INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

M.3120

Amendment 2
(03/2003)

SERIES M: TMN AND NETWORK MAINTENANCE: INTERNATIONAL TRANSMISSION SYSTEMS, TELEPHONE CIRCUITS, TELEGRAPHY, FACSIMILE AND LEASED CIRCUITS

Telecommunications management network

CORBA generic network and network element level information model

Amendment 2

ITU-T Recommendation M.3120 (2001) – Amendment 2

ITU-T M-SERIES RECOMMENDATIONS

TMN AND NETWORK MAINTENANCE: INTERNATIONAL TRANSMISSION SYSTEMS, TELEPHONE CIRCUITS, TELEGRAPHY, FACSIMILE AND LEASED CIRCUITS

Introduction and general principles of maintenance and maintenance organization	M.10–M.299
International transmission systems	M.300–M.559
International telephone circuits	M.560–M.759
Common channel signalling systems	M.760–M.799
International telegraph systems and phototelegraph transmission	M.800–M.899
International leased group and supergroup links	M.900–M.999
International leased circuits	M.1000–M.1099
Mobile telecommunication systems and services	M.1100–M.1199
International public telephone network	M.1200–M.1299
International data transmission systems	M.1300–M.1399
Designations and information exchange	M.1400–M.1999
International transport network	M.2000–M.2999
Telecommunications management network	**M.3000–M.3599**
Integrated services digital networks	M.3600–M.3999
Common channel signalling systems	M.4000–M.4999

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

ITU-T Recommendation M.3120

CORBA generic network and network element level information model

Amendment 2

Summary

This amendment provides several enhancements to the CORBA generic network and network element level information model. First, it details a mechanism that supports reporting attribute value ranges across the CORBA interface. Second, it defines a new Generic Transport TTP object class which is intended to represent a physical port or endpoints of transport connections. Third, it defines a new object class, ManagedElementR2, a subclass of ManagedElement with three additional attributes added. These attributes include one to hold the "model code" of a piece of equipment. Another new attribute is used to represent network element aliases, or names used by the EMS to refer to Network Elements. Also defined is an attribute to hold the generic "type" of a network element.

Another enhancement included in this amendment relates to expanding the CharacteristicInfo constants module so that it can adequately represent as much of the currently available signal rates as possible.

Source

Amendment 2 to ITU-T Recommendation M.3120 (2001) was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 29 March 2003.

FOREWORD

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

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

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

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

NOTE

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

INTELLECTUAL PROPERTY RIGHTS

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

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

CONTENTS

ITU-T Recommendation M.3120

CORBA generic network and network element level information model

Amendment 2

1 Scope

This amendment provides several enhancements to the CORBA generic network and network element level information model. First, it details a mechanism that supports reporting attribute value ranges across the CORBA interface. Second, it defines a new Generic Transport TTP object class which is intended to represent a physical port or endpoints of transport connections. Third, it defines a new object class, ManagedElementR2, a subclass of ManagedElement with three additional attributes added. These attributes include one to hold the "model code" of a piece of equipment. Another new attribute is used to represent network element aliases, or names used by the EMS to refer to Network Elements. Also defined is an attribute to hold the generic "type" of a network element.

Another enhancement included in this amendment relates to expanding the CharacteristicInfo constants module so that it can adequately represent as much of the currently available signal rates as possible.

2 References

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

[1] ITU-T Recommendation Q.816 (2001), *CORBA-Based TMN Services*.

[2] ITU-T Recommendation Q.816.1 (2001), *CORBA-Based TMN Services Extensions to Support Coarse-Grained Interfaces*.

[3] ITU-T Recommendation X.780 (2001), *TMN Guidelines for Defining CORBA Managed Objects*.

[4] ITU-T Recommendation X.780.1 (2001), *TMN Guidelines for Defining Coarse-Grained CORBA Managed Objects*.

[5] ITU-T Recommendation M.3120 (2001), *CORBA Generic Network and NE Level Information Model*.

[6] ITU-T Recommendation M.3100 (1995), *Generic Network Information Model* plus Amendment 1 (1999).

[7] ITU-T Recommendation Q.822.1 (2001), *Coarse-Grained CORBA Generic Network and NE Level Information Model*.

