignmentProfileNameType
                        alarmSeverityAssignmentProfilePointer;
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
        }; // valuetype ScanPointValueType


        interface ScanPoint: ExternalPoint
        {
                CurrentProblemSetType currentProblemListGet ()
                        raises (itut_x780::ApplicationError);

/**
以下の属性は、モニタされる外部装置の警報状態が、サービスに影響するか否かを示している。
*/
                boolean serviceAffectedGet ()
                        raises (itut_x780::ApplicationError);

                AlarmSeverityAssignmentProfileNameType
                        alarmSeverityAssignmentProfilePointerGet ()
                        raises (itut_x780::ApplicationError,
                                NOalarmSeverityAssignmentPointerPackage);

                void alarmSeverityAssignmentProfilePointerSet
                        (in AlarmSeverityAssignmentProfileNameType profile)
                        raises (itut_x780::ApplicationError,
                                NOalarmSeverityAssignmentPointerPackage);

                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, environmentalAlarm)

        }; // interface ScanPoint
```

```
interface ScanPointFactory: itut_x780::ManagedObjectFactory
{
       itut_x780::ManagedObject create
              (in NameBindingType nameBinding,
              in MONameType superior,
              in string reqID,
              out MONameType name,
              in StringSetType packageNameList,
              in AdministrativeStateType adminstrativeState,
                     // externalPointPackage
                     // GET-REPLACE
              in long exteralPointId,
                     // externalPointPackage
                     // GET, SET-BY-CREATE
              in Istring externalPointMessage,
                     // externalPointPackage
                     // GET-REPLACE
              in Istring locationName,
                     // conditional
                     // locationNamePackage
                     // GET-REPLACE
              in AlarmSeverityAssignmentProfileNameType profile)
                     // conditional,
                     // alarmSeverityAssignmentPointerPackage,
                     // GET-REPLACE
              raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

};// interface ScanPointFactory


/**
```

6.5.10 Fabric

このインタフェースは、ファブリックと M.3100 の FabricR1(ファブリックのサブクラスである)の機能性の両方をサポートする。
これは, 切替動作をサポートすることを意味しており,オプションとして、基本ファブリック（Fabric）機能性と共にオブジェクトの生成と削除通知を発出する。

ファブリックオブジェクトは、クロスコネクションの設定と解放を管理する機能を表す。 それはまた TP プールと GTP への端点の割付を管理する。

運用状態（Administrative State）:

- Unlocked:ファブリックはその通常機能を実行することができる。クロスコネクトを設定あるいは解除するため、また TP Pools を再調整するため、ＧＴＰに／から端点を追加/削除するためにアクションが許可される。

- Locked: ファブリックはその通常機能を実行することができない。どんなアクションも受け入れられない。どんな新しいクロスコネクトも設定または解除することができず、どんな TP プールも再調整することができない、そして、どんな端点も、ＧＴＰに／から、追加／削除することができない。

操作状態（Operational State）:

- Enabled:ファブリックが操作状態イネーブルド(enabled)であるとき、それは完全に動作中であり、または部分的に動作中であることがある。(部分的な動作は、可用性状態属性によって示される)

- Disabled:ファブリックはその通常機能を実行するのができない。例えば、管理システムは、(1) どんなクロスコネクトも設定、または解除できない、(2) TP Pools を再調整することができない、(3)ＧＴＰに／から、端点を追加／削除することができないであろう。

可用性状態（Availability Status）:

この属性がサポートしている値は以下のとおりである。

- Degraded:ファブリックは何らかの点において劣化している。例えば、ファブリックは、TP Pools を再調整するためのアクションをまだ受付けることができる一方、新しいクロスコネクトを確立する機能を実行することができない。ファブリックは、劣化しているが、サービスが(即ち、その操作状態はイネーブルドである)利用可能なままである。

- 空集合。

```
*/
       valuetype FabricValueType: itut_x780::ManagedObjectValueType
       {
              public OperationalStateType      operationalState;
                     // GET
              public AdministrativeStateType   administrativeState;
                     // GET-REPLACE
              public AvailabilityStatusSetType availabilityStatus;
                     // GET
              public CharacteristicInfoSetType characteristicInfoList;
                     // GET, SET-BY-CREATE
              public MONameSetType       supportedByObjectList;
                     // GET-REPLACE, ADD-REMOVE
       }; // valuetype FabricValueType


       interface Fabric: itut_x780::ManagedObject
       {
              OperationalStateType operationalStateGet ()
                     raises (itut_x780::ApplicationError);

              AdministrativeStateType administrativeStateGet ()
                     raises (itut_x780::ApplicationError);

              void administrativeStateSet
                     (in AdministrativeStateType adminstrativeState)
                     raises (itut_x780::ApplicationError);

              AvailabilityStatusSetType availabilityStatusGet ()
                     raises (itut_x780::ApplicationError);

/**
Set-By-Create:インタフェースのこの属性値は生成動作のために入力パラメータに特定される。
*/
              CharacteristicInfoSetType characteristicInfoListGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは所定の管理オブジェクトによってサポートされるオブジェクトのインスタンスのリストを返す。オブジェクトのインスタンスは物理的なものと同様に論理的なオブジェクトを含んでいる。この属性は内部の詳細を特定することを強制するものではなく、管理に必要なレベルの詳細だけを特定するものである。管理オブジェクトをサポートするオブジェクトのインスタンスがそのオブジェクトに対して未知であるならば、このメソッドは空のリストを返す。
*/
              MONameSetType supportedByObjectListGet ()
                     raises (itut_x780::ApplicationError);

/**
次のメソッドは、このオブジェクトをサポートしているオブジェクトインスタンスのリストを置き換えるために利用される。
*/
              void supportedByObjectListSet
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError);

/**
次のメソッドは、このオブジェクトをサポートしているオブジェクトインスタンスのリストに追加するために利用される。
*/
```

```
            void supportedByObjectListAdd
                  (in MONameSetType objectList)
                  raises (itut_x780::ApplicationError);

/**
次のメソッドは、このオブジェクトをサポートしているオブジェクトインスタンスのリストからオブジェクトを削除するために利用される。
*/
            void supportedByObjectListRemove
                  (in MONameSetType objectList)
                  raises (itut_x780::ApplicationError);

/**

以下のメソッドは、GTP に端点を調整するために使用される。グループ端点のインスタンスが存在していないならば、新しいもが自動的に作成され、そして、その識別子が結果として返される。さもなければ、端点が既に GTP にあるそれらに加えられる。端点はゼロかあるいは、一つの GTP のメンバであることがある。GTP がクロスコネクトにかかわり、GTP が TP プールのメンバであるか、端点が既に GTP のメンバであるならば、この動作は失敗するであろう。独立している片方向の接続性を提供することができる双方向の端点は、接続性の各方向について、ゼロか一つの GTP のメンバであることがある。

AddTPsToGTPInformation と AddTPsToGTPResult の構文は、両方とも"sequence of"のタイプであることに注意せよ。AddTPsToGTPResult における第 n 要素は AddTPsToGTPInformation の第 n 要素に関連する。

このパラメータの TPsAddToGTPInfoType 構文の"agtp"フィールドの値は、空であることがある。
*/
            void addTPsToGTP
                  (in TPsAddToGTPInfoSeqType request,
                  out TPsAddToGTPResultSeqType result)
                  raises (itut_x780::ApplicationError);

/**
removeTPsFromGTP 動作は GTP から端点を取り除くために使用されている。この動作は GTP がクロスコネクトにかかわるか、それが TP プールのメンバであるならば失敗する。GTP から最後の端点を取り除くことは、GTP オブジェクトを削除するという効果がある。GTP が削除されるならば、GTP という名前は動作（ACTION）応答で返送される。

RemoveTPsFromGTPInformation と RemoveTPsFromGTPResult の構文はともに、"sequence of"のタイプですることに注意せよ。RemoveTPsFromGTPResult における第 n 要素は RemoveTPsFromGTPInformation の第 n 要素に関連する。
*/
            void removeTPsFromGTP
                  (in TPsRemoveInfoSeqType request,
                  out TPsRemoveResultSeqType result)
                  raises (itut_x780::ApplicationError);

/*
以下のメソッドは、たとえばルーティングなどのように、何らかの管理目的のために、すべて同等である端点か、GTP のプールに端点か GTP を調整するために使用される。tpPool のインスタンスが存在していないならば、新しいものが自動的に作成され、そして、その識別子が結果として返される。さもなければ、端点か GTP は、既に tpPool にあるそれらに加えられる。間接アダプタ（Indirect Adaptor）が指定されるならば、間接アダプタ（Indirect Adaptor）から含まれる CTP を表す GTP が生成され、それは tpPool に加えられるであろう。

AddTPsToTPPoolInformation と AddTPsToTPPoolResult の構文は、両方ともに"sequence of"のタイプであることに注意せよ。AddTPsToTPPoolResult における第 n 要素は AddTPsToTPPoolInformation の第 n 要素に関連する。

本パラメータの TPsAddToTPPoolInfoType 構文の"atppool"フィールドの値は、空であることがある。
*/
            void addTPsToTPPool
                  (in TPsAddToTPPoolInfoSeqType request,
                  out TPsAddToTPPoolResultSeqType result)
                  raises (itut_x780::ApplicationError);
/**
```

removeTPsFromTPPool 動作は、端点プールから端点を取り除くために使用される。プールから最後の端点を取り除くことは、TP Pool オブジェクトを削除する効果がある。TP プールが削除されるならば、TP Pool という名前が ACTION 応答で返される。

RemoveTPsFromTPPoolInformation と RemoveTPsFromTPPoolResult の構文は両方とも"sequence of"のタイプであることに注意せよ。RemoveTPsFromTPPoolResult における第 n 要素は、RemoveTPsFromTPPoolInformation の第 n 要素に関連する。
*/

```
            void removeTPsFromTPPool
                  (in TPsRemoveInfoSeqType request,
                  out TPsRemoveResultSeqType result)
                  raises (itut_x780::ApplicationError);
```

/**
"connect"メソッドは、M.3100 の"connect"動作に基づく。 それは、端点か GTP 間の接続を設定するために使用される。 接続されるべき端点は、次の 2 つの方法のうちの一つで指定されることができる:

1) 明示的に 2 つの端点か GTP を特定することによって;

2) 一つの端点か GTP を指定し、任意のアイドル中の端点/GTP が使用されることがある tpPool を指定することによって。 もし成功しているならば、結果として、いつも端点か GTP の明示的なリストを返す。

クロスコネクト調整にはポイントツーポイントとポイントツー多重ポイント(放送型) の 2 つの基本的な型がある。explicitPtoP か ptoTpPool オプションのどちらかがこの動作で選択されるならば、単一のクロスコネクションが生成される。このクロスコネクトオブジェクトは、クロスコネクトにかかわる端点か GTP を示す。接続は connectivityPointer 属性によって端点に示される。もし、crossConnection オブジェクトの administrativeState がアンロックドであるならば、この属性は、この動作の結果として、それが接続されている端点のローカル名に設定される。また、端点、GTP の crossConnectionObjectPointer は、クロスコネクトオブジェクトを示す。

ポイントツー多重ポイント・クロスコネクト(explicitPtoMp か ptoMPools オプションを選ぶことによって示される)については、一つの多重ポイント・クロスコネクトオブジェクトが生成され、toTps パラメタで指定されるそれぞれの端点単位に一個の crossConnection オブジェクトを含む。ソース TP では crossConnectionObjectPointer が、新たに生成された多重ポイント・クロスコネクトオブジェクトを指し示す。toTPs リスト(おそらく、指定された tpPool から選択された)の中に指定された各 Tp では、CrossConnectionObject ポインタが対応するクロスコネクトオブジェクトを示す。接続された端点の接続性ポインタは、新しい接続性を反映するためにアップデートされるであろう。tpPool オブジェクト(もしあるならば)の idleTPcount と connectedTPcount 属性は、動作の結果、更新される。もし GTP が一緒に接続されるために数個の端点を指定することによって暗示的に定義されるならば、GTP オブジェクトは自動的に生成され、そして、その ID がアクション応答として返される。

もし間接アダプタ（Indirect Adaptor）が指定されるならば、間接アダプタ（Indirect Adaptor）から含まれる CTP を表す GTP が生成され、そして、それが接続されるであろう。

生成されたクロスコネクトか多重ポイントクロスコネクトオブジェクトの運用状態が、この動作の任意のパラメタとして指定される。このパラメタが省略されるならば、運用状態は ' アンロックド ' に設定される(addLegs パラメタが指定されない場合)。もし指定された端点のどれかが既にクロスコネクトにかかわっているか、または既存の GTP の一部が指定されるならば、この動作は失敗する。

もし addLeg パラメタが指定されるならば、一つかそれ以上の区間(Leg)が既存の多重ポイント・クロスコネクトアレンジメントに加えられる。選択された端点か GTP は、既にアレンジメントに接続された端点のものと同様の信号タイプをサポートしなければならない。その結果は、成功であるならば、常にマルチポイントクロスコネクトにかかわった端点か GTP を返す。クロスコネクトオブジェクトはこの動作の結果、生成される。このオブジェクトは指定された mpCrossConnection オブジェクトのインスタンスから命名される。生成されたクロスコネクトオブジェクトの運用状態は、特に動作パラメタに指定されない限り、含まれている多重ポイント・クロスコネクトオブジェクトのものと同じである。

ConnectInformtion と ConnectResult の構文は両方とも"sequence of"のタイプであることに注意せよ。ConnectResult における第 n 要素は ConnectInformation の第 n 要素に関連する。

namedCrossConnection パラメタが存在しているとき、namedCrossConnection(または、namedCrossConnection のサブクラス)インスタンスが生成される。userLabel、そして/又は、運用限界が存在して

いるとき、crossConnectionR1(または、crossConnectionR1 のサブクラス)インスタンスが生成される。これらのパラメタのどれも存在していないとき、どのインスタンスが生成されるかを推論することは、ローカルシステム次第である。namedCrossConnection フィールドは、同じ動作要求で、userLabel フィールドか運用限界フィールドのどちらかと共に使用されないものとする。
*/
```
            void connect
                  (in ConnectInfoSeqType request,
                  out ConnectResultSeqType result)
                  raises (itut_x780::ApplicationError);
```

/**
"disconnect"というメソッドは、M.3100 の"disconnect"動作に基づく。それは、クロスコネクトを解除するのに使用される。解除すべきである接続は、接続の端点あるいは GTP(s)を特定することによって指定される。もし接続がポイントツーポイントであったならば、もう片方の端点か GTP もまた暗示的に開放され、そして、クロスコネクトオブジェクトは削除される。もし接続がポイントツー多重ポイントであり、動作がマスタであるならば、すべての端点か区間である GTPs もまた暗示的に、開放され、そして、多重ポイントクロスコネクトとクロスコネクトオブジェクトは削除される。

もし接続がポイントツーマルチポイントであり、動作が区間であるならば、それが最後の区間でなければ、まさしくその区間は開放され、マスタ端点がまた暗示的に開放されて、多重ポイント・クロスコネクトとクロスコネクトオブジェクトが削除される。tpPool オブジェクト(もしあるならば)の idleTPcount と connectedTPcount 属性は動作の結果、更新される。開放された端点の接続性ポインタは、この動作の結果、空に設定される。

この動作は GTP の構成にどんな影響も与えないし、GTP はこの動作の結果、削除されない。GTP の一部が指定されるならば、この動作は失敗する。

DisconnectInformtion と DisconnectResult の構文は、両方とも"sequence of"のタイプであることに注意せよ。DisconnectResult における第 n 要素は、DisconnectInformation の第 n 要素に関連する。
*/
```
            void disconnect
                  (in MONameSeqType tps,
                  out DisconnectResultSeqType result)
                  raises (itut_x780::ApplicationError);
```

/**
"switchOver"メソッドは、M.3100 の"switchOver"動作に基づく。それは以下の能力を、アトミックな方法で、提供する。

(1)元のクロスコネクトされた端点の一つを維持しながら、既存の接続を同タイプの別の一つに切り換える。もし操作が成功ならば、結果的に、動作情報で示される古い接続の削除と新しい端点の接続をもたらす。動作情報で示される新しい端点は、新しいクロスコネクトを確立するために、利用可能でなければならない(それぞれの方向が、この動作の前に開放される)。個々の接続の切り換えは、アトミック操作であるとみなされる。

(2) 既存の接続の束を切り換える。これらの個々の接続は、上記のように切り換えられる。この場合、動作は、ベストエフォート・ポリシにより振る舞い、首尾よく切り換えすることができた接続だけが、互いに独立に、実行される。

いずれにせよ、接続とは、片方向か双方向のポイントツーポイント接続（即ち、crossConnection）や、マルチポイント接続の区間(即ち、mpCrossConnection に含まれる crossConnection)、または、勧告 G.774.04 で定義されるマルチポイント接続保護の区間を指定する。

SwitchOverInformation と SwitchOverResult の構文の両方とも"sequence of"のタイプであることに注意せよ。SwitchOverResult における第 n 要素は SwitchOverInformation の第 n 要素に関連する。
*/
```
            void switchover
                  (in ConnectSwitchOverInfoSeqType request,
                  out ConnectResultSeqType result)
                  raises (itut_x780::ApplicationError);
```

/**
同じ条件付きパッケージ中に objectCreation と objectDeletion 通知がある。 両方の通知がファブリックインタフェースのインスタンスによってサポートされているならば、このパッケージは存在する。
*/
```
            CONDITIONAL_NOTIFICATION(
```

```
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)

};  // interface Fabric


interface FabricFactory: itut_x780::ManagedObjectFactory
{
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in MONameSetType supportedByObjectList,
                        // may be empty set type
                        // fabricPackage
                        // GET-REPLACE, ADD-REMOVE
                in CharacteristicInfoSetType characteristicInfoList,
                        // fabricPackage
                        // GET, SET-BY-CREATE
                in AdministrativeStateType administrativeState)
                        // fabricPackage
                        // GET-REPLACE
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

};  // interface FabricFactory

/**
```

6.5.11　GTP (グループ端点オブジェクト)

GTP(グループ端点オブジェクト)はクロスコネクトの様な管理目的のシングルユニットとして扱われる端点のグループを意味する。SignalType 属性は GTP の構成を記述する。端点が GTP に含まれるとき、それはその GTP と無関係にクロスコネクトされる事はない。
*/

```
	valuetype GTPValueType: itut_x780::ManagedObjectValueType
	{
		public CrossConnectionPointerType crossConnectionPointer;
			// GET
		public SignalIdType		signalId;
			// GET
		public MONameSeqType			tpsInGTPList;
			// GET
	}; // valuetype GTPValueType


	interface GTP: itut_x780::ManagedObject
	{

/**
次に述べる属性はクロスコネクト、GTP もしくはファブリック(Fabric)の様な管理オブジェクトを指している。端点が接続されているのでも接続の為にリザーブされているのでもない時、その crossConnectionObjectPointer はその接続に責任のあるファブリックオブジェクトを指している。

次に述べるメソッドは GTP が終点であるクロスコネクトへの参照を返す。もし GTP が接続に含まれない場合には、空が返る。
*/

		CrossConnectionPointerType crossConnectionPointerTypeGet ()
			raises (itut_x780::ApplicationError);

/**
次のメソッドはその接続に責任を持つファブリックオブジェクトへの参照を返す。これは M.3100 を基にしている。M.3100 では GTP が接続中でない時、"crossConnectionObjectPointerGet"はファブリックを返すがそれは混乱をもたらし、構文的に複雑となる。
*/

		FabricNameType fabricPointerGet ()
			raises (itut_x780::ApplicationError);

/**
次の操作は GTP の信号タイプを返す。信号タイプはシンプル、バンドル、複合の何れかを取る事ができる。信号タイプがシンプルであったならば、それは一つの特性情報からなる。信号タイプがバンドルであったならば、それは全てのタイプが同じ特性情報を持つ幾つかの信号タイプからなる。信号タイプが複合であったならば、それはバンドル信号タイプの列となる。複合信号タイプの順番は実際の信号の組成を意味する。
*/
		SignalIdType signalIdTypeGet ()
			raises (itut_x780::ApplicationError);

/**
次のメソッドは GTP によって表される TP の参照リストを返す。
*/
		TPNameSeqType tpsInGTPListGet ()
			raises (itut_x780::ApplicationError);

	}; // interface GTP


	interface GTPFactory: itut_x780::ManagedObjectFactory
	{
```

```
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID,    // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList)
                  raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

      }; // GTPFactory

/**
```

6.5.12 ManagedElement

ManagedElement オブジェクトは通信機器または加入者へのサポートかつ/またはサービスを提供する管理された要素機能を行う通信ネットワーク内の TMN エンティティ(グループまたはパーツ)を表す管理オブジェクトである。
管理された要素は付加的に調停や OS の機能を行っても良いし行わなくとも構わない。管理された要素はモニタされ、かつ/またはコントロールされる目的のための標準 CORBA インタフェースを通して管理される。管理された要素は地理的に分散しているか、もしくは分散していない機器を含んでいる。

管理された要素のオブジェクトが属性値の変化の通知をサポートしている時、attributeValueChange 通知は、警報様態、ユーザレベル、版数、場所名、現在の問題リスト、視聴覚局所警報有効化の一つの属性が変化した時に発行されなければならない。
サポートされないことがある上記の属性の為に、属性値変化通知の発行の振る舞いは属性が管理オブジェクトによってサポートされている時のみ適用される。
オブジェクトが状態変更通知をサポートしている場合、stateChangeNotification は管理状態、操作状態、使用上の状態の変化があれば、発行しなければならない。

管理プロトコルによる削除は許可されない。(オブジェクトは削除操作のレスポンスとして DeletenotAllowd 例外を送信すべきである。)

このインタフェースは既出の管理エレメントオブジェクトのサブクラスである M.3100 管理エレメント Rl オブジェクトを基本としている。

この値の型は一つの操作内の全ての ManagedElement 属性を検索するために使われる。
それらがサポートされていない場合は、ほとんどのサポートされていない属性は空の文字列またはリストとして返される。もっとも、空の文字列の受信はその属性がサポートされていない事を意味しない。

このインタフェースは次の条件付パッケージ（CONDITIONAL PACKAGES）を含んでいる。

- audibleVisualLocalAlarmPackage: インスタンスがそれをサポートしている場合に存在する。
- alarmSeverityAssignmentPointerPackage: 管理オブジェクトが、警報サービスの構成をサポートしている場合に存在する。
- userLabelPackage: インスタンスがこれをサポートしている場合に存在する。
- vendorNamePackage: インスタンスがこれをサポートしている場合に存在する。
- versionPackage: インスタンスがこれをサポートしている場合に存在する。
- locationNamePackage: インスタンスがこれをサポートしている場合に存在する。
- externalTimePackage: インスタンスがこれをサポートしている場合に存在する。
- systemTimingSourcePackage: インスタンスがこれをサポートしている場合に存在する。
- arcPackage: インスタンスがこれをサポートしている場合に存在する。
- arcRetrieveAlarmDetailPackage: インスタンスがこれをサポートし、arcPackage もまたサポートしている場合に存在する。

```
*/

      valuetype ManagedElementValueType: itut_x780::ManagedObjectValueType
      {
            public AlarmStatusType           alarmStatus;
                  // GET
            public CurrentProblemSetType     currentProblemList;
                  // GET
            public AdministrativeStateType   administrativeState;
```

```
            // GET-REPLACE
        public OperationalStateType      operationalState;
            // GET
        public UsageStateType            usageState;
            // GET
        public boolean                   enableAudibleVisualLocalAlarm;
            // conditional
            // audibleVisualLocalAlarmPackage
            // GET-REPLACE
        public AlarmSeverityAssignmentProfileNameType
            alarmSeverityAssignmentProfilePointer;
            // conditional
            // alarmSeverityAssignmentPointerPackage
            // GET-REPLACE
        public Istring                   userLabel;
            // conditional
            // userLabelPackage
            // GET-REPLACE
        public Istring                   vendorName;
            // conditional
            // vendorNamePackage
            // GET-REPLACE
        public Istring                   version;
            // conditional
            // versionPackage
            // GET-REPLACE
        public Istring                   locationName;
            // conditional
            // locationNamePackage
            // GET-REPLACE
        public ExternalTimeType          externalTime;
            // conditional
            // externalTimePackage
            // GET-REPLACE
        public SystemTimingSourceType    systemTimingSource;
            // conditional
            // systemTimingSourcePackage
            // GET-REPLACE
        public ArcStateType       arcState;
            // conditional
            // arcPackage
            // GET
        public ArcQIStatusType           arcQIStatus;
            // conditional
            // arcPackage
            // GET
        public ArcProbableCauseSetType   arcProbableCauseList;
            // conditional
            // arcPackage
            // GET-REPLACE, ADD-REMOVE
        public ArcIntervalProfileNameType arcIntervalProfilePointer;
            // conditional
            // arcPackage
            // GET-REPLACE
        public ArcTimeType        arcManagementRequestedInterval;
            // conditional
            // arcPackage
            // GET-REPLACE
        public ArcTimeType        arcTimeRemaining;
            // conditional
            // arcPackage
            // GET
}; // valuetype ManagedElementValueType


interface ManagedElement: itut_x780::ManagedObject
{
        AlarmStatusType alarmStatusGet ()
```

```
                raises (itut_x780::ApplicationError);

/**

次のメソッドは管理オブジェクトと関連して重要度を含んだ現在の存在する問題を返す。

*/
            CurrentProblemSetType currentProblemListGet ()
                raises (itut_x780::ApplicationError);

            AdministrativeStateType administrativeStateGet ()
                raises (itut_x780::ApplicationError);

            void administrativeStateSet
                (in AdministrativeStateType administrativeState)
                raises (itut_x780::ApplicationError);

            OperationalStateType operationalStateGet ()
                raises (itut_x780::ApplicationError);

            UsageStateType usageStateGet ()
                raises (itut_x780::ApplicationError);

/**

次のメソッドは視聴覚局所警報がイネーブルド(enabled)の場合に”true”を返す。値に”false”を設定すると視聴覚
局所警報を抑止する。
(実行中の警報は静止されるべきである。)聴覚警報のリセットは、次の警報状態の発生まで警報を静止する。これは
M.3100 の仕様である”allow”と”inhibit”アクションから変更された。
現在の設定を読む事ができ、変更が発生した際に属性値変更通知が発行されるという理由でこの属性が選ばれている。
*/

            boolean enableAudibleVisualLocalAlarmGet ()
                raises (itut_x780::ApplicationError,
                        NOaudibleVisualLocalAlarmPackage);

/**

次のメソッドは視聴覚局所警報を有効または無効にするために使用される。

*/
            void enableAudibleVisualLocalAlarmSet
                (in boolean enable)
                raises (itut_x780::ApplicationError,
                        NOaudibleVisualLocalAlarmPackage);

/**
次のメソッドは視聴覚局所警報を静止するために使用される。
*/
            void resetAudibleAlarm ()
                raises (itut_x780::ApplicationError,
                        NOaudibleVisualLocalAlarmPackage);

/**
次のメソッドは警報重要度割付プロファイルポインタを検索するために使用される。
警報重要度割付プロファイルポインタが空の場合、警報を報告する際に次の 2 つの選択のうち一つが適用される。
a)管理されたシステムが重要度を割り付ける。 b)’不定’の値が使用される。
*/
            AlarmSeverityAssignmentProfileNameType
                alarmSeverityAssignmentProfilePointerGet ()
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

/**
次のメソッドは警報重要度割付プロファイルポインタを設定するために使用される。
*/
```

```
            void alarmSeverityAssignmentProfilePointerSet
                  (in AlarmSeverityAssignmentProfileNameType profile)
                  raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

/**
次のメソッドはオブジェクトを特定するために管理システムで使用され得るラベルを返す。
*/
            Istring userLabelGet ()
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

/**
次のメソッドはインスタンスへのラベルの割り当てに使用される。この値はクライアントにとって意味を持ちうるがオブ
ジェクトについては持たない。
この属性がサポートされるならば、これはこのオブジェクトによって発行される通知に含まれるべきである。
*/
            void userLabelSet
                  (in Istring userLabel)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

/**
次のメソッドは関連する資源の上位の名前を返す
*/
            Istring vendorNameGet ()
                  raises (itut_x780::ApplicationError,
                        NOvendorNamePackage);

/**
次のメソッドは関連する資源の上位の名前を設定する。
*/
            void vendorNameSet
                  (in Istring vendorName)
                  raises (itut_x780::ApplicationError,
                        NOvendorNamePackage);

/**
次のメソッドは関連する資源の版数を返す。
*/
            Istring versionGet ()
                  raises (itut_x780::ApplicationError,
                        NOversionPackage);

/**
次のメソッドは関連する資源の版数を設定する。
*/
            void versionSet
                  (in Istring version)
                  raises (itut_x780::ApplicationError,
                        NOversionPackage);

/**
次のメソッドは関連する資源の物理的な位置を返す。
*/
            Istring locationNameGet ()
                  raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

/**
次のメソッドは関連する資源の物理的な位置を設定する。
*/
            void locationNameSet
                  (in Istring locationName)
                  raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);
```

```
/**
次のメソッドは管理エレメントの time-of-day システム時間を返す。この属性は管理エレメントでの全てのタイムスタンプ動作の参照用として機能する。
*/
              ExternalTimeType externalTimeGet ()
                     raises (itut_x780::ApplicationError,
                            NOexternalTimePackage);

/**
次のメソッドは管理された要素の time-of-day システム時間を設定する。この属性は管理された要素での全てのタイムスタンプ動作の参照用として機能する。
*/
              void externalTimeSet
                     (in ExternalTimeType externalTime)
                     raises (itut_x780::ApplicationError,
                            NOexternalTimePackage);

/**
次のメソッドは管理エレメントのシステムタイミングソースを返す。これは同期のために、プライマリとセカンダリの管理エレメントのタイミングソースを指定するために利用される。
*/
              SystemTimingSourceType systemTimingSourceGet ()
                     raises (itut_x780::ApplicationError,
                            NOsystemTimingSourcePackage);

/**
次のメソッドは管理エレメントのシステムタイミングソースを設定するために利用される。これは時間同期のためにプライマリとセカンダリの管理エレメントのタイミングソースを指定するために利用される。
*/
              void systemTimingSourceSet
                     (in SystemTimingSourceType systemTimingSource)
                     raises (itut_x780::ApplicationError,
                            NOsystemTimingSourcePackage);

/**
状態変化通知はこの属性値の変化を示すために使用されなければならない。
*/
              ArcStateType arcStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOarcPackage);

/**
状態変化通知も属性値変化通知もまた、この属性値の変化を示すために使用されてはならない。
*/

              ArcQIStatusType     arcQIStatusGet ()
                     raises (itut_x780::ApplicationError,
                            NOarcPackage);

              ArcProbableCauseSetType    arcProbableCauseListGet ()
                     raises (itut_x780::ApplicationError,
                            NOarcPackage);

              void arcProbableCauseListSet
                     (in ArcProbableCauseSetType arcProbableCauseList)
                     raises (itut_x780::ApplicationError,
                            NotSupportedProbableCause,
                            NOarcPackage);

              void arcProbableCauseListAdd
                     (in ArcProbableCauseSetType arcProbableCauseList)
                     raises (itut_x780::ApplicationError,
                            NotSupportedProbableCause,
                            NOarcPackage);
```

```
            void arcProbableCauseListRemove
                  (in ArcProbableCauseSetType arcProbableCauseList)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
次のメソッドは、関連した ARC インターバルプロファイル(Interval Profile)オブジェクトを識別する ARC インターバルプロファイルポインタ(Interval Profile Pointer)の値を検索するために使用される。設定可能な持続間隔と時間間隔が使用されない場合、この属性値は空である。(即ち nalm-qi と nalm-ti の状態では用いられない)
*/

            ArcIntervalProfileNameType arcIntervalProfilePointerGet ()
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
次のメソッドは、関連した ARC インターバルプロファイルオブジェクトを識別する ARC インターバルプロファイルポインタの値を変更するために使用される。設定可能な持続間隔及び時間間隔が使用されない場合、この属性値は空である。(即ち nalm-qi と nalm-ti の状態では用いられない)
*/

            void arcIntervalProfilePointerSet
                  (in ArcIntervalProfileNameType profile)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
次のメソッドは、ARC 間隔の管理要求時間を識別する ARC 管理要求間隔の値を取り出すために使用される。この属性は管理リクエストまたは資源が自動的に alm 状態に遷移する時にのみ値が変化する。この属性の値の変更を要求する管理要求はそうする事が正しくない時に拒否される。例えば、管理された資源が alm か nalm の状態にある時などである。この属性の値は ARC 間隔が与えられた瞬間に管理リクエスト経由で調整されたかそうでないかを反映する。
*/

            ArcTimeType arcManagementRequestedIntervalGet ()
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
次のメソッドは、ARC 間隔の管理要求時間を識別する、ARC 管理の要求された間隔の値を変更するために使用される。
*/

            void arcManagementRequestedIntervalSet
                  (in ArcTimeType time)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**

以下のメソッドは、ARC 間隔(即ち、nalm-qi 状態の持続間隔や、nalm-ti 状態の時間間隔)の残りの時間を検索するために利用される。それは必ずしもその状態の残り時間を示すというわけではないことに注意せよ。例えば、arcTimeRemaining が nalm-qi 状態において 30 分であるとして、ARC 間隔 タイマが期間が切れる前に条件付の問題が管理された資源のために上げられるな ら、それは、再度、条件付の問題が解消し、タイマを再起動して、再び残りの時間 を減少させ始めるまで、タイマを中止して無期限に待つことになる。nalm-ti、nalm、または nalm-qi 状態に資源の遷移があるとき、この属性値は管 理要求間隔に初期化される。動作中のタイマが全く無いとき、その値はタイマが動作していないことを示す (即ち、時間調整が全く行われない)。何れの属性値の変化通知も、この属性値における変化のために送られないこと に注意せよ。
*/

            ArcTimeType arcTimeRemainingGet ()
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
次のメソッドは警報報告を有効にするか無効にするかの何れかに使用される。これは望ましい ARC 状態の特定することによって実現される。幾つかの場合では、状態が与えられた資源タイプでサポートされていないためにその動作が拒否さ
```

れる。状態を特定するのに加えて、管理者は一度限りの使用の為にデフォルト以外に arcInterval を要求する。そのような書き換えは nalmqi と naml-ti 状態への遷移でのみ適用される。
*/

```
            boolean arcControl
                  (in ArcControlRequestType request)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下の操作は申告された警告状況に対して警告の詳細を返す。この機能をサポートするために、インタフェースは
arcPackage もまたサポートしなければならない。
*/

            ArcAlarmDetailSetType arcRetrieveAlarmDetail ()
                  raises (itut_x780::ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);

            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, environmentalAlarm)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, equipmentAlarm)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, communicationsAlarm)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, processingErrorAlarm)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)

      }; // interface ManagedElement


      interface ManagedElementFactory: itut_x780::ManagedObjectFactory
      {
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID,    // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList,
                  in AdministrativeStateType administrativeState,
                        // managedElementPackage
                        // GET-REPLACE
                  in boolean enableAudibleVisualLocalAlarm,
                        // conditional
                        // audibleVisualLocalAlarmPackage
                        // GET-REPLACE
                  in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                  in Istring userLabel,
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                  in Istring vendorName,
                        // conditional
```

```
                        // vendorNamePackage
                        // GET-REPLACE
                    in Istring version,
                        // conditional
                        // versionPackage
                        // GET-REPLACE
                    in Istring locationName,
                        // conditional
                        // locationNamePackage
                        // GET-REPLACE
                    in ExternalTimeType externalTime,
                        // conditional
                        // externalTimePackage
                        // GET-REPLACE
                    in SystemTimingSourceType systemTimingSource,
                        // conditional
                        // systemTimingSourcePackage
                        // GET-REPLACE
                    in ArcProbableCauseSetType arcProbableCauseList,
                        // conditional
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval)
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface ManagedElementFactory

/**
```

6.5.13 ManagedElementComplex

ManagedElementComplex オブジェクトクラスは、ネットワークエレメントの集まりを表す管理オブジェクトのクラスである。OS が、このオブジェクトクラスのインスタンスによって表される複合体が持つ一つまたはそれ以上の NE を、参照し、管理する事ができる。

このインタフェース、は M.3100 managedElemantComplex の仕様を基本としている。

ManagedElementComplexValueType 構造体は、一つの操作における全ての ManagedElementComplex の属性を検索するために使用される。殆どのサポートされていない属性はそれらがサポートされていない場合、空の文字列またはリストとして返される。しかし、空文字列の値の受信は属性がサポートされていない事を意味しない。

```
*/

        valuetype ManagedElementComplexValueType:
                itut_x780::ManagedObjectValueType
        {
        }; // valuetype ManagedElementComplexValueType


        interface ManagedElementComplex: itut_x780::ManagedObject
        {
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectCreation,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion,
                        createDeleteNotificationsPackage)
        }; // interface ManagedElementComplex
```

```
interface ManagedElementComplexFactory: itut_x780::ManagedObjectFactory
{
       itut_x780::ManagedObject create
              (in NameBindingType nameBinding,
              in MONameType superior,
              in string reqID,    // auto naming if empty string
              out MONameType name,
              in StringSetType packageNameList)
              raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