[8] ANSI Standard T1.231.1997 (1997), *Digital Hierarchy – Layer 1 In-Service Digital Transmission Performance Monitoring*.

3 Definitions

This amendment has no new definitions in addition to those found in the main Recommendation.

4 Abbreviations

This amendment has no new abbreviations in addition to those found in the main Recommendation.

5 Conventions

This amendment has no new conventions in addition to those found in the main Recommendation.

6 Overview of attribute value ranges

This clause provides a mechanism that allows managed systems using the M.3120 paradigm, to automatically report acceptable value ranges for attributes associated with a network element in the model. Such mechanism would be a valuable asset for equipment discovery and configuration, since a managing system would automatically be aware of the acceptable value ranges for each configurable parameter in the network before attempting to set these values.

For this mechanism to be implemented, we define a new AttributeRanges object class. The AttributeRanges class allows the managed system to report the minimum and maximum values a certain attribute accepts, as well as the granularity, or step increments, of the range. Each AttributeRanges instance contains ranges for attributes belonging to one object class. The "*kind*" attribute in AttributeRanges denotes the object class for which ranges are being defined. "*attributeName*" specifies the name of the attribute for which a range is being defined. The range is then defined using the "*minimum*", "*maximum*", and "*granularity*" attributes. "*granularity*" is not needed for attributes containing floating numbers.

For each ManagedElement instance representing a network element, one or more AttributeRanges instances may be created. AttributeRanges instances are bound to the ManagedElement instance via a containment relationship.

Ranges are defined per ManagedElement instance. This allows for an attribute to have different ranges when it belongs to different network elements. In other words, the scope of each AttributeRanges instance is the relevant objects associated with the ManagedElement which contains the AttributeRanges instance.

Figure 1 illustrates the scoping concept more clearly. In the figure, we see two different instances of ManagedElement (A and B). Contained under ManagedElement A are two AttributeRanges instances named A and B. Similarly, contained under ManagedElement B are two other AttributeRanges instances named C and D. AttributeRanges A defines ranges for all AalProfileTypeOne instances associated with ManagedElementA, while AttributeRanges C defines ranges for AalProfileTypeOne instances associated with ManagedElement B. Similarly, AttributeRanges B defines ranges for all CesServiceProfile instances associated with ManagedElementA, and AttributeRanges D defines ranges for CesServiceProfile instances associated with ManagedElement B. In other words, the managed system instantiates one AttributeRanges instance (portrayed as a table in the figure) per class per Managed Element instance.

Hence, if the managing system needs to modify the parameters of an AalProfileTypeOne instance associated with ManagedElement A (such as the instance *ProfileA* in the figure), it can query AttributeRanges A before modifying the values.

In order to set ranges for attributes defined inside data structures, the dot notation is used. For instance, consider the following data structure:

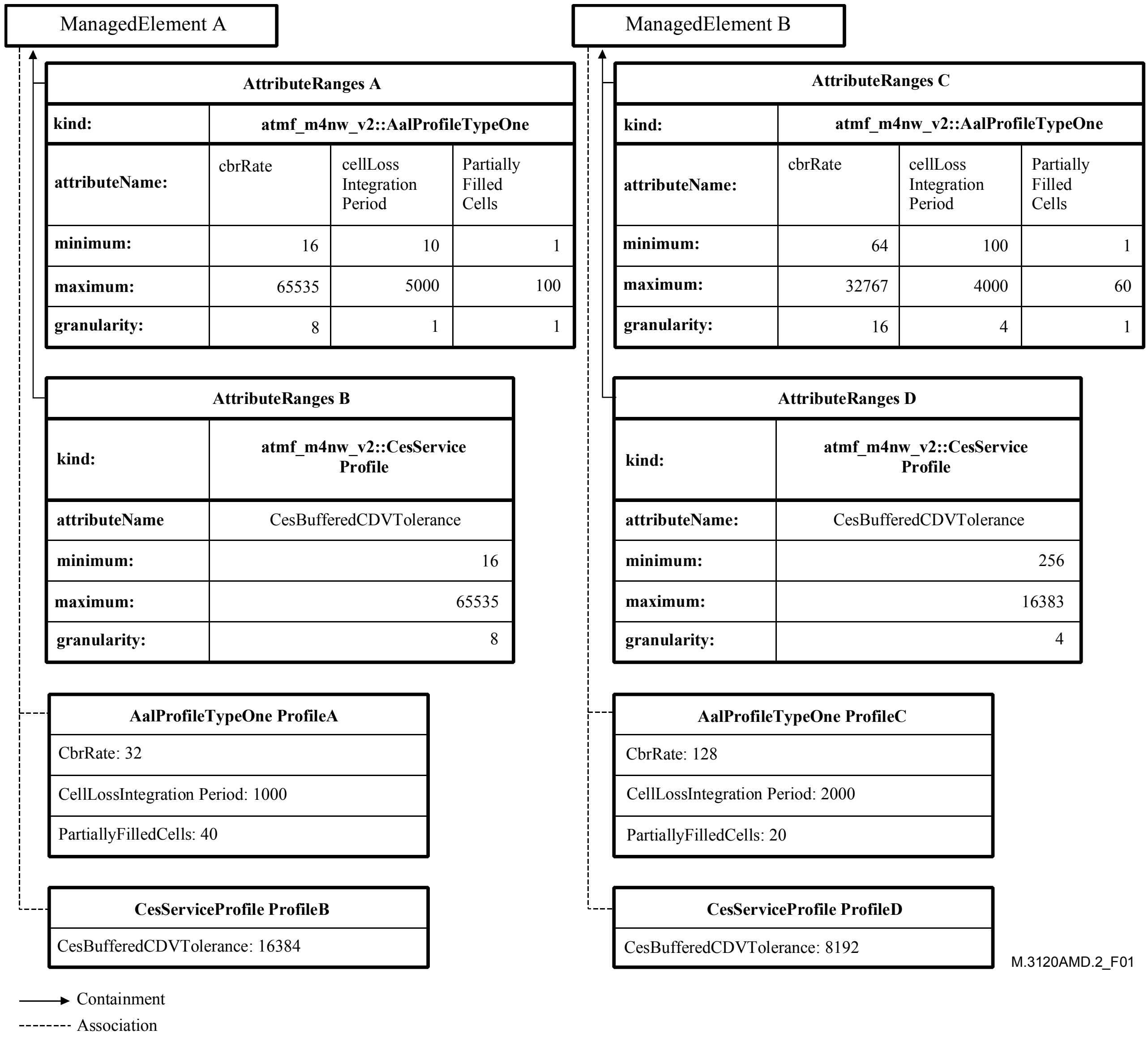
```
struct SampleStructureType {
long xyx,
long abc,
float def };
```

In order to set an attribute range on attribute xyz, we may refer to attribute *xyz* by setting the *attributeName* attribute in *AttributeRangeType* to "*SampleStructureType.xyz*".

Clause 10.2.1 defines a set of CORBA IDL interfaces for the attribute value ranges information model. These interfaces are translated manually from a set of Amendment 7/M.3100 GDMO managed object classes following the TMN CORBA framework and guidelines given in ITU-T Recs Q.816 and X.780 for fine-grained CORBA interface.

In addition to the fine-grained interface in clause 10.2.1, a companion Facade interface is defined in clause 10.3.1. This facade interface is defined according to the coarse-grained framework and guidelines given in Q.816.1 and X.780.1 for supporting coarse-grained CORBA interfaces. The name of this facade interface is the name of the corresponding fine-grained interface appended with "_F" (an underscore followed by a capital "F").

Figure 2 and Figure 3 show the inheritance and containment relationship of the CORBA interfaces defined in this clause. Note that facade interfaces follow the same inheritance hierarchy relationship as the corresponding fine-grained interfaces.