}; // interface ManagedElementComplexFactory


/**
```

6.5.14 MPCrossConnection (マルチポイントクロスコネクト)

MPCrossConnection (マルチポイントクロスコネクト(Multipoint Cross Connection))オブジェクトクラスは端点または終端属性によって記載された GTP オブジェクトと、端点または crossConnection 管理オブジェクトを含んだ終端属性にリストされた GTP オブジェクト間の割り当ての関連性を表す。

マルチポイントクロスコネクトは CTP シンク、 双方向 CTP、TTP ソース、双方向 TTP 、または GTP のうちの何れかと、CTP ソース、双方向 CTP、TTP シンク、双方向 TTP、 または GTP である設定との間に確立する事ができる。

fromTermination 属性は常に非空となる。端点または fromTermination 属性によって指されている GTP オブジェクトは全ての端点またはそれらの管理オブジェクトによって表される端点間を流れるトラヒックの様な方法で crossConnection 管理オブジェクトを含む toTermination 属性によって指されている GTP オブジェクトに関連する。

情報は From Termination から To Termination へとクロスコネクトオブジェクトを含んで流れる。

もし crossConnection オブジェクトを含む fromTermination 属性オブジェクトと toTermination 属性にリストされたオブジェクトが GTP ならば、From Termination GTP の n 番目の要素は To Termination GTP の n 番目の要素と(全ての n に対して)関連される。

From Termination の全てのレートは crossConnection オブジェクトを含んだそれぞれの To Termination の全てのレートと同じでなくてはならない。

Signal Type 属性はクロスコネクトされた信号を記述する。端点またはクロスコネクトされた GTP は互換性のある信号タイプを持たなければならない。

次は管理上の状態と動作状態の属性を定義する。

管理上の状態

- Unlocked: mpCrossConnection オブジェクトは管理上解除される。これはその管理上の状態に依存する接続を含むトラヒックの通過を許可する。

- Locked: クロスコネクトした端点間の Cross-Connection の通過するトラヒックは許可されない。

動作状態:
マルチポイントクロスコネクトオブジェクトの動作状態はマルチポイントクロスコネクトに含まれる全てのクロスコネクトオブジェクトを含むクロスコネクトの全ての健全さを反映する。

- Enabled: クロスコネクトはその通常の機能を行う。幾つかの(しかし全てではない)マルチポイントクロスコネクトに含まれるクロスコネクトオブジェクトは無効としうる。

- Disabled: クロスコネクトはその通常のクロスコネクト機能の動作を許可されない。マルチポイントクロスコネクトに含まれる全てのクロスコネクトオブジェクトは無効となる。

可用性様態:

この属性でサポートする値は以下のとおりである。

- In test

- Degraded: マルチポイントクロスコネクトは何れかの個所が劣化される。例えば、もしマルチポイントクロスコネクトに含まれる一つまたは多くの(しかし全てではない)クロスコネクトオブジェクトが無効化されているならば、マルチポイントクロスコネクトは劣化と判断される。マルチポイントクロスコネクトはそれが劣化している間にサービス(即ち 動作状態が有効化する)に応じられるよう留まる。

- Empty SET

このインタフェースは M.3100 mpCrossConnection, mpCrossConnectionR1, namedMPCrossConnection 仕様を基本としている。

このインタフェースは”redline”, “userLable”そして M.3100 の MPCrossConnection を NamedMPCrossConnection へと拡張した”crossConnectionName”を含んでいる。

MPCrossConnectionValueType 構造体は一つの操作における全ての MPCrossConnection 属性を検索するために使用される。ほとんどのサポートされない属性はそれがサポートされていないならば、空の文字列またはリストとして返される。空の文字列の値の受信はその属性がサポートされない事を意味しない。

このインタフェースは次の条件付パッケージを含む。

- redlinePackage: インスタンスがそれをサポートする場合に現れる。

- NamedCrossConnectionPackage: インスタンスがそれをサポートする場合に現れる。

- userLabelPackage: ユーザラベルがサポートされる場合に現れる。
*/

```
valuetype MPCrossConnectionValueType: itut_x780::ManagedObjectValueType
{
        public AdministrativeStateType    admininstrativeState;
                // GET-REPLACE
        public OperationalStateType       operationalState;
                // GET
        public AvailabilityStatusSetType availabilityStatus;
                // GET
        public SignalIdType         signalId;
                // GET
        public MONameType           fromTermination;
                // GET
                // May be nil
        public Istring                    userLabel;
                // conditional
                // userLabelPackage
                // GET-REPLACE
        public boolean                    redline;
                // conditional
                // redlinePackage
                // GET-REPLACE
        public Istring                    crossConnectionName;
                // conditional
                // namedCrossConnectionPackage
                // GET-REPLACE
}; // valuetype MPCrossConnectionValueType


interface MPCrossConnection: itut_x780::ManagedObject
{
        OperationalStateType operationalStateGet ()
                raises (itut_x780::ApplicationError);

        AdministrativeStateType administrativeStateGet ()
                raises (itut_x780::ApplicationError);
```

```
            void administrativeStateSet
                  (in AdministrativeStateType administrativeState)
                  raises (itut_x780::ApplicationError);

            AvailabilityStatusSetType availabilityStatusGet ()
                  raises (itut_x780::ApplicationError);

/**
次の操作は MPCrossConnection の信号タイプを返す。信号タイプはシンプル(simple)、バンドル(bundle)、複合(complex)の何れかとなる。信号タイプがシンプルであれば、それはシングルタイプの特性情報からなる。信号タイプがバンドルであれば、全ての同じ特性情報の信号タイプから構成される。
信号タイプが複合であれば、それは一連のバンドル信号タイプからなる。複合信号タイプの順序は実際の信号の構成を表す。
*/

            SignalIdType signalIdTypeGet ()
                  raises (itut_x780::ApplicationError);

/**
次のメソッドは空値またはそれらのカテゴリの一つのメンバを構成する TTP(ソースまたは双方向)、CTP(シンクまたは双方向)または GTP の参照を返す。
*/

            MONameType fromTerminationGet ()
                  raises (itut_x780::ApplicationError);
                  // the return value may be nil

/**
次の属性は関連した管理オブジェクトが運用限界を超えたかどうか特定する。例えば 敏感な回路の一部であると特定された。
*/
            boolean redlineGet ()
                  raises (itut_x780::ApplicationError,
                        NOredlinePackage);

            void redlineSet
                  (in boolean redline)
                  raises (itut_x780::ApplicationError,
                        NOredlinePackage);

            Istring userLabelGet ()
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

            void userLabelSet
                  (in Istring userLable)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

/**
次の属性はクロスコネクトの為の記述的な名前である。
*/

            Istring crossConnectionNameGet ()
                  raises (itut_x780::ApplicationError,
                        NOnamedCrossConnectionPackage);

            void crossConnectionNameSet
                  (in Istring crossConnectionName)
                  raises (itut_x780::ApplicationError,
                        NOnamedCrossConnectionPackage);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
```

```
        CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, objectDeletion,
               createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, attributeValueChange,
               attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, stateChange,
               stateChangeNotificationPackage)

    }; // interface MPCrossConnection


    interface MPCrossConnectionFactory: itut_x780::ManagedObjectFactory
    {
        itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID,    // auto naming if empty string
               out MONameType name,
               in StringSetType packageNameList,
               in AdministrativeStateType administrativeState,
                      // mpCrossConnectionPackage
                      // GET-REPLACE
               in Istring userLabel,
                      // conditional
                      // userLabelPackage
                      // GET-REPLACE
               in boolean redline,
                      // conditional
                      // redlinePackage
                      // GET-REPLACE
               in Istring crossConnectionName)
                      // conditional
                      // namedCrossConnectionPackage
                      // GET-REPLACE
               raises (itut_x780::ApplicationError,
                      itut_x780::CreateError);

    }; // MPCrossConnectionFactory



/**
```

6.5.15 Network

Network 管理オブジェクトは相関した通信と(論理的または物理的に)情報交換の能力がある管理オブジェクトの集合である。これらのオブジェクトは一つ以上の共通の特性(例えばそれらは一つの顧客かプロバイダに所有され得る、または特定のサービスネットワークに関係しうる)を持っている。ネットワークはそれによって包含関係を形作る他の(大きな)ネットワークの入れ子となってもよい。他のネットワークに含まれたネットワークの例は通信サブネットワークである。それは単一の管理者によって所有され、通信機能のみ行う事ができる。

管理プロトコルによる削除は許可されない。(オブジェクトは削除操作のレスポンスとして DeleteNotAllowd 例外を送信すべきである。)

このインタフェースは M.3100 NetworkRl 仕様を基本としている。

```
*/

    valuetype NetworkValueType: itut_x780::ManagedObjectValueType
    {
        public Istring      userLabel;
    }; // valuetype NetworkValueType
```

```
	interface Network: itut_x780::ManagedObject
	{

/**
次のメソッドはオブジェクトを特定するために管理システムで使用され得るラベルを返す。
*/

		Istring userLabelGet ()
			raises (itut_x780::ApplicationError);

/**
次のメソッドはインスタンスへのラベルを割り当てるために使用される。その値はクライアントに対して意味があるがオブジェクトに対しては意味が無い。この属性がサポートされるならば、それはこのオブジェクトによって発行される通知に含まれるべきである。
*/

		void userLabelSet
			(in Istring userLabel)
			raises (itut_x780::ApplicationError);

	}; // interface Network


	interface NetworkFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID,    // auto naming if empty string
			out MONameType name,
			in StringSetType packageNameList,
			in Istring userLabel)
			raises (itut_x780::ApplicationError,
				itut_x780::CreateError);

	}; // interface NetworkFactory



/**
```

6.5.15.1 LayerND (レイヤ･ネットワーク･ドメイン)

LayerND(レイヤネットワークドメイン)インタフェースをサポートする管理オブジェクトは、同じ G.805 レイヤに関する全ての資源にあるトランスポート管理ドメインを表す。

それはトランスポートネットワークレイヤの位相的な特徴を表す。

注: レイヤネットワークドメインオブジェクトに束縛されたネットワークレイヤ資源オブジェクトはそれらの管理されたシステムでのネットワークレイヤ資源オブジェクトの表示のためにレイヤネットワークドメインに含まれる幾つかのサブドメインオブジェクトの生成を必要とされ得る。これはレイヤネットワークドメインがネットワークのサブクラスであるため許可され、ネットワークからネットワークへの名前の結合がある。
*/

```
	valuetype LayerNDValueType: NetworkValueType
	{
		public SignalIdType signalId;
			// GET, SET-BY-CREATE
	}; // valuetype LayerNDValueType


	interface LayerND: Network
	{
```

```
/**
次の属性は、考慮中のエンティティが属するレイヤの(G.805 の文脈中の)の特有の情報を定義する。それはサブネットワーク接続(Sub-network Connecton)/接続が可能かどうかを決定するために使用される。信号識別子は単純なレートおよびフォーマットであることができるし、あるいは信号の集合を形成する同じ特有の情報を備えた一纏りのエンティティであることができる。
*/
              SignalIdType signalIdGet ()
                     raises (itut_x780::ApplicationError);

       }; // interface LayerND


       interface LayerNDFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID, // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList,
                     in SignalIdType signalId,
                            // layerNetworkDomainPackage
                            // GET, SET-BY-CREATE
                     in Istring userLabel)
                            // conditional
                            // userLabelPackage
                            // GET-REPLACE
                     raises (itut_x780::ApplicationError,
                            itut_x780::CreateError);

       }; // interface LayerNDFactory


/**
```

6.5.15.2 BasicLayerND (ベーシックレイヤネットワークドメイン)

BasicLayerND (ベーシックレイヤネットワークドメイン)インタフェースはトレールの即時設定及び開放を管理する管理オブジェクトのクラスである。それは次の機能性を提供する:

1) 即時のトレール設定
2) トレールの開放

basicTrailHandlerPackage:
即時のトレール設定。それが setupTrail リクエストを受け取る場合、エージェントは次のものに対して責任を持つ。:

1) トレールのためのルートを見つける
2) どのような必要なサブネットワーク接続でも設定する
3) トレールオブジェクトインスタンスが正確な初期の値で作成された事を確認する。
4) そのリクエストの結果をサービスユーザに通知する。

トレール開放。それが releaseTrail 要求を受け取る場合、エージェントは次のものに対して責任を持つ。:

1) 任意の用いられているサブネットワーク接続を開放する。最新版ネットワーク資源使用法(設定)情報のアップデートを行う。
2) そのリクエストの結果をサービスユーザに通知する。

リンクの管理がサポートされる場合、logicalLinkHandlerPackage が存在する。このパッケージは、論理的なリンクの順番の生成および削除のサポートを提供する。

リンク端の管理がサポートされる場合、logicalLinkEndHandlerPackage が存在する。このパッケージは、論理的なリンク端の通常の生成および削除のサポートを提供する。

トポロジのリンクの管理がサポートされる場合、topologicalLinkHandlerPackage が存在する。このパッケージは、トポロジリンクの通常の生成および削除のサポートを提供する。さらに、それはサーバトレールと割り当てられたトポロジのリンクへの割り当てられていないトポロジのリンクを可能にする動作を提供する。

トポロジのリンク端の管理がサポートされる場合、topologicalLinkEndHandlerPackage が存在する。このパッケージは、トポロジリンクの終了の通常の生成および削除のサポートを提供する。さらに、それは、サーバネットワークTTP に割り当てられるまだ割り当てられていないトポロジのリンク終端および、割り当てを解除される割り当てられたトポロジのリンク終端を可能にする動作を提供する。
*/

```
valuetype BasicLayerNDValueType: LayerNDValueType
{
}; // valuetype BasicLayerNDValueType


interface BasicLayerND: LayerND
{
```

/**
次の動作はネットワークトレール端点間のトレール(Trail)を設定するために使用される。接続されるトレール端点は 2 つの方法(ネットワークトレール端点を明示的に定義する事、または一つ以上のアクセスグループ(Access Groups)の指定により、明示的にネットワークトレール端点を識別する事) のうちの一つで指定する事ができる。

もし仕様が既にトレールに関与しているネットワーク端点のうちのどれかが指定されると、この動作は失敗する。トレールは動作パラメタ指向性の中で指定されるような指向性(一方向か双方向)を持つ。クライアントの識別子はサーバに渡され、作成されたトレールの識別子に対するサーバによって記録される。
*/

```
        void setupTrail
              (in ConnectivityEndPointSetType aEnd,
              in ConnectivityEndPointSetType zEnd,
              in DirectionalityType direction,
              in MONameType qos, // may be empty
              in Istring userId, // may be empty string
              in Istring userLabel, // may be empty string
              out TrailNameType resultTrail,
              out ConnectivityEndPointSetType resultAEnd,
              out ConnectivityEndPointSetType resultZEnd)
              raises (itut_x780::ApplicationError,
                    NetworkTTPsNotPartOfLayerND,
                    AEndNetworkTPConnected,
                    NetworkTTPsInAEndAccessGroupConnected,
                    ZEndNetworkTPConnected,
                    NetworksInZEndAccessGroupConnected,
                    UserIdentifierNotUnique,
                    FailureToSetUserIdentifier,
                    InvalidTPType,
                    InvalidTrail,
                    WrongAEndDirectionality,
                    WrongZEndDirectionality,
                    InvalidTransportServiceCharacteristics,
                    InvalidTrafficDescriptor);
```

/**
次の動作はトレールを開放するために使用される。ClientConnectionList によって指されたリンクコネクションとレイヤコネクションリストパッケージによって指されたサブネットワークコネクションもまた、この動作によって開放される。

成功する場合、分離されたネットワークトレール端点の connectivityPointer はこの動作の結果、空に設定される。
*/

```
        void releaseTrail
              (in TrailNameType atrail,
              in Istring userId) // may be empty string
              raises (itut_x780::ApplicationError,
```

```
                        UnknownTrail,
                        TrailConnected);

/**
次の動作が、次の何れかの間の論理的なリンクを作成する:
* 2 つのサブネットワーク
* 2 つのアクセスグループ
* サブネットワークおよびアクセスグループ
* アクセスグループおよびサブネットワーク
2 つの端点は動作リクエストの情報で指定される。

関連するサブネットワーク管理オブジェクトの linkPointerList 属性は論理的なリンクの生成を反映するために修正
される。

論理的なリンク管理オブジェクトはこの動作の結果作成される。論理的なリンクの名前は動作結果の中で返される。
*/

              void establishLogicalLink
                     (in LinkEndType aEnd,
                     in LinkEndType zEnd,
                     in Istring userId, // may be empty string
                     in Istring userLable, // may be empty string
                     in LinkDirectionalityTypeOpt direction,
                     out LogicalLinkNameType link)
                     raises (itut_x780::ApplicationError,
                            IncorrectLinkEnds,
                            UserIdentifierNotUnique,
                            FailureToSetUserIdentifier,
                            FailureToCreateLink,
                            FailureToBindLink,
                            FailureToSetDirectionality,
                            NOlogicalLinkHandlerPackage);

/**
次の動作が論理的なリンクを削除する。

関連するサブネットワークの linkPointerList 属性は論理的なリンクの削除を反映するために修正される。
*/

              void removeLogicalLink
                     (in LogicalLinkNameType link)
                     raises (itut_x780::ApplicationError,
                            IncorrectLink,
                            LinkConnectionsExisting,
                            FailureToDeleteLink,
                            NOlogicalLinkHandlerPackage);

/**
次の動作が、以下の間の論理的なリンクを作成する。
* 2 つのサブネットワーク
* 2 つのアクセスグループ
* サブネットワークおよびアクセスグループ
* アクセスグループおよびサブネットワーク

論理的なリンクおよび 2 つの論理的なリンク端管理オブジェクトがこの動作の結果として作成される。論理的なリンクの
名前および論理的なリンク端の名前は、動作結果の中で返される。
*/

              void establishLogicalLinkAndEnds
                     (in LinkEndType aEnd,
                     in LinkEndType zEnd,
                     in Istring userId, // may be empty string
                     in Istring userLable, // may be empty string
                     in LinkDirectionalityTypeOpt direction,
                     out LogicalLinkNameType link,
```

```
                    out LogicalLinkEndNameType resultAEnd,
                    out LogicalLinkEndNameType resultZEnd)
                    raises (itut_x780::ApplicationError,
                           UserIdentifierNotUnique,
                           IncorrectSubnetwork,
                           FailureToCreateLinkEnd,
                           FailureToBindLinkEnd,
                           FailureToSetUserIdentifier,
                           FailureToSetDirectionality,
                           NOlogicalLinkEndHandlerPackage);

/**
次の動作は、論理的なリンクおよびリンクを表わす論理的なリンク端管理オブジェクトを削除する。

関連するサブネットワークの linkPointerList 属性はリンクの削除を反映するために修正される。
*/
             void removeLogicalLinkAndEnds
                    (in LogicalLinkNameType link)
                    raises (itut_x780::ApplicationError,
                           IncorrectLink,
                           IncorrectLinkEnds,
                           NetworkCTPsExisting,
                           LinkConnectionsExisting,
                           FailureToDeleteLink,
                           NOlogicalLinkEndHandlerPackage);
/**
次の動作はサーバレイヤの中のトレールとクライアントレイヤ中のトポロジのリンクを関連付ける。一つのトポロジのリンクに関連した唯一のサーバトレールがありうる。
*/

             void associateTrailWithTopologicalLink
                    (in TopLinkNameType link,
                    in TrailNameType atrail,
                    out CapacityType potentialCapacity,
                    out LinkConnectionNameSetType resultLCs)
                    raises (itut_x780::ApplicationError,
                           NoSuchLink,
                           NoSuchTrail,
                           LinkAndTrailsNotCompatible,
                           InitialCapacitiesFailure,
                           TrailAlreadyAssociated,
                           FinalCapacitiesFailure,
                           ConsistencyFailure,
                           FailureToAssociate,
                           NOtopologicalLinkHandlerPackage);
/**
次の動作は、それがサポートするクライアントレイヤの中のトポロジリンクからのサーバレイヤトレールを分離する。
*/
             void disassociateTrailFromTopologicalLink
                    (in TopLinkNameType link,
                    in TrailNameType atrail) // may be nil
                    raises (itut_x780::ApplicationError,
                           NoSuchLink,
                           NoSuchTrail,
                           TrailNotAssociated,
                           CapacityProvisionned,
                           FinalCapacitiesFailure,
                           FailureToDisassociate,
                           NOtopologicalLinkHandlerPackage);

/**
次の動作が、以下の間のトポロジのリンクを作成する。
* 2 つのサブネットワーク
* 2 つのアクセスグループ
* サブネットワークおよびアクセスグループ
* アクセスグループおよびサブネットワーク
```

トポロジのリンク管理オブジェクトはこの動作の結果として作成される。トポロジのリンクの名前は動作結果の中で返される。
*/

```
              void establishTopologicalLink
                     (in LinkEndType aEnd,
                     in LinkEndType zEnd,
                     in Istring userId, // may be empty string
                     in Istring userLabel, // may be empty string
                     in DirectionalityTypeOpt direction,
                     out TopLinkNameType link)
                     raises (itut_x780::ApplicationError,
                            IncorrectLinkEnds,
                            UserIdentifierNotUnique,
                            FailureToSetUserIdentifier,
                            FailureToCreateTopologicalLink,
                            FailureToBindTopologicalLink,
                            FailureToSetDirectionality,
                            NOtopologicalLinkHandlerPackage);

/**
```
次の動作はトポロジのリンクを削除する。

関連するサブネットワークの linkPointerList 属性はトポロジのリンクの削除を反映するために修正される。
*/

```
              void removeTopologicalLink
                     (in TopLinkNameType link)
                     raises (itut_x780::ApplicationError,
                            NoSuchLink,
                            LinkConnectionsExisting,
                            FailureToDeleteLink,
                            NOtopologicalLinkHandlerPackage);

/**
```
次の動作はサーバレイヤの networkTTP とクライアントレイヤのトポロジのリンク端を関連させる。一つのトポロジのリンクの終了に関連した唯一の networkTTP がありうる。

動作の結果は、リンクの潜在能力および利用可能なネットワーク CTP のリストを返す。
*/

```
              void associateNetworkTTPWithTopologicalLinkEnd
                     (in TopLinkEndNameType le,
                     in NetworkTTPAbstractNameType ttp,
                     out CapacityType potentialCapacity,
                     out NetworkCTPAbstractNameSetType ctpList)
                     raises (itut_x780::ApplicationError,
                            NoSuchLinkEnd,
                            NoSuchNetworkTTP,
                            LinkEndAndNetworkTTPsNotCompatible,
                            InitialCapacitiesFailure,
                            NetworkTTPAlreadyAssociated,
                            FinalCapacitiesFailure,
                            ConsistencyFailure,
                            FailureToAssociate,
                            NOtopologicalLinkEndHandlerPackage);

/**
```
次の動作は、クライアントレイヤ中のトポロジのリンク端からサーバレイヤネットワーク TTP を分離する。
*/

```
              void disassociateNetworkTTPFromTopologicalLinkEnd
                     (in TopLinkEndNameType le,
                     in NetworkTTPAbstractNameType ttp) // may be nil
                     raises (itut_x780::ApplicationError,
                            NoSuchLinkEnd,
```

```
                        NoSuchNetworkTTP,
                        NetworkTTPNotAssociated,
                        CapacityProvisionned,
                        FinalCapacitiesFailure,
                        NOtopologicalLinkEndHandlerPackage);

/**
次の動作が、以下の間のトポロジのリンクを作成する。
* 2 つのサブネットワーク
* 2 つのアクセスグループ
* サブネットワークおよびアクセスグループ
* アクセスグループおよびサブネットワーク

トポロジのリンクおよび 2 つのトポロジのリンク端管理オブジェクトがこの動作の結果として作成される。トポロジのリンクの名前およびトポロジのリンク端の名前は、動作結果の中で返される。
*/

              void establishTopologicalLinkAndEnds
                     (in LinkEndType aEnd,
                     in LinkEndType zEnd,
                     in Istring userId, // may be empty string
                     in Istring userLabel, // may be empty string
                     in DirectionalityTypeOpt direction,
                     out TopLinkNameType link,
                     out TopLinkEndNameType resultAEnd,
                     out TopLinkEndNameType resultZEnd)
                     raises (itut_x780::ApplicationError,
                            IncorrectLinkEnds,
                            UserIdentifierNotUnique,
                            IncorrectSubnetwork,
                            FailureToCreateLinkEnd,
                            FailureToBindLinkEnd,
                            FailureToSetUserIdentifier,
                            FailureToSetDirectionality,
                            NOtopologicalLinkEndHandlerPackage);

/**
次の動作はトポロジのリンク、およびトポロジのリンクを表わす管理オブジェクトのトポロジのリンク終了を削除する。
*/

              void removeTopologicalLinkAndEnds
                     (in TopLinkNameType link)
                     raises (itut_x780::ApplicationError,
                            NoSuchLinkEnd,
                            NetworkCTPsExisting,
                            FailureToDeleteTopologicalLinkEnd,
                            NOtopologicalLinkEndHandlerPackage);

       }; // interface BasicLayerND


       interface BasicLayerNDFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID,    // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList,
                     in SignalIdType signalId,
                            // layerNetworkDomainPackage
                            // GET, SET-BY-CREATE
                     in Istring userLabel)
                            // conditional
                            // userLabelPackage
                            // GET-REPLACE
                     raises (itut_x780::ApplicationError,
```

```
                        itut_x780::CreateError);

    }; // interface BasicLayerNDFactory

/**
```

6.5.16 Pipe

パイプ(Pipe)インタフェースをサポートする管理オブジェクトは、2 つ以上の端点間の情報の転送を保証する。

このクラスのインスタンスが双方向である場合、a-および z-端点はさらに双方向でなければならない。このクラスのインスタンスが一方向の場合、a-点ソース TP か双方向 TP でなければならない。そして z-端点はシンク TP か双方向 TP でなければならない。

一方向の接続については、aEndNWTPList 属性がソース終了を特定しなければならない。

転送がトレールおよびリンク接続経由で達成されるという理由では、パイプクラスはインスタンス化可能ではない。

このインタフェースは次の条件付パッケージを含んでいる。

- administrativeStatePackage:インスタンスがそれをサポートサポートする場合は存在する。
- operationalStatePackage:インスタンスがそれをサポートする場合は、存在する。
- alarmSeverityAssignmentPointerPackage:コミュニケーション警報がサポートされ、管理オブジェクトが警報重要度の設定をサポートする場合は存在する。
- availabilityStatusPackage:インスタンスがそれをサポートする場合は存在する。
- protectedPackage:インスタンスがそれをサポートする場合は存在する。
- qualityOfConnectivityServicePackage:インスタンスがそれをサポートする場合は存在する。
- supportedByPackage:インスタンスがそれをサポートする場合は存在する。
- tmnCommunicationsAlarmInformationR1Package:インスタンスがコミュニケーション警報(Communication Alarm)をサポートする場合は存在する。
- userLabelPackage:インスタンスがそれをサポートする場合は存在する。

```
*/

    valuetype PipeValueType: itut_x780::ManagedObjectValueType
    {
        public DirectionalityType  directionality;
            // GET
        public SignalIdType        signalId;
            // GET, SET-BY-CREATE
        public MONameSetType       aEndNetworkTPList;
            // GET, SET-BY-CREATE
        public MONameSetType       zEndNetworkTPList;
            // GET, SET-BY-CREATE
        public AdministrativeStateType   administrativeState;
            // conditional
            // administrativeStatePackage
            // GET-REPLACE
        public OperationalStateType      operationalState;
            // conditional
            // operationalStatePackage
            // GET
        public AlarmSeverityAssignmentProfileNameType
            alarmSeverityAssignmentProfilePointer;
            // conditional
            // alarmSeverityAssignmentPointerPackage
            // GET-REPLACE
        public AvailabilityStatusSetType availabilityStatus;
            // conditional
            // availabilityStatusPackage
            // GET
        public boolean                   protected;
            // conditional
            // protectedPackage
```

```
                // GET, SET-BY-CREATE
        public MONameType          qualityOfConnectivityService;
                // conditional
                // qualityOfConnectivityServicePackage
                // GET
        public MONameSetType       supportedByObjectList;
                // conditional
                // supportedByPackage
                // GET-REPLACE, ADD-REMOVE
        public AlarmStatusType           alarmStatus;
                // conditional
                // tmnCommunicationsAlarmInformationR1Package
                // GET
        public CurrentProblemSetType     currentProblemList;
                // conditional
                // tmnCommunicationsAlarmInformationR1Package
                // GET
        public Istring                   userLabel;
                // conditional
                // userLabelPackage
                // GET-REPLACE
    }; // valuetype PipeValueType


    interface Pipe : itut_x780::ManagedObject
    {

/**
方向性属性は、送信が一方向か双方向かを示す。
*/

        DirectionalityType directionalityGet ()
                raises (itut_x780::ApplicationError);

/**
信号識別子属性は、接続(Connectivity)インスタンスを横断して転送される信号について記述する。このインスタンスによって関連づけられるネットワーク端点を表す管理オブジェクトは、互換性をもつ信号識別子を持つ必要がある。
*/

        SignalIdType signalIdGet ()
                raises (itut_x780::ApplicationError);

/**
次の属性の値は、Pipe オブジェクトクラスのサブクラスのインスタンスの一つ以上のネットワーク端点を識別する。この属性は空にはなりえない。
*/

        MONameSetType aEndNetworkTPListGet ()
                raises (itut_x780::ApplicationError);

/**
次の属性の値は、Pipe オブジェクトクラスのサブクラスのインスタンスの一つ以上のネットワーク端点を識別する。
*/

        MONameSetType zEndNetworkTPListGet ()
                raises (itut_x780::ApplicationError);

        AdministrativeStateType administrativeStateGet ()
                raises (itut_x780::ApplicationError,
                        NOadministrativeStatePackage);

        void administrativeStateSet
                (in AdministrativeStateType administrativeState)
                raises (itut_x780::ApplicationError,
                        NOadministrativeStatePackage);

/**
```

```
運用上の状態は、信号を運ぶ能力を示す。
*/

              OperationalStateType operationalStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOoperationalStatePackage);

              AlarmSeverityAssignmentProfileNameType
                     alarmSeverityAssignmentProfilePointerGet ()
                     raises (itut_x780::ApplicationError,
                            NOalarmSeverityAssignmentPointerPackage);

              void alarmSeverityAssignmentProfilePointerSet
                     (in AlarmSeverityAssignmentProfileNameType profile)
                     raises (itut_x780::ApplicationError,
                            NOalarmSeverityAssignmentPointerPackage);

              AvailabilityStatusSetType availabilityStatusGet ()
                     raises (itut_x780::ApplicationError,
                            NOavailabilityStatusPackage);

/**
次の属性は関連する管理オブジェクトが保護されても保護されなくても識別する。値 TRUE は、それが保護される事を暗
示する。
*/

              boolean protectedGet ()
                     raises (itut_x780::ApplicationError,
                            NOprotectedPackage);

/**
次の属性は、パイプおよびそのサブクラスのためのサービス品質を識別し、さらなる定義を要求する。
*/

              MONameType quailityOfConnectivityServiceGet ()
                     raises (itut_x780::ApplicationError,
                            NOquailityOfConnectivityServicePackage);

              MONameSetType supportedByObjectListGet ()
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              void supportedByObjectListSet
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              void supportedByObjectListAdd
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              void supportedByObjectListRemove
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              AlarmStatusType alarmStatusGet ()
                     raises (itut_x780::ApplicationError,
                            NOtmnCommunicationsAlarmInformationR1Package);

/**
下記のメソッドは管理オブジェクトに関連して現在の既存の問題を重要度を含んだ形で返す。
*/

              CurrentProblemSetType currentProblemListGet ()
                     raises (itut_x780::ApplicationError,
```

```
                    NOtmnCommunicationsAlarmInformationR1Package);

        Istring userLabelGet ()
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        void userLabelSet
                (in Istring userLabel)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, communicationAlarm,
                tmnCommunicationsAlarmInformationR1Package)

};  // interface Pipe


interface PipeFactory: itut_x780::ManagedObjectFactory
{
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in SignalIdType signalId,
                        // pipeR2Package
                        // GET, SET-BY-CREATE
                in MONameSetType aEndNetworkTPList,
                        // pipeR2Package
                        // GET, SET-BY-CREATE
                in MONameSetType zEndNetworkTPList,
                        // pipeR2Package
                        // GET, SET-BY-CREATE
                in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
                in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                in boolean protected,
                        // conditional
                        // protectedPackage
                        // GET, SET-BY-CREATE
                in MONameSetType supportedByObjectList,
                        // conditional
                        // supportedByPackage
                        // GET-REPLACE, ADD-REMOVE
                in Istring userLabel)
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                raises (itut_x780::ApplicationError,
```

```
                        itut_x780::CreateError);
    }; // interface PipeFactory


/**
```

6.5.16.1 LinkConnection

LinkConnection インタフェースをサポートする管理オブジェクトは、ネットワーク接続端点(Network Connection Termination Points)間の情報の透過的な転送を担う。

リンク接続(Link Connection)はトレールのコンポーネントとなりうる。一つ以上のリンク接続(またサブネットワーク接続(Sub-network Connecton))のシーケンスは、トレールを形成するためにともにリンクされ得る。

リンク接続は一方向あるいは、双方向である。

ポイントツーポイントの一方向リンク接続は、双方向ネットワーク(Network)接続端点ソースあるいは双方向ネットワーク(Network)接続端点とネットワーク接続端点シンクあるいは双方向ネットワーク接続端点の間で確立する事ができる。

ポイントツーポイントの双方向リンク接続(LinkConnection)は、双方向ネットワーク(Network)接続端点と双方向ネットワーク(Network)接続端点間で確立する事ができる。

リンク接続(Link Connection)を作成する操作は成功せず、要求された端点がネットワークトレール端点(Network Trail Termination Point)である場合、無効の TP タイプで失敗する。

全てのタイプのリンク接続(Link Connection)については、A End 属性によって示されたネットワーク端点は Z End 属性によって示されたネットワーク端点と関連する。これは、トラヒックは方向性属性で示される一方向または双方向の様態で管理オブジェクトによって表されるネットワーク端点の間を流れることができるといったことによる。

このインタフェースは次の条件付パッケージを含んでいる。

- serverTrailListPackage:リンク接続がサーバトレールにサポートされる場合は存在する
- compositePointerPackage:リンク接続がそのサブネットワーク接続(Sub-network Connecton)のコンポーネントである場合は存在する。
- clientTrailPackage:リンク接続がクライアントトレールを供給する場合は存在する。

```
*/

    valuetype LinkConnectionValueType: PipeValueType
    {
        public TrailNameSetType    serverTrailList;
            // conditional
            // serverTrailListPackage
            // GET, SET-BY-CREATE
        public SNCNameType  compositePointer;
            // conditional
            // compositePointerPackage
            // GET
        public TrailNameTypeclientTrail;
            // conditional
            // clientTrailPackage
            // GET, SET-BY-CREATE
    }; // valuetype LinkConnectionValueType


    interface LinkConnection: Pipe
    {

/**
```

次の属性値は、接続オブジェクトを提供するために平行して使用され得る低位のネットワークレイヤの中でトレールオブジェクト(ほとんどの場合では一つ)を識別する。
*/
```
            TrailNameSetType serverTrailListGet ()
                  raises (itut_x780::ApplicationError,
                        NOserverTrailListPackage);

/**
```
次の属性は接続インスタンスが同じレイヤ内のサブネットワーク接続(Sub-network Connecton)のコンポーネントである場合に使用される。このパッケージは、サブネットワーク接続管理オブジェクトクラスのインスタンスを識別する。与えられたレイヤの内では、与えられたサブネットワーク接続が、リンク接続およびサブネットワーク接続のシーケンスから構成される。このポインタは一つのこれらのコンポーネントから複合サブネットワーク接続までを示す。
```
*/

            SNCNameType compositePointerGet ()
                  raises (itut_x780::ApplicationError,
                        NOcompositePointerPackage);

/**
次の属性値は、接続オブジェクトによって提供される接続と同じネットワークレイヤの中のトレールオブジェクトインスタンスを識別する。
*/

            TrailNameType clientTrailGet ()
                  raises (itut_x780::ApplicationError,
                        NOclientTrailPackage);

      }; // interface LinkConnection


      interface LinkConnectionFactory: itut_x780::ManagedObjectFactory
      {
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID,    // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList,
                  in SignalIdType signalId,
                        // pipeR2Package
                        // GET, SET-BY-CREATE
                  in MONameSetType aEndNetworkTPList,
                        // pipeR2Package
                        // GET, SET-BY-CREATE
                  in MONameSetType zEndNetworkTPList,
                        // pipeR2Package
                        // GET, SET-BY-CREATE
                  in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
                  in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                  in boolean protected,
                        // conditional
                        // protectedPackage
                        // GET, SET-BY-CREATE
                  in MONameSetType supportedByObjectList,
                        // conditional
                        // supportedByPackage
                        // GET-REPLACE, ADD-REMOVE
                  in Istring userLabel,
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
```

```
                    in TrailNameSetType serverTrailList,
                           // conditional
                           // serverTrailListPackage
                           // GET, SET-BY-CREATE
                    in TrailNameType clientTrail)
                           // conditional
                           // clientTrailPackage
                           // GET, SET-BY-CREATE
                    raises (itut_x780::ApplicationError,
                           itut_x780::CreateError);

      }; // interface LinkConnectionFactory

/*
```

6.5.16.2 SNC (サブネットワークコネクション)

SNC(サブネットワーク接続)インタフェースをサポートする管理オブジェクトは、A End 属性の中で識別されたネットワーク端点オブジェクト、および、この管理オブジェクトの Z End 属性にリストされたネットワーク端点オブジェクトを関連させる。
サブネットワーク接続は、明示的に指定されたネットワーク端点(あるいはネットワーク端点のグループ)間で設定されることができるし、あるいはどんな使用されていないネットワーク端点あるいはグループも使用されることができるネットワーク端点管理オブジェクトインスタンスのコンテナの役割を担う管理オブジェクト間で暗黙に設定され得る。

A End と Z End 属性にリストされた管理オブジェクトがグループを表わす場合、A End グループの n 番目の要素は全ての Z End グループの n 番目の要素と関連する。
サブネットワーク接続を含む各グループに n 要素がなくてはならない。

n 要素を備えたグループについては、信号識別子は個々の要素に特有な情報の n 回の纏まりからなる。

ポイントツーポイントの一方向のサブネットワーク接続は、ネットワーク接続端点シンク、双方向のネットワーク接続端点、ネットワークトレール端点のソース、双方向ネットワークトレール端点あるいはネットワークグループ端点の何れかと、ネットワーク接続端点ソース、双方向ネットワーク接続端点、ネットワークトレール端点シンク、双方向ネットワークトレール端点あるいはネットワークグループ端点のうちの何れかの間で成立する事ができる。

ポイントツーポイントの双方向サブネットワーク接続は、双方向のネットワーク接続端点、双方向ネットワークトレール端点あるいはネットワークグループ端点の何れかと、双方向のネットワーク接続端点、双方向ネットワークトレール端点あるいはネットワークグループ端点の何れかの間で成立する事が可能である。

ポイントツーマルチポイントの一方向サブネットワーク接続はネットワーク接続端点シンク、双方向ネットワーク接続端点、ネットワークトレール端点ソース、双方向のネットワークトレール端点あるいはネットワークグループ端点の何れかと、そのメンバがネットワーク接続終端点ソース、双方向のネットワーク接続端点、ネットワークトレール端点シンク、双方向のネットワークトレール端点、あるいはネットワークグループ端点の何れかである集合との間で成立することが可能である。

ポイントツーマルチポイントの双方向サブネットワーク接続は双方向のネットワーク接続端点、双方向のネットワークトレール端点あるいはネットワークグループ終端の何れかと、そのメンバが双方向のネットワーク接続端点、双方向のネットワークトレール端点、あるいはネットワークグループ端点の何れかである集合との間で成立することが可能である。

ComponentPointerPackage はサブネットワーク接続がコンポーネントサブネットワーク接続と同じレイヤ内でのリンク接続で作られる事によってサポートされる。

このインタフェースは次の条件付パッケージを含んでいる。

- compositePointerPackage:サブネットワーク接続が同じレイヤ(分割されたサブネットワーク)内の別のサブネットワーク接続のコンポーネントである場合は存在する。

- componentPointerPackage:サブネットワーク接続が、多くのコンポーネントサブネットワーク接続およびリンク接続から構成される場合は、同じレイヤ(分割されたサブネットワーク)の内に存在する

- RelatedRoutingProfilePackage:ルーチングプロフィールがサポートされる場合は存在する。
*/

```
	valuetype SNCValueType: PipeValueType
	{
		public SNCNameType  compositePointer;
			// conditional
			// compositePointerPackage
			// GET
		public PipeNameSetType     componentPointerList;
			// conditional
			// componentPointerPackage
			// GET
		public MONameType   relatedRoutingProfile;
			// conditional
			// relatedRoutingProfilePackage
			// GET
	}; // valuetype SNCValueType


	interface SNC: Pipe
	{

/**
接続インスタンスが同じレイヤ内のサブネットワーク接続のコンポーネントである場合、次の属性が使用される。
*/

		SNCNameType compositePointerGet ()
			raises (itut_x780::ApplicationError,
				NOcompositePointerPackage);

/**
サブネットワーク接続が、同じレイヤ内の多くの構成要素のサブネットワーク接続およびリンク接続から構成される場合、
次の属性が使用される。
*/

		PipeNameSetType componentPointerListGet ()
			raises (itut_x780::ApplicationError,
				NOcomponentPointerPackage);

		MONameType relatedRoutingProfileGet ()
			raises (itut_x780::ApplicationError,
				NOrelatedRoutingProfilePackage);

	}; // interface SNC


	interface SNCFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID,    // auto naming if empty string
			out MONameType name,
			in StringSetType packageNameList,
			in SignalIdType signalId,
				// pipeR2Package
				// GET, SET-BY-CREATE
			in MONameSetType aEndNetworkTPList,
				// pipeR2Package
				// GET, SET-BY-CREATE
			in MONameSetType zEndNetworkTPList,
				// pipeR2Package
				// GET, SET-BY-CREATE
			in AdministrativeStateType adminstrativeState,
				// conditional
				// administrativeStatePackage
				// GET-REPLACE
			in AlarmSeverityAssignmentProfileNameType profile,
```

```
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                    in boolean protected,
                        // conditional
                        // protectedPackage
                        // GET, SET-BY-CREATE
                    in MONameSetType supportedByObjectList,
                        // conditional
                        // supportedByPackage
                        // GET-REPLACE, ADD-REMOVE
                    in Istring userLabel)
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

    }; // interface SNCFactory


/**
```

6.5.16.3 Trail

レイヤネットワークの中でトレール(Trail)インタフェースをサポートする管理オブジェクトは、一つ以上の他のレイヤネットワークからの特有な情報の転送に対する完全性に責任がある。[このインタフェースは M.3100 trailR2 オブジェクトの後にモデル化する]。

トレールは、2 つ以上のネットワークトレール端点(Network Trail Termination Points)および一つ以上のリンク接続(Link Connection)あるいはサブネットワーク接続(Sub-network Connetctions)から構成され、ネットワーク接続端点(Network Connectin Termination Points)と関連される。

ポイントツーポイントの一方向トレール(Trail)は、ネットワーク TTP ソースあるいは双方向ネットワーク TTP の何れかと、ネットワーク(Network) TTP シンクか双方向ネットワーク(Network) TTP の間で成立する事ができる。

ポイントツーポイントの双方向トレール(Trail)は、双方向ネットワーク(Network) TTP と双方向ネットワーク (Network) TTP の間で成立する事ができる。

全てのタイプのトレール(Trail)については、A End 属性によって指されている端点は Z End 属性に指されているネットワーク端点と関係がある。これは、トラヒックがこれらの管理オブジェクトによって、方向性属性によって示されるような一方向か双方向の方法で表わされるネットワーク端点間に流れる事ができるといった方法によってである。

layerConnectionList 属性は、現在の場合、トレールを構成するサブネットワーク接続およびリンク接続(同じレイヤの中の)をリストする。これは、その構成要素であるサブネットワーク接続およびリンク接続へのトレールの分解の単一の分割された見方を表わす。

このインタフェースは次の条件付パッケージを含んでいる。

- layerConnectionListPackage:同じレイヤにトレールを構築するサブネットワーク接続およびリンク接続のシーケンスを見る要求がある場合は存在する。

- - trafficDescriptorPackage:柔軟な帯域幅割付けがサポートされる場合は存在する

- - clientLinkPointerPackage:このトレールにサポートされるより高いレイヤの中のリンクを見る要求がある場合は存在する。

- - clientLinkConnectionPointerListPackage:このトレールにサポートされるより高いレイヤの中でリンク接続を見る要求がある場合は存在する。

```
*/

        valuetype TrailValueType: PipeValueType
        {
```

```
            public PipeNameSetType           connectionList;
                   // conditional
                   // layerConnectionListPackage
                   // GET, SET-BY-CREATE
            public MONameType          trafficDescriptor;
                   // conditional
                   // trafficDescriptorPackage
                   // GET-REPLACE
            public TopLinkNameSetType  clientLinkPointerList;
                   // conditional
                   // clientLinkPointerPackage
                   // GET
            public LinkConnectionNameSetType
                   clientLinkConnectionPointerList;
                   // conditional
                   // clientLinkConnectionPointerListPackage
                   // GET
      }; // valuetype TrailValueType

      interface Trail: Pipe
      {

/**
次の属性は、同じレイヤの中のトレール(Trail)を構成する与えられたレイヤにリンク接続およびサブネットワーク接
続のリストを定義する。接続インスタンスのこの構成は単純なシーケンスであることができるし、あるいは、マルチポイ
ントの場合には木構造であることができる。
*/

            PipeNameSetType connectionListGet ()
                   raises (itut_x780::ApplicationError,
                          NOlayerConnectionListPackage);

/**
次の属性は、トレールのトラヒックディスクリプタを含んでいる。それは柔軟な帯域幅割付けと共に使用される。
*/

            MONameType trafficDescriptorGet ()
                   raises (itut_x780::ApplicationError,
                          NOtrafficDescriptorPackage);

            void trafficDescriptorSet
                   (in MONameType descriptor)
                   raises (itut_x780::ApplicationError,
                          NewServiceCharacteristicsExistsAlready,
                          NewTrafficDescriptorExistsAlready,
                          InvalidServiceCharacteristicsRequested,
                          InvalidTrafficDescriptorRequested,
                          NOtrafficDescriptorPackage);

/**
次の属性はクライアントレイヤネットワークドメインのトレールの収容能力を反映するトポロジのリンクへのポインタの
集合である。
*/

            TopLinkNameSetType clientLinkPointerListGet ()
                   raises (itut_x780::ApplicationError,
                          NOclientLinkPointerPackage);

/**
トレールの次の属性はトレールでサポートされるクライアントレイヤネットワークドメインのリンク接続へのポインタの
集合である。
*/

            LinkConnectionNameSetType clientLinkConnectionPointerListGet ()
                   raises (itut_x780::ApplicationError,
                          NOclientLinkConnectionPointerListPackage);
```

```
};  // interface Trail


interface TrailFactory: itut_x780::ManagedObjectFactory
{
       itut_x780::ManagedObject create
              (in NameBindingType nameBinding,
              in MONameType superior,
              in string reqID,    // auto naming if empty string
              out MONameType name,
              in StringSetType packageNameList,
              in SignalIdType signalId,
                     // pipeR2Package
                     // GET, SET-BY-CREATE
              in MONameSetType aEndNetworkTPList,
                     // pipeR2Package
                     // GET, SET-BY-CREATE
              in MONameSetType zEndNetworkTPList,
                     // pipeR2Package
                     // GET, SET-BY-CREATE
              in AdministrativeStateType adminstrativeState,
                     // conditional
                     // administrativeStatePackage
                     // GET-REPLACE
              in AlarmSeverityAssignmentProfileNameType profile,
                     // conditional
                     // alarmSeverityAssignmentPointerPackage
                     // GET-REPLACE
              in boolean protected,
                     // conditional
                     // protectedPackage
                     // GET, SET-BY-CREATE
              in MONameSetType supportedByObjectList,
                     // conditional
                     // supportedByPackage
                     // GET-REPLACE, ADD-REMOVE
              in Istring userLabel,
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
              in PipeNameSetType connectionList,
                     // conditional
                     // layerConnectionListPackage
                     // GET, SET-BY-CREATE
              in MONameType trafficDescriptor)
                     // conditional
                     // trafficDescriptorPackage
                     // GET-REPLACE
              raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

}; // interface TrailFactory


/**
```

6.5.17 Trail95

Trail95 は一つ以上の他のレイヤネットワークからの特性情報転送の完全性に責任を負うレイヤネットワークにおける管理オブジェクトクラスである。トレールは二つのトレール端点と一つ以上の接続と関連する接続端点からなる。
[このインタフェースは M.3100 trailR1 オブジェクトに基づいてモデル化している。]

接続方向は a と z 端点の方向性によって決定される。

クラスインスタンスが双方向ならば、a-と z-端点も双方向でなければならない。クラスインスタンスが片方向ならば、a-端点はソース TP、z-端点はシンク TP でなければならない。

操作状態は信号を運ぶ能力を示す。

このインタフェースは以下の条件付パッケージを含む。

- characteristicInformationPackage: インスタンスがサポートするならば存在する。

- protectedPackage: インスタンスがサポートするならば存在する。

- tmnCommunicationsAlarmInformationR1Package: インスタンスが通信警報通知をサポートするならば存在する。

- alarmSeverityAssignmentPointerPackage: インスタンスが通信警報通知をサポートしかつ警報重要度の設定をサポートするならば存在する。

- userLabelPackage:インスタンスがサポートするならば存在する。

- serverConnectionListPackage:インスタンスがサポートするならば存在する。

- clientConnectionListPackage:インスタンスがサポートするならば存在する。
*/

```
      valuetype Trail95ValueType: itut_x780::ManagedObjectValueType
      {
            public DirectionalityType  directionality;
                  // GET
            public AdministrativeStateType   administrativeState;
                  // GET-REPLACE
            public OperationalStateType      operationalState;
                  // GET
            public MONameType          aTP;
                  // GET, SET-BY-CREATE
            public MONameType          zTP;
                  // GET, SET-BY-CREATE
            public CharacteristicInfoType    characteristicInfo;
                  // conditional
                  // characteristicInformationPackage
                  // GET, SET-BY-CREATE
            public boolean                   protected;
                  // conditional
                  // protectedPackage
                  // GET, SET-BY-CREATE
            public AlarmStatusType           alarmStatus;
                  // conditional
                  // tmnCommunicationsAlarmInformationR1Package
                  // GET
            public CurrentProblemSetType     currentProblemList;
                  // conditional
                  // tmnCommunicationsAlarmInformationR1Package
                  // GET
            public AlarmSeverityAssignmentProfileNameType
                  alarmSeverityAssignmentProfilePointer;
                  // conditional
```

```
                    // alarmSeverityAssignmentPointerPackage
                    // GET-REPLACE
             public Istring                    userLabel;
                    // conditional
                    // userLabelPackage
                    // GET-REPLACE
             public MONameSetType       serverConnectionList;
                    // conditional
                    // serverConnectionListPackage
                    // GET, SET-BY-CREATE
             public MONameSetType       clientConnectionList;
                    // conditional
                    // clientConnectionListPackage
                    // GET, SET-BY-CREATE
      }; // valuetype Trail95ValueType


      interface Trail95: itut_x780::ManagedObject
      {
/**
方向性属性は通信が片方向か双方向かを示す。
*/
             DirectionalityType directionalityGet ()
                    raises (itut_x780::ApplicationError);

             AdministrativeStateType administrativeStateGet ()
                    raises (itut_x780::ApplicationError);

             void administrativeStateSet
                    (in AdministrativeStateType adminstrativeState)
                    raises (itut_x780::ApplicationError);

             OperationalStateType operationalStateGet ()
                    raises (itut_x780::ApplicationError);

             MONameType aTPGet ()
                    raises (itut_x780::ApplicationError);

             MONameType zTPGet ()
                    raises (itut_x780::ApplicationError);

/**
特性情報は端点サブクラスのインスタンスの接続可能性を確認するために使われる。
*/
             CharacteristicInfoType characteristicInfoGet ()
                    raises (itut_x780::ApplicationError,
                           NOcharacteristicInformationPackage);

/**
以下の属性は関連する管理オブジェクトが保護されているかどうかを特定する。値 TRUE は保護されることを示す。
*/
             boolean protectedGet ()
                    raises (itut_x780::ApplicationError,
                           NOprotectedPackage);

             AlarmStatusType alarmStatusGet ()
                    raises (itut_x780::ApplicationError,
                           NOtmnCommunicationsAlarmInformationR1Package);

/**
以下のメソッドは管理オブジェクトに関する現存する問題を重要度つきで返す。
*/
             CurrentProblemSetType currentProblemListGet ()
                    raises (itut_x780::ApplicationError,
                           NOtmnCommunicationsAlarmInformationR1Package);

             AlarmSeverityAssignmentProfileNameType
```

```
                alarmSeverityAssignmentProfilePointerGet ()
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

        void alarmSeverityAssignmentProfilePointerSet
                (in AlarmSeverityAssignmentProfileNameType profile)
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

        Istring userLabelGet ()
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        void userLabelSet
                (in Istring userLabel)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

/**
以下の属性値は、同一ネットワークレイヤ内のトレールを構成するために直列に接続された一つ以上の接続オブジェクトを特定する。
*/
        MONameSetType serverConnectionListGet ()
                raises (itut_x780::ApplicationError,
                        NOserverConnectionListPackage);

/**
以下の属性値はトレールによって提供されるクライアント接続を特定する。これらのクライアント接続はトレールの速度よりも低いか同等である。後者の場合、リストは唯一つのクライアント接続から成る。
*/
        MONameSetType clientConnectionListGet ()
                raises (itut_x780::ApplicationError,
                        NOclientConnectionListPackage);

        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, communicationAlarm,
                tmnCommunicationsAlarmInformationR1Package)

};  // interface Trail95


interface Trail95Factory: itut_x780::ManagedObjectFactory
{
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in AdministrativeStateType administrativeState,
                        // GET-REPLACE
                in MONameType aTP,
                        // GET, SET-BY-CREATE
                in MONameType zTP,
                        // GET, SET-BY-CREATE
```

```
                    in CharacteristicInfoType characteristicInfo,
                           // conditional
                           // characteristicInformationPackage
                           // GET, SET-BY-CREATE
                    in boolean protected,
                           // conditional
                           // protectedPackage
                           // GET, SET-BY-CREATE
                    in AlarmSeverityAssignmentProfileNameType profile,
                           // conditional
                           // alarmSeverityAssignmentPointerPackage
                           // GET-REPLACE
                    in Istring userLabel,
                           // conditional
                           // userLabelPackage
                           // GET-REPLACE
                    in MONameSetType serverConnectionList,
                           // conditional
                           // serverConnectionListPackage
                           // GET, SET-BY-CREATE
                    in MONameSetType clientConnectionList)
                           // conditional
                           // clientConnectionListPackage
                           // GET, SET-BY-CREATE
                    raises (itut_x780::ApplicationError,
                           itut_x780::CreateError);

      }; // interface Trail95Factory

/**
```

6.5.18 ソフトウェア（Software）

ソフトウェア管理オブジェクトは装置に格納されているプログラムとデータテーブルといった論理情報を表す。ソフトウェアは他のソフトウェアに入れ子となり得るため包含関係を形成する。

オブジェクトが attributeValueChange 通知をサポートしているとき、属性値変化通知は次のような属性値の一つの変化で発行されなければならない：
警報状態、影響オブジェクト、使用者ラベル、バージョン、現状問題一覧。
以上の属性対応は任意であるため、属性値変化通知発出のための振舞いは管理オブジェクトが対応している場合に限り適用される。オブジェクトが状態変化通知をサポートしているとき、管理状態または操作状態の変化で stateChangeNotification が発行されなければならない。

このインタフェースは M.3100 SoftwareR1 仕様に基づく。

この構造は全てのソフトウェア属性を一回の操作で取り出すのに使われる。サポートされていない属性の多くは空の文字列または空のリストとして返される。しかしながら、空の文字列の受信は属性がサポートされていないということを意味しない。

このインタフェースは以下の条件付パッケージを含む。

- administrativeOperationalStatePackage: インスタンスがサポートするならば存在する。

- affectedObjectListPackage:インスタンスがサポートするならば存在する。

- alarmSeverityAssignmentPointerPackage:インスタンスが警報重要度の設定をサポートするならば存在する。

- softwareProcessingErrorAlarmR2Package:インスタンスがサポートするならば存在する。

- currentProblemListPackage:インスタンスがサポートするならば存在する。

- userLabelPackage:インスタンスがサポートするならば存在する。

インスタンスがサポートするならば存在する。

- versionPackage:インスタンスがサポートするならば存在する。

- arcPackage:インスタンスが警報報告制御をサポートするならば存在する。

- arcRetrieveAlarmDetailPackage:インスタンスがこのパッケージと arcPackage もまたサポートするならば存在する。
*/

```
       valuetype SoftwareValueType: itut_x780::ManagedObjectValueType
       {
              public OperationalStateType      operationalState;
                     // conditional
                     // administrativeOperationalStatePackage
                     // GET
              public AdministrativeStateType   administrativeState;
                     // conditional
                     // administrativeOperationalStatePackage
                     // GET-REPLACE
              public MONameSetType             affectedObjects;
                     // conditional
                     // affectedObjectListPackage
                     // GET
              public AlarmSeverityAssignmentProfileNameType
                     alarmSeverityAssignmentProfilePointer;
                     // conditional
                     // alarmSeverityAssignmentPointerPackage
                     // GET-REPLACE
              public AlarmStatusType           alarmStatus;
                     // must present if arcPackage presented
                     // conditional
                     // softwareProcessingErrorAlarmR2Package
                     // GET
              public CurrentProblemSetType     currentProblemList;
                     // must present if arcPackage presented
                     // conditional
                     // currentProblemListPackage
                     // GET
              public Istring                   userLabel;
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
              public Istring                   vendorName;
                     // conditional
                     // vendorNamePackage
                     // GET-REPLACE
              public Istring                   version;
                     // conditional
                     // versionPackage
                     // GET-REPLACE
              public ArcStateType        arcState;
                     // conditional
                     // arcPackage
                     // GET
              public ArcQIStatusType           arcQIStatus;
                     // conditional
                     // arcPackage
                     // GET
              public ArcProbableCauseSetType   arcProbableCauseList;
                     // conditional
                     // arcPackage
                     // GET-REPLACE, ADD-REMOVE
              public ArcIntervalProfileNameType arcIntervalProfilePointer;
                     // conditional
                     // arcPackage
                     // GET-REPLACE
              public ArcTimeType         arcManagementRequestedInterval;
                     // conditional
```

```
                // arcPackage
                // GET-REPLACE
        public ArcTimeType         arcTimeRemaining;
                // conditional
                // arcPackage
                // GET
};  // valuetype SoftwareValueType


interface Software: itut_x780::ManagedObject
{
        OperationalStateType operationalStateGet ()
                raises (itut_x780::ApplicationError,
                        NOadministrativeOperationalStatesPackage);

        AdministrativeStateType administrativeStateGet ()
                raises (itut_x780::ApplicationError,
                        NOadministrativeOperationalStatesPackage);

        void administrativeStateSet
                (in AdministrativeStateType administrativeState)
                raises (itut_x780::ApplicationError,
                        NOadministrativeOperationalStatesPackage);

/**
影響オブジェクトリスト属性は状態の変化または管理オブジェクト削除によって、直接影響を受けるオブジェクトインスタンスを規定する。属性は内部詳細を規定するのではなく、管理に必要とされるレベルの詳細を規定する。
*/
        MONameSetType affectedObjectsGet ()
                raises (itut_x780::ApplicationError,
                        NOaffectedObjectListPackage);

/**
以下のメソッドは警報重要度割当プロファイルポインタを取り出すのに使用される。もし警報重要度プロファイルポインタが空であるなら、次の二つの内の一つの選択肢が適用される：
a) 管理システムが重要度を割り当てる、または
b) ‘不定’の値を使用する。
*/
        AlarmSeverityAssignmentProfileNameType
                alarmSeverityAssignmentProfilePointerGet ()
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

/**
以下のメソッドは警報重要度プロファイルポインタを設定するのに使用する。
*/
        void alarmSeverityAssignmentProfilePointerSet
                (in AlarmSeverityAssignmentProfileNameType profile)
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

/**
以下のメソッドはオブジェクトの現在警報状態を返す。
*/
        AlarmStatusType alarmStatusGet ()
                raises (itut_x780::ApplicationError,
                        NOsoftwareProcessingErrorAlarmR2Package);

/**
以下のメソッドは管理オブジェクトに関連する現存問題を重要度付きで返す。
*/
        CurrentProblemSetType currentProblemListGet ()
                raises (itut_x780::ApplicationError,
                        NOcurrentProblemListPackage);

/**
```

```
以下のメソッドは、オブジェクトを特定するために管理システムによって使われ得るラベルを返す。
*/
              Istring userLabelGet ()
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

/**
以下のメソッドはインスタンスにラベルを割り当てるために使用される。その値はオブジェクトではなく、クライアントにとって重要であり得る。この属性がサポートされるなら、このオブジェクトによって発出される通知に含まれるべきである。
*/
              void userLabelSet
                     (in Istring userLabel)
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

/**
以下のメソッドは関連する資源供給者の名前を返す。
*/
              Istring vendorNameGet ()
                     raises (itut_x780::ApplicationError,
                            NOvendorNamePackage);

/**
以下のメソッドは関連する資源供給者の名前を設定する。
*/
              void vendorNameSet
                     (in Istring vendorName)
                     raises (itut_x780::ApplicationError,
                            NOvendorNamePackage);

/**
以下のメソッドは関連する資源のバージョンを返す。
*/
              Istring versionGet ()
                     raises (itut_x780::ApplicationError,
                            NOversionPackage);

/**
以下のメソッドは関連する資源のバージョンを設定する。
*/
              void versionSet
                     (in Istring version)
                     raises (itut_x780::ApplicationError,
                            NOadministrativeOperationalStatesPackage);

/**
状態変化通知はこの属性の値変化を示すために使用される。
*/
              ArcStateType arcStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOarcPackage);

/**
状態変化通知も属性変化通知もまたこの属性の値変化を示すことに使用されてはならない。
*/
              ArcQIStatusType     arcQIStatusGet ()
                     raises (itut_x780::ApplicationError,
                            NOarcPackage);

              ArcProbableCauseSetType    arcProbableCauseListGet ()
                     raises (itut_x780::ApplicationError,
                            NOarcPackage);

              void arcProbableCauseListSet
                     (in ArcProbableCauseSetType arcProbableCauseList)
```

```
                raises (itut_x780::ApplicationError,
                       NotSupportedProbableCause,
                       NOarcPackage);

            void arcProbableCauseListAdd
                   (in ArcProbableCauseSetType arcProbableCauseList)
                   raises (itut_x780::ApplicationError,
                          NotSupportedProbableCause,
                          NOarcPackage);

            void arcProbableCauseListRemove
                   (in ArcProbableCauseSetType arcProbableCauseList)
                   raises (itut_x780::ApplicationError,
                          NOarcPackage);

/**
以下のメソッドは関連する ARC 間隔プロファイルオブジェクトを特定する ARC 間隔プロファイルポインタ値の取り出しに使用される。この属性値は、設定可能な持続間隔と時限間隔が使用されないならば、空である（即ち、nalm-qi と nalm-ti 状態に使用されない）。
*/
            ArcIntervalProfileNameType arcIntervalProfilePointerGet ()
                   raises (itut_x780::ApplicationError,
                          NOarcPackage);

/**
以下のメソッドは関連する ARC 間隔プロファイルオブジェクトを特定する ARC 間隔プロファイルポインタ値の変更に使用される。この属性値は、設定可能な持続間隔と時限間隔が使用されないならば、空である（すなわち、nalm-qi と nalm-ti 状態に使用されない）。
*/
            void arcIntervalProfilePointerSet
                   (in ArcIntervalProfileNameType profile)
                   raises (itut_x780::ApplicationError,
                          NOarcPackage);

/**
以下のメソッドは、ARC 間隔の管理要求時間を特定する ARC 管理要求間隔値を取り出すのに使用される。この属性は、管理要求によるかまたは資源が自動的に警報状態に移行する時にのみ値を変える。この属性値を変える管理要求はそうすることが無効である場合拒否される。例えば、管理資源が alm または nalm 状態のとき。この属性値は ARC 間隔がある瞬間に管理要求を経由して調節可能か否かを反映する。
*/
            ArcTimeType arcManagementRequestedIntervalGet ()
                   raises (itut_x780::ApplicationError,
                          NOarcPackage);

/**
以下のメソッドは ARC 間隔の管理要求時間を特定する ARC 管理要求間隔値の変更に使用される。
*/
            void arcManagementRequestedIntervalSet
                   (in ArcTimeType time)
                   raises (itut_x780::ApplicationError,
                          NOarcPackage);

/**
以下のメソッドは、ARC 間隔(即ち、nalm-qi 状態の持続間隔や、nalm-ti 状態の 時間間 隔)の残りの時間を検索するために利用される。それは必ずしもその状態の残り時間を 示すというわけではないことに注意せよ。例えば、arcTimeRemaining が nalm-qi 状態に おいて 30 分であるとして、ARC 間隔 タイマが期間が切れる前に条件付の問題が管理された資源のために上げられるなら、それは、再度、条件付の問題が解消し、タイマを再起動して、再び残りの時間を減少させ始めるまで、タイマを中止して無期限に待つことになる。nalm-ti、nalm、または nalm-qi 状態に資源の遷移があるとき、この属性値は管理要求間隔に初期化される。動作中のタイマが全く無いとき、その値はタイマが動作していないことを示す (即ち、時間調整が全く行われない)。何れの属性値の変化通知も、この属性値における変化のために送られないことに注意せよ。
*/
            ArcTimeType arcTimeRemainingGet ()
                   raises (itut_x780::ApplicationError,
                          NOarcPackage);
```

```
/**
以下のメソッドは警報報告の作動または停止いずれにも使用される。これは望ましい ARC 状態の特定によって達成される。場合によっては与えられた資源タイプでは状態がサポートされないという理由で、動作は拒否される。状態を特定することに加えて、マネージャは一回の使用に対して既定値以外の arcInterval を要求する。このような書き換えは nalm-qi と naml-ti 状態へ遷移するときのみ適用される。
*/
            boolean arcControl
                  (in ArcControlRequestType request)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
以下の操作は宣言された警報状態に対して警報詳細を返す。この機能をサポートするために、インタフェースは arcPackage もまたサポートしなければならない。
*/
            ArcAlarmDetailSetType arcRetrieveAlarmDetail ()
                  raises (itut_x780::ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, procesingErrorAlarm,
                  softwareProcessingErrorAlarmR2Package)

      }; // interface Software


      interface SoftwareFactory: itut_x780::ManagedObjectFactory
      {
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID,    // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList,
                  in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeOperationalStatePackage
                        // GET-REPLACE
                  in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                  in Istring userLabel,
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                  in Istring vendorName,
                        // conditional
                        // vendorNamePackage
                        // GET-REPLACE
                  in Istring version,
                        // conditional
                        // versionPackage
                        // GET-REPLACE
```

```
                    in ArcProbableCauseSetType arcProbableCauseList,
                           // conditional
                           // arcPackage
                           // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                           // conditional
                           // arcPackage
                           // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval)
                           // conditional
                           // arcPackage
                           // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                           itut_x780::CreateError);
      }; // SoftwareFactory

/**
```

6.5.19 サブネットワーク（Subnetwork）

サブネットワークインタフェースをサポートする管理オブジェクトはネットワーク端点の論理的な集まりを表す。

もし包含されるサブネットワークが存在しなければ、ContainedSubnetworkList 属性は空となる。もし包含する（親の）サブネットワークが存在しない場合は、ContainedInSubnetworkList 属性もまた空となる。

このインタフェースは以下の条件付パッケージを含む。

- availabilityStatusPackage:インスタンスがサポートするならば存在する。

- containedAccessGroupListPackage: アクセスグループインスタンスがサブネットワークに含まれるならば存在する。

- containedInSubnetworkListPackage:サブネットワークオブジェクトインスタンスがサブネットワーク（分割がサポートされている）に含まれるならば存在する。

- containedLinkEndListPackage:サブネットワークオブジェクトインスタンス（分割がサポートされている）に含まれるリンク端インスタンスがあるならば存在する。

- containedLinkListPackage: サブネットワークオブジェクトインスタンス（分割がサポートされている）に含まれるリンク端インスタンスがあるならば存在する。

このパッケージはサブネットワークが分割を通して含むリンクを特定する。

リンクはポリシによって許されるならば、集約サブネットワークとは異なる layerNetworkDomain （互換性のある信号識別を持つ異なる networkR1 管理領域に関連付けられた）から指定され得る。

- containedNetworkTPListPackage:サブネットワークオブジェクトインスタンスに含まれるネットワーク端点があるならば存在する。

- containedSubnetworkListPackage: サブネットワークオブジェクトインスタンス（分割がサポートされている）に含まれるサブネットワークがあるならば存在する。

このパッケージは集約サブネットワークが分割を通して包含する要素サブネットワークを示す。

- linkPointerListPackage: リンク、サブネットワーク、アクセスグループを用いたトポロジ的な見え方がサポートされるならば存在する。

- supportedByPackage:インスタンスがサポートするならば存在する。

- administrativeOperationalStatePackage:インスタンスがサポートするならば存在する。

- usageStatePackage:インスタンスがサポートするならば存在する。

```
- userLabelPackage:インスタンスがサポートするならば存在する。
*/
      valuetype SubnetworkValueType: itut_x780::ManagedObjectValueType
      {
            public SignalIdType        signalId;
                  // GET, SET-BY-CREATE
            public AvailabilityStatusSetType availabilityStatus;
                  // conditional
                  // availabilityStatusPackage
                  // GET
            public AccessGroupNameSetType    containedAccessGroupList;
                  // conditional
                  // containedAccessGroupListPackage
                  // GET-REPLACE, ADD-REMOVE
            public SubnetworkNameSetType     containedInSubnetworkList;
                  // conditional
                  // containedInSubnetworkListPackage
                  // GET-REPLACE, ADD-REMOVE
            public AbstractLinkEndNameSetType containedLinkEndList;
                  // conditional
                  // containedLinkEndListPackage
                  // GET-REPLACE, ADD-REMOVE
            public AbstractLinkNameSetType   containedLinkList;
                  // conditional
                  // containedLinkListPackage
                  // GET-REPLACE, ADD-REMOVE
            public NetworkTPNameSetType      containedNetworkTPList;
                  // conditional
                  // containedNetworkTPListPackage
                  // GET-REPLACE, ADD-REMOVE
            public SubnetworkNameSetType     containedSubnetworkList;
                  // conditional
                  // containedSubnetworkListPackage
                  // GET-REPLACE, ADD-REMOVE
            public AbstractLinkNameSetType   linkPointerList;
                  // conditional
                  // linkPointerListPackage
                  // GET
            public MONameSetType       supportedByObjectList;
                  // conditional
                  // supportedByPackage
                  // GET-REPLACE, ADD-REMOVE
            public AdministrativeStateType   adminstrativeState;
                  // conditional
                  // administrativeOperationalStatePackage
                  // GET-REPLACE
            public OperationalStateType      operationalState;
                  // conditional
                  // administrativeOperationalStatePackage
                  // GET
            public UsageStateType            usageState;
                  // conditional
                  // usageStatePackage
                  // GET
            public Istring                   userLabel;
                  // conditional
                  // userLabelPackage
                  // GET-REPLACE
      }; // valuetype SubnetworkValueType


      interface Subnetwork : itut_x780::ManagedObject
      {
/**
```

```
以下の属性は、議論しているエンティティが属するレイヤ（G.805 の意味合いで）の特性情報を定義する。サブネットワーク接続／接続性が可能であるか否かの決定に使用される。信号 Id は単純な速度と形式であるかまたは集約信号を形成する同一の特性情報を持つエンティティの束であり得る。
*/
              SignalIdType signalIdGet ()
                     raises (itut_x780::ApplicationError);

              AvailabilityStatusSetType availabilityStatusGet ()
                     raises (itut_x780::ApplicationError,
                            NOavailabilityStatusPackage);

/**
以下の属性は、サブネットワークに含まれるアクセスグループインスタンスリストを定義する。
*/
              AccessGroupNameSetType containedAccessGroupListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedAccessGroupListPackage);

              void containedAccessGroupListSet
                     (in AccessGroupNameSetType accessGroupList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedAccessGroupListPackage);

              void containedAccessGroupListAdd
                     (in AccessGroupNameSetType accessGroupList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedAccessGroupListPackage);

              void containedAccessGroupListRemove
                     (in AccessGroupNameSetType accessGroupList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedAccessGroupListPackage);

/**
以下の属性は与えられたレイヤのサブネットワークを含む親のサブネットワークリストを定義する。このサブネットワークオブジェクトインスタンスがサブネットワーク（分割がサポートされている）に含まれるならばサポートされる。

要素サブネットワークは、ポリシによって許されるならば、集約サブネットワークよりも異なる layerNetworkDomain（互換性のある信号識別を持つ異なる networkR1 管理領域に関連付けられた）から指定され得る。
*/
              SubnetworkNameSetType containedInSubnetworkListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListSet
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListAdd
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

              void containedInSubnetworkListRemove
                     (in SubnetworkNameSetType containedInSubnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedInSubnetworkListPackage);

/**
以下の属性は（与えられたレイヤにおける）透視的観点からサブネットワークの内部トポロジを記述するのに使用される。このトポロジはリンク端とサブネットワークから成る。リンク端はこの属性に記載される。
*/
              AbstractLinkEndNameSetType containedLinkEndListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedLinkEndListPackage);
```

```
                void containedLinkEndListSet
                       (in AbstractLinkEndNameSetType linkEndList)
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkEndListPackage);

                void containedLinkEndListAdd
                       (in AbstractLinkEndNameSetType linkEndList)
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkEndListPackage);

                void containedLinkEndListRemove
                       (in AbstractLinkEndNameSetType linkEndList)
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkEndListPackage);

/**
以下の属性は（与えられたレイヤの）サブネットワークの内部トポロジを記述するのに使用される。このトポロジはリンクとサブネットワークから構成される。リンクはこの属性に記載される。
*/
                AbstractLinkNameSetType containedLinkListGet ()
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkListPackage);

                void containedLinkListSet
                       (in AbstractLinkNameSetType linkList)
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkListPackage);

                void containedLinkListAdd
                       (in AbstractLinkNameSetType linkList)
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkListPackage);

                void containedLinkListRemove
                       (in AbstractLinkNameSetType linkList)
                       raises (itut_x780::ApplicationError,
                              NOcontainedLinkListPackage);

/**
以下の属性はサブネットワークに含まれるネットワーク TP へのポインタリストである。
*/
                NetworkTPNameSetType containedNetworkTPListGet ()
                       raises (itut_x780::ApplicationError,
                              NOcontainedNetworkTPListPackage);

                void containedNetworkTPListSet
                       (in NetworkTPNameSetType networkTPList)
                       raises (itut_x780::ApplicationError,
                              NetworkTTPAndSubnetworkNotCompatible,
                              FailureToAssociateNetworkTTP,
                              FailureToDisassociateNetworkTTP,
                              NOcontainedNetworkTPListPackage);

                void containedNetworkTPListAdd
                       (in NetworkTPNameSetType networkTPList)
                       raises (itut_x780::ApplicationError,
                              NetworkTTPAndSubnetworkNotCompatible,
                              FailureToAssociateNetworkTTP,
                              NOcontainedNetworkTPListPackage);

                void containedNetworkTPListRemove
                       (in NetworkTPNameSetType networkTPList)
                       raises (itut_x780::ApplicationError,
                              FailureToDisassociateNetworkTTP,
                              NOcontainedNetworkTPListPackage);
```

```
/**
以下の属性は（与えられたレイヤの）サブネットワークの内部トポロジを記述するのに使用される。このトポロジはリンクとサブネットワークから構成される。サブネットワークはこの属性に記載される。
*/
              SubnetworkNameSetType containedSubnetworkListGet ()
                     raises (itut_x780::ApplicationError,
                            NOcontainedSubnetworkListPackage);

              void containedSubnetworkListSet
                     (in SubnetworkNameSetType subnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedSubnetworkListPackage);

              void containedSubnetworkListAdd
                     (in SubnetworkNameSetType subnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedSubnetworkListPackage);

              void containedSubnetworkListRemove
                     (in SubnetworkNameSetType subnetworkList)
                     raises (itut_x780::ApplicationError,
                            NOcontainedSubnetworkListPackage);

/**
以下の属性はサブネットワークの終端するリンクまたはアクセスグループの終端するリンクを指す。
*/
              AbstractLinkNameSetType linkPointerListGet ()
                     raises (itut_x780::ApplicationError,
                            NOlinkPointerListPackage);

              MONameSetType supportedByObjectListGet ()
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              void supportedByObjectListSet
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              void supportedByObjectListAdd
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              void supportedByObjectListRemove
                     (in MONameSetType objectList)
                     raises (itut_x780::ApplicationError,
                            NOsupportedByPackage);

              AdministrativeStateType administrativeStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOadministrativeOperationalStatesPackage);

              void administrativeStateSet
                     (in AdministrativeStateType administrativeState)
                     raises (itut_x780::ApplicationError,
                            NOadministrativeOperationalStatesPackage);

              OperationalStateType operationalStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOadministrativeOperationalStatesPackage);

              UsageStateType usageStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOusageStatePackage);

              Istring userLabelGet ()
```

```
            raises (itut_x780::ApplicationError,
                  NOuserLabelPackage);

      void userLabelSet
            (in Istring userLabel)
            raises (itut_x780::ApplicationError,
                  NOuserLabelPackage);

      MANDATORY_NOTIFICATION(
            itut_x780::Notifications, objectCreation)
      MANDATORY_NOTIFICATION(
            itut_x780::Notifications, objectDeletion)
      CONDITIONAL_NOTIFICATION(
            itut_x780::Notifications, attributeValueChange,
            attributeValueChangeNotificationPackage)
      CONDITIONAL_NOTIFICATION(
            itut_x780::Notifications, stateChange,
            stateChangeNotificationPackage)