AttributeRanges A			
kind:	atmf_m4nw_v2::AalProfileTypeOne		
attributeName:	cbrRate	cellLoss Integration Period	Partially Filled Cells
minimum:	16	10	1
maximum:	65535	5000	100
granularity:	8	1	1

AttributeRanges B	
kind:	atmf_m4nw_v2::CesService Profile
attributeName	CesBufferedCDVTolerance
minimum:	16
maximum:	65535
granularity:	8

AalProfileTypeOne ProfileA
CbrRate: 32
CellLossIntegration Period: 1000
PartiallyFilledCells: 40

CesServiceProfile ProfileB
CesBufferedCDVTolerance: 16384

AttributeRanges C			
kind:	atmf_m4nw_v2::AalProfileTypeOne		
attributeName:	cbrRate	cellLoss Integration Period	Partially Filled Cells
minimum:	64	100	1
maximum:	32767	4000	60
granularity:	16	4	1

AttributeRanges D	
kind:	atmf_m4nw_v2::CesService Profile
attributeName:	CesBufferedCDVTolerance
minimum:	256
maximum:	16383
granularity:	4

AalProfileTypeOne ProfileC
CbrRate: 128
CellLossIntegration Period: 2000
PartiallyFilledCells: 20

CesServiceProfile ProfileD
CesBufferedCDVTolerance: 8192

Figure 1/M.3120/Amd.2 – Instance diagram portraying the use of AttributeRanges

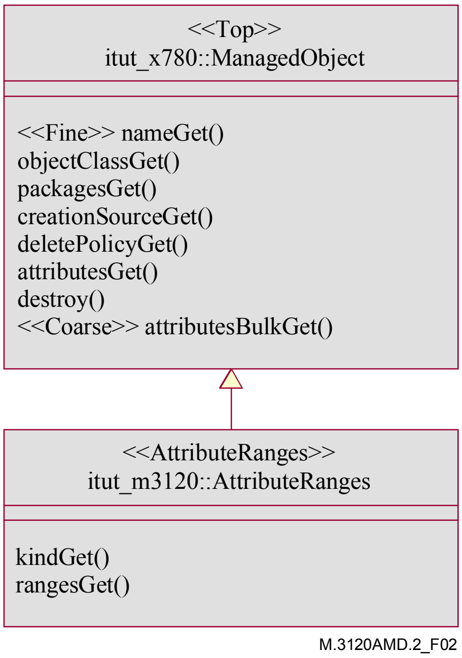


Figure 2/M.3120/Amd.2 – Attribute value ranges inheritance relationship

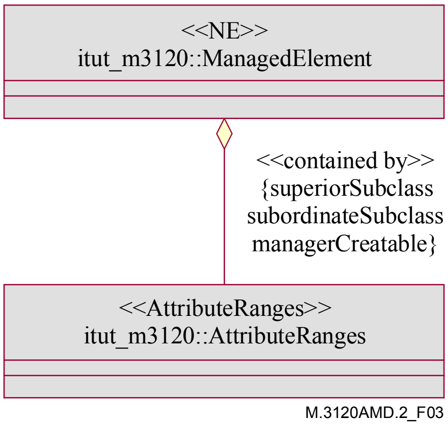


Figure 3/M.3120/Amd.2 – Attribute value ranges containment relationship

7 Overview of the generic transport TTP

This clause defines a new Generic Transport TTP object class. This new object is used to represent a physical port or endpoints of transport connections. It may be used by technology-specific models as an abstraction of an underlying transport layer.

A new GenericTransportTTP interface is defined. This object is a subclass of NetworkTP. It is related to ManagedElement using a containment relationship. It is associated with CircuitPack using the PortAssociationList attribute, and with LinkEnd using the ClientLinkEndPointerList attribute.

Clause 10.2.2 defines a set of CORBA IDL interfaces for the GenericTransportTTP and GenericTransportPmCD object classes. These interfaces are translated manually from a set of Amendment 8/M.3100 GDMO managed object classes following the TMN CORBA framework and guidelines given in ITU-T Recs Q.816 and X.780 for fine-grained CORBA interface.

In addition to the fine-grained interfaces in 10.2.2, a companion set of Facade interfaces are defined in 10.3.2. These facade interfaces are defined according to the coarse-grained framework and guidelines given in Q.816.1 and X.780.1 for supporting coarse-grained CORBA interface. The name of these facade interfaces are the name of the corresponding fine-grained interface appended with "_F" (an underscore followed by a capital "F").