}; // interface Subnetwork

interface SubnetworkFactory: itut_x780::ManagedObjectFactory
{
      itut_x780::ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID,    // auto naming if empty string
            out MONameType name,
            in StringSetType packageNameList,
            in SignalIdType signalId,
                  // subnetworkPackage
                  // GET, SET-BY-CREATE
            in AccessGroupNameSetType containedAccessGroupList,
                  // conditional
                  // containedAccessGroupListPackage
                  // GET-REPLACE, ADD-REMOVE
            in SubnetworkNameSetType containedInSubnetworkList,
                  // conditional
                  // containedInSubnetworkListPackage
                  // GET-REPLACE, ADD-REMOVE
            in AbstractLinkEndNameSetType containedLinkEndList,
                  // conditional
                  // containedLinkEndListPackage
                  // GET-REPLACE, ADD-REMOVE
            in AbstractLinkNameSetType containedLinkList,
                  // conditional
                  // containedLinkListPackage
                  // GET-REPLACE, ADD-REMOVE
            in NetworkTPNameSetType containedNetworkTPList,
                  // conditional
                  // containedNetworkTPListPackage
                  // GET-REPLACE, ADD-REMOVE
            in SubnetworkNameSetType containedSubnetworkList,
                  // conditional
                  // containedSubnetworkListPackage
                  // GET-REPLACE, ADD-REMOVE
            in MONameSetType supportedByObjectList,
                  // conditional
                  // supportedByPackage
                  // GET-REPLACE, ADD-REMOVE
            in AdministrativeStateType adminstrativeState,
                  // conditional
                  // administrativeOperationalStatePackage
                  // GET-REPLACE
            in Istring userLabel)
                  // conditional
                  // userLabelPackage
                  // GET-REPLACE
```

```
                    raises (itut_x780::ApplicationError,
                           itut_x780::CreateError,
                           FailureToAssociateNetworkTTP,
                           FailureToCreateNetworkTTP,
                           FailureToCreateSubnetwork);

             // DELETE_ERROR:
             //     SubnetworkInUse
             //     BoundSubnetwork
             //     FailureToRemoveSubnetwork

      }; // interface SubnetworkFactory

/**
```

6.5.19.1 BasicSubnetwork

BasicSubnetwork インタフェースはマネージャの制御下で、サブネットワーク接続の設定と解放を管理する管理オブジェクトクラスである。

基本接続実行（Basic Connection Performer）パッケージは基本接続設定機能を提供する。
SetupSubNetworkConnection 動作はサブネットワーク接続の設定を行い、releaseSubNetworkConnection はサブネットワーク接続を解除する。

```
*/
      valuetype BasicSubnetworkValueType: SubnetworkValueType
      {
      }; // valuetype BasicSubnetworkValueType


      interface BasicSubnetwork: Subnetwork
      {
/**
```

以下の動作はネットワーク端点間のサブネットワーク接続の設定に使用される。

もしリンクエンドがサブネットワーク接続に関わるのであれば、それの属性である idleNWCTPCount と connectedNWCTPCount は本動作の結果として更新される。

```
*/
             void setupSnc
                    (in ConnectivityEndPointSetType aEnd,
                    in ConnectivityEndPointSetType zEnd,
                    in DirectionalityType direction,
                    in SignalIdType signalId, // may be empty sequence
                    in MONameType qos, // may be empty
                    in boolean implicitTPCreation,
                    out SNCNameType connection,
                    out ConnectivityEndPointType resultAEnd,
                    out ConnectivityEndPointType resultZEnd,
                    out Istring userId) // may be empty string
                    raises (itut_x780::ApplicationError,
                           InvalidTransportServiceCharacteristics,
                           IncorrectSubnetworkTerminationPoints,
                           AEndNetworkTPConnected,
                           ZEndNetworkTPConnected,
                           WrongAEndDirectionality,
                           WrongZEndDirectionality,
                           FailureToConnect,
                           FailureToSetUserIdentifier,
                           NObasicConnectionPerformerPackage);

/**
```

以下の動作はサブネットワーク接続の開放に使用される。compositePointer 属性によって指定されるサブネットワーク接続もまたこの動作によってクリアされる。もしリンク端がサブネットワーク接続に関わるのであれば、それの属性である idleNWCTPCount と connectedNWCTPCount は本動作の結果として更新される。もし TP 生成が暗黙的に使用されると、関連する TP はサブネットワーク接続が開放されるときに削除される。

```
*/
```

```
        void releaseSnc
                (in SNCNameType connection,
                in Istring userId) // may be empty string
                raises (itut_x780::ApplicationError,
                        NoSuchSnc,
                        SncConnected,
                        FailureToRelease,
                        NObasicConnectionPerformerPackage);

    }; // interface BasicSubnetwork

    interface BasicSubnetworkFactory: itut_x780::ManagedObjectFactory
    {
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in SignalIdType signalId,
                        // subnetworkPackage
                        // GET, SET-BY-CREATE
                in AccessGroupNameSetType containedAccessGroupList,
                        // conditional
                        // containedAccessGroupListPackage
                        // GET-REPLACE, ADD-REMOVE
                in SubnetworkNameSetType containedInSubnetworkList,
                        // conditional
                        // containedInSubnetworkListPackage
                        // GET-REPLACE, ADD-REMOVE
                in AbstractLinkEndNameSetType containedLinkEndList,
                        // conditional
                        // containedLinkEndListPackage
                        // GET-REPLACE, ADD-REMOVE
                in AbstractLinkNameSetType containedLinkList,
                        // conditional
                        // containedLinkListPackage
                        // GET-REPLACE, ADD-REMOVE
                in NetworkTPNameSetType containedNetworkTPList,
                        // conditional
                        // containedNetworkTPListPackage
                        // GET-REPLACE, ADD-REMOVE
                in SubnetworkNameSetType containedSubnetworkList,
                        // conditional
                        // containedSubnetworkListPackage
                        // GET-REPLACE, ADD-REMOVE
                in MONameSetType supportedByObjectList,
                        // conditional
                        // supportedByPackage
                        // GET-REPLACE, ADD-REMOVE
                in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeOperationalStatePackage
                        // GET-REPLACE
                in Istring userLabel)
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError,
                        FailureToAssociateNetworkTTP,
                        FailureToCreateNetworkTTP,
                        FailureToCreateSubnetwork);

    }; // interface BasicSubnetworkFactory


/**
```

6.5.20 TP (端点（Termination Point）)

TP (端点)管理オブジェクトは伝送エンティティ、例えばトレールまたは接続、の終端を表す。特性情報属性は、クロスコネクションまたは接続が可能か否かを決定するために端点サブクラス間の同等性特定に使われる。操作状態は正当な信号を生成し、かつまた受信する感知される能力を反映する。端点サブクラスは属性値変化通知と状態変化通知が生成される属性と状態を規定すべきである。

TP インタフェースはインスタンス化可能ではない。

このインタフェースは以下の条件付パッケージを含む。

- operationalStatePackage:インスタンスがサポートするならば存在する。

- crossConnectionPointerPackage:TP が柔軟に割り当てられている（即ち、クロスコネクトされている）ならば存在する。

- characteristicInformationPackage:インスタンスがサポートするならば存在する。

- networkLevelPackage:インスタンスがサポートするならば存在する。

- tmnCommunicationsAlarmInformationR1Package: TP が通信警報通知をサポートするならば存在する。

- alarmSeverityAssignmentPointerPackage:インスタンスが警報重要度の設定をサポートするならば存在する。

- arcPackage:インスタンスが警報報告制御をサポートするならば存在する。

- ArcRetrieveAlarmDetailPackage:インスタンスがこれをサポートし、arcPackage もインスタンスによってサポートされるならば存在する。
*/

```
       valuetype TPValueType: itut_x780::ManagedObjectValueType
       {
              public MONameSetType       supportedByObjectList;
                     // GET
              public OperationalStateType      operationalState;
                     // conditional
                     // operationalStatePackage
                     // GET
              public CrossConnectionPointerType crossConnectionPointer;
                     // conditional
                     // crossConnectionPointerPackage
                     // GET
              public CharacteristicInfoType    characteristicInfo;
                     // conditional
                     // characteristicInformationPackage
                     // GET, SET-BY-CREATE
              public MONameType          networkLevelPointer;
                     // conditional
                     // networkLevelPackage
                     // GET-REPLACE
              public AlarmStatusType           alarmStatus;
                     // conditional
                     // tmnCommunicationsAlarmInformationR1Package
                     // GET
              public CurrentProblemSetType     currentProblemList;
                     // conditional
                     // tmnCommunicationsAlarmInformationR1Package
                     // GET
              public AlarmSeverityAssignmentProfile
                     alarmSeverityAssignmentProfilePointer;
                     // conditional
                     // alarmSeverityAssignmentPointerPackage
                     // GET-REPLACE
```

```
            public ArcStateType        arcState;
                  // conditional
                  // arcPackage
                  // GET
            public ArcQIStatusType           arcQIStatus;
                  // conditional
                  // arcPackage
                  // GET
            public ArcProbableCauseSetType   arcProbableCauseList;
                  // conditional
                  // arcPackage
                  // GET-REPLACE, ADD-REMOVE
            public ArcIntervalProfileNameType arcIntervalProfilePointer;
                  // conditional
                  // arcPackage
                  // GET-REPLACE
            public ArcTimeType         arcManagementRequestedInterval;
                  // conditional
                  // arcPackage
                  // GET-REPLACE
            public ArcTimeType         arcTimeRemaining;
                  // conditional
                  // arcPackage
                  // GET
      }; // valuetype TPValueType


      interface TP : itut_x780::ManagedObject
      {
/**
サポーティッドバイリスト（Supported By List） は、与えられた管理オブジェクトに直接的に影響を与えることのできるオブジェクトインスタンスの集合を特定する。オブジェクトインスタンスには物理及び論理オブジェクトともに含まれる。この属性は内部詳細を規定するのではなく、管理に必要とされるレベルの詳細を規定する。管理オブジェクトをサポートするオブジェクトインスタンスがそのオブジェクトにとって未知であるなら、この属性は空のリストである。
*/
            MONameSetType supportedByObjectListGet ()
                  raises (itut_x780::ApplicationError);

            OperationalStateType operationalStateGet ()
                  raises (itut_x780::ApplicationError,
                        NOoperationalStatePackage);

/**
以下のメソッドは端点が終点であるクロスコネクションへの参照を返す。もし TP が接続に関わらないならば、空が返される。
*/
            CrossConnectionPointerType crossConnectionPointerGet ()
                  raises (itut_x780::ApplicationError,
                        NOcrossConnectionPointerPackage);

/**
特性情報は端点サブクラスインスタンスの接続可能性の確認に使用される。
*/
            CharacteristicInfoType characteristicInfoGet ()
                  raises (itut_x780::ApplicationError,
                        NOcharacteristicInformationPackage);

/**
ネットワークレベルポインタはネットワークレベルオブジェクトを特定する。ネットワークレベルポインタ値は管理システムによってのみ修正される。
*/
            MONameType networkLevelPointerGet ()
                  raises (itut_x780::ApplicationError,
                        NOnetworkLevelPackage);

            void networkLevelPointerSet
                  (in MONameType networkLevelPointer)
```

```
                raises (itut_x780::ApplicationError,
                        NOnetworkLevelPackage);

        AlarmStatusType alarmStatusGet ()
                raises (itut_x780::ApplicationError,
                        NOtmnCommunicationsAlarmInformationR1Package);

/**
以下のメソッドは管理オブジェクトに関連する現存する問題を重要度付きで返す。
*/
        CurrentProblemSetType currentProblemListGet ()
                raises (itut_x780::ApplicationError,
                        NOtmnCommunicationsAlarmInformationR1Package);

/**
もし警報重要度割当プロファイルポインタが空であるなら、警報報告時に以下の二つの選択が適用される。
a) 被管理システムが重要度を割り当てる、または
b）'不定'の値が使用される。
*/
        AlarmSeverityAssignmentProfile
                alarmSeverityAssignmentProfilePointerGet ()
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

        void alarmSeverityAssignmentProfilePointerSet
                (in AlarmSeverityAssignmentProfile profile)
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

/**
状態変化通知はこの属性値変化を示すのに使用されなくてはならない。
*/
        ArcStateType arcStateGet ()
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
状態変化通知、属性値変化通知のどちらもこの属性の値変化を示すために使用されるべきではない。
*/
        ArcQIStatusType     arcQIStatusGet ()
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

        ArcProbableCauseSetType    arcProbableCauseListGet ()
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

        void arcProbableCauseListSet
                (in ArcProbableCauseSetType arcProbableCauseList)
                raises (itut_x780::ApplicationError,
                        NotSupportedProbableCause,
                        NOarcPackage);

        void arcProbableCauseListAdd
                (in ArcProbableCauseSetType arcProbableCauseList)
                raises (itut_x780::ApplicationError,
                        NotSupportedProbableCause,
                        NOarcPackage);

        void arcProbableCauseListRemove
                (in ArcProbableCauseSetType arcProbableCauseList)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```

以下のメソッドは関連する ARC 間隔プロファイルオブジェクトを特定する ARC 間隔プロファイルポインタ値の取り出しに使用する。この属性値は設定可能な持続間隔と時間間隔が使用されない（即ち、nalm-qi と nalm-ti 状態に使用されない）場合は空である。
*/
```
            ArcIntervalProfileNameType arcIntervalProfilePointerGet ()
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```
以下のメソッドは関連する ARC 間隔プロファイルオブジェクトを特定する ARC 間隔プロファイルポインタ 値の変更に使用される。この属性値は、設定可能な持続間隔と時間間隔が使用されない（即ち、nalm-qi と nalm-ti 状態に使用されない）ならば、空である。
*/
```
            void arcIntervalProfilePointerSet
                  (in ArcIntervalProfileNameType profile)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```
以下のメソッドは、ARC 間隔の管理要求時間を特定する ARC 管理要求間隔値を取り出すのに使用される。この属性は、管理要求によるかまたは資源が自動的に警報状態に移行する時にのみ値を変える。この属性値を変える管理要求はそうすることが無効である場合拒否される。例えば、管理資源が alm または nalm 状態のとき。この属性値は ARC 間隔がある瞬間に管理要求を通じて調節可能か否かを反映する。
*/
```
            ArcTimeType arcManagementRequestedIntervalGet ()
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```
以下のメソッドは ARC 間隔の管理要求時間を特定する ARC 管理要求間隔値の変更に使用される。
*/
```
            void arcManagementRequestedIntervalSet
                  (in ArcTimeType time)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```
以下のメソッドは、ARC 間隔(即ち、nalm-qi 状態の持続間隔や、nalm-ti 状態の 時間間 隔)の残りの時間を検索するために利用される。それは必ずしもその状態の残り時間を 示すというわけではないことに注意せよ。例えば、arcTimeRemaining が nalm-qi 状態に おいて 30 分であるとして、ARC 間隔 タイマが期間が切れる前に条件付の問題が管理さ れた資源のために上げられるな ら、それは、再度、条件付の問題が解消し、タイマを 再起動して、再び残りの時間 を減少させ始めるまで、タイマを中止して無期限に待つ ことになる。nalm-ti、nalm、または nalm-qi 状態に資源の遷移があるとき、この属性 値は管 理要求間隔に初期化される。動作中のタイマが全く無いとき、その値はタイマ が動作していないことを示す (即ち、時間調整が全く行われない)。何れの属性値の変 化通知も、この属性値における変化のために送られないこと に注意せよ。
*/
```
            ArcTimeType arcTimeRemainingGet ()
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```
以下のメソッドは警報報告の作動または停止いずれにも使用される。これは望ましい ARC 状態の特定によって達成される。場合によっては与えられた資源タイプでは状態がサポートされないという理由で、動作は拒否される。状態を特定することに加えて、マネージャは一回の使用に対して既定値以外の arcInterval を要求する。このような書き換えは nalm-qi と naml-ti 状態へ遷移するときのみ適用される。
*/
```
            boolean arcControl
                  (in ArcControlRequestType request)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

/**
```
以下の操作は宣言された警報状態に対して警報詳細を返す。この機能をサポートするために、インタフェースは arcPackage もまたサポートしなければならない。

```
*/
              ArcAlarmDetailSetType arcRetrieveAlarmDetail ()
                     raises (itut_x780::ApplicationError,
                            NOarcRetrieveAlarmDetailPackage);

              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, objectCreation,
                     createDeleteNotificationsPackage)
              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion,
                     createDeleteNotificationsPackage)
              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, attributValueChange,
                     attributeValueChangeNotificationPackage)
              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, stateChange,
                     stateChangeNotificationPackage)
              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, communicationAlarm,
                     tmnCommunicationsAlarmInformationR1Package)

       }; // interface TP

/**
```

6.5.20.1　CTPAbstract (接続端点抽象（Connection Termination Point Abstruct）)

CTPAbstract (接続端点抽象)インタフェースはインスタンス化されない仮想インタフェースである。その唯一の目的は、シンクオブジェクトとソースオブジェクトに共通な操作を一度にグループ化する。それ故、双方向オブジェクトはCORBA で許されている菱形多重継承を使うこともあり得る。

このインタフェースは以下の条件付パッケージを含む。

```
- channelNumberPackage: インスタンスがサポートするならば存在する。
*/
       valuetype CTPAbstractValueType: TPValueType
       {
              public long  channelNumber;
                     // conditional
                     // channelNumberPackage
                     // GET, SET-BY-CREATE
       }; // CTPAbstractValueType


       interface CTPAbstract: TP
       {
/**
チャネル番号属性がこの管理オブジェクトクラスインスタンスによってサポートされるならば、channelNumberGet
メソッドがサポートされる。
*/
              long channelNumberGet ()
                     raises (itut_x780::ApplicationError,
                            NOchannelNumberPackage);

       }; // interface CTPAbstract



/**
```

6.5.20.1.1CTPSink (接続端点シンク(Connection Termination Point Sink))

CTPSink (接続端点シンク)インタフェースはM.3100 connectionTerminationPointSink オブジェクトクラスに基づく。

この管理オブジェクトはリンク接続を終端する。

下流接続性ポインタ属性は、同一レイヤの端点から情報（トラヒック）を受信する同一管理要素内で、端点管理オブジェクトを指すか、空である。参照されるオブジェクトは以下のクラス又はサブクラスのひとつのインスタンスでなければならない：
トレール端点、接続端点。
下流接続性ポインタは信号が同報されるかどうかによって一つ以上のオブジェクトを特定し得る。
*/

```
	valuetype CTPSinkValueType: CTPAbstractValueType
	{
		public DownstreamConnectivityPointerType
			downstreamConnectivityPointer;
			// GET, SET-BY-CREATE
	}; // valuetype CTPSinkValueType


	interface CTPSink: CTPAbstract
	{
/**
DownstreamConnectivityPoint
```

同等性の照合は全ての構文選択に適用可能である。SET-COMPARISON と SET-INTERSECTION に対する照合は構文選択が同報か連結同報のいずれかに対応する場合にのみ許される。

Set-By-Create: このインタフェース属性値は生成操作に対する入力パラメタの中で規定される。
*/

```
		DownstreamConnectivityPointerType
			downstreamConnectivityPointerGet ()
			raises (itut_x780::ApplicationError);

		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectCreation)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectDeletion)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, stateChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, attributeValueChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, communicationsAlarm)

	}; // interface CTPSink


	interface CTPSinkFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID, // auto naming if empty string
			out MONameType name,
			in StringSetType packageNameList,
			in MONameSetType supportedByObjectList,
				// may be nil
				// equipmentR1Package
				// GET-REPLACE, ADD-REMOVE
			in CharacteristicInfoType characteristicInfo,
				// conditional
				// characteristicInformationPackage
				// GET, SET-BY-CREATE
			in MONameType networkLevelPointer,
				// conditional
				// networkLevelPackage
				// GET-REPLACE
			in AlarmSeverityAssignmentProfileNameType profile,
				// conditional
```

```
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                    in long channelNumber,
                        // conditional
                        // channelNumberPackage
                        // GET, SET-BY-CREATE
                    in DownstreamConnectivityPointerType
                        downstreamConnectivityPointer,
                        // GET, SET-BY-CREATE
                    in ArcProbableCauseSetType arcProbableCauseList,
                        // conditional
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval)
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface CTPSinkFactory


/**
```

6.5.20.1.2CTPSource (接続端点ソース(Connection Termination Point Source))

CTPSource (接続端点ソース)インタフェースは M.3100 connectionTerminationPointSource オブジェクトクラスに基づく。

この管理オブジェクトはリンク接続の起点となる。

上流接続性ポインタ属性は、同一レイヤの端点へ情報（トラヒック）を送信する同一管理要素内で、端点管理オブジェクトを指すか、空である。参照されるオブジェクトは以下のクラス又はサブクラスのひとつのインスタンスでなければならない：
トレール端点、接続端点。

ConnectivityPointer の元の ASN.1 は
ConnectivityPointer ::= SEQUENCE OF ObjectInstance
のように単純化されており、ここでは中身の無い系列は空を示し、また一つのエントリ系列は単一であることを示す。その結果 ConnectivityPointerType の単純化された IDL は順序が重要な意味を持つ TPSeqType である。

```
*/
        valuetype CTPSourceValueType: CTPAbstractValueType
        {
                public TPNameSeqTypeupstreamConnectivityPointer;
                        // GET, SET-BY-CREATE
        }; // valuetype CTPSourceValueType

        interface CTPSource: CTPAbstract
        {
/**
UpstreamConnectivityPoint
```

同等性照合は全ての構文選択に適用可能である。

```
Set-By-Create: このインタフェース属性値は生成操作に対する入力パラメタの中で規定される。
*/
                TPNameSeqType upstreamConnectivityPointerGet ()
                        raises (itut_x780::ApplicationError);

                MANDATORY_NOTIFICATION(
```

```
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, stateChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, communicationsAlarm)

};  // interface CTPSource

interface CTPSourceFactory: itut_x780::ManagedObjectFactory
{
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID, // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in MONameSetType supportedByObjectList,
                        // may be nil
                        // equipmentR1Package
                        // GET-REPLACE, ADD-REMOVE
                in CharacteristicInfoType characteristicInfo,
                        // conditional
                        // characteristicInformationPackage
                        // GET, SET-BY-CREATE
                in MONameType networkLevelPointer,
                        // conditional
                        // networkLevelPackage
                        // GET-REPLACE
                in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                in long channelNumber,
                        // conditional
                        // channelNumberPackage
                        // GET, SET-BY-CREATE
                in TPNameSeqType upstreamConnectivityPointer,
                        // GET, SET-BY-CREATE
                in ArcProbableCauseSetType arcProbableCauseList,
                        // conditional
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                in ArcTimeType arcManagementRequestedInterval)
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);
};  // interface CTPSourceFactory


/**
```

6.5.20.1.2.1 CTPBid (接続端点双方向（Connection Termination Point Bidirectional）)

CTPBid (接続端点双方向)インタフェースは M.3100 connectionTerminationPointBidirectional オブジェクトクラスに基づく。

この管理オブジェクトはリンク接続の起点または終端点となる。

ConnectivityPointer の元の ASN.1 は
ConnectivityPointer ::= SEQUENCE OF ObjectInstance
のように単純化されており、ここでは中身の無い系列は空を示し、一つのエントリ系列は単一であることを示す。その結果 ConnectivityPointerType の単純化された IDL は順序が重要な意味を持つ TPSeqType である。
*/

```
	valuetype CTPBidValueType: CTPAbstractValueType
	{
		public DownstreamConnectivityPointerType
			downstreamConnectivityPointer;
			// GET, SET-BY-CREATE
		public TPNameSeqTypeupstreamConnectivityPointer;
			// GET, SET-BY-CREATE
	}; // valuetype CTPBidValueType


	interface CTPBid: CTPSink, CTPSource
	{
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectCreation)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectDeletion)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, stateChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, attributeValueChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, communicationsAlarm)
	}; // interface CTPBid


	interface CTPBidFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID, // auto naming if empty string
			out MONameType name,
			in StringSetType packageNameList,
			in MONameSetType supportedByObjectList,
				// may be nil
				// equipmentR1Package
				// GET-REPLACE, ADD-REMOVE
			in CharacteristicInfoType characteristicInfo,
				// conditional
				// characteristicInformationPackage
				// GET, SET-BY-CREATE
			in MONameType networkLevelPointer,
				// conditional
				// networkLevelPackage
				// GET-REPLACE
			in AlarmSeverityAssignmentProfileNameType profile,
				// conditional
				// alarmSeverityAssignmentPointerPackage
				// GET-REPLACE
			in long channelNumber,
				// conditional
				// channelNumberPackage
				// GET, SET-BY-CREATE
			in DownstreamConnectivityPointerType
				downstreamConnectivityPointer,
				// GET, SET-BY-CREATE
			in TPNameSeqType upstreamConnectivityPointer,
				// GET, SET-BY-CREATE
			in ArcProbableCauseSetType arcProbableCauseList,
				// conditional
```

```
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval)
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface CTPBidFactory


/**
```

6.5.20.2 TTPAbstract (トレール端点抽象（Trail Termination Point Abstract）)

TTPAbstract (トレール端点抽象)インタフェースはインスタンス化可能でない仮想インタフェースである。その唯一の目的は、シンクオブジェクトとソースオブジェクトに共通な操作を一度にグループ化する。それ故、双方向オブジェクトは CORBA で許されている菱形多重継承を使うこともあり得る。

このインタフェースは以下の条件付パッケージを含む。

- supportableClientListPackage: インスタンスが一つ以上のクライアントタイプをサポートするならば存在する。

- aqdministrativeStatePackage: インスタンスが管理上サービス中、サービス停止中と認識され得るならば存在する。

```
*/
        valuetype TTPAbstractValueType: TPValueType
        {
/**
TPValueType における以下の属性は条件付であったが、現在は必須である。関連操作は条件付パッケージ例外を上げてはならない。
*/
                public OperationalStateType      operationalState;
                        // operationalStatePackage
                        // GET
                public ObjectClassSetType  supportableClientList;
                        // conditional
                        // supportableClientListPackage
                        // GET, SET-BY-CREATE
                public AdministrativeStateType   administrativeState;
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
        };  // valuetype TTPAbstractValueType


        interface TTPAbstract: TP
        {
/**
オブジェクトがクライアントタイプを一つ以上サポート可能であれば、supportableClientListPackage はサポートされる。
```

以下の属性値は、特定の管理オブジェクトがサポート可能なクライアントを表すオブジェクトクラスのリストである。これは、Rec. G.803 における特定のサーバレイヤの管理オブジェクトによって特定されるクライアントレイヤの部分集合であり得る。

Set-By-Create: このインタフェース属性値は生成操作に対する入力パラメタの中で規定される。

```
*/
```

```
              ObjectClassSetType supportableClientListGet ()
                     raises (itut_x780::ApplicationError,
                            NOsupportableClientListPackage);

              AdministrativeStateType administrativeStateGet ()
                     raises (itut_x780::ApplicationError,
                            NOadministrativeStatePackage);

              void administrativeStateSet
                     (in AdministrativeStateType adminstrativeState)
                     raises (itut_x780::ApplicationError,
                            NOadministrativeStatePackage);

       }; // interface TTPAbstract


/**
```

6.5.20.2.1 TTPSink (トレール端点シンク（Trail Termination Point Sink）)

TTPSink (トレール端点シンク)インタフェースは M.3100 trailTerminationPointSink オブジェクトクラスに基づいてモデル化されている。

TrailTerminationPointSink オブジェクトクラスはトレールを終端する管理オブジェクトクラスである。それはトレール関係およびクライアント・サーバ関係に焦点を当てているレイヤネットワークにおいてアクセスポイントを表す。

シンク方向に関しては、操作状態は正当な信号を受信する感知される能力を反映する。もし信号受信に失敗したか入力信号処理が不可能であることを端点が検出すると、操作状態はディスエーブルド（disabled）値を持つ。

運用状態がロックであるとき、端点は運用上サービスからはずされる。管理状態がロック解除されているとき、端点は運用上サービスに供される。運用状態の変化は接続性ポインタに何ら影響しない。

操作状態の変化は状態変更通知を引き起こす。トレール端点クラスインスタンスに運用状態が存在するならば、それは状態変更通知を発出してはならない。しかしながら、トレール端点クラスのサブクラスはこの通知を必要とするように振舞いを修正し得る。トレール端点のサブクラスは、どの属性値変更通知が生成されるかを明確にするために属性の規定を行うべきである。

上流接続性ポインタ属性は、同一レイヤの端点へ情報（トラヒック）を送信する同一管理要素内の端点管理オブジェクトを指すか、空である。参照されるオブジェクトは以下のクラス又はそのサブクラスのひとつのインスタンスでなければならない：
シンク又は双方向接続端点（単一又は連結系列）、又はソース又は双方向トレール端点。

```
*/
       valuetype TTPSinkValueType: TTPAbstractValueType
       {
              public TPNameSeqTypeupstreamConnectivityPointer;
                     // GET, SET-BY-CREATE
       };  // valuetype TTPSinkValueType


       interface TTPSink: TTPAbstract
       {
/**
UpstreamConnectivityPoint
```

同等性の照合は全ての構文選択に適用可能である。

```
Set-By-Create: このインタフェース属性値は生成操作に対する入力パラメタの中で規定される。
*/
              TPNameSeqType upstreamConnectivityPointerGet ()
                     raises (itut_x780::ApplicationError);

              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
```

```
                itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, stateChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, communicationsAlarm)

};  // interface TTPSink


interface TTPSinkFactory: itut_x780::ManagedObjectFactory
{
        itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID,    // auto naming if empty string
                out MONameType name,
                in StringSetType packageNameList,
                in MONameSetType supportedByObjectList,
                        // may be nil
                        // GET-REPLACE, ADD-REMOVE
                in CharacteristicInfoType  characteristicInfo,
                        // conditional
                        // characteristicInformationPackage
                        // GET, SET-BY-CREATE
                in MONameType networkLevelPointer,
                        // conditional
                        // networkLevelPackage
                        // GET-REPLACE
                in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                in DownstreamConnectivityPointerType
                        downstreamConnectivityPointer,
                        // conditional
                        // downstreamConnectivityPointerPackage
                        // GET, SET-BY-CREATE
                in TPNameSeqType upstreamConnectivityPointer,
                        // conditional
                        // upstreamConnectivityPointerPackage
                        // GET, SET-BY-CREATE
                in ObjectClassSetType supportableClientList,
                        // conditional
                        // supportableClientListPackage
                        // GET, SET-BY-CREATE
                in AdministrativeStateType adminstrativeState)
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

};  // interface TTPSinkFactory

/**
```

6.5.20.2.2TTPSource (トレール端点ソース（Trail Termination Point Source）)

TTPSource (トレール端点ソース)インタフェースは M.3100 trailTerminationPointSource オブジェクトクラスに基づいてモデル化されている。

TrailTerminationPointSource オブジェクトクラスはトレールを起点する管理オブジェクトクラスである。それはトレール関係およびクライアント・サーバ関係に焦点を当てるレイヤネットワークにおいてアクセスポイントを表す。

ソース方向に関して、操作状態は正当な信号を生成する感知される能力を反映する。もし正当な信号生成が不可能であることを端点が検出すると、操作状態はディスエーブルド（disabled）値を持つ。

運用状態がロックド（locked）であるとき、端点は運用上サービスからはずされる。運用状態がアンロックド（unlocked）されているとき、端点は運用上サービスに供される。運用状態の変化は接続性ポインタに何ら影響しない。

運用状態の変化は状態変更通知を引き起こす。トレール端点クラスインスタンスに運用状態が存在するならば、それは状態変更通知を発出してはならない。しかしながら、トレール端点クラスのサブクラスはこの通知を必要とするように振舞いを修正し得る。トレール端点のサブクラスは、どの属性値変更通知が生成されるかを明確にするために属性の規定を行うべきである。

下流接続性ポインタ属性は、同一レイヤの端点から情報（トラヒック）を受信する同一管理要素内で、端点管理オブジェクトを指すか、空である。参照されるオブジェクトは以下のクラス又はそのサブクラスのひとつのインスタンスでなければならない：
ソース又は双方向接続端点（単一又は一つ以上の接続端点ソースオブジェクトに接続された場合は連結系列又は集合）、又はシンク又は双方向トレール端点（単一または一つ以上のシンクトレール端点オブジェクトに接続される場合は集合）。
*/

```
	valuetype TTPSourceValueType: TPValueType
	{
		public DownstreamConnectivityPointerType
			downstreamConnectivityPointer;
			// GET, SET-BY-CREATE
	};  // valuetype TTPSourceValueType


	interface TTPSource: TP
	{
/**
DownstreamConnectivityPoint
```

同等性の照合は全ての構文選択に適用可能である。SET-COMPARISON と SET-INTERSECTION に対する照合は構文選択が同報か連結同報のいずれかに対応する場合にのみ許される。

Set-By-Create: このインタフェース属性値は生成操作に対する入力パラメタの中で規定される。
*/

```
		DownstreamConnectivityPointerType
			downstreamConnectivityPointerGet ()
			raises (itut_x780::ApplicationError);

		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectCreation)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectDeletion)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, stateChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, attributeValueChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, communicationsAlarm)

	}; // interface TTPSource


	interface TTPSourceFactory: itut_x780::ManagedObjectFactory
	{
		itut_x780::ManagedObject create
			(in NameBindingType nameBinding,
			in MONameType superior,
			in string reqID,    // auto naming if empty string
			out MONameType name,
			in StringSetType packageNameList,
			in MONameSetType supportedByObjectList,
				// may be nil
				// GET-REPLACE, ADD-REMOVE
			in CharacteristicInfoType  characteristicInfo,
				// conditional
```

```
                        // characteristicInformationPackage
                        // GET, SET-BY-CREATE
                    in MONameType networkLevelPointer,
                        // conditional
                        // networkLevelPackage
                        // GET-REPLACE
                    in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                    in DownstreamConnectivityPointerType
                        downstreamConnectivityPointer,
                        // conditional
                        // downstreamConnectivityPointerPackage
                        // GET, SET-BY-CREATE
                    in TPNameSeqType upstreamConnectivityPointer,
                        // conditional
                        // upstreamConnectivityPointerPackage
                        // GET, SET-BY-CREATE
                    in ObjectClassSetType supportableClientList,
                        // conditional
                        // supportableClientListPackage
                        // GET, SET-BY-CREATE
                    in AdministrativeStateType adminstrativeState)
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface TTPSourceFactory

/**
```

6.5.20.2.2.1 TTPBid (トレール端点双方向（Trail Termination Point Bidirectional）)

TTPBid (トレール端点双方向)インタフェースは M.3100 trailTerminationPointSource オブジェクトクラスに基づいてモデル化されている。

トレール端点双方向オブジェクトクラスはトレールを起点または終端する管理オブジェクトクラスである。それはトレール関係およびクライアント・サーバ関係に焦点を当てるレイヤネットワークにおいてアクセスポイントを表す。

双方向端点については、もし端点のシンク部またはソース部が操作ディスエーブルド（disabled）状態であるならば、操作状態はディスエーブルド（disabled）である。

```
*/
        valuetype TTPBidValueType: TTPAbstractValueType
        {
                public DownstreamConnectivityPointerType
                        downstreamConnectivityPointer;
                        // GET, SET-BY-CREATE
                public TPNameSeqType       upstreamConnectivityPointer;
                        // GET, SET-BY-CREATE
        };  // valuetype TTPBidValueType


        interface TTPBid: TTPSink, TTPSource
        {
/**
supportableClientList, TTPSink 及び TTPSource の administrativeState の操作は TTPBid におけるのと同等である。
*/
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectCreation)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, stateChange)
```

```
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, attributeValueChange)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, communicationsAlarm)

        }; // interface TTPBid


        interface TTPBidFactory: itut_x780::ManagedObjectFactory
        {
                itut_x780::ManagedObject create
                        (in NameBindingType nameBinding,
                        in MONameType superior,
                        in string reqID,        // auto naming if empty string
                        out MONameType name,
                        in StringSetType packageNameList,
                        in MONameSetType supportedByObjectList,
                                // may be nil
                                // GET-REPLACE, ADD-REMOVE
                        in CharacteristicInfoType       characteristicInfo,
                                // conditional
                                // characteristicInformationPackage
                                // GET, SET-BY-CREATE
                        in MONameType networkLevelPointer,
                                // conditional
                                // networkLevelPackage
                                // GET-REPLACE
                        in AlarmSeverityAssignmentProfileNameType profile,
                                // conditional
                                // alarmSeverityAssignmentPointerPackage
                                // GET-REPLACE
                        in DownstreamConnectivityPointerType
                                downstreamConnectivityPointer,
                                // conditional
                                // downstreamConnectivityPointerPackage
                                // GET, SET-BY-CREATE
                        in TPNameSeqType upstreamConnectivityPointer,
                                // conditional
                                // upstreamConnectivityPointerPackage
                                // GET, SET-BY-CREATE
                        in ObjectClassSetType supportableClientList,
                                // conditional
                                // supportableClientListPackage
                                // GET, SET-BY-CREATE
                        in AdministrativeStateType adminstrativeState)
                                // conditional
                                // administrativeStatePackage
                                // GET-REPLACE
                        raises (itut_x780::ApplicationError,
                                itut_x780::CreateError);

        }; // interface TTPBidFactory

/**
```

6.5.20.3 NetworkTP (ネットワーク端点（Network Termination Point）)

`NetworkTP` (ネットワーク端点)インタフェースは、例えばトレールまたはリンク接続を表すインスタンスのような伝送エンティティの終端を表す。

`sncPointer` はサブネットワーク接続を示す。しかしながら、全てのネットワーク端点が柔軟な接続を持つとは限らず、もう一つのネットワーク端点を指し示すことはより妥当である。例えば再生器において二つのネットワーク接続端点間に柔軟性はなく、お互いを指している。この例においては `networkTPPointer` が使われるべきである。両ポインタ共条件付きである。

接続性ポインタ属性はネットワーク端点を表すインスタンス同士を関連付けるリンク接続またはトレールを表す管理オブジェクトを指し示す。

NetworkTP インタフェースはインスタンス化可能ではない。

このインタフェースは以下の条件付パッケージを含む。

- configuredConnectivityPackage: インスタンスがサポートするならば存在する。

- connectivityPointerPackage: ネットワーク端点がリンク接続またはトレールを終端するならば存在する。

- administrativeStatePackage: この管理オブジェクトによって代表される資源が運用上サービス（の観点）から除外され得るならば存在する。

- availabilityStatusPackage: この管理オブジェクトによって代表される資源がその可用性（の観点）を表すことが出来るならば存在する。

- localtionNamePackage: インスタンスがサポートするならば存在する。

- userLabelPackage: インスタンスがサポートするならば存在する。

- neAssignmentPackage: ネットワーク要素端点ビューが使用可能であるならば存在する。

- sncPointerPackage: ネットワーク端点が他のネットワーク端点へ柔軟に接続され得る場合は存在する。

- networkTPPointerPackage: ネットワーク端点間に柔軟性がない（縮退の場合のみ）ならば存在する。
*/

```
       valuetype NetworkTPValueType: TPValueType
       {
              public PointDirectionalityType   pointDirectionality;
                     // GET
              public SignalIdType        signalId;
                     // GET, SET-BY-CREATE
              public ConfiguredConnectivityType configuredConnectivity;
                     // conditional
                     // configuredConnectivityPackage
                     // GET
              public PipeNameType        connectivityPointer;
                     // conditional
                     // connectivityPointerPackage
                     // GET
              public AdministrativeStateType   administrativeState;
                     // conditional
                     // administrativeStatePackage
                     // GET-REPLACE
              public AvailabilityStatusSetType availabilityStatus;
                     // conditional
                     // availabilityStatusPackage
                     // GET
              public Istring                   locationName;
                     // conditional
                     // localtionNamePackage
                     // GET-REPLACE
              public Istring                   userLabel;
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
              public MONameType          neAssignmentPointer;
                     // conditional
                     // neAssignmentPackage
                     // GET
              public SNCNameSetType            sncPointer;
                     // conditional
                     // sncPointerPackage
                     // GET
              public NetworkTPNameType   networkTPPointer;
```

```
                    // conditional
                    // networkTPPointerPackage
                    // GET
        }; // valuetype NetworkTPValueType


        interface NetworkTP: TP
        {
/**
以下の属性は networkTP 管理オブジェクトインスタンスの方向性を示す。
*/
                PointDirectionalityType pointDirectionalityGet ()
                        raises (itut_x780::ApplicationError);

/**
以下の属性は、議論しているエンティティが属するレイヤの特性情報を定義する。サブネットワーク接続／接続性が可能
か決定するのに使用される。
*/
                SignalIdType signalIdGet ()
                        raises (itut_x780::ApplicationError);

/**
以下の属性はネットワーク端点管理オブジェクト（またはサブクラス）の設定された接続性を示す。この属性の取り得る
値は sourceConnect, sinkConnect, bidirectionalConnect 及び noConnect である。

シンクと同等な pointDirectionality を伴った ネットワーク端点管理オブジェクトについて、この属性に許される
値は noConnect 及び sinkConnect である。

ソースと同等な pointDirectionality を伴った ネットワーク端点管理オブジェクトについて、この属性に許される
値は noConnect 及び sourceConnect である。

双方向に同等な pointDirectionality を伴った ネットワーク端点管理オブジェクトについて、この属性に許される
値は noConnect と bidirectionalConnect である。

いくつかの技術では、双方向ネットワーク端点管理オブジェクトに sinkConnect 及び sourceConnect が許され得
る。
*/
                ConfiguredConnectivityType configuredConnectivityGet ()
                        raises (itut_x780::ApplicationError,
                                NOconfiguredConnectivityPackage);

/**
以下の属性は、ネットワーク端点によって終端されるリンク接続またはトレール管理オブジェクトのインスタンスを特定
する。

この属性は、ネットワーク端点によって終端されるリンク接続またはトレールを指す。
*/
                PipeNameType connectivityPointerGet ()
                        raises (itut_x780::ApplicationError,
                                NOconnectivityPointerPackage);

/**
GDMO においてこのオブジェクトは administrativeOperationalStatesPackage と
operationalStatePackage の両方を持つ。CORBA において、それは administrativeStatePackage と
operationalStatePackage によって最適化される。
*/
                AdministrativeStateType administrativeStateGet ()
                        raises (itut_x780::ApplicationError,
                                NOadministrativeStatePackage);
                void administrativeStateSet
                        (in AdministrativeStateType administrativeState)
                        raises (itut_x780::ApplicationError,
                                NOadministrativeStatePackage);

                AvailabilityStatusSetType availabilityStatusGet ()
```

```
                raises (itut_x780::ApplicationError,
                        NOavailabilityStatusPackage);

        Istring locationNameGet ()
                raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

        void locationNameSet
                (in Istring locationName)
                raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

        Istring userLabelGet ()
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        void userLabelSet
                (in Istring userLabel)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

/**
以下の属性は、分割階層における最下位レベルの Network TP から、Network TP をサポートする機能を表す NE TP
までを指し示す。NE 割当ポインタを利用する Network CTP の従属分割ポインタは空である。
*/
        MONameType neAssignmentPointerGet ()
                raises (itut_x780::ApplicationError,
                        NOneAssignmentPointerPackage);

/**
以下の属性は、ネットワーク端点と関係を有するサブネットワーク接続の順序リストを指す。サブネットワーク接続が存
在しない場合、このポインタはサブネットワークを指すか空である。このリストは、一対一アプリケーションでは単一エ
ントリを持ち、一対多アプリケーションでは複数エントリを持ち得る。
*/
        SNCNameSetType sncPointerGet ()
                raises (itut_x780::ApplicationError,
                        NOsncPointerPackage);

/**
以下の属性はネットワーク端点を指す。
*/
        NetworkTPNameType networkTPPointerGet ()
                raises (itut_x780::ApplicationError,
                        NOnetworkTPPointerPackage);

        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)

    }; // interface NetworkTP

/**
```

6.5.20.3.1NetworkCTPAbstract (ネットワーク接続端点抽象（Network Connection Termination Point Abstract))

NetworkCTPAbstruct (ネットワーク接続端点抽象)インタフェースはインスタンス化可能でない仮想インタフェースである。その唯一の目的は、シンクオブジェクトとソースオブジェクトに共通な操作を一度にグループ化する。それ故、双方向オブジェクトは CORBA で許されている菱形多重継承を使うこともあり得る。

このインタフェースは以下の条件付パッケージを含む。

- channelNumberPackage: インスタンスがサポートするなら存在する。

- networkCTPPackage: サブネットワーク分割の上位または下位レベルのネットワーク端点インスタンスへのポインタがこの管理オブジェクトクラスによってサポートされるならば存在する。

このパッケージは（与えられたレイヤ内で）より上位および、より下位のサブネットワーク分割でのネットワーク CTP 管理オブジェクトクラスインスタンスを、分割ポインタを使用することによって特定する。

- serverTTPPointerPackage: サーバトレール端点ポインタ属性がこの管理オブジェクトクラスによってサポートされるならば存在する。
*/

```
	valuetype NetworkCTPAbstractValueType: NetworkTPValueType
	{
		public long                         channelNumber;
			// conditional
			// channelNumberPackage
			// GET, SET-BY-CREATE
		public NetworkCTPAbstractNameType superPartitionPointer;
			// conditional
			// networkCTPPackage
			// GET
		public NetworkCTPAbstractNameType subPartitionPointer;
			// conditional
			// networkCTPPackage
			// GET
		public NetworkTTPAbstractNameSetType    serverTTPPointer;
			// conditional
			// serverTTPPointerPackage
			// GET
	}; // valuetype NetworkCTPAbstractValueType


	interface NetworkCTPAbstract: NetworkTP
	{
		long channelNumberGet ()
			raises (itut_x780::ApplicationError,
				NOchannelNumberPackage);

/**
以下の属性は、より上位レベルの分割におけるネットワーク CTP へのポインタである。それは、より下位レベルの分割において、より上位レベルのネットワーク CTP に直接対応するネットワーク CTP に対してのみ存在する。それは空であり得る。
*/
		NetworkCTPAbstractNameType superPartitionPointerGet ()
			raises (itut_x780::ApplicationError,
				NOnetworkCTPPackage);

/**
以下の属性はより下位レベルの分割にあるネットワーク CTP へのポインタである。NE 割当ポインタ経由で最下位レベルのネットワーク CTP が NE CTP を指す場合、従属分割ポインタは空である。
*/
		NetworkCTPAbstractNameType subPartitionPointerGet ()
			raises (itut_x780::ApplicationError,
				NOnetworkCTPPackage);

/**
以下の属性は、もう一方のレイヤにおいて CTP かつまた linkEnd を扱い得る TTP を定義する。通常、TTP またはより上位レイヤの TTP は、CTP またはより下位レイヤの CTP を扱う。
*/
		NetworkTTPAbstractNameSetType serverTTPPointerGet ()
			raises (itut_x780::ApplicationError,
				NOserverTTPPointerPackage);

	}; // interface NetworkCTPAbstract


/**
```

6.5.20.3.1.1 NetworkCTPSink (ネットワーク接続端点シンク（Network Connection Termination Point Sink）)

NetworkCTPSink (ネットワーク接続端点シンク)インタフェースはリンク接続を終端、かつ／またサブネットワーク接続を起点する。資源はリンク接続を経由してネットワーク接続端点を表すインスタンスから情報（トラヒック）を受け、それをサブネットワーク接続経由でサブネットワーク内のネットワーク CTP ソースかまたはネットワーク TTP シンクを表すインスタンスへ送る。

このクラスのインスタンスは同一レイヤにあるネットワーク接続端点、ソースまたは双方向を表すインスタンスと接続性関係（リンク接続又はサブネットワーク接続）のみを有し得る。

このクラスインスタンスはサブネットワーク接続経由して、同一レイヤのネットワークトレール端点、シンクまたは双方向を表す単一のインスタンスへ接続されたサブネットワークであり得る。

サブネットワーク接続ポインタ属性は、同じサブネットワーク内で、ネットワーク端点との関連を表す管理オブジェクトを指し、このネットワーク端点から（へ）受信するか空である。

参照された管理オブジェクトはサブネットワーク接続を表すべきである。ネットワーク CTP シンクが異なるサブネットワークのために多くのサブネットワーク接続に関与する場面ではサブネットワーク接続ポインタは空である。

関連するサブネットワーク接続によって特定されるいかなるネットワーク端点も関係が存在することを示すが、これは情報がネットワーク端点間を流れることができることを示してはいない。この能力は操作状態を含んだ状態属性の組み合わせで示される。

接続性ポインタ属性は、このネットワーク端点へ情報（トラヒック）を送るネットワーク接続端点ソースまたは双方向を表すインスタンスにこのインスタンスを関連付ける接続を表す管理オブジェクトを指すか空である。
*/

```
valuetype NetworkCTPSinkValueType: NetworkCTPAbstractValueType
{
}; // valuetype NetworkCTPSinkValueType


interface NetworkCTPSink: NetworkCTPAbstract
{
}; // interface NetworkCTPSink


interface NetworkCTPSinkFactory: itut_x780::ManagedObjectFactory
{
      itut_x780::ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID,    // auto naming if empty string
            out MONameType name,
            in StringSetType packageNameList,
            in SignalIdType signalId,
                  // networkTerminationPointPackage
                  // GET, SET-BY-CREATE
            in CharacteristicInfoType characteristicInfo,
                  // conditional
                  // characteristicInformationPackage
                  // GET, SET-BY-CREATE
            in MONameType networkLevelPointer,
                  // conditional
                  // networkLevelPackage
                  // GET-REPLACE
            in AlarmSeverityAssignmentProfileNameType profile,
                  // conditional
                  // alarmSeverityAssignmentPointerPackage
                  // GET-REPLACE
            in ArcProbableCauseSetType arcProbableCauseList,
                  // conditional
                  // arcPackage
                  // GET-REPLACE, ADD-REMOVE
```

```
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                           // conditional
                           // arcPackage
                           // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval,
                           // conditional
                           // arcPackage
                           // GET-REPLACE
                    in AdministrativeStateType administrativeState,
                           // conditional
                           // administrativeStatePackage
                           // GET-REPLACE
                    in Istring locationName,
                           // conditional
                           // localtionNamePackage
                           // GET-REPLACE
                    in Istring userLabel,
                           // conditional
                           // userLabelPackage
                           // GET-REPLACE
                    in long channelNumber)
                           // conditional
                           // channelNumberPackage
                           // GET, SET-BY-CREATE
                    raises (itut_x780::ApplicationError,
                           itut_x780::CreateError);

      }; // interface NetworkCTPSinkFactory

/**
```

6.5.20.3.1.2 NetworkCTPSource (ネットワーク接続端点ソース（Network Connection Termination Point Source）)

NetworkCTPSource（ネットワーク接続端点ソース）インタフェースはリンク接続を終端、かつまたサブネットワーク接続を起点する管理オブジェクトクラスである。資源は同一サブネットワーク内で、リンク接続を経由してネットワーク接続端点を表すインスタンスへ情報（トラヒック）を送り、かつサブネットワーク接続を経由して、ネットワーク CTP シンクかまたはネットワーク TTP ソースを表すインスタンスから受ける。

このクラスのインスタンスは同一レイヤにあるネットワーク接続端点、シンクまたは双方向を表すインスタンスと接続性関係（リンク接続又はサブネットワーク接続）のみを持ち得る。

このクラスインスタンスはサブネットワーク接続を経由して、同一レイヤのネットワークトレール端点、ソースまたは双方向を表す単一のインスタンスへ接続されたサブネットワークであり得る。

サブネットワーク接続ポインタ属性は同じサブネットワーク内で、このネットワーク端点へ情報（トラヒック）を送るネットワーク端点との関連を表す管理オブジェクトを指すかまたは空である。

参照された管理オブジェクトはサブネットワーク接続を表すべきである。ネットワーク CTP ソースが異なるサブネットワークの多くのサブネットワーク接続に関与する場面ではサブネットワーク接続ポインタは空である。

関連するサブネットワーク接続によって特定されるネットワーク端点は関係が存在することを示すが、これは情報がネットワーク端点間を流れることができることを示してはいない。この能力は操作属性を含んだ状態属性の組み合わせで示される。

接続性ポインタ属性はこのネットワーク端点から情報（トラヒック）を受信するシンクまたは双方向のネットワーク接続端点を表すインスタンスにこのインスタンスを関連付ける接続を表す管理オブジェクトを指すか空である。

```
*/
      valuetype NetworkCTPSourceValueType: NetworkCTPAbstractValueType
      {
      }; // valuetype NetworkCTPSourceValueType
```

```
interface NetworkCTPSource: NetworkCTPAbstract
{
}; // interface NetworkCTPSource


interface NetworkCTPSourceFactory: itut_x780::ManagedObjectFactory
{
       itut_x780::ManagedObject create
              (in NameBindingType nameBinding,
              in MONameType superior,
              in string reqID,    // auto naming if empty string
              out MONameType name,
              in StringSetType packageNameList,
              in SignalIdType signalId,
                     // networkTerminationPointPackage
                     // GET, SET-BY-CREATE
              in CharacteristicInfoType characteristicInfo,
                     // conditional
                     // characteristicInformationPackage
                     // GET, SET-BY-CREATE
              in MONameType networkLevelPointer,
                     // conditional
                     // networkLevelPackage
                     // GET-REPLACE
              in AlarmSeverityAssignmentProfileNameType profile,
                     // conditional
                     // alarmSeverityAssignmentPointerPackage
                     // GET-REPLACE
              in ArcProbableCauseSetType arcProbableCauseList,
                     // conditional
                     // arcPackage
                     // GET-REPLACE, ADD-REMOVE
              in ArcIntervalProfileNameType arcIntervalProfilePointer,
                     // conditional
                     // arcPackage
                     // GET-REPLACE
              in ArcTimeType arcManagementRequestedInterval,
                     // conditional
                     // arcPackage
                     // GET-REPLACE
              in AdministrativeStateType administrativeState,
                     // conditional
                     // administrativeStatePackage
                     // GET-REPLACE
              in Istring locationName,
                     // conditional
                     // localtionNamePackage
                     // GET-REPLACE
              in Istring userLabel,
                     // conditional
                     // userLabelPackage
                     // GET-REPLACE
              in long channelNumber)
                     // conditional
                     // channelNumberPackage
                     // GET, SET-BY-CREATE
              raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

}; // interface NetworkCTPSourceFactory


/**
```

6.5.20.3.1.2.1 NetworkCTPBid (ネットワーク接続端点双方向（Network Connection Termination Point Bidirectional）)

NetworkCTPBud (ネットワーク接続端点双方向)インタフェースはリンク接続を終端し、且つ又サブネットワーク接続を起点する管理オブジェクトクラスである。

このオブジェクトクラスインスタンスを片方向として構成する必要があるならば、サブクラスはどの方向性が設定可能であることを許されるか規定され得る。
*/

```
valuetype NetworkCTPBidValueType: NetworkCTPAbstractValueType
{
}; // valuetype NetworkCTPBidValueType


interface NetworkCTPBid: NetworkCTPSink, NetworkCTPSource
{
}; // interface NetworkCTPBid


interface NetworkCTPBidFactory: itut_x780::ManagedObjectFactory
{
      itut_x780::ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID,    // auto naming if empty string
            out MONameType name,
            in StringSetType packageNameList,
            in SignalIdType signalId,
                  // networkTerminationPointPackage
                  // GET, SET-BY-CREATE
            in CharacteristicInfoType characteristicInfo,
                  // conditional
                  // characteristicInformationPackage
                  // GET, SET-BY-CREATE
            in MONameType networkLevelPointer,
                  // conditional
                  // networkLevelPackage
                  // GET-REPLACE
            in AlarmSeverityAssignmentProfileNameType profile,
                  // conditional
                  // alarmSeverityAssignmentPointerPackage
                  // GET-REPLACE
            in ArcProbableCauseSetType arcProbableCauseList,
                  // conditional
                  // arcPackage
                  // GET-REPLACE, ADD-REMOVE
            in ArcIntervalProfileNameType arcIntervalProfilePointer,
                  // conditional
                  // arcPackage
                  // GET-REPLACE
            in ArcTimeType arcManagementRequestedInterval,
                  // conditional
                  // arcPackage
                  // GET-REPLACE
            in AdministrativeStateType administrativeState,
                  // conditional
                  // administrativeStatePackage
                  // GET-REPLACE
            in Istring locationName,
                  // conditional
                  // localtionNamePackage
                  // GET-REPLACE
            in Istring userLabel,
                  // conditional
```

```
                        // userLabelPackage
                        // GET-REPLACE
                    in long channelNumber)
                        // conditional
                        // channelNumberPackage
                        // GET, SET-BY-CREATE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface NetwortCTPBidFactory



/**
```

6.5.20.3.2NetworkTTPAbstract (ネットワークトレール端点抽象（Network Trail Termination Point Abstract）)

NetworkTTPAbstruct (ネットワークトレール端点抽象)インタフェースはインスタンス化可能でない仮想インタフェースである。その唯一の目的は、シンクオブジェクトとソースオブジェクトに共通な操作を一度にグループ化する。それ故、双方向オブジェクトは CORBA で許されている菱形多重継承を使うこともあり得る。

このインタフェースは以下の条件付パッケージを含む。

- supportableClientListPackage: インスタンスがサポートするならば存在する。

- ClientCTPListPackage: この管理オブジェクトのクライアント networkCTP の管理がサポートされるならば存在する。

- ClientLinkEndPointerPackage: リンクエンドがクライアントレイヤのサブネットワークによってサポートされるならば存在する。

```
*/
        valuetype NetworkTTPAbstractValueType: NetworkTPValueType
        {
            public ObjectClassSetType         supportableClientList;
                // conditional
                // supportableClientListPackage
                // GET, SET-BY-CREATE
            public NetworkCTPAbstractNameSetType    clientCTPList;
                // conditional
                // clientCTPListPackage
                // GET
            public AbstractLinkEndNameSetType clientLinkEndPointerList;
                // conditional
                // clientLinkEndPointerPackage
                // GET
        }; // valuetype NetworkTTPAbstractValueType


        interface NetworkTTPAbstract: NetworkTP
        {
/**
```

以下の属性値は特定の管理オブジェクトがサポート可能なクライアントを示すオブジェクトクラスリストである。これは Rec. G.803 において特定のサーバレイヤ管理オブジェクトによって特定されるクライアントレイヤの部分集合であり得る。

```
*/
            ObjectClassSetType supportableClientListGet ()
                raises (itut_x780::ApplicationError,
                    NOsupportableClientListPackage);

/**
```

以下の属性は、もう一方のレイヤで TTP のクライアントである CTP または CTP リストを定義する。通常、上位レイヤにおける単一の TTP は下位レイヤにおける多数の CTP をサポートする。あるいは、連結が使用されるところでは下位レイヤにおける多数の TTP は上位レイヤの CTP をサポートする。
*/
```
            NetworkCTPAbstractNameSetType clientCTPListGet ()
                  raises (itut_x780::ApplicationError,
                        NOclientCTPListPackage);

/**
```
以下の属性は、クライアントレイヤネットワーク領域においてネットワークトレール端点のプロパティを反映するリンク末端部へのポインタの集合である。
*/
```
            AbstractLinkEndNameSetType clientLinkEndPointerListGet ()
                  raises (itut_x780::ApplicationError,
                        NOclientLinkEndPointerPackage);

      };  // interface NetworkTTPAbstract

/**
```

6.5.20.3.2.1 NetworkTTPSink ネットワークトレール端点シンク（(Network Trail Termination Point Sink))

NetworkTTPSink (ネットワークトレール端点)インタフェースはネットワークの観点においてトレールおよびサブネットワーク接続を終端する管理オブジェクトクラスである。

このクラスのインスタンスは同一レイヤにあるもう一方のソースまたは双方向のネットワークトレール端点を表すインスタンスとトレール関係のみを有し得る。

このクラスインスタンスはサブネットワーク接続を経由して、同一レイヤの単一シンクまたは双方向のネットワーク接続端点またはネットワークトレール端点ソースへ接続されたサブネットワークであり得る。

サブネットワーク接続ポインタ属性は同じサブネットワーク内で、情報（トラヒック）をこのネットワーク端点へ送信する一つ以上のネットワーク接続端点との関連を表す管理オブジェクトを指すか空である。

関連するサブネットワーク接続によって特定されるネットワーク端点は関係が存在することを示すが、これは情報がネットワーク端点間を流れることができることを示してはいない。この能力は操作属性を含んだ状態属性の組み合わせで示される。

接続性ポインタ属性は情報（トラヒック）を同一レイヤのこのネットワーク端点へ送信するネットワークトレール端点を表すインスタンスにこのインスタンスを関連付けるトレールを表す管理オブジェクトを指すか空である。
*/
```
      valuetype NetworkTTPSinkValueType: NetworkTTPAbstractValueType
      {
      }; // valuetype NetworkTTPSinkValueType


      interface NetworkTTPSink: NetworkTTPAbstract
      {
      };  // interface NetworkTTPSink


      interface NetworkTTPSinkFactory: itut_x780::ManagedObjectFactory
      {
            itut_x780::ManagedObject create
                  (in NameBindingType nameBinding,
                  in MONameType superior,
                  in string reqID, // auto naming if empty string
                  out MONameType name,
                  in StringSetType packageNameList,
                  in SignalIdType signalId,
                        // networkTerminationPointPackage
                        // GET, SET-BY-CREATE
                  in CharacteristicInfoType characteristicInfo,
```

```
                        // conditional
                        // characteristicInformationPackage
                        // GET, SET-BY-CREATE
                    in MONameType networkLevelPointer,
                        // conditional
                        // networkLevelPackage
                        // GET-REPLACE
                    in AlarmSeverityAssignmentProfileNameType profile,
                        // conditional
                        // alarmSeverityAssignmentPointerPackage
                        // GET-REPLACE
                    in ArcProbableCauseSetType arcProbableCauseList,
                        // conditional
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
                    in Istring locationName,
                        // conditional
                        // localtionNamePackage
                        // GET-REPLACE
                    in Istring userLabel,
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                    in ObjectClassSetType supportableClientList)
                        // conditional
                        // supportableClientListPackage
                        // GET, SET-BY-CREATE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError,
                        FailureToCreateNetworkTTP);

                // DELETE_ERROR:
                //     NetworkTTPTerminatesTrail
                //     NetwrokTTPAssociatedWithSubnetwork
                //     NetworkTTPAssociatedWithAccessGroup
                //     FailureToRemoveNeworkTTP

        }; // interface NetworkTTPSinkFactory

/**
```

6.5.20.3.2.2 NetworkTTPSource (ネットワークトレール端点ソース（Network Trail Termination Point Source）)

NetworkTTPSource (ネットワークトレール端点ソース)インタフェースはネットワークの観点においてトレールおよびサブネットワーク接続を終端する管理オブジェクトクラスである。

このクラスのインスタンスは同一レイヤにあるもう一方のネットワークトレール端点、シンクまたは双方向とトレール関係のみを有し得る。

このクラスインスタンスはサブネットワーク接続を経由して、同一レイヤの単一ソースまたは双方向のネットワーク接続端点またはネットワークトレール端点シンクへサブネットワーク接続され得る。それはまた同じ信号の多重コピーを転送

するために同報モードで運用されている時、サブネットワーク接続を経由して同一レイヤのネットワーク CTP の多重インスタンスへ接続され得る。

サブネットワーク接続ポインタ属性は同じサブネットワーク内で情報（トラヒック）をこのネットワーク端点から受信する一つ以上のネットワーク接続端点との関連を表す管理オブジェクトを指すか空である。

関連するサブネットワーク接続によって特定されるネットワーク端点は関係が存在することを示すが、これは情報がネットワーク端点間を流れることができることを示してはいない。この能力は操作属性を含んだ状態属性の組み合わせで示される。

接続性ポインタ属性は同一レイヤにおいて情報（トラヒック）をこのネットワーク端点から受信するネットワークトレール端点を表すインスタンスにこのインスタンスに関連付けるトレールを表す管理オブジェクトを指すか空である。
*/

```
valuetype NetworkTTPSourceValueType: NetworkTTPAbstractValueType
{
}; // valuetype NetworkTTPSourceValueType


interface NetworkTTPSource: NetworkTTPAbstract
{
};  // interface NetworkTTPSource


interface NetworkTTPSourceFactory: itut_x780::ManagedObjectFactory
{
      itut_x780::ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID, // auto naming if empty string
            out MONameType name,
            in StringSetType packageNameList,
            in SignalIdType signalId,
                  // networkTerminationPointPackage
                  // GET, SET-BY-CREATE
            in CharacteristicInfoType characteristicInfo,
                  // conditional
                  // characteristicInformationPackage
                  // GET, SET-BY-CREATE
            in MONameType networkLevelPointer,
                  // conditional
                  // networkLevelPackage
                  // GET-REPLACE
            in AlarmSeverityAssignmentProfileNameType profile,
                  // conditional
                  // alarmSeverityAssignmentPointerPackage
                  // GET-REPLACE
            in ArcProbableCauseSetType arcProbableCauseList,
                  // conditional
                  // arcPackage
                  // GET-REPLACE, ADD-REMOVE
            in ArcIntervalProfileNameType arcIntervalProfilePointer,
                  // conditional
                  // arcPackage
                  // GET-REPLACE
            in ArcTimeType arcManagementRequestedInterval,
                  // conditional
                  // arcPackage
                  // GET-REPLACE
            in AdministrativeStateType adminstrativeState,
                  // conditional
                  // administrativeStatePackage
                  // GET-REPLACE
            in Istring locationName,
                  // conditional
                  // localtionNamePackage
                  // GET-REPLACE
            in Istring userLabel,
```

```
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                in ObjectClassSetType supportableClientList)
                        // conditional
                        // supportableClientListPackage
                        // GET, SET-BY-CREATE
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError,
                        FailureToCreateNetworkTTP);

        // DELETE_ERROR:
        //     NetworkTTPTerminatesTrail
        //     NetwrokTTPAssociatedWithSubnetwork
        //     NetworkTTPAssociatedWithAccessGroup
        //     FailureToRemoveNeworkTTP

    }; // interface NetworkTTPSourceFactory


/**
```

6.5.20.3.2.2.1 NetworkTTPBid (ネットワークトレール端点双方向（Network Trail Termination Point Bidirectional）)

NetworkTTPBid (ネットワーク端点双方向)インタフェースはネットワークの観点においてトレールおよびサブネットワーク接続を終端する管理オブジェクトクラスである。

このオブジェクトクラスインスタンスを片方向として構成する必要があるならば、サブクラスはどの方向性が設定可能であることが許されるか規定し得る。

```
*/
    valuetype NetworkTTPBidValueType: NetworkTTPAbstractValueType
    {
    }; // valuetype NetworkTTPBidValueType


    interface NetworkTTPBid: NetworkTTPSink, NetworkTTPSource
    {
    };  // interface NetworkTTPBid


    interface NetworkTTPBidFactory: itut_x780::ManagedObjectFactory
    {
        itut_x780::ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID,    // auto naming if empty string
            out MONameType name,
            in StringSetType packageNameList,
            in SignalIdType signalId,
                // networkTerminationPointPackage
                // GET, SET-BY-CREATE
            in CharacteristicInfoType characteristicInfo,
                // conditional
                // characteristicInformationPackage
                // GET, SET-BY-CREATE
            in MONameType networkLevelPointer,
                // conditional
                // networkLevelPackage
                // GET-REPLACE
            in AlarmSeverityAssignmentProfileNameType profile,
                // conditional
                // alarmSeverityAssignmentPointerPackage
                // GET-REPLACE
            in ArcProbableCauseSetType arcProbableCauseList,
                // conditional
```

```
                        // arcPackage
                        // GET-REPLACE, ADD-REMOVE
                    in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in ArcTimeType arcManagementRequestedInterval,
                        // conditional
                        // arcPackage
                        // GET-REPLACE
                    in AdministrativeStateType adminstrativeState,
                        // conditional
                        // administrativeStatePackage
                        // GET-REPLACE
                    in Istring locationName,
                        // conditional
                        // localtionNamePackage
                        // GET-REPLACE
                    in Istring userLabel,
                        // conditional
                        // userLabelPackage
                        // GET-REPLACE
                    in ObjectClassSetType supportableClientList)
                        // conditional
                        // supportableClientListPackage
                        // GET, SET-BY-CREATE
                    raises (itut_x780::ApplicationError,
                        itut_x780::CreateError,
                        FailureToCreateNetworkTTP);

                // DELETE_ERROR:
                //     NetworkTTPTerminatesTrail
                //     NetwrokTTPAssociatedWithSubnetwork
                //     NetworkTTPAssociatedWithAccessGroup
                //     FailureToRemoveNeworkTTP

        }; // interface NetworkTTPBidFactory

/**
```

6.5.21 TPPool (端点プール（Termination Point Pool）)

TPPool (端点プール)オブジェクトは端点の集合またはある管理目的、例えばルーティング、のために使用される GTP を表す。GTP のメンバである端点は GTP の残りの部分に関係なく TPPool のメンバになりえない。

このインタフェースは M.3100 tpPool の仕様に基づいている。

TPPoolValueType 構造体は一回の操作で TPPool 属性の全てを取り出すために使用される。多くのサポートされない属性は、空の文字列またはリストとして返される。空の文字列値の受け取りは、しかしながらその属性がサポートされないことを意味しない。

```
*/
        valuetype TPPoolValueType: itut_x780::ManagedObjectValueType
        {
                public TPNameSetType        tpsInTPPool;
                        // GET
                public short        totalTpCount;
                        // GET
                public short        connectedTpCount;
                        // GET
                public short        idleTpCount;
                        // GET
        }; // valuetype TPPoolType


        interface TPPool: itut_x780::ManagedObject
        {
```

```
/**
以下のメソッドは TP Pool によって代表される端点参照リストを返す。
*/
              TPNameSetType tpsInTPPoolGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは TP Plool に関連する端点の総数を返す。
*/
              short totalTpCountGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは接続された TP Pool に関連する端点の総数を返す。
*/
              short connectedTpCountGet ()
                     raises (itut_x780::ApplicationError);

/**
以下のメソッドは操作可能状態でかつクロスコネクションが使用可能な TP Pool に関連する端点の総数を返す。
*/
              short idleTpCountGet ()
                     raises (itut_x780::ApplicationError);

       }; // interface TPPool


       interface TPPoolFactory: itut_x780::ManagedObjectFactory
       {
              itut_x780::ManagedObject create
                     (in NameBindingType nameBinding,
                     in MONameType superior,
                     in string reqID,    // auto naming if empty string
                     out MONameType name,
                     in StringSetType packageNameList)
                     raises (itut_x780::ApplicationError,
                            itut_x780::CreateError);

       }; // TPPoolFactory


/**
```

```
/**
```

6.6　インタフェースーファサード（Interfaces – Facade）

ファサードインタフェースの振る舞いは対応する細粒度インタフェースと同一である。それ故、コメントはファサードインタフェース中に含まれない。読者はファサードインタフェースの振る舞いについて、6.6 節（TTC 注：6.5 節の誤り）の細粒度インタフェースを参照のこと。

管理システムが細粒度インタフェースのみをサポートするならば、この節は IDL から省略することができる。

```
*/

/**
```

6.6.1 AbstractLink_F

```
*/

       interface AbstractLink_F: itut_x780::ManagedObject_F
       {
              MONameType aEndGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError);

              CapacityType availableLinkCapacityGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError);

              SignalIdType signalIdGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError);

              MONameType zEndGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError);

              UsageCostType usageCostGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError,
                            NOusageCostPackage);

              void usageCostSet
                     (in MONameType name,
                     in UsageCostType usageCost)
                     raises (itut_x780::ApplicationError,
                            NOusageCostPackage);

              Istring userLabelGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

              void userLabelSet
                     (in MONameType name,
                      in Istring label)
                     raises (itut_x780::ApplicationError,
                            NOuserLabelPackage);

              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion)

              CONDITIONAL_NOTIFICATION(
                     itut_x780::Notifications, attributeValueChange,
                     attributeValueChangeNotificationPackage)
```

```
};  // interface AbstractLink_F

/**
```

6.6.1.1 LogicalLink_F

```
*/

interface LogicalLink_F : AbstractLink_F
{
	LinkDirectionalityType linkDirectionalityGet
		(in MONameType name)
		raises (itut_x780::ApplicationError);

	LinkConnectionNameSetType linkConnectionPointerListGet
		(in MONameType name)
		raises (itut_x780::ApplicationError,
			NOlinkConnectionPointerListPackage);

	void linkConnectionPointerListSet
		(in MONameType name,
		in LinkConnectionNameSetType connectionList)
		raises (itut_x780::ApplicationError,
			NOlinkConnectionPointerListPackage);

	void linkConnectionPointerListAdd
		(in MONameType name,
		in LinkConnectionNameSetType connectionList)
		raises (itut_x780::ApplicationError,
			NOlinkConnectionPointerListPackage);

	void linkConnectionPointerListRemove
		(in MONameType name,
		in LinkConnectionNameSetType connectionList)
		raises (itut_x780::ApplicationError,
			NOlinkConnectionPointerListPackage);


	void assignLinkConnectionOnLogicalLink
		(in MONameType name,
		in LayerNDNameType layer,
		in LinkConnectionNameSetType requestedConnectionList,
		out LinkConnectionNameSetType resultingConnectionList)
		raises (itut_x780::ApplicationError,
			LinkAndLinkConnectionNotCompatible,
			InvalidLinkConnection,
			NotEnoughLinkConnection,
			LinkConnectionAlreadyAssigned,
			InconsistentSignalIdentification,
			InconsistentDirectionality,
			FailureToSetLinkConnectionCallerId,
			FailureToDecreaseCapacity);

	void deassignLinkConnectionFromLogicalLink
		(in MONameType name,
		in LinkConnectionNameSetType requestedConnectionList)
		raises (itut_x780::ApplicationError,
			LinkAndLinkConnectionNotCompatible,
			InvalidLinkConnection,
			NotAssignedToCaller,
			FailureToDeassignLinkConnection,
			FailureToIncreaseCapacity);

}; // interface LogicalLink_F

/**
```

6.6.1.2 TopLink_F (Topological Link)

```
*/

        interface TopLink_F : AbstractLink_F
        {

                DirectionalityType directionalityGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                TrailNameType serverTrailGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                CapacityType totalLinkCapacityGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOtotalLinkCapacityPackage);

                long maximumLinkConnectionCountGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOmaximumLinkConnectionCountPackage);

                CapacityType potentialLinkCapacityPackageGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOpotentialLinkCapacityPackage);

                CapacityType provisionedLinkCapacityGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOprovisionedLinkCapacityPackage);

                long provisionedLinkConnectionCountGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOprovisionedLinkConnectionCountPackage);


                void addCapacityToTopLink
                        (in MONameType name,
                        in RequestedCapacityType requestedCapacity,
                        out CapacityType resultingCapacity,
                        out LinkConnectionNameSetType resultingLinkConnection)
                        raises (itut_x780::ApplicationError,
                                NoSuchLink,
                                InsufficientCapacity,
                                InvalidChannelsNumber,
                                ChannelsAlreadyProvisioned,
                                FailureToCreateLC,
                                FailureToAssociateLC,
                                FailureToSupportLC,
                                FailureToIncreaseCapacity);

                void removeCapacityFromTopLink
                        (in MONameType name,
                        in RequestedCapacityType requestedCapacity,
                        out CapacityType resultingCapacity)
                        raises (itut_x780::ApplicationError,
                                NoSuchLink,
                                InsufficientCapacity,
                                InvalidChannelsNumber,
                                FailureToRemoveLC,
                                FailureToDecreaseCapacity);
```

```
};  // interface TopLink_F



/**
```

6.6.2 AbstractLinkEnd_F

```
*/

        interface AbstractLinkEnd_F: itut_x780::ManagedObject_F
        {
                PointCapacityType availableLinkEndCapacityGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                AbstractLinkNameType linkPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                SubnetworkNameSetType containedInSubnetworkListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOcontainedInSubnetworkListPackage);

                void containedInSubnetworkListSet
                        (in MONameType name,
                        in SubnetworkNameSetType containedInSubnetworkList)
                        raises (itut_x780::ApplicationError,
                                NOcontainedInSubnetworkListPackage);

                void containedInSubnetworkListAdd
                        (in MONameType name,
                        in SubnetworkNameSetType containedInSubnetworkList)
                        raises (itut_x780::ApplicationError,
                                NOcontainedInSubnetworkListPackage);

                void containedInSubnetworkListRemove
                        (in MONameType name,
                        in SubnetworkNameSetType containedInSubnetworkList)
                        raises (itut_x780::ApplicationError,
                                NOcontainedInSubnetworkListPackage);

                Istring userLabelGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                void userLabelSet
                        (in MONameType name,
                        in Istring label)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectCreation)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, attributeValueChange)

        };  // interface AbstractLinkEnd_F


/**
```

6.6.2.1 LogicalLinkEnd_F

```
*/

	interface LogicalLinkEnd_F: AbstractLinkEnd_F
	{
		PointDirectionalityType logicalEndDirectionalityGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		NetworkCTPAbstractNameSetType networkCTPInLinkEndListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOnetworkCTPsInLinkEndListPackage);

		void assignNetworkCTPOnLogicalLinkEnd
			(in MONameType name,
			in NetworkCTPAbstractNameSetType
				requestedNetworkCTPList,
			out NetworkCTPAbstractNameSetType
				resultingNetworkCTPList)
			raises (itut_x780::ApplicationError,
				LinkEndAndNetworkCTPNotCompatible,
				InvalidNetworkCTP,
				NotEnoughNetworkCTP,
				NetworkCTPAlreadyAssigned,
				InconsistentSignalIdentification,
				InconsistentDirectionality,
				FailureToSetNetworkCTPCallerId,
				FailureToDecreaseCapacity);

		void deassignNetworkCTPFromLogicalLinkEnd
			(in MONameType name,
			in NetworkCTPAbstractNameSetType
				requestedNetworkCTPList)
			raises (itut_x780::ApplicationError,
				LinkEndAndNetworkCTPNotCompatible,
				InvalidNetworkCTP,
				NotAssignedToCaller,
				FailureToDeassignNetworkCTP,
				FailureToIncreaseCapacity);

	}; // interface LogicalLinkEnd_F


/**
```

6.6.2.2 TopLinkEnd_F (Topological Link End)

```
*/


	interface TopLinkEnd_F : AbstractLinkEnd_F
	{

		PointDirectionalityType pointDirectionalityGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		NetworkTTPAbstractNameSetType serverTTPPointerGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		PointCapacityType totalLinkEndCapacityGet
			(in MONameType name)
```

```
                raises (itut_x780::ApplicationError,
                        NOtotalLinkEndCapacityPackage);

        long maximumNetworkCTPCountGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOmaximumNetworkCTPCountPackage);

        PointCapacityType potentialLinkEndCapacityPackageGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOpotentialLinkEndCapacityPackage);

        PointCapacityType provisionedLinkEndCapacityGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOprovisionedLinkEndCapacityPackage);

        long provisionedNetworkCTPCountGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOprovisionedNetworkCTPCountPackage);


        void addCapacityToTopLinkEnd
                (in MONameType name,
                in RequestedPointCapacityType requestedCapacity,
                out PointCapacityType resultingCapacity,
                out NetworkCTPAbstractNameSetType
                        resultingNetworkCTPList,
                out PointCapacityType resultingProvisionedCapacity)
                raises (itut_x780::ApplicationError,
                        NoSuchLinkEnd,
                        InsufficientCapacity,
                        InvalidChannelsNumber,
                        ChannelsAlreadyProvisioned,
                        FailureToCreateLC,
                        FailureToAssociateLC,
                        FailureToSupportLC,
                        FailureToIncreaseCapacity);

        void removeCapacityFromTopLinkEnd
                (in MONameType name,
                in RequestedPointCapacityType requestedCapacity,
                out PointCapacityType resultingCapacity,
                out LinkConnectionNameSetType
                        resultingLinkConnectionList)
                raises (itut_x780::ApplicationError,
                        NoSuchLinkEnd,
                        InsufficientCapacity,
                        InvalidChannelsNumber,
                        FailureToRemoveLC,
                        FailureToDecreaseCapacity);

    };  // interface TopLinkEnd_F

/**
```

6.6.3 AccessGroup_F

```
*/


    interface AccessGroup_F: itut_x780::ManagedObject_F
    {
        NetworkTPNameSetType accessPointListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);
```

```
void accessPointListSet
       (in MONameType name,
       in NetworkTPNameSetType accessPointList)
       raises (itut_x780::ApplicationError,
              NetworkTTPAndAccessGroupNotCompatible,
              FailureToAssociateNetworkTTP,
              FailureToDisassociateNetworkTTP);

void accessPointListAdd
       (in MONameType name,
       in NetworkTPNameSetType accessPointList)
       raises (itut_x780::ApplicationError,
              NetworkTTPAndAccessGroupNotCompatible,
              FailureToAssociateNetworkTTP);

void accessPointListRemove
       (in MONameType name,
       in NetworkTPNameSetType accessPointList)
       raises (itut_x780::ApplicationError,
              NetworkTTPAndAccessGroupNotCompatible,
              FailureToDisassociateNetworkTTP);

TopEndDirectionalityType topEndDirectionalityGet
       (in MONameType name)
       raises (itut_x780::ApplicationError);


SignalIdType signalIdGet
       (in MONameType name)
       raises (itut_x780::ApplicationError);

SubnetworkNameSetType containedInSubnetworkListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

void containedInSubnetworkListSet
       (in MONameType name,
       in SubnetworkNameSetType containedInSubnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

void containedInSubnetworkListAdd
       (in MONameType name,
       in SubnetworkNameSetType containedInSubnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

void containedInSubnetworkListRemove
       (in MONameType name,
       in SubnetworkNameSetType containedInSubnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

AbstractLinkNameSetType linkPointerListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOlinkPointerListPackage);

Istring userLabelGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOuserLabelPackage);

void userLabelSet
       (in MONameType name,
       in Istring label)
```

```
                raises (itut_x780::ApplicationError,
                       NOuserLabelPackage);

        }; // interface AccessGroup_F


/**
```

6.6.4 AlarmSeverityAssignmentProfile_F

```
*/

        interface AlarmSeverityAssignmentProfile_F: itut_x780::ManagedObject_F
        {
                AlarmSeverityAssignmentSetType alarmSeverityAssignmentListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                void alarmSeverityAssignmentsAdd
                        (in MONameType name,
                        in AlarmSeverityAssignmentSetType
                                alarmSeverityAssignmentList)
                        raises (itut_x780::ApplicationError);

                void alarmSeverityAssignmentsRemove
                        (in MONameType name,
                        in AlarmSeverityAssignmentSetType
                                alarmSeverityAssignmentList)
                        raises (itut_x780::ApplicationError);

                void alarmSeverityAssignmentListSet
                        (in MONameType name,
                        in AlarmSeverityAssignmentSetType
                                alarmSeverityAssignmentList)
                        raises (itut_x780::ApplicationError);

                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectCreation,
                                objectManagementNotificatoinsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion,
                                objectManagementNotificatoinsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, attributeValueChange,
                                objectManagementNotificatoinsPackage)

        }; // interface AlarmSeverityAssignmentProfile_F


/**
```

6.6.5 ArcIntervalProfile_F (Alarm Report Control Interval Profile)

```
*/


        interface ArcIntervalProfile_F: itut_x780::ManagedObject_F
        {
                ArcTimeType arcDefaultTimedIntervalGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                void arcDefaultTimedIntervalSet
                        (in MONameType name,
                        in ArcTimeType arcDefaultTimedInterval)
```

```
                raises (itut_x780::ApplicationError);

        ArcTimeType arcDefaultPersistenceIntervalGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        void arcDefaultPersistenceIntervalSet
                (in MONameType name,
                in ArcTimeType arcDefaultPersistenceInterval)
                raises (itut_x780::ApplicationError);

        Istring userLabelGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        void userLabelSet
                (in MONameType name,
                in Istring userLabel)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)

};  // interface ArcIntervalProfile_F

/**
```

6.6.6 CircuitEndPointSubgroup_F

```
*/


interface CircuitEndPointSubgroup_F: itut_x780::ManagedObject_F
{
        long numberOfCircuitsGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        Istring labelOfFarEndExchangeGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        SignallingCapabilityType signallingCapabilityGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        InformationTransferCapabilityType
                informationTransferCapabilityGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        CircuitDirectionalityType circuitDirectionalityGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        TransmissionCharacteristicSetType
                transmissionCharacteristicsGet
                (in MONameType name)
```

```
                raises (itut_x780::ApplicationError);

        Istring userLabelGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        void userLabelSet
                (in MONameType name,
                in Istring userLabel)
                raises (itut_x780::ApplicationError);

        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)

};  // interface CircuitEndPointSubgroup_F

/**
```

6.6.7 CrossConnection_F

```
*/


interface CrossConnection_F : itut_x780::ManagedObject_F
{
        AdministrativeStateType administrativeStateGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        void administrativeStateSet
                (in MONameType name,
                in AdministrativeStateType adminstrativeState)
                raises (itut_x780::ApplicationError);

        OperationalStateType operationalStateGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        SignalIdType signalIdGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        MONameType fromTerminationGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        MONameType toTerminationGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        DirectionalityType directionalityGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        boolean redlineGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOredlinePackage);

        void redlineSet
                (in MONameType name,
                in boolean redline)
                raises (itut_x780::ApplicationError,
```

```
                        NOredlinePackage);

                Istring userLabelGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                void userLabelSet
                        (in MONameType name,
                        in Istring userLable)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                Istring crossConnectionNameGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOnamedCrossConnectionPackage);

                void crossConnectionNameSet
                        (in MONameType name,
                        in Istring crossConnectionName)
                        raises (itut_x780::ApplicationError,
                                NOnamedCrossConnectionPackage);

                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectCreation,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, attributValueChange,
                        attributeValueChangeNotificationPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, stateChange,
                        stateChangeNotificationPackage)

        }; // interface CrossConnection_F


/**
```

6.6.8 Equipment_F

```
*/


        interface Equipment_F: itut_x780::ManagedObject_F
        {
                ReplaceableType replaceableGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                Istring serialNumberGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                Istring typeGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                MONameSetType supportedByObjectListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);
```

```
void supportedByObjectListSet
       (in MONameType name,
       in MONameSetType objectList)
       raises (itut_x780::ApplicationError);

void supportedByObjectListAdd
       (in MONameType name,
       in MONameSetType objectList)
       raises (itut_x780::ApplicationError);

void supportedByObjectListRemove
       (in MONameType name,
       in MONameSetType objectList)
       raises (itut_x780::ApplicationError);

AlarmStatusType alarmStatusGet
       (in MONameType name)
       raises (itut_x780::ApplicationError);

CurrentProblemSetType currentProblemListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError);

OperationalStateType operationalStateGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOadministrativeOperationalStatesPackage);

AdministrativeStateType administrativeStateGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOadministrativeOperationalStatesPackage);

void administrativeStateSet
       (in MONameType name,
       in AdministrativeStateType administrativeState)
       raises (itut_x780::ApplicationError,
              NOadministrativeOperationalStatesPackage);

MONameSetType affectedObjectsGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOaffectedObjectListPackage);

AlarmSeverityAssignmentProfileNameType
       alarmSeverityAssignmentProfilePointerGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOalarmSeverityAssignmentPointerPackage);

void alarmSeverityAssignmentProfilePointerSet
       (in MONameType name,
       in AlarmSeverityAssignmentProfileNameType profile)
       raises (itut_x780::ApplicationError,
              NOalarmSeverityAssignmentPointerPackage);

Istring userLabelGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOuserLabelPackage);

void userLabelSet
       (in MONameType name,
       in Istring userLabel)
       raises (itut_x780::ApplicationError,
              NOuserLabelPackage);
```

```
Istring vendorNameGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOvendorNamePackage);

void vendorNameSet
       (in MONameType name,
       in Istring vendorName)
       raises (itut_x780::ApplicationError,
              NOvendorNamePackage);

Istring versionGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOversionPackage);

void versionSet
       (in MONameType name,
       in Istring version)
       raises (itut_x780::ApplicationError,
              NOversionPackage);

Istring locationNameGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOlocationNamePackage);

void locationNameSet
       (in MONameType name,
       in Istring locationName)
       raises (itut_x780::ApplicationError,
              NOlocationNamePackage);

ArcStateType arcStateGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcQIStatusType     arcQIStatusGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcProbableCauseSetType    arcProbableCauseListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

void arcProbableCauseListSet
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NotSupportedProbableCause,
              NOarcPackage);

void arcProbableCauseListAdd
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NotSupportedProbableCause,
              NOarcPackage);

void arcProbableCauseListRemove
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NOarcPackage);
```

```
            ArcIntervalProfileNameType arcIntervalProfilePointerGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            void arcIntervalProfilePointerSet
                  (in MONameType name,
                  in ArcIntervalProfileNameType profile)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            ArcTimeType arcManagementRequestedIntervalGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            void arcManagementRequestedIntervalSet
                  (in MONameType name,
                  in ArcTimeType time)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            ArcTimeType arcTimeRemainingGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            boolean arcControl
                  (in MONameType name,
                  in ArcControlRequestType request)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            ArcAlarmDetailSetType arcRetrieveAlarmDetail
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);


            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, equipmentAlarm,
                  equipmentsEquipmentAlarmR2Package)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, environmentalAlarm,
                  environmentalAlarmR2Package)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, processingErrorAlarm,
                  processingErrorAlarmR2Package)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, communicationsAlarm,
                  tmnCommunicationsAlarmInformationR1Package)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)

      }; // interface Equipment_F

/**
```

6.6.8.1 CircuitPack_F

```
*/

        interface CircuitPack_F: Equipment_F
        {
                AvailabilityStatusSetType availabilityStatusGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                short numOfPortsGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOnumberOfPortPackage);

                PortAssociationSetType portAssociationListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOportAssociationsPackage);

                SignalRateSetType availableSignalRateSetTypeGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOcircuitPackConfigurationPackage);

                PortSignalRateAndMappingSetType
                        portSignalRateAndMappingListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOcircuitPackConfigurationPackage);

                void portSignalRateAndMappingListSet
                        (in MONameType name,
                        in PortSignalRateAndMappingSetType rateAndMappingList)
                        raises (itut_x780::ApplicationError,
                                ServiceAffectedError,
                                NOcircuitPackConfigurationPackage);

                void portSignalRateAndMappingListAdd
                        (in MONameType name,
                        in PortSignalRateAndMappingSetType rateAndMappingList)
                        raises (itut_x780::ApplicationError,
                                ServiceAffectedError,
                                NOcircuitPackConfigurationPackage);

                void portSignalRateAndMappingListRemove
                        (in MONameType name,
                        in PortSignalRateAndMappingSetType rateAndMappingList)
                        raises (itut_x780::ApplicationError,
                                ServiceAffectedError,
                                NOcircuitPackConfigurationPackage);

                IstringSetType acceptableCircuitPackTypeListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOcontainedBoardPackage);

                void acceptableCircuitPackTypeListSet
                        (in MONameType name,
                        in IstringSetType acceptableCircuitPackTypeList)
                        raises (itut_x780::ApplicationError,
                                NOcontainedBoardPackage);

                void acceptableCircuitPackTypeListAdd
                        (in MONameType name,
                        in IstringSetType acceptableCircuitPackTypeList)
                        raises (itut_x780::ApplicationError,
```

```
                        NOcontainedBoardPackage);

                void acceptableCircuitPackTypeListRemove
                        (in MONameType name,
                        in IstringSetType acceptableCircuitPackTypeList)
                        raises (itut_x780::ApplicationError,
                                NOcontainedBoardPackage);

                boolean reset
                        (in MONameType name,
                        in short resetLevel)
                        raises (itut_x780::ApplicationError,
                                CircuitPackResetError,
                                NOcircuitPackResetPackage);

                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectCreation)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, staeChange)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, equipmentAlarm)

        }; // interface CircuitPack_F

/**
```

6.6.8.2 EquipmentHolder_F

```
*/


        interface EquipmentHolder_F: Equipment_F
        {
                IstringSetType equipmentHolderAddressGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                IstringSetType acceptableCircuitPackTypeListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOsubordinateCircuitPackPackage);

                void acceptableCircuitPackTypeListSet
                        (in MONameType name,
                        in IstringSetType list)
                        raises (itut_x780::ApplicationError,
                                NOsubordinateCircuitPackPackage);

                void acceptableCircuitPackTypeListAdd
                        (in MONameType name,
                        in IstringSetType list)
                        raises (itut_x780::ApplicationError,
                                NOsubordinateCircuitPackPackage);

                void acceptableCircuitPackTypeListRemove
                        (in MONameType name,
                        in IstringSetType list)
                        raises (itut_x780::ApplicationError,
                                NOsubordinateCircuitPackPackage);

                HolderStatusType holderStatusGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOsubordinateCircuitPackPackage);
```

```
            SoftwareNameSetType subordinateCircuitPackSoftwareLoadGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOsubordinateCircuitPackPackage);

            void subordinateCircuitPackSoftwareLoadSet
                  (in MONameType name,
                  in SoftwareNameSetType list)
                  raises (itut_x780::ApplicationError,
                        NOsubordinateCircuitPackPackage);

      }; // interface EquipmentHolder_F


/**
```

6.6.9 ExternalPoint_F

```
*/

      interface ExternalPoint_F: itut_x780::ManagedObject_F
      {

            OperationalStateType operationalStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            AdministrativeStateType administrativeStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            void administrativeStateSet
                  (in MONameType name,
                  in AdministrativeStateType adminstrativeState)
                  raises (itut_x780::ApplicationError);

            MONameSetType supportedByObjectListGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            long externalPointIdGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            Istring externalPointMessageGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            void externalPointMessageSet
                  (in MONameType name,
                  in Istring message)
                  raises (itut_x780::ApplicationError);

            Istring locationNameGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

            void locationNameSet
                  (in MONameType name,
                  in Istring locationName)
                  raises (itut_x780::ApplicationError,
                        NOlocationNamePackage);

            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, objectCreation)
            MANDATORY_NOTIFICATION(
```

```
                itut_x780::Notifications, objectDeletion)
            MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)
            MANDATORY_NOTIFICATION(
                itut_x780::Notifications, stateChange)

      }; // interface ExternalPoint_F

/**
```

6.6.9.1 ControlPoint_F

```
*/


      interface ControlPoint_F: ExternalPoint_F
      {
            ControlStateType currentControlStateGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

            ValidControlType validControlGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

            void validControlSet
                (in MONameType name,
                in ValidControlType validControl)
                raises (itut_x780::ApplicationError);

            ControlStateType normalControlStateGet
                (in MONameType name)
                raises (NOnormalControlStatePackage);

            void normalControlStateSet
                (in MONameType name,
                in ControlStateType controlState)
                raises (itut_x780::ApplicationError,
                      NOnormalControlStatePackage);

            ControlResultType externalControl
                (in MONameType name,
                in ControlActionType controlAction)
                raises (itut_x780::ApplicationError);

      }; // interface ControlPoint_F

/**
```

6.6.9.2 ScanPoint_F

```
*/


      interface ScanPoint_F: ExternalPoint_F
      {
            CurrentProblemSetType currentProblemListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

            boolean serviceAffectedGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

            AlarmSeverityAssignmentProfileNameType
```

```
                alarmSeverityAssignmentProfilePointerGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

        void alarmSeverityAssignmentProfilePointerSet
                (in MONameType name,
                in AlarmSeverityAssignmentProfileNameType profile)
                raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, environmentalAlarm)

    }; // interface ScanPoint_F


/**
```

6.6.10 Fabric_F

```
*/


    interface Fabric_F: itut_x780::ManagedObject_F
    {
        OperationalStateType operationalStateGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        AdministrativeStateType administrativeStateGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        void administrativeStateSet
                (in MONameType name,
                in AdministrativeStateType adminstrativeState)
                raises (itut_x780::ApplicationError);

        AvailabilityStatusSetType availabilityStatusGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        CharacteristicInfoSetType characteristicInfoListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        MONameSetType supportedByObjectListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        void supportedByObjectListSet
                (in MONameType name,
                in MONameSetType objectList)
                raises (itut_x780::ApplicationError);

        void supportedByObjectListAdd
                (in MONameType name,
                in MONameSetType objectList)
                raises (itut_x780::ApplicationError);

        void supportedByObjectListRemove
                (in MONameType name,
                in MONameSetType objectList)
                raises (itut_x780::ApplicationError);
```

```
            void addTPsToGTP
                  (in MONameType name,
                  in TPsAddToGTPInfoSeqType request,
                  out TPsAddToGTPResultSeqType result)
                  raises (itut_x780::ApplicationError);

            void removeTPsFromGTP
                  (in MONameType name,
                  in TPsRemoveInfoSeqType request,
                  out TPsRemoveResultSeqType result)
                  raises (itut_x780::ApplicationError);

            void addTPsToTPPool
                  (in MONameType name,
                  in TPsAddToTPPoolInfoSeqType request,
                  out TPsAddToTPPoolResultSeqType result)
                  raises (itut_x780::ApplicationError);
            void removeTPsFromTPPool
                  (in MONameType name,
                  in TPsRemoveInfoSeqType request,
                  out TPsRemoveResultSeqType result)
                  raises (itut_x780::ApplicationError);

            void connect
                  (in MONameType name,
                  in ConnectInfoSeqType request,
                  out ConnectResultSeqType result)
                  raises (itut_x780::ApplicationError);

            void disconnect
                  (in MONameType name,
                  in MONameSeqType tps,
                  out DisconnectResultSeqType result)
                  raises (itut_x780::ApplicationError);

            void switchover
                  (in MONameType name,
                  in ConnectSwitchOverInfoSeqType request,
                  out ConnectResultSeqType result)
                  raises (itut_x780::ApplicationError);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)

      }; // interface Fabric_F

/**
```

6.6.11 GTP_F (Group Termination Point)

```
*/

	interface GTP_F: itut_x780::ManagedObject_F
	{
		CrossConnectionPointerType crossConnectionPointerTypeGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		FabricNameType fabricPointerGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		SignalIdType signalIdTypeGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		TPNameSeqType tpsInGTPListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

	}; // interface GTP_F

/**
```

6.6.12 ManagedElement_F

```
*/


	interface ManagedElement_F: itut_x780::ManagedObject_F
	{
		AlarmStatusType alarmStatusGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		CurrentProblemSetType currentProblemListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		AdministrativeStateType administrativeStateGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		void administrativeStateSet
			(in MONameType name,
			in AdministrativeStateType administrativeState)
			raises (itut_x780::ApplicationError);

		OperationalStateType operationalStateGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		UsageStateType usageStateGet
			(in MONameType name)
			raises (itut_x780::ApplicationError);

		boolean enableAudibleVisualLocalAlarmGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOaudibleVisualLocalAlarmPackage);
```

```
void enableAudibleVisualLocalAlarmSet
       (in MONameType name,
       in boolean enable)
       raises (itut_x780::ApplicationError,
              NOaudibleVisualLocalAlarmPackage);

void resetAudibleAlarm
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOaudibleVisualLocalAlarmPackage);

AlarmSeverityAssignmentProfileNameType
       alarmSeverityAssignmentProfilePointerGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOalarmSeverityAssignmentPointerPackage);

void alarmSeverityAssignmentProfilePointerSet
       (in MONameType name,
       in AlarmSeverityAssignmentProfileNameType profile)
       raises (itut_x780::ApplicationError,
              NOalarmSeverityAssignmentPointerPackage);

Istring userLabelGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOuserLabelPackage);

void userLabelSet
       (in MONameType name,
       in Istring userLabel)
       raises (itut_x780::ApplicationError,
              NOuserLabelPackage);

Istring vendorNameGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOvendorNamePackage);

void vendorNameSet
       (in MONameType name,
       in Istring vendorName)
       raises (itut_x780::ApplicationError,
              NOvendorNamePackage);

Istring versionGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOversionPackage);

void versionSet
       (in MONameType name,
       in Istring version)
       raises (itut_x780::ApplicationError,
              NOversionPackage);

Istring locationNameGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOlocationNamePackage);

void locationNameSet
       (in MONameType name,
       in Istring locationName)
       raises (itut_x780::ApplicationError,
              NOlocationNamePackage);
```

```
ExternalTimeType externalTimeGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOexternalTimePackage);

void externalTimeSet
       (in MONameType name,
       in ExternalTimeType externalTime)
       raises (itut_x780::ApplicationError,
              NOexternalTimePackage);

SystemTimingSourceType systemTimingSourceGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOsystemTimingSourcePackage);

void systemTimingSourceSet
       (in MONameType name,
       in SystemTimingSourceType systemTimingSource)
       raises (itut_x780::ApplicationError,
              NOsystemTimingSourcePackage);

ArcStateType arcStateGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcQIStatusType     arcQIStatusGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcProbableCauseSetType    arcProbableCauseListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

void arcProbableCauseListSet
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NotSupportedProbableCause,
              NOarcPackage);

void arcProbableCauseListAdd
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NotSupportedProbableCause,
              NOarcPackage);

void arcProbableCauseListRemove
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcIntervalProfileNameType arcIntervalProfilePointerGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

void arcIntervalProfilePointerSet
       (in MONameType name,
       in ArcIntervalProfileNameType profile)
       raises (itut_x780::ApplicationError,
              NOarcPackage);
```

```
            ArcTimeType arcManagementRequestedIntervalGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            void arcManagementRequestedIntervalSet
                  (in MONameType name,
                  in ArcTimeType time)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            ArcTimeType arcTimeRemainingGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            boolean arcControl
                  (in MONameType name,
                  in ArcControlRequestType request)
                  raises (itut_x780::ApplicationError,
                        NOarcPackage);

            ArcAlarmDetailSetType arcRetrieveAlarmDetail
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);

            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, environmentalAlarm)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, equipmentAlarm)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, communicationsAlarm)
            MANDATORY_NOTIFICATION(
                  itut_x780::Notifications, processingErrorAlarm)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)

      }; // interface ManagedElement_F




/**
```

6.6.13 ManagedElementComplex_F

```
*/


      interface ManagedElementComplex_F: itut_x780::ManagedObject_F
      {
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
```

```
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
      }; // interface ManagedElementComplex_F

/**
```

6.6.14 MPCrossConnection_F (Multipoint Cross Connection)

```
*/

      interface MPCrossConnection_F: itut_x780::ManagedObject_F
      {
            OperationalStateType operationalStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            AdministrativeStateType administrativeStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            void administrativeStateSet
                  (in MONameType name,
                  in AdministrativeStateType administrativeState)
                  raises (itut_x780::ApplicationError);

            AvailabilityStatusSetType availabilityStatusGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            SignalIdType signalIdTypeGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            MONameType fromTerminationGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);
                  // the return value may be nil

            boolean redlineGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOredlinePackage);

            void redlineSet
                  (in MONameType name,
                  in boolean redline)
                  raises (itut_x780::ApplicationError,
                        NOredlinePackage);

            Istring userLabelGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

            void userLabelSet
                  (in MONameType name,
                  in Istring userLable)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

            Istring crossConnectionNameGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOnamedCrossConnectionPackage);
```

```
            void crossConnectionNameSet
                  (in MONameType name,
                  in Istring crossConnectionName)
                  raises (itut_x780::ApplicationError,
                        NOnamedCrossConnectionPackage);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)

      }; // interface MPCrossConnection_F


/**
```

6.6.15 Network_F

```
*/

      interface Network_F: itut_x780::ManagedObject_F
      {
            Istring userLabelGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            void userLabelSet
                  (in MONameType name,
                  in Istring userLabel)
                  raises (itut_x780::ApplicationError);

      }; // interface Network_F


/**
```

6.6.15.1 LayerND_F (Layer Network Domain)

```
*/

      interface LayerND_F: Network_F
      {
            SignalIdType signalIdGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

      }; // interface LayerND_F


/**
```

6.6.15.1.1 BasicLayerND_F (Basic Layer Network Domain)

```
*/

      interface BasicLayerND_F: LayerND_F
      {
```

```
void setupTrail
       (in MONameType name,
       in ConnectivityEndPointSetType aEnd,
       in ConnectivityEndPointSetType zEnd,
       in DirectionalityType direction,
       in MONameType qos, // may be empty
       in Istring userId, // may be empty string
       in Istring userLabel, // may be empty string
       out TrailNameType resultTrail,
       out ConnectivityEndPointSetType resultAEnd,
       out ConnectivityEndPointSetType resultZEnd)
       raises (itut_x780::ApplicationError,
              NetworkTTPsNotPartOfLayerND,
              AEndNetworkTPConnected,
              NetworkTTPsInAEndAccessGroupConnected,
              ZEndNetworkTPConnected,
              NetworkTTPsInZEndAccessGroupConnected,
              UserIdentifierNotUnique,
              FailureToSetUserIdentifier,
              InvalidTPType,
              InvalidTrail,
              WrongAEndDirectionality,
              WrongZEndDirectionality,
              InvalidTransportServiceCharacteristics,
              InvalidTrafficDescriptor);

void releaseTrail
       (in MONameType name,
       in TrailNameType atrail,
       in Istring userId) // may be empty string
       raises (itut_x780::ApplicationError,
              UnknownTrail,
              TrailConnected);

void establishLogicalLink
       (in MONameType name,
       in LinkEndType aEnd,
       in LinkEndType zEnd,
       in Istring userId, // may be empty string
       in Istring userLable, // may be empty string
       in LinkDirectionalityTypeOpt direction,
       out LogicalLinkNameType link)
       raises (itut_x780::ApplicationError,
              IncorrectLinkEnds,
              UserIdentifierNotUnique,
              FailureToSetUserIdentifier,
              FailureToCreateLink,
              FailureToBindLink,
              FailureToSetDirectionality,
              NOlogicalLinkHandlerPackage);

void removeLogicalLink
       (in MONameType name,
       in LogicalLinkNameType link)
       raises (itut_x780::ApplicationError,
              IncorrectLink,
              LinkConnectionsExisting,
              FailureToDeleteLink,
              NOlogicalLinkHandlerPackage);

void establishLogicalLinkAndEnds
       (in MONameType name,
       in LinkEndType aEnd,
       in LinkEndType zEnd,
       in Istring userId, // may be empty string
       in Istring userLable, // may be empty string
       in LinkDirectionalityTypeOpt direction,
       out LogicalLinkNameType link,
```

```
        out LogicalLinkEndNameType resultAEnd,
        out LogicalLinkEndNameType resultZEnd)
        raises (itut_x780::ApplicationError,
                UserIdentifierNotUnique,
                IncorrectSubnetwork,
                FailureToCreateLinkEnd,
                FailureToBindLinkEnd,
                FailureToSetUserIdentifier,
                FailureToSetDirectionality,
                NOlogicalLinkEndHandlerPackage);

    void removeLogicalLinkAndEnds
        (in MONameType name,
        in LogicalLinkNameType link)
        raises (itut_x780::ApplicationError,
                IncorrectLink,
                IncorrectLinkEnds,
                NetworkCTPsExisting,
                LinkConnectionsExisting,
                FailureToDeleteLink,
                NOlogicalLinkEndHandlerPackage);

    void associateTrailWithTopologicalLink
        (in MONameType name,
        in TopLinkNameType link,
        in TrailNameType atrail,
        out CapacityType potentialCapacity,
        out LinkConnectionNameSetType resultLCs)
        raises (itut_x780::ApplicationError,
                NoSuchLink,
                NoSuchTrail,
                LinkAndTrailsNotCompatible,
                InitialCapacitiesFailure,
                TrailAlreadyAssociated,
                FinalCapacitiesFailure,
                ConsistencyFailure,
                FailureToAssociate,
                NOtopologicalLinkHandlerPackage);

    void disassociateTrailFromTopologicalLink
        (in MONameType name,
        in TopLinkNameType link,
        in TrailNameType atrail) // may be nil
        raises (itut_x780::ApplicationError,
                NoSuchLink,
                NoSuchTrail,
                TrailNotAssociated,
                CapacityProvisionned,
                FinalCapacitiesFailure,
                FailureToDisassociate,
                NOtopologicalLinkHandlerPackage);

    void establishTopologicalLink
        (in MONameType name,
        in LinkEndType aEnd,
        in LinkEndType zEnd,
        in Istring userId, // may be empty string
        in Istring userLabel, // may be empty string
        in DirectionalityTypeOpt direction,
        out TopLinkNameType link)
        raises (itut_x780::ApplicationError,
                IncorrectLinkEnds,
                UserIdentifierNotUnique,
                FailureToSetUserIdentifier,
                FailureToCreateTopologicalLink,
                FailureToBindTopologicalLink,
                FailureToSetDirectionality,
                NOtopologicalLinkHandlerPackage);
```

```
void removeTopologicalLink
       (in MONameType name,
       in TopLinkNameType link)
       raises (itut_x780::ApplicationError,
              NoSuchLink,
              LinkConnectionsExisting,
              FailureToDeleteLink,
              NOtopologicalLinkHandlerPackage);

void associateNetworkTTPWithTopologicalLinkEnd
       (in MONameType name,
       in TopLinkEndNameType le,
       in NetworkTTPAbstractNameType ttp,
       out CapacityType potentialCapacity,
       out NetworkCTPAbstractNameSetType ctpList)
       raises (itut_x780::ApplicationError,
              NoSuchLinkEnd,
              NoSuchNetworkTTP,
              LinkEndAndNetworkTTPsNotCompatible,
              InitialCapacitiesFailure,
              NetworkTTPAlreadyAssociated,
              FinalCapacitiesFailure,
              ConsistencyFailure,
              FailureToAssociate,
              NOtopologicalLinkEndHandlerPackage);

void disassociateNetworkTTPFromTopologicalLinkEnd
       (in MONameType name,
       in TopLinkEndNameType le,
       in NetworkTTPAbstractNameType ttp) // may be nil
       raises (itut_x780::ApplicationError,
              NoSuchLinkEnd,
              NoSuchNetworkTTP,
              NetworkTTPNotAssociated,
              CapacityProvisionned,
              FinalCapacitiesFailure,
              NOtopologicalLinkEndHandlerPackage);

void establishTopologicalLinkAndEnds
       (in MONameType name,
       in LinkEndType aEnd,
       in LinkEndType zEnd,
       in Istring userId, // may be empty string
       in Istring userLabel, // may be empty string
       in DirectionalityTypeOpt direction,
       out TopLinkNameType link,
       out TopLinkEndNameType resultAEnd,
       out TopLinkEndNameType resultZEnd)
       raises (itut_x780::ApplicationError,
              IncorrectLinkEnds,
              UserIdentifierNotUnique,
              IncorrectSubnetwork,
              FailureToCreateLinkEnd,
              FailureToBindLinkEnd,
              FailureToSetUserIdentifier,
              FailureToSetDirectionality,
              NOtopologicalLinkEndHandlerPackage);

void removeTopologicalLinkAndEnds
       (in MONameType name,
       in TopLinkNameType link)
       raises (itut_x780::ApplicationError,
              NoSuchLinkEnd,
              NetworkCTPsExisting,
              FailureToDeleteTopologicalLinkEnd,
              NOtopologicalLinkEndHandlerPackage);
```

```
};  // interface BasicLayerND_F

/**
```

6.6.16 Pipe_F

```
*/

        interface Pipe_F : itut_x780::ManagedObject_F
        {
                DirectionalityType directionalityGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                SignalIdType signalIdGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                MONameSetType aEndNetworkTPListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                MONameSetType zEndNetworkTPListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);

                AdministrativeStateType administrativeStateGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOadministrativeStatePackage);

                void administrativeStateSet
                        (in MONameType name,
                        in AdministrativeStateType administrativeState)
                        raises (itut_x780::ApplicationError,
                                NOadministrativeStatePackage);

                OperationalStateType operationalStateGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOoperationalStatePackage);

                AlarmSeverityAssignmentProfileNameType
                        alarmSeverityAssignmentProfilePointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOalarmSeverityAssignmentPointerPackage);

                void alarmSeverityAssignmentProfilePointerSet
                        (in MONameType name,
                        in AlarmSeverityAssignmentProfileNameType profile)
                        raises (itut_x780::ApplicationError,
                                NOalarmSeverityAssignmentPointerPackage);

                AvailabilityStatusSetType availabilityStatusGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOavailabilityStatusPackage);

                boolean protectedGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOprotectedPackage);

                MONameType quailityOfConnectivityServiceGet
                        (in MONameType name)
```

```
                raises (itut_x780::ApplicationError,
                        NOquailityOfConnectivityServicePackage);

        MONameSetType supportedByObjectListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOsupportedByPackage);

        void supportedByObjectListSet
                (in MONameType name,
                in MONameSetType objectList)
                raises (itut_x780::ApplicationError,
                        NOsupportedByPackage);

        void supportedByObjectListAdd
                (in MONameType name,
                in MONameSetType objectList)
                raises (itut_x780::ApplicationError,
                        NOsupportedByPackage);

        void supportedByObjectListRemove
                (in MONameType name,
                in MONameSetType objectList)
                raises (itut_x780::ApplicationError,
                        NOsupportedByPackage);

        AlarmStatusType alarmStatusGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOtmnCommunicationsAlarmInformationR1Package);

        CurrentProblemSetType currentProblemListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOtmnCommunicationsAlarmInformationR1Package);

        Istring userLabelGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        void userLabelSet
                (in MONameType name,
                in Istring userLabel)
                raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, communicationAlarm,
                tmnCommunicationsAlarmInformationR1Package)

    }; // interface Pipe_F

/**
```

6.6.16.1 LinkConnection_F

```
*/


	interface LinkConnection_F: Pipe_F
	{
		TrailNameSetType serverTrailListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOserverTrailListPackage);

		SNCNameType compositePointerGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOcompositePointerPackage);

		TrailNameType clientTrailGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOclientTrailPackage);

	}; // interface LinkConnection_F


/**
```

6.6.16.2 SNC_F (Subnetwork Connection)

```
*/


	interface SNC_F: Pipe_F
	{
		SNCNameType compositePointerGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOcompositePointerPackage);

		PipeNameSetType componentPointerListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOcomponentPointerPackage);

		MONameType relatedRoutingProfileGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOrelatedRoutingProfilePackage);

	}; // interface SNC_F



/**
```

6.6.16.3 Trail_F

```
*/


	interface Trail_F: Pipe_F
	{
		PipeNameSetType connectionListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOlayerConnectionListPackage);
```

```
            MONameType trafficDescriptorGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOtrafficDescriptorPackage);

            void trafficDescriptorSet
                  (in MONameType name,
                  in MONameType descriptor)
                  raises (itut_x780::ApplicationError,
                        NewServiceCharacteristicsExistsAlready,
                        NewTrafficDescriptorExistsAlready,
                        InvalidServiceCharacteristicsRequested,
                        InvalidTrafficDescriptorRequested,
                        NOtrafficDescriptorPackage);

            TopLinkNameSetType clientLinkPointerListGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOclientLinkPointerPackage);

            LinkConnectionNameSetType clientLinkConnectionPointerListGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOclientLinkConnectionPointerListPackage);

      };  // interface Trail_F


/**
```

6.6.17 Trail95_F

```
*/

      interface Trail95_F: itut_x780::ManagedObject_F
      {
            DirectionalityType directionalityGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            AdministrativeStateType administrativeStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            void administrativeStateSet
                  (in MONameType name,
                  in AdministrativeStateType adminstrativeState)
                  raises (itut_x780::ApplicationError);

            OperationalStateType operationalStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            MONameType aTPGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            MONameType zTPGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            CharacteristicInfoType characteristicInfoGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOcharacteristicInformationPackage);
```

```
            boolean protectedGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOprotectedPackage);

            AlarmStatusType alarmStatusGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOtmnCommunicationsAlarmInformationR1Package);

            CurrentProblemSetType currentProblemListGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOtmnCommunicationsAlarmInformationR1Package);

            AlarmSeverityAssignmentProfileNameType
                  alarmSeverityAssignmentProfilePointerGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

            void alarmSeverityAssignmentProfilePointerSet
                  (in MONameType name,
                  in AlarmSeverityAssignmentProfileNameType profile)
                  raises (itut_x780::ApplicationError,
                        NOalarmSeverityAssignmentPointerPackage);

            Istring userLabelGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

            void userLabelSet
                  (in MONameType name,
                  in Istring userLabel)
                  raises (itut_x780::ApplicationError,
                        NOuserLabelPackage);

            MONameSetType serverConnectionListGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOserverConnectionListPackage);

            MONameSetType clientConnectionListGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError,
                        NOclientConnectionListPackage);

            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, attributeValueChange,
                  attributeValueChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectCreation,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, objectDeletion,
                  createDeleteNotificationsPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, stateChange,
                  stateChangeNotificationPackage)
            CONDITIONAL_NOTIFICATION(
                  itut_x780::Notifications, communicationAlarm,
                  tmnCommunicationsAlarmInformationR1Package)

      };  // interface Trail95_F


/**
```

6.6.18 Software_F

```
*/


        interface Software_F: itut_x780::ManagedObject_F
        {
                OperationalStateType operationalStateGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOadministrativeOperationalStatesPackage);

                AdministrativeStateType administrativeStateGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOadministrativeOperationalStatesPackage);

                void administrativeStateSet
                        (in MONameType name,
                        in AdministrativeStateType administrativeState)
                        raises (itut_x780::ApplicationError,
                                NOadministrativeOperationalStatesPackage);

                MONameSetType affectedObjectsGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOaffectedObjectListPackage);

                AlarmSeverityAssignmentProfileNameType
                        alarmSeverityAssignmentProfilePointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOalarmSeverityAssignmentPointerPackage);

                void alarmSeverityAssignmentProfilePointerSet
                        (in MONameType name,
                        in AlarmSeverityAssignmentProfileNameType profile)
                        raises (itut_x780::ApplicationError,
                                NOalarmSeverityAssignmentPointerPackage);

                AlarmStatusType alarmStatusGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOsoftwareProcessingErrorAlarmR2Package);

                CurrentProblemSetType currentProblemListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOcurrentProblemListPackage);

                Istring userLabelGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                void userLabelSet
                        (in MONameType name,
                        in Istring userLabel)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                Istring vendorNameGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOvendorNamePackage);
```

```
                void vendorNameSet
                       (in MONameType name,
                       in Istring vendorName)
                       raises (itut_x780::ApplicationError,
                              NOvendorNamePackage);

                Istring versionGet
                       (in MONameType name)
                       raises (itut_x780::ApplicationError,
                              NOversionPackage);

/**
The following method sets the version of the associated resource.
*/
                void versionSet
                       (in MONameType name,
                       in Istring version)
                       raises (itut_x780::ApplicationError,
                              NOadministrativeOperationalStatesPackage);

                ArcStateType arcStateGet
                       (in MONameType name)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

                ArcQIStatusType     arcQIStatusGet
                       (in MONameType name)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

                ArcProbableCauseSetType    arcProbableCauseListGet
                       (in MONameType name)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

                void arcProbableCauseListSet
                       (in MONameType name,
                       in ArcProbableCauseSetType arcProbableCauseList)
                       raises (itut_x780::ApplicationError,
                              NotSupportedProbableCause,
                              NOarcPackage);

                void arcProbableCauseListAdd
                       (in MONameType name,
                       in ArcProbableCauseSetType arcProbableCauseList)
                       raises (itut_x780::ApplicationError,
                              NotSupportedProbableCause,
                              NOarcPackage);

                void arcProbableCauseListRemove
                       (in MONameType name,
                       in ArcProbableCauseSetType arcProbableCauseList)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

                ArcIntervalProfileNameType arcIntervalProfilePointerGet
                       (in MONameType name)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

                void arcIntervalProfilePointerSet
                       (in MONameType name,
                       in ArcIntervalProfileNameType profile)
                       raises (itut_x780::ApplicationError,
                              NOarcPackage);

                ArcTimeType arcManagementRequestedIntervalGet
                       (in MONameType name)
```

```
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

        void arcManagementRequestedIntervalSet
                (in MONameType name,
                in ArcTimeType time)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

        ArcTimeType arcTimeRemainingGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

        boolean arcControl
                (in MONameType name,
                in ArcControlRequestType request)
                raises (itut_x780::ApplicationError,
                        NOarcPackage);

        ArcAlarmDetailSetType arcRetrieveAlarmDetail
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);

        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectCreation,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, objectDeletion,
                createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, procesingErrorAlarm,
                softwareProcessingErrorAlarmR2Package)

};  // interface Software_F


/**
```

6.6.19 Subnetwork_F

```
*/


interface Subnetwork_F : itut_x780::ManagedObject_F
{
        SignalIdType signalIdGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        AvailabilityStatusSetType availabilityStatusGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOavailabilityStatusPackage);

        AccessGroupNameSetType containedAccessGroupListGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOcontainedAccessGroupListPackage);
```

```
void containedAccessGroupListSet
       (in MONameType name,
       in AccessGroupNameSetType accessGroupList)
       raises (itut_x780::ApplicationError,
              NOcontainedAccessGroupListPackage);

void containedAccessGroupListAdd
       (in MONameType name,
       in AccessGroupNameSetType accessGroupList)
       raises (itut_x780::ApplicationError,
              NOcontainedAccessGroupListPackage);

void containedAccessGroupListRemove
       (in MONameType name,
       in AccessGroupNameSetType accessGroupList)
       raises (itut_x780::ApplicationError,
              NOcontainedAccessGroupListPackage);

SubnetworkNameSetType containedInSubnetworkListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

void containedInSubnetworkListSet
       (in MONameType name,
       in SubnetworkNameSetType containedInSubnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

void containedInSubnetworkListAdd
       (in MONameType name,
       in SubnetworkNameSetType containedInSubnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

void containedInSubnetworkListRemove
       (in MONameType name,
       in SubnetworkNameSetType containedInSubnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedInSubnetworkListPackage);

AbstractLinkEndNameSetType containedLinkEndListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkEndListPackage);

void containedLinkEndListSet
       (in MONameType name,
       in AbstractLinkEndNameSetType linkEndList)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkEndListPackage);

void containedLinkEndListAdd
       (in MONameType name,
       in AbstractLinkEndNameSetType linkEndList)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkEndListPackage);

void containedLinkEndListRemove
       (in MONameType name,
       in AbstractLinkEndNameSetType linkEndList)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkEndListPackage);

AbstractLinkNameSetType containedLinkListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkListPackage);
```

```
void containedLinkListSet
       (in MONameType name,
       in AbstractLinkNameSetType linkList)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkListPackage);

void containedLinkListAdd
       (in MONameType name,
       in AbstractLinkNameSetType linkList)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkListPackage);

void containedLinkListRemove
       (in MONameType name,
       in AbstractLinkNameSetType linkList)
       raises (itut_x780::ApplicationError,
              NOcontainedLinkListPackage);

NetworkTPNameSetType containedNetworkTPListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcontainedNetworkTPListPackage);

void containedNetworkTPListSet
       (in MONameType name,
       in NetworkTPNameSetType networkTPList)
       raises (itut_x780::ApplicationError,
              NetworkTTPAndSubnetworkNotCompatible,
              FailureToAssociateNetworkTTP,
              FailureToDisassociateNetworkTTP,
              NOcontainedNetworkTPListPackage);

void containedNetworkTPListAdd
       (in MONameType name,
       in NetworkTPNameSetType networkTPList)
       raises (itut_x780::ApplicationError,
              NetworkTTPAndSubnetworkNotCompatible,
              FailureToAssociateNetworkTTP,
              NOcontainedNetworkTPListPackage);

void containedNetworkTPListRemove
       (in MONameType name,
       in NetworkTPNameSetType networkTPList)
       raises (itut_x780::ApplicationError,
              FailureToDisassociateNetworkTTP,
              NOcontainedNetworkTPListPackage);

SubnetworkNameSetType containedSubnetworkListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcontainedSubnetworkListPackage);

void containedSubnetworkListSet
       (in MONameType name,
       in SubnetworkNameSetType subnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedSubnetworkListPackage);

void containedSubnetworkListAdd
       (in MONameType name,
       in SubnetworkNameSetType subnetworkList)
       raises (itut_x780::ApplicationError,
              NOcontainedSubnetworkListPackage);

void containedSubnetworkListRemove
       (in MONameType name,
       in SubnetworkNameSetType subnetworkList)
```

```
        raises (itut_x780::ApplicationError,
                NOcontainedSubnetworkListPackage);

    AbstractLinkNameSetType linkPointerListGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOlinkPointerListPackage);

    MONameSetType supportedByObjectListGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOsupportedByPackage);

    void supportedByObjectListSet
        (in MONameType name,
        in MONameSetType objectList)
        raises (itut_x780::ApplicationError,
                NOsupportedByPackage);

    void supportedByObjectListAdd
        (in MONameType name,
        in MONameSetType objectList)
        raises (itut_x780::ApplicationError,
                NOsupportedByPackage);

    void supportedByObjectListRemove
        (in MONameType name,
        in MONameSetType objectList)
        raises (itut_x780::ApplicationError,
                NOsupportedByPackage);

    AdministrativeStateType administrativeStateGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOadministrativeOperationalStatesPackage);

    void administrativeStateSet
        (in MONameType name,
        in AdministrativeStateType administrativeState)
        raises (itut_x780::ApplicationError,
                NOadministrativeOperationalStatesPackage);

    OperationalStateType operationalStateGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOadministrativeOperationalStatesPackage);

    UsageStateType usageStateGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOusageStatePackage);

    Istring userLabelGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOuserLabelPackage);

    void userLabelSet
        (in MONameType name,
        in Istring userLabel)
        raises (itut_x780::ApplicationError,
                NOuserLabelPackage);

    MANDATORY_NOTIFICATION(
        itut_x780::Notifications, objectCreation)
    MANDATORY_NOTIFICATION(
        itut_x780::Notifications, objectDeletion)
    CONDITIONAL_NOTIFICATION(
```

```
                itut_x780::Notifications, attributeValueChange,
                attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
                itut_x780::Notifications, stateChange,
                stateChangeNotificationPackage)

};  // interface Subnetwork_F


/**
```

6.6.19.1 BasicSubnetwork_F

```
*/

        interface BasicSubnetwork_F: Subnetwork_F
        {
                void setupSnc
                        (in MONameType name,
                        in ConnectivityEndPointSetType aEnd,
                        in ConnectivityEndPointSetType zEnd,
                        in DirectionalityType direction,
                        in SignalIdType signalId, // may be empty sequence
                        in MONameType qos, // may be empty
                        in boolean implicitTPCreation,
                        out SNCNameType connection,
                        out ConnectivityEndPointType resultAEnd,
                        out ConnectivityEndPointType resultZEnd,
                        out Istring userId) // may be empty string
                        raises (itut_x780::ApplicationError,
                                InvalidTransportServiceCharacteristics,
                                IncorrectSubnetworkTerminationPoints,
                                AEndNetworkTPConnected,
                                ZEndNetworkTPConnected,
                                WrongAEndDirectionality,
                                WrongZEndDirectionality,
                                FailureToConnect,
                                FailureToSetUserIdentifier,
                                NObasicConnectionPerformerPackage);

                void releaseSnc
                        (in MONameType name,
                        in SNCNameType connection,
                        in Istring userId) // may be empty string
                        raises (itut_x780::ApplicationError,
                                NoSuchSnc,
                                SncConnected,
                                FailureToRelease,
                                NObasicConnectionPerformerPackage);

        }; // interface BasicSubnetwork_F


/*
```

6.6.20 TP_F (Termination Point)

```
*/


        interface TP_F : itut_x780::ManagedObject_F
        {
                MONameSetType supportedByObjectListGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError);
```

```
OperationalStateType operationalStateGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOoperationalStatePackage);

CrossConnectionPointerType crossConnectionPointerGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcrossConnectionPointerPackage);

CharacteristicInfoType characteristicInfoGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOcharacteristicInformationPackage);

MONameType networkLevelPointerGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOnetworkLevelPackage);

void networkLevelPointerSet
       (in MONameType name,
       in MONameType networkLevelPointer)
       raises (itut_x780::ApplicationError,
              NOnetworkLevelPackage);

AlarmStatusType alarmStatusGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOtmnCommunicationsAlarmInformationR1Package);

CurrentProblemSetType currentProblemListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOtmnCommunicationsAlarmInformationR1Package);

AlarmSeverityAssignmentProfile
       alarmSeverityAssignmentProfilePointerGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOalarmSeverityAssignmentPointerPackage);

void alarmSeverityAssignmentProfilePointerSet
       (in MONameType name,
       in AlarmSeverityAssignmentProfile profile)
       raises (itut_x780::ApplicationError,
              NOalarmSeverityAssignmentPointerPackage);

ArcStateType arcStateGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcQIStatusType     arcQIStatusGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

ArcProbableCauseSetType    arcProbableCauseListGet
       (in MONameType name)
       raises (itut_x780::ApplicationError,
              NOarcPackage);

void arcProbableCauseListSet
       (in MONameType name,
       in ArcProbableCauseSetType arcProbableCauseList)
       raises (itut_x780::ApplicationError,
              NotSupportedProbableCause,
```

```
            NOarcPackage);

void arcProbableCauseListAdd
      (in MONameType name,
      in ArcProbableCauseSetType arcProbableCauseList)
      raises (itut_x780::ApplicationError,
            NotSupportedProbableCause,
            NOarcPackage);

void arcProbableCauseListRemove
      (in MONameType name,
      in ArcProbableCauseSetType arcProbableCauseList)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

ArcIntervalProfileNameType arcIntervalProfilePointerGet
      (in MONameType name)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

void arcIntervalProfilePointerSet
      (in MONameType name,
      in ArcIntervalProfileNameType profile)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

ArcTimeType arcManagementRequestedIntervalGet
      (in MONameType name)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

void arcManagementRequestedIntervalSet
      (in MONameType name,
      in ArcTimeType time)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

ArcTimeType arcTimeRemainingGet
      (in MONameType name)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

boolean arcControl
      (in MONameType name,
      in ArcControlRequestType request)
      raises (itut_x780::ApplicationError,
            NOarcPackage);

ArcAlarmDetailSetType arcRetrieveAlarmDetail
      (in MONameType name)
      raises (itut_x780::ApplicationError,
            NOarcRetrieveAlarmDetailPackage);

CONDITIONAL_NOTIFICATION(
      itut_x780::Notifications, objectCreation,
      createDeleteNotificationsPackage)
CONDITIONAL_NOTIFICATION(
      itut_x780::Notifications, objectDeletion,
      createDeleteNotificationsPackage)
CONDITIONAL_NOTIFICATION(
      itut_x780::Notifications, attributValueChange,
      attributeValueChangeNotificationPackage)
CONDITIONAL_NOTIFICATION(
      itut_x780::Notifications, stateChange,
      stateChangeNotificationPackage)
CONDITIONAL_NOTIFICATION(
      itut_x780::Notifications, communicationAlarm,
      tmnCommunicationsAlarmInformationR1Package)
```

```
};  // interface TP_F

/**
```

6.6.20.1 CTPAbstract_F (Connection Termination Point Abstract)

```
*/

interface CTPAbstract_F: TP_F
{
        long channelNumberGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                        NOchannelNumberPackage);

};  // interface CTPAbstract_F

/**
```

6.6.20.1.1 CTPSink_F (Connection Termination Point Sink)

```
*/

interface CTPSink_F: CTPAbstract_F
{
        DownstreamConnectivityPointerType
                downstreamConnectivityPointerGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, stateChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, communicationsAlarm)

};  // interface CTPSink_F

/**
```

6.6.20.1.2 CTPSource_F (Connection Termination Point Source)

```
*/

interface CTPSource_F: CTPAbstract_F
{
        TPNameSeqType upstreamConnectivityPointerGet
                (in MONameType name)
                raises (itut_x780::ApplicationError);

        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)
```

```
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, stateChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, attributeValueChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, communicationsAlarm)

	}; // interface CTPSource_F



/**
```

6.6.20.1.2.1 CTPBid_F (Connection Termination Point Bidirectional)

```
*/


	interface CTPBid_F: CTPSink_F, CTPSource_F
	{
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectCreation)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, objectDeletion)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, stateChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, attributeValueChange)
		MANDATORY_NOTIFICATION(
			itut_x780::Notifications, communicationsAlarm)
	}; // interface CTPBid_F


/**
```

6.6.20.2 TTPAbstract_F (Trail Termination Point Abstract)

```
*/


	interface TTPAbstract_F: TP_F
	{
		ObjectClassSetType supportableClientListGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOsupportableClientListPackage);

		AdministrativeStateType administrativeStateGet
			(in MONameType name)
			raises (itut_x780::ApplicationError,
				NOadministrativeStatePackage);

		void administrativeStateSet
			(in MONameType name,
			in AdministrativeStateType adminstrativeState)
			raises (itut_x780::ApplicationError,
				NOadministrativeStatePackage);

	}; // interface TTPAbstract_F


/**
```

6.6.20.2.1 TTPSink_F (Trail Termination Point Sink)

```
*/


       interface TTPSink_F: TTPAbstract_F
       {
              TPNameSeqType upstreamConnectivityPointerGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError);

              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, stateChange)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, attributeValueChange)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, communicationsAlarm)

       }; // interface TTPSink_F


/**
```

6.6.20.2.2 TTPSource_F (Trail Termination Point Source)

```
*/


       interface TTPSource_F: TP_F
       {
              DownstreamConnectivityPointerType
                     downstreamConnectivityPointerGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError);

              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectCreation)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, objectDeletion)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, stateChange)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, attributeValueChange)
              MANDATORY_NOTIFICATION(
                     itut_x780::Notifications, communicationsAlarm)

       }; // interface TTPSource_F


/**
```

6.6.20.2.2.1 TTPBid_F (Trail Termination Point Bidirectional)

```
*/


       interface TTPBid_F: TTPSink_F, TTPSource_F
       {
              MANDATORY_NOTIFICATION(
```

```
                itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, stateChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, attributeValueChange)
        MANDATORY_NOTIFICATION(
                itut_x780::Notifications, communicationsAlarm)

    }; // interface TTPBid_F



/**
```

6.6.20.3 NetworkTP_F (Network Termination Point)

```
*/


    interface NetworkTP_F: TP_F
    {
        PointDirectionalityType pointDirectionalityGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);

        SignalIdType signalIdGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);

        ConfiguredConnectivityType configuredConnectivityGet
            (in MONameType name)
            raises (itut_x780::ApplicationError,
                NOconfiguredConnectivityPackage);

        PipeNameType connectivityPointerGet
            (in MONameType name)
            raises (itut_x780::ApplicationError,
                NOconnectivityPointerPackage);

        AdministrativeStateType administrativeStateGet
            (in MONameType name)
            raises (itut_x780::ApplicationError,
                NOadministrativeStatePackage);

        void administrativeStateSet
            (in MONameType name,
            in AdministrativeStateType administrativeState)
            raises (itut_x780::ApplicationError,
                NOadministrativeStatePackage);

        AvailabilityStatusSetType availabilityStatusGet
            (in MONameType name)
            raises (itut_x780::ApplicationError,
                NOavailabilityStatusPackage);

        Istring locationNameGet
            (in MONameType name)
            raises (itut_x780::ApplicationError,
                NOlocationNamePackage);

        void locationNameSet
            (in MONameType name,
            in Istring locationName)
            raises (itut_x780::ApplicationError,
```

```
                        NOlocationNamePackage);

                Istring userLabelGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                void userLabelSet
                        (in MONameType name,
                        in Istring userLabel)
                        raises (itut_x780::ApplicationError,
                                NOuserLabelPackage);

                MONameType neAssignmentPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOneAssignmentPointerPackage);

                SNCNameSetType sncPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOsncPointerPackage);

                NetworkTPNameType networkTPPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOnetworkTPPointerPackage);

                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectCreation)
                MANDATORY_NOTIFICATION(
                        itut_x780::Notifications, objectDeletion)

        }; // interface NetworkTP_F


/**
```

6.6.20.3.1 NetworkCTPAbstract_F (Network Connection Termination Point Abstract)

```
*/

        interface NetworkCTPAbstract_F: NetworkTP_F
        {
                long channelNumberGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOchannelNumberPackage);

                NetworkCTPAbstractNameType superPartitionPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOnetworkCTPPackage);

                NetworkCTPAbstractNameType subPartitionPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOnetworkCTPPackage);

                NetworkTTPAbstractNameSetType serverTTPPointerGet
                        (in MONameType name)
                        raises (itut_x780::ApplicationError,
                                NOserverTTPPointerPackage);

        }; // interface NetworkCTPAbstract_F
```

```
/**
```

6.6.20.3.1.1 NetworkCTPSink_F (Network Connection Termination Point Sink)

```
*/

       interface NetworkCTPSink_F: NetworkCTPAbstract_F
       {
       }; // interface NetworkCTPSink




/**
```

6.6.20.3.1.2 NetworkCTPSource_F (Network Connection Termination Point Source)

```
*/

       interface NetworkCTPSource_F: NetworkCTPAbstract_F
       {
       }; // interface NetworkCTPSource_F


/**
```

6.6.20.3.1.2.1 NetworkCTPBid_F (Network Connection Termination Point Bidirectional)

```
*/

       interface NetworkCTPBid_F: NetworkCTPSink_F, NetworkCTPSource_F
       {
       }; // interface NetworkCTPBid


/**
```

6.6.20.3.2 NetworkTTPAbstract_F (Network Trail Termination Point Abstract)

```
*/

       interface NetworkTTPAbstract_F: NetworkTP_F
       {
              ObjectClassSetType supportableClientListGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError,
                            NOsupportableClientListPackage);

              NetworkCTPAbstractNameSetType clientCTPListGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError,
                            NOclientCTPListPackage);

              AbstractLinkEndNameSetType clientLinkEndPointerListGet
                     (in MONameType name)
                     raises (itut_x780::ApplicationError,
```

```
                        NOclientLinkEndPointerPackage);

      };  // interface NetworkTTPAbstract_F


/**
```

6.6.20.3.2.1 NetworkTTPSink_F (Network Trail Termination Point Sink)

```
*/


      interface NetworkTTPSink_F: NetworkTTPAbstract_F
      {
      };  // interface NetworkTTPSink


/**
```

6.6.20.3.2.2 NetworkTTPSource_F (Network Trail Termination Point Source)

```
*/


      interface NetworkTTPSource_F: NetworkTTPAbstract_F
      {
      };  // interface NetworkTTPSource_F


/**
```

6.6.20.3.2.2.1 NetworkTTPBid_F (Network Trail Termination Point Bidirectional)

```
*/


      interface NetworkTTPBid_F: NetworkTTPSink_F, NetworkTTPSource_F
      {
      };  // interface NetworkTTPBid_F


/**
```

6.6.21 TPPool_F (Termination Point Pool)

```
*/


      interface TPPool_F: itut_x780::ManagedObject_F
      {
            TPNameSetType tpsInTPPoolGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            short totalTpCountGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            short connectedTpCountGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            short idleTpCountGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);
```

```
        }; // interface TPPool_F

/**
```

6.7 通知（Notifications）

本ドキュメントではモデル特有の通知は定義されない。
*/

/**

6.8 ネームバインディング（Name Binding）

*/

/**
以下のモジュールはネームバインディング情報を含む。
*/

```
       module NameBinding
       {
```

/**

6.8.1 AccessGroup

生成操作中に topologicalEndDirectionality 属性のセットに失敗またはアクセスグループオブジェクトの生成に失敗すると、生成操作は failureToSetDirectionality 値かまたは failureToCreateAccessGroup 値をそれぞれ持つ固有エラーを伴って失敗するであろう。

削除操作中に accessPointList が空でなければ、削除操作は、値 networkTTPsExisting を持つ固有エラーを伴って失敗するであろう。アクセスグループ管理オブジェクトが削除されないと、 削除操作は値 failureToRemoveAccessGroup を持つ固有エラーを伴って失敗するであろう。
*/

```
              module AccessGroup_LayerND
              {
                     const string  superiorClass =
                            "itut_m3120::LayerND";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::AccessGroup";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = TRUE;
                     const DeletePolicyType deletePolicy =
                            itut_x780::deleteOnlyIfNoContainedObjects;
                     const string  kind = "AccessGroup";
              }; // module AccessGroup_LayerND
```

/**

6.8.2 AlarmSeverityAssignmentProfile

*/

```
              module AlarmSeverityAssignmentProfile_ManagedElement
              {
                     const string  superiorClass =
                            "itut_m3120::ManagedElement";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::AlarmSeverityAssignmentProfile";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = TRUE;
                     const DeletePolicyType deletePolicy =
```

```
                    itut_x780::deleteOnlyIfNoContainedObjects;
              const string  kind = "AlarmSeverityAssignmentProfile";
       }; // module AlarmSeverityAssignmentProfile_ManagedElement


/**
```

6.8.3 ArcIntervalProfile

```
*/
       module ArcIntervalProfile_ManagedElement
       {
              const string  superiorClass =
                     "itut_m3120::ManagedElement";
              const boolean superiorSubclassesAllowed = TRUE;
              const string  subordinateClass =
                     "itut_m3120::ArcIntervalProfile";
              const boolean subordinateSubclassesAllowed = TRUE;
              const boolean managerCreatesAllowed = FALSE;
              const DeletePolicyType deletePolicy =
                     itut_x780::notDeletable;
              const string  kind = "ArcIntervalProfile";
       }; // module ArcIntervalProfile_ManagedElement


       module ArcIntervalProfile_ManagedElementComplex
       {
              const string  superiorClass =
                     "itut_m3120::ManagedElementComplex";
              const boolean superiorSubclassesAllowed = TRUE;
              const string  subordinateClass =
                     "itut_m3120::ArcIntervalProfile";
              const boolean subordinateSubclassesAllowed = TRUE;
              const boolean managerCreatesAllowed = FALSE;
              const DeletePolicyType deletePolicy =
                     itut_x780::notDeletable;
              const string  kind = "ArcIntervalProfile";
       }; // module ArcIntervalProfile_ManagedElementComplex


       module ArcIntervalProfile_Network
       {
              const string  superiorClass =
                     "itut_m3120::Network";
              const boolean superiorSubclassesAllowed = TRUE;
              const string  subordinateClass =
                     "itut_m3120::ArcIntervalProfile";
              const boolean subordinateSubclassesAllowed = TRUE;
              const boolean managerCreatesAllowed = FALSE;
              const DeletePolicyType deletePolicy =
                     itut_x780::notDeletable;
              const string  kind = "ArcIntervalProfile";
       }; // module ArcIntervalProfile_Network




/**
```

6.8.4 CircuitPack

以下のネームバインディングは、circuitPack が内蔵されるボード（例：低次終端）にスロットを与えるときにのみ使用される。内蔵するボードに circuitPack が挿入される時、挿入されたボードを表す circuitPack オブジェクトが自動生成される。

```
*/
       module CircuitPack_CircuitPack_AutoCreated
```

```
        {
                const string  superiorClass =
                        "itut_m3120::CircuitPack";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::CircuitPack";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteContainedObjects;
                const string  kind = "CircuitPack";
        }; // module CircuitPack_CircuitPack_AutoCreated


/**
以下のネームバインディングは他の equipmentHolder インスタンスに関連して circuitPack インスタンスを指定するために使用される。circuitPack オブジェクトの生成は上位オブジェクトによって代表される資源に物理サーキットパック挿入の結果である。

管理システムはサーキットパックの特定タイプを計画するために、explicitlyCreated ネームバインディングを使ってこのサーキットパックを削除し新しいそれを再生成し得る。

サーキットパックは含まれるオブジェクトが無いときにのみ、システム管理の結果として削除することができる。
*/
        module CircuitPack_EquipmentHolder_AutoCreated
        {
                const string  superiorClass =
                        "itut_m3120::EquipmentHolder";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::CircuitPack";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "CircuitPack";
        }; // module CircuitPack_EquipmentHolder_AutoCreated


/**
以下のネームバインディングは equipmentHolder インスタンスに関連して、circuitPack インスタンスを指定するために使用される。circuitPack オブジェクトの生成は上位オブジェクトによって代表される資源に物理サーキットパック挿入の結果である。

サーキットパックは含まれるオブジェクト込みで、システム管理の結果として削除することができる。
*/
        module CircuitPack_EquipmentHolder_AutoCreated_Delete
        {
                const string  superiorClass =
                        "itut_m3120::EquipmentHolder";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::CircuitPack";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteContainedObjects;
                const string  kind = "CircuitPack";
        }; // module CircuitPack_EquipmentHolder_AutoCreated_Delete


/**
以下のネームバインディングは他の equipmentHolder インスタンスに関連して、circuitPack インスタンスを指定するために使用される。circuitPack オブジェクトの生成はシステム管理プロトコルの結果である。もし circuitPackType が equipmentHolder によってサポートされている型と互換性がない場合、生成要求は結果的に処
```

理失敗エラーとなる。generalErrorParameter はエラーを報告するために使用され、circuitPackType 属性の値を与え得る。このパラメタは必要に応じて他の処理失敗に使用され得る。

サーキットパックは含まれるオブジェクトが無いときにのみ、システム管理の結果として削除することができる。
*/
```
module CircuitPack_EquipmentHolder_ExplicitylyCreated
{
	const string  superiorClass =
		"itut_m3120::EquipmentHolder";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CircuitPack";
	const boolean subordinateSubclassesAllowed = TRUE;
	const boolean managerCreatesAllowed = TRUE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CircuitPack";
}; // module CircuitPack_EquipmentHolder_ExplicitlyCreated


/**
```
以下のネームバインディングは他の equipmentHolder インスタンスに関連して、circuitPack インスタンスを指定するために使用される。circuitPack オブジェクトの生成はシステム管理の結果である。

サーキットパックは含まれるオブジェクト込みで、システム管理の結果として削除することができる。
*/
```
module CircuitPack_EquipmentHolder_ExplicitylyCreated_Delete
{
	const string  superiorClass =
		"itut_m3120::EquipmentHolder";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CircuitPack";
	const boolean subordinateSubclassesAllowed = TRUE;
	const boolean managerCreatesAllowed = TRUE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteContainedObjects;
	const string  kind = "CircuitPack";
}; // CircuitPack_EquipmentHolder_ExplicitlyCreated_Delete


/**
```

6.8.5 CTPSink

以下のネームバインディングは TTP が情報（トラヒック）を 双方向 CTP に送るという関係を表す。

自動インスタンス命名が使用されるとき、ネームバインディングの選択はローカルマターとされる。
*/
```
module CTPSink_TTPBid
{
	const string  superiorClass =
		"itut_m3120::TTPBid";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CTPSink";
	const boolean subordinateSubclassesAllowed = TRUE;
	const boolean managerCreatesAllowed = TRUE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CTPSink";
}; // module CTPSink_TTPBid


/**
```
以下のネームバインディングは TTP が情報（トラヒック）を シンク CTP に送るという関係を表す。

自動インスタンス命名が使用されるとき、ネームバインディングの選択はローカルマターとされる。
*/
```
module CTPSink_TTPSink
{
	const string  superiorClass =
		"itut_m3120::TTPSink";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CTPSink";
	const boolean subordinateSubclassesAllowed = FALSE;
	const boolean managerCreatesAllowed = TRUE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CTPSink";
}; // module CTPSink_TTPSink
```

```
/**
```

6.8.6 CTPSource

以下のネームバインディングは TTP が情報（トラヒック）を 双方向 CTP から受信するという関係を表す。

自動インスタンス命名が使用されるとき、ネームバインディングの選択はローカルマターとされる。
*/
```
module CTPSource_TTPBid
{
	const string  superiorClass =
		"itut_m3120::TTPBid";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CTPSource";
	const boolean subordinateSubclassesAllowed = TRUE;
	const boolean managerCreatesAllowed = TRUE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CTPSource";
}; // module CTPSource_TTPBid
```

```
/**
```
以下のネームバインディングは TTP が情報（トラヒック）を ソース CTP から受信するという関係を表す。

自動インスタンス命名が使用されるとき、ネームバインディングの選択はローカルマターとされる。
*/
```
module CTPSource_TTPSource
{
	const string  superiorClass =
		"itut_m3120::TTPSource";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CTPSource";
	const boolean subordinateSubclassesAllowed = FALSE;
	const boolean managerCreatesAllowed = TRUE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CTPSource";
}; // module CTPSource_TTPSource
```

```
/**
```

6.8.7 CrossConnection

クロスコネクションオブジェクトにおける fromTermination 属性値が空であってはならない。クロスコネクションインスタンスが削除されるとき、次のような属性が影響を受ける。端点オブジェクトまたは GTP オブジェクトにおいて、削除されたクロスコネクションインスタンスを指していた crossConnectionObjectPointer 属性は、端点接続に責任を負うファブリックを指すようにセットされなければならない。

適当な TP Pool オブジェクトにおいて、（適用可能ならば）カウンタは更新されなければならない。切断された端点において、connectivityPointer 属性は空にセットされなければならない。クロスコネクションオブジェクトインスタンスの削除はいかなる GTP の構成にも影響を与えない。
*/

```
module CrossConnection_Fabric
{
	const string  superiorClass =
		"itut_m3120::Fabric";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CrossConnection";
	const boolean subordinateSubclassesAllowed = TRUE;
	const boolean managerCreatesAllowed = FALSE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CrossConnection";
}; // module CrossConnection_Fabric
```

```
/**
```

クロスコネクションオブジェクトにおける fromTermination 属性値は空でなくてはならない。クロスコネクションインスタンスが削除されるとき、次のような属性が影響を受ける。端点オブジェクトまたは GTP オブジェクトにおいて、削除されたクロスコネクションインスタンスを指していた crossConnectionObjectPointer 属性は、端点接続に責任を負うファブリックを指すようにセットされなければならない。適当な TP Pool オブジェクトにおいて、（適用可能ならば）カウンタは更新されなければならない。切断された端点において、connectivityPointer 属性は空にセットされなければならない。

多重点クロスコネクションオブジェクトインスタンスに含まれるクロスコネクションの最終削除は、多重点クロスコネクションオブジェクトインスタンス の削除（及び適当なポインタの更新）にも影響を与える。クロスコネクションオブジェクトインスタンスの削除はいかなる GTP の構成にも影響を与えない。
*/

```
module CrossConnection_MPCrossConnection
{
	const string  superiorClass =
		"itut_m3120::MPCrossConnection";
	const boolean superiorSubclassesAllowed = TRUE;
	const string  subordinateClass =
		"itut_m3120::CrossConnection";
	const boolean subordinateSubclassesAllowed = TRUE;
	const boolean managerCreatesAllowed = FALSE;
	const DeletePolicyType deletePolicy =
		itut_x780::deleteOnlyIfNoContainedObjects;
	const string  kind = "CrossConnection";
}; // module CrossConnection_MPCrossConnection
```

```
/**
```

6.8.8 装置（Equipment）

自動インスタンス命名が使用されるとき、ネームバインディングの選択はローカルマターとされる。
*/

```
module Equipment_Equipment
{
	const string  superiorClass =
```

```
                        "itut_m3120::Equipment";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::Equipment";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "Equipment";
        }; // module Equipment_Equipment


/**
自動インスタンス命名が使用されるとき、ネームバインディングの選択はローカルマターとされる。
*/
        module Equipment_ManagedElement
        {
                const string  superiorClass =
                        "itut_m3120::ManagedElement";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::Equipment";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "Equipment";
        }; // module Equipment_ManagedElement




/**
```

6.8.9 EquipmentHolder

```
以下のネームバインディングは他の equipmentHolder インスタンスに関連して equipmentHolder インスタンスを
指定するのに使用される。
*/
        module EquipmentHolder_EquipmentHolder
        {
                const string  superiorClass =
                        "itut_m3120::EquipmentHolder";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::EquipmentHolder";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "EquipmentHolder";
        }; // module EquipmentHolder_EquipmentHolder

        module EquipmentHolder_ManagedElement
        {
                const string  superiorClass =
                        "itut_m3120::ManagedElement";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::EquipmentHolder";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "EquipmentHolder";
        }; // module EquipmentHolder_ManagedElement
/**
```

6.8.10 ExternalPoint

```
*/
            module ExternalPoint_Equipment
            {
                  const string  superiorClass =
                        "itut_m3120::Equipment";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::ExternalPoint";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "ExternalPoint";
            }; // module ExternalPoint_Equipment


            module ExternalPoint_ManagedElement
            {
                  const string  superiorClass =
                        "itut_m3120::ManagedElement";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::ExternalPoint";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "ExternalPoint";
            }; // module ExternalPoint_ManagedElement


            module ExternalPoint_ManagedElementComplex
            {
                  const string  superiorClass =
                        "itut_m3120::ManagedElementComplex";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::ExternalPoint";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780:: deleteOnlyIfNoContainedObjects;
                  const string  kind = "ExternalPoint";
            }; // module ExternalPoint_ManagedElementComplex


/**
```

6.8.11 ファブリック（Fabric）

```
*/
            module Fabric_ManagedElement
            {
                  const string  superiorClass =
                        "itut_m3120::ManagedElement";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::Fabric";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "Fabric";
            }; // module Fabric_ManagedElement
```

```
/**
```

6.8.12 GTP

```
*/
            module GTP_Fabric
            {
                  const string  superiorClass =
                         "itut_m3120::Fabric";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                         "itut_m3120::GTP";
                  const boolean subordinateSubclassesAllowed = FALSE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                         itut_x780:: notDeletable;
                  const string  kind = "GTP";
            }; // module GTP_Fabric


/**
```

6.8.13 LayerND

```
*/
            module LayerND_Network
            {
                  const string  superiorClass =
                         "itut_m3120::Network";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                         "itut_m3120::LayerND";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                         itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "LayerND";
            }; // module LayerND_Network


/**
```

6.8.14 LinkConnection

```
*/
            module LinkConnection_LayerND
            {
                  const string  superiorClass =
                         "itut_m3120::LayerND";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                         "itut_m3120::LinkConnection";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                         itut_x780:: notDeletable;
                  const string  kind = "LinkConnection";
            }; // module LinkConnection_LayerND


            module LinkConnection_TopLink
            {
                  const string  superiorClass =
```

```
                        "itut_m3120::TopLink";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::LinkConnection";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                        itut_x780:: notDeletable;
                const string  kind = "LinkConnection";
        }; // module LinkConnection_TopLink



/**
```

6.8.15 LogicalLink

```
logicalLink 管理オブジェクトは establishLink または establishLinkAndLinkEnds 動作によって生成される。

logicalLink 管理オブジェクトは removeLink または removeLinkAndLinkEnd によって削除される。
*/
        module LogicalLink_LayerND
        {
                const string  superiorClass =
                        "itut_m3120::LayerND";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::LogicalLink";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                        itut_x780::notDeletable;
                const string  kind = "LogicalLink";
        }; // module LogicalLink_LayerND


/**
```

6.8.16 LogicalLinkEnd

```
*/
        module LogicalLinkEnd_LayerND
        {
                const string  superiorClass =
                        "itut_m3120::LayerND";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::LogicalLinkEnd";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                        itut_x780:: notDeletable;
                const string  kind = "LogicalLinkEnd";
        }; // module LogicalLinkEnd_LayerND


        module LogicalLinkEnd_Subnetwork
        {
                const string  superiorClass =
                        "itut_m3120::Subnetwork";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::LogicalLinkEnd";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
```

```
		const DeletePolicyType deletePolicy =
			itut_x780:: notDeletable;
		const string kind = "LogicalLinkEnd";
	}; // module LogicalLinkEnd_LayerND


/**
```

6.8.17 ManagedElement

```
管理エレメントオブジェクトはシステム管理プロトコルによっては生成または削除されない。
オブジェクトは管理エレメントを初期化したときに生成される。
*/
	module ManagedElement_ManagedElementComplex
	{
		const string superiorClass =
			"itut_m3120::ManagedElementComplex";
		const boolean superiorSubclassesAllowed = TRUE;
		const string subordinateClass =
			"itut_m3120::ManagedElement";
		const boolean subordinateSubclassesAllowed = TRUE;
		const boolean managerCreatesAllowed = FALSE;
		const DeletePolicyType deletePolicy =
			itut_x780::notDeletable;
		const string kind = "ManagedElement";
	}; // module ManagedElement_ManagedElementComplex


/**
以下のネームバインディングは managedElementComplex オブジェクトに関連する managedElement オブジェクト
インスタンスの指定に使用される。managedElement オブジェクトは管理プロトコルによって明示的に生成される。
managedElement の生成は、managedElement に含まれるある特定オブジェクトの自動生成を始動し、
managedElement の削除は managedElement に含まれる全てのオブジェクトの自動削除を始動し得る。
*/
	module ManagedElement_ManagedElementComplex_ExplicitlyCreated
	{
		const string superiorClass =
			"itut_m3120::ManagedElementComplex";
		const boolean superiorSubclassesAllowed = TRUE;
		const string subordinateClass =
			"itut_m3120::ManagedElement";
		const boolean subordinateSubclassesAllowed = TRUE;
		const boolean managerCreatesAllowed = TRUE;
		const DeletePolicyType deletePolicy =
			itut_x780::deleteContainedObjects;
		const string kind = "ManagedElement";
	}; // ManagedElement_ManagedElementComplex_ExplicitlyCreated


/**
管理エレメントオブジェクトはシステム管理プロトコルによっては生成または削除されない。オブジェクトは管理エレメ
ントを初期化したときに生成される。
*/
	module ManagedElement_Network
	{
		const string superiorClass =
			"itut_m3120::Network";
		const boolean superiorSubclassesAllowed = FALSE;
		const string subordinateClass =
			"itut_m3120::ManagedElement";
		const boolean subordinateSubclassesAllowed = TRUE;
		const boolean managerCreatesAllowed = FALSE;
		const DeletePolicyType deletePolicy =
			itut_x780::notDeletable;
		const string kind = "ManagedElement";
	}; // module ManagedElement_Network
```

```
/**
以下のネームバインディングは外部命名オブジェクトへの managedElement オブジェクトの指定に使用される。
managedElement オブジェクトはシステム管理プロトコルによっては生成または削除されない。管理エレメントオブジ
ェクトはネットワークエレメントの初期化時に生成される。
*/
              module ManagedElement_Organization
              {
                     const string  superiorClass =
                            "";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::ManagedElement";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = FALSE;
                     const DeletePolicyType deletePolicy =
                            itut_x780::notDeletable;
                     const string  kind = "ManagedElement";
              }; // module ManagedElement_Organization


/**
```

6.8.18 ManagedElementComplex

```
以下のネームバインディングはネットワークオブジェクトへ managedElementComplex オブジェクトを指定するため
に使用する。managedElementComplex オブジェクトはシステム管理プロトコルによって生成または削除されない。
*/
              module ManagedElementComplex_Network
              {
                     const string  superiorClass =
                            "itut_m3120::Network";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::ManagedElementComplex";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = FALSE;
                     const DeletePolicyType deletePolicy =
                            itut_x780::notDeletable;
                     const string  kind = "ManagedElementComplex";
              }; // module ManagedElementComplex_Network


/**
以下のネームバインディングは managedElementComplex オブジェクトを外部命名オブジェクトに指定するために使
用される。managedElementComplex オブジェクトはシステム管理プロトコルによっては生成または削除されない。
*/
              module ManagedElementComplex_Organization
              {
                     const string  superiorClass =
                            "";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::ManagedElementComplex";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = FALSE;
                     const DeletePolicyType deletePolicy =
                            itut_x780::notDeletable;
                     const string  kind = "ManagedElementComplex";
              }; // module ManagedElementComplex_Organization


/**
```

6.8.19 MPCrossConnection

```
*/
            module MPCrossConnection_Fabric
            {
                  const string  superiorClass =
                         "itut_m3120::Fabric";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                         "itut_m3120::MPCrossConnection";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                         itut_x780:: notDeletable;
                  const string  kind = "MPCrossConnection";
            }; // module MPCrossConnection_Fabric


/**
```

6.8.20 ネットワーク（Network）

ネットワークオブジェクトはシステム管理プロトコルによって生成または削除されない。オブジェクトはネットワークを初期化したときに生成される。

```
*/
            module Network_Network
            {
                  const string  superiorClass =
                         "itut_m3120::Network";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                         "itut_m3120::Network";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                         itut_x780::notDeletable;
                  const string  kind = "Network";
            }; // module Network_Network


/**
```

以下のネームバインディングはネットワークオブジェクトを外部命名オブジェクトへ指定するために使用される。ネットワークオブジェクトはシステム管理プロトコルによって生成または削除されない。オブジェクトはネットワークを初期化したときに生成される。

```
*/
            module Network_Organization
            {
                  const string  superiorClass =
                         "";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                         "itut_m3120::Network";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                         itut_x780::notDeletable;
                  const string  kind = "Network";
            }; // module Network_Organization



/**
```

6.8.21 NetworkCTPSink

```
*/
            module NetworkCTPSink_LayerND
            {
                  const string  superiorClass =
                        "itut_m3120::LayerND";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::NetworkCTPSink";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                        itut_x780:: notDeletable;
                  const string  kind = "NetworkCTPSink";
            }; // module NetworkCTPSink_LayerND


/**
下位の管理オブジェクトは、サブネットワークの構成に従って上位管理オブジェクトのインスタンス化、またはサブネットワークに追加資源（計画中の資源を含む）の追加または取り外しの際に自動的にインスタンス化され、削除される。
*/
            module NetworkCTPSink_Subnetwork
            {
                  const string  superiorClass =
                        "itut_m3120::Subnetwork";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::NetworkCTPSink";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::notDeletable;
                  const string  kind = "NetworkCTPSink";
            }; // module NetworkCTPSink_Subnetwork


/**
```

6.8.22 NetworkCTPSource

```
*/
            module NetworkCTPSource_LayerND
            {
                  const string  superiorClass =
                        "itut_m3120::LayerND";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::NetworkCTPSource";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = FALSE;
                  const DeletePolicyType deletePolicy =
                        itut_x780:: notDeletable;
                  const string  kind = "NetworkCTPSource";
            }; // module NetworkCTPSource_LayerND


/**
下位の管理オブジェクトは、サブネットワークの構成に従って上位管理オブジェクトのインスタンス化、またはサブネットワークに追加資源（計画中の資源を含む）の追加または取り外しの際に自動的にインスタンス化され削除される。
*/
            module NetworkCTPSource_Subnetwork
            {
                  const string  superiorClass =
                        "itut_m3120::Subnetwork";
                  const boolean superiorSubclassesAllowed = TRUE;
```

```
                const string  subordinateClass =
                       "itut_m3120::NetworkCTPSource";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = FALSE;
                const DeletePolicyType deletePolicy =
                       itut_x780::notDeletable;
                const string  kind = "NetworkCTPSource";
        }; // module NetworkCTPSource_Subnetwork



/**
```

6.8.23 NetworkTTPSink

削除操作時、networkTTP がトレールを終端しているならば、削除操作は networkTTPTerminatesTrail 値をもった特定のエラーを伴って失敗する。

削除操作時、networkTTP がサブネットワークまたはアクセスグループに関連付けられているならば、削除操作は networkTTPAssociatedWithSubnetwork 値または networkTTPAssociatedWithAccessGroup 値をそれぞれ持ったエラーを伴って失敗する。

```
*/
        module NetworkTTPSink_LayerND
        {
                const string  superiorClass =
                       "itut_m3120::LayerND";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                       "itut_m3120::NetworkTTPSink";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                       itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "NetworkTTPSink";
        }; // module NetworkTTPSink_LayerND


/**
```

削除操作時、networkTTP がトレールを終端しているならば、削除操作は networkTTPTerminatesTrail 値をもった特定のエラーを伴って失敗する。

```
*/
        module NetworkTTPSink_Subnetwork
        {
                const string  superiorClass =
                       "itut_m3120::Subnetwork";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                       "itut_m3120::NetworkTTPSink";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                       itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "NetworkTTPSink";
        }; // module NetworkTTPSink_Subnetwork



/**
```

6.8.24 NetworkTTPSource

削除操作時、networkTTP がトレールを終端しているならば、削除操作は networkTTPTerminatesTrail 値を持った特定のエラーを伴って失敗する。

削除操作時、networkTTP がサブネットワークまたはアクセスグループに関連付けられているならば、削除操作は networkTTPAssociatedWithSubnetwork 値または networkTTPAssociatedWithAccessGroup 値をそれぞれ持ったエラーを伴って失敗する。
*/

```
            module NetworkTTPSource_LayerND
            {
                  const string  superiorClass =
                        "itut_m3120::LayerND";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::NetworkTTPSource";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "NetworkTTPSource";
            }; // module NetworkTTPSource_LayerND


/**
削除操作時、networkTTP がトレールを終端しているならば、削除操作は networkTTPTerminatesTrail 値をもった特定のエラーを伴って失敗する。
*/
            module NetworkTTPSource_Subnetwork
            {
                  const string  superiorClass =
                        "itut_m3120::Subnetwork";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::NetworkTTPSource";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "NetworkTTPSource";
            }; // module NetworkTTPSource_Subnetwork




/**
```

6.8.25 ソフトウェア（Software）

自動インスタンス命名が用いられるときは、ネームバインディングの選択はローカルマターとされる。
*/

```
            module Software_Equipment
            {
                  const string  superiorClass =
                        "itut_m3120::Equipment";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::Software";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "Software";
            }; // module Software_Equipment


/**
自動インスタンス命名が用いられるときは、ネームバインディングの選択はローカルマターとされる。
*/
            module Software_ManagedElement
```

```
        {
                const string  superiorClass =
                        "itut_m3120::ManagedElement";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::Software";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "Software";
        }; // module Software_ManagedElement


/**
自動インスタンス命名が用いられるときは、ネームバインディングの選択はローカルマターとされる。
*/
        module Software_Software
        {
                const string  superiorClass =
                        "itut_m3120::Software";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::Software";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "Software";
        }; // module Software_Software




/**
```

6.8.26 サブネットワーク（Subnetwork）

networkTTP 管理オブジェクトインスタンスが生成されるまたはサブネットワークに関連付けられる必要があるような生成操作時、networkTTP 管理オブジェクトインスタンスの生成または関連付けに失敗した場合、それぞれ failureToCreateNetworkTTP 値または failureToAssociateNetworkTTP 値を伴って特定のエラーが返され生成操作は失敗する。

削除操作時、サブネットワークが使用中である（サブネットワーク接続が存在する）ことが判明するか、他の資源に結び付けられているならば、それぞれ subnetworkInUse 値または boundSubnetwork 値を伴った特定のエラーが返却され、生成（TTC 注：削除の誤り）操作は失敗する。

```
*/
        module Subnetwork_LayerND
        {
                const string  superiorClass =
                        "itut_m3120::LayerND";
                const boolean superiorSubclassesAllowed = TRUE;
                const string  subordinateClass =
                        "itut_m3120::Subnetwork";
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                const string  kind = "Subnetwork";
        }; // module Subnetwork_LayerND




/**
```

6.8.27 SNC

```
*/
              module SNC_Subnetwork
              {
                     const string  superiorClass =
                            "itut_m3120::Subnetwork";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::SNC";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = FALSE;
                     const DeletePolicyType deletePolicy =
                            itut_x780:: notDeletable;
                     const string  kind = "SNC";
              }; // module SNC_Subnetwork



/**
```

6.8.28 TopLink

topologicalLink 管理オブジェクトは、リンクをサポートするサーバネットワークレイヤ領域においてトレールが生成されるときに自動生成されるか、establishTopologicalLink 動作または establishTopologicalLinkAndLinkEnds 動作によって生成されるかのいずれかである。

topologicalLink 管理オブジェクトは、removeTopologicalLink 動作または removeTopologicalLinkAndLinkEnds 動作、または topologicalLink 管理オブジェクトが以前に自動生成されていた場合はトレールの削除のいずれかによって削除される。

```
*/
              module TopLink_LayerND
              {
                     const string  superiorClass =
                            "itut_m3120::LayerND";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::TopLink";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = FALSE;
                     const DeletePolicyType deletePolicy =
                            itut_x780::notDeletable;
                     const string  kind = "TopLink";
              }; // module TopLink_LayerND



/**
```

6.8.29 TopLinkEnd

```
*/
              module TopLinkEnd_LayerND
              {
                     const string  superiorClass =
                            "itut_m3120::LayerND";
                     const boolean superiorSubclassesAllowed = TRUE;
                     const string  subordinateClass =
                            "itut_m3120::TopLinkEnd";
                     const boolean subordinateSubclassesAllowed = TRUE;
                     const boolean managerCreatesAllowed = FALSE;
                     const DeletePolicyType deletePolicy =
                            itut_x780:: notDeletable;
```

```
		const string  kind = "TopLinkEnd";
	}; // module TopLinkEnd_LayerND


	module TopLinkEnd_Subnetwork
	{
		const string  superiorClass =
			"itut_m3120::Subnetwork";
		const boolean superiorSubclassesAllowed = TRUE;
		const string  subordinateClass =
			"itut_m3120::TopLinkEnd";
		const boolean subordinateSubclassesAllowed = TRUE;
		const boolean managerCreatesAllowed = FALSE;
		const DeletePolicyType deletePolicy =
			itut_x780:: notDeletable;
		const string  kind = "TopLinkEnd";
	}; // module TopLinkEnd_Subnetwork



/**
```

6.8.30 TPPool

```
*/
	module TPPool_Fabric
	{
		const string  superiorClass =
			"itut_m3120::Fabric";
		const boolean superiorSubclassesAllowed = TRUE;
		const string  subordinateClass =
			"itut_m3120::TPPool";
		const boolean subordinateSubclassesAllowed = FALSE;
		const boolean managerCreatesAllowed = FALSE;
		const DeletePolicyType deletePolicy =
			itut_x780:: notDeletable;
		const string  kind = "TPPool";
	}; // module TPPool_Fabric



/**
```

6.8.31 Trail

```
*/
	module Trail_LayerND
	{
		const string  superiorClass =
			"itut_m3120::LayerND";
		const boolean superiorSubclassesAllowed = FALSE;
		const string  subordinateClass =
			"itut_m3120::Trail";
		const boolean subordinateSubclassesAllowed = FALSE;
		const boolean managerCreatesAllowed = FALSE;
		const DeletePolicyType deletePolicy =
			itut_x780:: notDeletable;
		const string  kind = "Trail";
	}; // module Trail_LayerND



/**
```

6.8.32 Trail95

```
*/
            module Trail95_Network
            {
                  const string  superiorClass =
                        "itut_m3120::Network";
                  const boolean superiorSubclassesAllowed = FALSE;
                  const string  subordinateClass =
                        "itut_m3120::Trail95";
                  const boolean subordinateSubclassesAllowed = FALSE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "Trail95";
            }; // module Trail95_Network



/**
```

6.8.33 TTPSource

```
自動インスタンス命名が用いられるときは、ネームバインディングの選択はローカルマターとされる。
*/
            module TTPSource_ManagedElement
            {
                  const string  superiorClass =
                        "itut_m3120::ManagedElement";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::TTPSource";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "TTPSource";
            }; // module TTPSource_ManagedElement



/**
```

6.8.34 TTPSink

```
自動インスタンス命名が用いられるときは、ネームバインディングの選択はローカルマターとされる。
*/
            module TTPSink_ManagedElement
            {
                  const string  superiorClass =
                        "itut_m3120::ManagedElement";
                  const boolean superiorSubclassesAllowed = TRUE;
                  const string  subordinateClass =
                        "itut_m3120::TTPSink";
                  const boolean subordinateSubclassesAllowed = TRUE;
                  const boolean managerCreatesAllowed = TRUE;
                  const DeletePolicyType deletePolicy =
                        itut_x780::deleteOnlyIfNoContainedObjects;
                  const string  kind = "TTPSink";
            }; // module TTPSink_ManagedElement


      }; // module NameBinding
```

```
}; // module itut_m3120


#endif // _itut_m3120_idl_

/**
```

7. 情報モデル IDL: 定数値(Information Model IDL :Constants)

```
#ifndef _itut_m3120const_idl_
#define _itut_m3120const_idl_
#pragma prefix "itu.int"
/**
本 IDL コードは 6 節の IDL コードを含むファイルと同一のディレクトリに置かれている"itut_m3120const.idl"
という名称のファイルに格納される様、意図されている。
/**
本モジュールの itut_m3120 は、M.3100 と G.855.1 にて定義されるオブジェクトを元にした IDL 定義を含んでいる。 本
ファイルの IDL 定義は単に、本モジュール内で使われる定数値である。オブジェクトインタフェースは別個のファイルに
含まれるが、同じ名前の"itut_m3120" モジュールに含まれる。
*/
*/
module itut_m3120
{
       const string moduleName = "itut_m3120";

/**
```

7.1 AdditionalInformationConst

```
本定数値はサービスに影響があったかどうかを示す ManagementExtensionType の any フィールドの論理値を伴う。。
*/
              const short alarmEffectOnService = 1;
/**
本定数は疑わしいオブジェクトを識別する ManagementExtensionType の “ any ”
フィールドの MOsetType 値を伴う。
*/
              const short suspectObjectList = 2;

/**
本定数値はユーザラベルを表すために ManagementExtensionType の “ any ”
フィールドの文字列値を伴う。
*/
              const short userLabel = 3;

       }; // module AdditionalInformationConst



/**
```

7.2 CharacteristicInfoConst

```
本モジュールは CharacteristicInfo  UID 向けに定義される定数値を含む。本値は M.3100 から流用された。
 */
       module CharacteristicInfoConst
       {
              const string moduleName =
                     "itut_m3120::CharacteristicInfoConst";
/**
stmLevel attribute = 1 である opticalSPITTP*  オブジェクトインスタンス
*/
              const short opticalSTM1SPICI = 1;
/**
stmLevel attribute = 4 である opticalSPITTP*オブジェクトインスタンス */
              const short opticalSTM4SPICI = 2;

/**
opticalSPITTP*オブジェクトインスタンス object instances with stmLevel attribute = 16
*/
              const short opticalSTM16SPICI = 3;

/**
```

```
stmLevel attribute = 1 である electricalSPITTP* オブジェクトインスタンス*/
              const short electricalSTM1SPICI = 4;

/**
stmLevel attribute = 1 である rsCTP* オブジェクトインスタンス
*/
              const short rsSTM1SPICI = 5;

/**
stmLevel attribute = 4 である rsCTP*オブジェクトインスタンス
*/
              const short rsSTM4SPICI = 6;

/**
stmLevel attribute = 16 である rsCTP*オブジェクトインスタンス
*/
              const short rsSTM16SPICI = 7;
/**
stmLevel attribute = 1 である msCTP* オブジェクトインスタンス
*/
              const short msSTM1SPICI = 8;

/**
stmLevel attribute = 4 である msCTP* オブジェクトインスタンス
*/
              const short msSTM4SPICI = 9;

/**
stmLevel attribute = 16 である msCTP* オブジェクトインスタンス
*/
              const short msSTM16SPICI = 10;

              const short au3TU3VC3CI = 11;
              const short au4VC4CI = 12;
              const short tu11VC11CI = 13;
              const short tu12VC12CI = 14;
              const short tu2VC2CI = 15;
              const short tu12VC11CI = 16;
              const short vpCI = 17;
              const short vcCI = 18;
              const short e0CI = 19;
              const short e1CI = 20;
              const short e2CI = 21;
              const short e3CI = 22;
              const short e4CI = 23;

       }; // module CharacteristicInfoConst


/**
```

7.3CreateErrorConst

本モジュールは、生成障害要因 UID（Create Error Cause UID）向けに定義される定数値を含む。本値は、M.3100 corridendum 一般障害要因（General Error Cause）型定義から流用された。

```
*/
       module CreateErrorConst
       {
              const string moduleName =
                     "itut_m3120::CreateErrorConst";

/**
ObjectInIncompatibleState は、オブジェクトが提供された状態にあることを規定するために使われる。
*/
              const short objectInIncompatibleState = 1;

/**
NoValidRelatedObject は MIB 内に存在しない関連オブジェクトを規定するために使われる。
*/
```

```
                const short noValidRelatedObject = 2;

/**
InvolvedInOffering は競合するサービス提供に、既に関わっているオブジェクトを識別するために使われる。
*/
                const short involvedInOffering = 3;

/**

ServiceNotSupported は、装置によりサポートされていないサービスを操作が起動しようと試みている事を示すために使われる。
.*/

const short serviceNotSupported = 4;

/**
ProvisioningOrderConflict はサービスが装置によりサポートされていない順に提供されている事を識別するために使われる。
*/
                const short provisioningOrderConflict = 5;

/**
EquipmentFailure は、装置障害が操作中に発生した事を示すために使われる。
*/
                const short equipmentFailure = 6;

/**
MaxNumberExceeded は要求された生成操作がインスタンスの最大値に達して完了できなかった事を示すために使われる。
*/
                const short maxNumberExceeded = 7;

/**
ContainedObjects は要求された削除操作が包含されるインスタンスが存在する事により完了できなかった事を示すために使われる。
*/
                const short containedObjects = 8;

        }; // module CreateErrorConst


/**
```

7.4 DeleteErrorConst

```
ManagementExtensionType の"any"フィールド は値なしである。
*/
                const short boundSubnetwork = 1;
                const short failureToRemoveAccessGroup = 2;
                const short failureToRemoveNetworkCTPs = 3;
                const short failureToRemoveNetworkTTP = 4;
                const short failureToRemoveSubnetwork = 5;
/**
ManagementExtensionType の"any"フィールドは関連するアクセスグループの MO である。
*/
                const short networkTTPAssociatedWithAccessGroup = 6;
/**
ManagementExtensionType の"any"フィールドは関連するサブネットワークの MO である。

*/
                const short networkTTPAssociatedWithSubnetwork = 7;
/**
ManagementExtensionType の"any"フィールドは値なしである。
*/
                const short networkTTPsExisting = 8;
/**
```

```
ManagementExtensionType の”any”フィールドは終端点トレールの MO である。
*/
                const short networkTTPTerminatesTrail = 9;
/**
ManagementExtensionType の”any”フィールドは値なしである。

*/
                const short SubnetworkInUse = 10;

        }; // module DeleteErrorConst


/**
```

7.5 Probable CauseCons

本モジュールは推定原因 ID（Probable Cause UID）として定義された定数値を含む。これらの値は M.3100 から流用された。

```
*/
        module ProbableCauseConst
        {
                const string moduleName =
                        "itut_m3120::ProbableCauseConst";

                const short indeterminate = 0;

/**
以下の通信警報に使われる。
*/
                const short aIS = 1 ;
                const short callSetUpFailure = 2;
                const short degradedSignal = 3;
                const short farEndReceiverFailure = 4;
                const short framingError = 5;
                const short lossOfFrame = 6;
                const short lossOfPointer = 7;
                const short lossOfSignal = 8;
                const short payloadTypeMismatch = 9;
                const short transmissionError = 10;
                const short remoteAlarmInterface = 11;
                const short excessiveBER = 12;
                const short pathTraceMismatch = 13;
                const short unavailable = 14;
                const short signalLabelMismatch = 15;
                const short lossOfMultiFrame = 16;
                const short receiveFailure = 17;
                const short transmitFailure = 18;
                const short modulationFailure = 19;
                const short demodulationFailure = 20;
                const short broadcastChannelFailure = 21;
                const short connectionEstablishmentError = 22;
                const short invalidMessageReceived = 23;
                const short localNodeTransmissionError = 24;
                const short remoteNodeTransmissionError = 25;
                const short routingFailure = 26;

/**
27-50 の値は通信警報に関連する想定原因用に確保されている。
*/

/**
以下は装置警報に使われる。
*/
                const short backplaneFailure = 51;
                const short dataSetProblem = 52;
                const short equipmentIdentifierDuplication = 53;
                const short externalIFDeviceProblem = 54;
                const short lineCardProblem = 55;
                const short multiplexerProblem = 56;
                const short nEIdentifierDuplication = 57;
                const short powerProblem = 58;
                const short processorProblem = 59;
                const short protectionPathFailure = 60;
                const short receiverFailure = 61;
```

```
            const short replaceableUnitMissing = 62;
            const short replaceableUnitTypeMismatch = 63;
            const short synchronizationSourceMismatch = 64;
            const short terminalProblem = 65;
            const short timingProblem = 66;
            const short transmitterFailure = 67;
            const short trunkCardProblem = 68;
            const short replaceableUnitProblem = 69;

/**
実時間クロックが故障した事をシステムが検知した場合に発行される装置警報である。

*/
            const short realTimeClockFailure = 70;
            const short antennaFailure = 71;
            const short batteryChargingFailure = 72;
            const short diskFailure = 73;
            const short frequencyHoppingFailure = 74;
            const short iODeviceError = 75;
            const short lossOfSynchronisation = 76;
            const short lossOfRedundancy = 77;
            const short powerSupplyFailure = 78;
            const short signalQualityEvaluationFailure = 79;
            const short tranceiverFailure = 80;

/**
81-100 の値は装置警報想定原因用に確保されている。
*/
/**
以下は環境警報として使用される。
*/
            const short airCompressorFailure = 101;
            const short airConditioningFailure = 102;
            const short airDryerFailure = 103;
            const short batteryDischarging = 104;
            const short batteryFailure = 105;
            const short commercialPowerFailure = 106;
            const short coolingFanFailure = 107;
            const short engineFailure = 108;
            const short fireDetectorFailure = 109;
            const short fuseFailure = 110;
            const short generatorFailure = 111;
            const short lowBatteryThreshold = 112;
            const short pumpFailure = 113;
            const short rectifierFailure = 114;
            const short rectifierHighVoltage = 115;
            const short rectifierLowFVoltage = 116;
            const short ventilationsSystemFailure = 117;
            const short enclosureDoorOpen = 118;
            const short explosiveGas = 119;
            const short fire = 120;
            const short flood = 121;
            const short highHumidity = 122;
            const short highTemperature = 123;
            const short highWind = 124;
            const short iceBuildUp = 125;
            const short intrusionDetection = 126;
            const short lowFuel = 127;
            const short lowHumidity = 128;
            const short lowCablePressure = 129;
            const short lowTemperature = 130;
            const short lowWater = 131;
            const short smoke = 132;
            const short toxicGas = 133;
            const short coolingSystemFailure = 134;
            const short externalEquipmentFailure = 135;
            const short externalPointFailure = 136;

/**
137-150 の値は環境警報関連想定原因用に確保されている。
*/

/**
以下は処理障害警報として使用される。
*/
            const short storageCapacityProblem = 151;
```

```
            const short memoryMismatch = 152;
            const short corruptData = 153;
            const short outOfCPUCycles = 154;
            const short sfwrEnvironmentProblem = 155;
            const short sfwrDownloadFailure = 156;

/**
実時間クロック内の時刻が失われたが、クロック自身は動作している事をシステムが検出した場合に発行される処理障害警報である。これは例えば、電源バックアップを実時間クロック用に持たない小さい NE に電源断が発生した場合等に起こり得る。
*/
            const short lossOfRealTime = 157;

/**
システムが再初期化された場合に発行される処理障害警報である。これは管理システムの持つ管理対象システムのビューがもはや有効でない事を管理システムに示す。利用例として、管理されるシステムが管理システムに再初期化事象を通知するために重要度警告（Warning）である本警報を再初期化の後に発行する場合がある。
クリア通知は発信されないであろう。
*/
            const short reinitialized = 158;
            const short applicationSubsystemFailure = 159;
            const short configurationOrCustomisationError = 160;
            const short databaseInconsistency = 161;
            const short fileError = 162;
            const short outOfMemory = 163;
            const short softwareError = 164;
            const short timeoutExpired = 165;
            const short underlayingResourceUnavailable = 166;
            const short versionMismatch = 167;

/**
168-200 の値は処理障害警報関連の推定原因用に確保されている。
*/

            const short bandwidthReduced = 201;
            const short congestion = 202;
            const short excessiveErrorRate = 203;
            const short excessiveResponseTime = 204;
            const short excessiveRetransmissionRate = 205;
            const short reducedLoggingCapability = 206;
            const short systemResourcesOverload = 207;

      }; // module ProbableCauseConst

/**
```

7.6 北米固有情報（North America Specific Characteritic Information）

```
本モジュールは北米固有の CharacteristicInfo UID を定義した定数値を含んでいる。
*/
      module NorthAmericaCharacteristicInfoConst
      {
            const string moduleName =
                  "itut_m3120:: NorthAmericaCharacteristicInfoConst";

            const short ds0CI = 1;           // 64 KB/s
            const short ds1CI = 2;           // 1.544 MB/s
            const short ds1cCI = 3;          // 3.152 MB/s
            const short ds2CI = 4;           // 6.312 MB/s
            const short ds3CI = 5;           // 44.736 MB/s
            const short qds0CI = 6;    // 16 KB/s
            const short sr24CI = 7;    // 2.4 KB/s
            const short sr48CI = 8;    // 4.8 KB/s
            const short sr96CI = 9;    // 9.6 KB/s
            const short sr192CI = 10;  // 19.2 KB/s
            const short sr384CI = 11;  // 38.4 KB/s
            const short sr560CI = 12;  // 56 KB/s
            const short tu13VC13CI = 13;     // vt3

      }; // module NorthAmericaCharacteristicInfoConst
```

```
}; // module itut_m3120

#endif // _itut_m3120const_idl_
```

8 情報モデル IDL：警報報告制御（Information Model IDL: Alarm Reporting Control）

```
/**
```

8.1 The arcPackage IDL

本節は、arcPackage GDMO パッケージに対する IDL を定義する。本 IDL コードは、ARC 機能をサポートする全ての IDL インタフェースに含まれる事になっている。

```
*/

/**
本パッケージは警報報告制御（ARC）をサポートするオブジェクトに要求される特長を定義する。
ARC のフィーチャは警報報告をサポートする全てのオブジェクトによりサポートされる。
arcProbableCauseList への変化は即時に有効となる。 各想定原因の変化の振舞いは ALM 状態から／への遷移の際に現れる振舞いと同様である。想定原因報告はその想定原因が追加されると停止して、リストから削除された後に開始する。
ArcProbableCauseList が変更された場合、変化はベストエフォートで行われる。サポートされていない想定原因のみ、エラーを返し、リストに追加されない。有効な想定原因は受け入れられてリストに追加される。デフォルトの時間間隔（ARC プロファイルに２つ有り）は、上書き値が arcControl アクションにて指定されない限り、何か他の状態から QI 状態又は TI 状態に入った場合にのみ有効である。 ArcManagementRequestedInterval の変化は、（noAdjustment に設定されたのと反対に）これに時間値が設定された場合にのみ許可される。これは即時に有効となる。
属性値変化通知は arcState、arcProbableCauseList、 arcIntervalProfilePointer、arcManagementRequestedInterval の変化に対して送信される。
*/


/**
以下は ARC 機能をサポートするインタフェースの追加値の型である。
*/
              public ArcStateType        arcState;
                     // conditional
                     // arcPackage
                     // GET
              public ArcQIStatusType            arcQIStatus;
                     // conditional
                     // arcPackage
                     // GET
              public ArcProbableCauseSetType    arcProbableCauseList;
                     // conditional
                     // arcPackage
                     // GET-REPLACE, ADD-REMOVE
              public ArcIntervalProfileNameType arcIntervalProfilePointer;
                     // conditional
                     // arcPackage
                     // GET-REPLACE
              public ArcTimeType         arcManagementRequestedInterval;
                     // conditional
                     // arcPackage
                     // GET-REPLACE
              public ArcTimeType         arcTimeRemaining;
                     // conditional
                     // arcPackage
                     // GET
              public CurrentProblemSetType      currentProblemList;
                     // include only if managed object not have this
                     // conditional
                     // arcPackage
                     // GET


/**
以下は ARC 機能をサポートするインタフェースの追加操作である。
```

```
*/

/**
状態変化通知は本属性の値変化を示すために使われる。
*/
              ArcStateType arcStateGet ()
                     raises (ApplicationError,
                            NOarcPackage);

/**
状態変化通知も、属性値変化通知も本属性の値の変化を示すためには使われない。
*/
              ArcQIStatusType     arcQIStatusGet ()
                     raises (ApplicationError,
                            NOarcPackage);

              ArcProbableCauseSetType    arcProbableCauseListGet ()
                     raises (ApplicationError,
                            NOarcPackage);

              void arcProbableCauseListSet
                     (in ArcProbableCauseSetType arcProbableCauseList)
                     raises (ApplicationError,
                            NotSupportedProbableCause,
                            NOarcPackage);

              void arcProbableCauseListAdd
                     (in ArcProbableCauseSetType arcProbableCauseList)
                     raises (ApplicationError,
                            NotSupportedProbableCause,
                            NOarcPackage);

              void arcProbableCauseListRemove
                     (in ArcProbableCauseSetType arcProbableCauseList)
                     raises (ApplicationError,
                            NOarcPackage);

/**
以下のメソッドは、ARC 時間間隔プロファイルポインタの値を取得するために使われる。本ポインタは関連する ARC 時
間間隔プロファイルオブジェクトを示す。本属性値は、設定可能な持続間隔と時間間隔が使われなかった場合、空である。
（即ち、nalm-qi と nalm-ti 状態に使われない場合である。）
*/
              ArcIntervalProfileNameType arcIntervalProfilePointerGet ()
                     raises (ApplicationError,
                            NOarcPackage);

/**
以下のメソッドは、ARC 時間間隔プロファイルポインタの値を変更するために使われる。本ポインタは関連する ARC 時
間間隔プロファイルオブジェクトを示す。本属性値は、設定可能な持続間隔と時間間隔が使われなかった場合、空である。
（即ち、nalm-qi と nalm-ti 状態に使われない場合である。）
*/
              void arcIntervalProfilePointerSet
                     (in ArcIntervalProfileNameType profile)
                     raises (ApplicationError,
                            NOarcPackage);

/**
以下のメソッドは、ARC 管理要求時間間隔の値を取得するために使われる。本時間間隔は ARC 時間間隔として管理要求
された時刻を示す。本属性は、管理要求の際にのみ、又は資源を自動的に alm 状態に遷移させる際にのみ値を変化させ
る。
本属性値を変更させる管理要求は、それを行う事が無効の場合には否定される。例えば、管理資源が alm、又は nalm 状
態にある場合である。本属性値は、ARC 時間間隔が、特定の時点で管理要求経由で調整可能かどうかの結果を反映して
いる。
*/
              ArcTimeType arcManagementRequestedIntervalGet ()
                     raises (ApplicationError,
                            NOarcPackage);

/**
以下のメソッドは ARC 管理要求時間間隔の値の変更に使われる。管理要求時間間隔は、ARC 時間間隔の管理要求時間を
示している。
*/
```

```
                void arcManagementRequestedIntervalSet
                        (in ArcTimeType time)
                        raises (ApplicationError,
                                NOarcPackage);

/**
以下のメソッドは、ARC 時間間隔（即ち、nalm-qi 状態用持続時間間隔と nalm-ti 状態の時間間隔）
として残っている時間を取得するために使われる。
本取得値は必ずしも本状態の残り時間を示していない事は注意が必要である。
例えば、arcTimeRemaining は nalm-qi 状態で 30 分とできるが、もし制限された問題が、ARC 時間間隔がタイムア
ウトする前に管理資源に対して発生した場合、タイマを終了して再び制限された問題がなくなる迄無限に待ちタイマを再
開して残り時間を再び減算し始める。もし資源が nalm-ti、nalm、nalm-qi へ遷移した場合、本属性値は管理要求時
間間隔に初期化される。タイマが作動していない場合、本値はタイマが作動していない事を示す。（つまり時間内で調整
が行われていない事を示す。）本属性値の変化については属性値変化通知が送られない事は注意が必要である。
 */
                ArcTimeType arcTimeRemainingGet ()
                        raises (ApplicationError,
                                NOarcPackage);

/**
以下のメソッドは管理オブジェクトに関連して現在存在する問題を重要度とともに取得するために使用される。
管理オブジェクトクラスが本メソッドをまだ保持していない場合にのみ含まれる。
*/
                CurrentProblemSetType currentProblemListGet ()
                        raises (ApplicationError,
                                NOarcPackage);/**
以下のメソッドは警告報告を開始又は停止する場合に使われる。これは要求される ARC 状態を示す事により実現される。
場合によっては、状態が与えられた資源タイプによりサポートされなかったためにアクションが拒否される事も想定され
る。状態を特定する事に加えて、マネージャは１回だけの利用のためにデフォルト以外の arcInterval を要求する事も
あり得る。そのようなオーバライドは、nalm-qi と  naml-ti 状態への遷移のみに適用される。
*/
                boolean arcControl
                        (in ArcControlRequest request)
                        raises (ApplicationError,
                                NOarcPackage);
/**
以下は ARC 機能をサポートするインタフェースに対する追加通知である。
*/
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, attributeValueChange,
                        arcPackage)
                CONDITIONAL_NOTIFICATION(
                        itut_x780::Notifications, stateChange,
                        arcPackage)

/**

ArcPackage の終了
*/

/**
```

8.2 The arcRetrieveAlarmDetailPackage IDL

```
本節は、arcRetrieveAlarmDetail パッケージ  GDMO パッケージに対する IDL を定義する。本 IDL コードは警報詳
細取得機能をサポートする全ての IDL インタフェースに含まれる事になっている。本機能サポートは、8.1 節にて規定
された ARC 機能のサポートが前提である。
*/

/**
ARC 警報詳細取得をサポートする全てのインタフェースは、IDL インタフェース定義において以下の操作を含む。本機
能をサポートするために、インタフェースは 8.1 節に定義された ARC  IDL も含む。
本操作の返り値は警報状態の詳細警報リストである。各警報状態につき、詳細警報情報は想定原因、関知重要度、事象発
生時間（選択可能）、警報状態を含む。宣言された警報状態の想定原因が ARC 制御下にある場合、（即ち、資源オブジ
ェクトの  ArcProbableCauseSetType に含まれる場合）、本警報状態の状態値は保留中となる。従って、
ArcAlarmDetailSetType 中の想定原因は、警報状態の値が関知重要度と同じでない場合、本想定原因は ARC 制御下
に入る。
*/
```

```
            ArcAlarmDetailSetType arcRetrieveAlarmDetail ()
                  raises (ApplicationError,
                        NOarcRetrieveAlarmDetailPackage);
/**
```

ArcRetreiveAlarmDetailPackage