Figures 4 and 5 show the inheritance, containment, and association relationships of the CORBA interfaces defined in this Recommendation. Note that facade interfaces follow the same inheritance hierarchy relationship as the corresponding fine-grained interfaces.

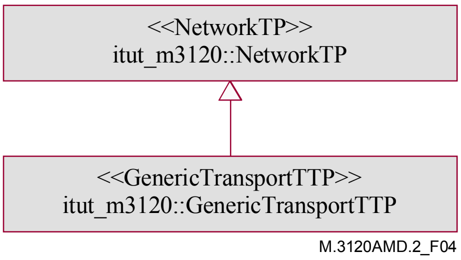


Figure 4/M.3120/Amd.2 – Generic transport TTP inheritance relationship

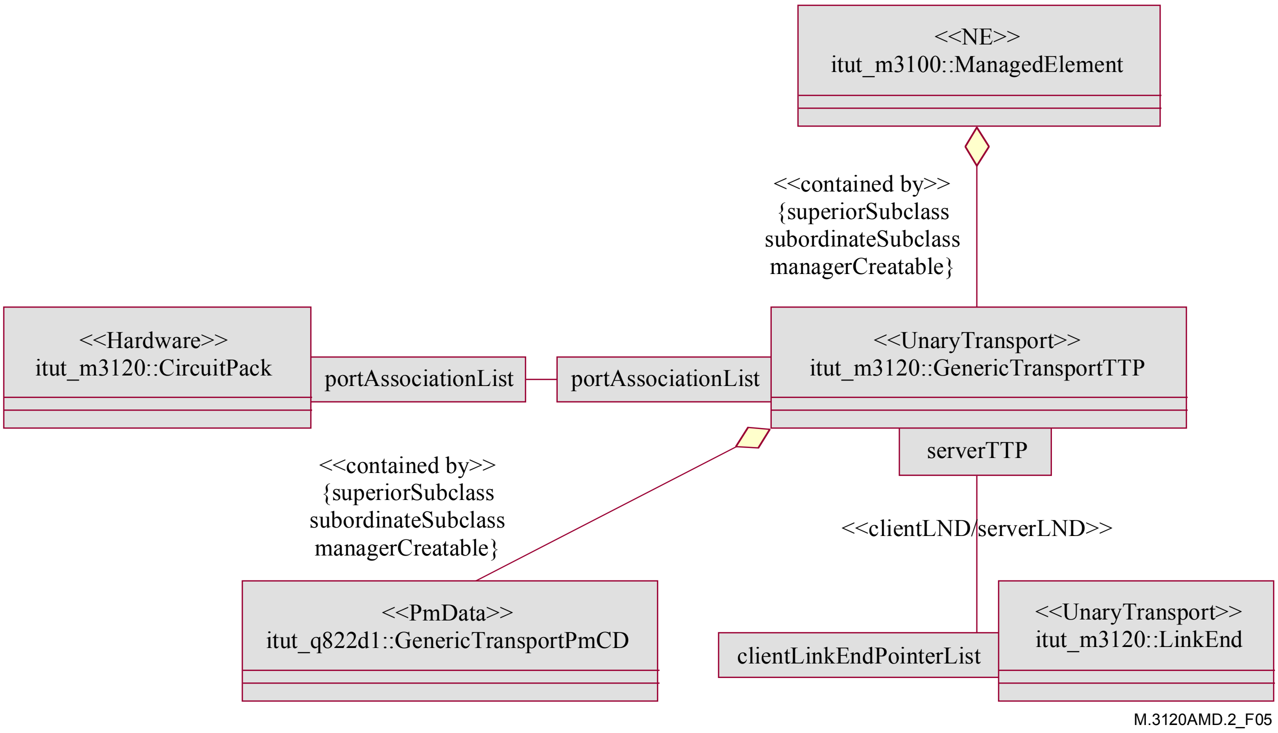


Figure 5/M.3120/Amd.2 – Generic transport TTP containment and association relationships

8 Enhancements to ManagedElement object class

This clause describes new attributes to be added to the ManagedElement class. In order to preserve backward compatibility, these new attributes are placed in a subclass of ManagedElement, named ManagedElementR2. The original ManagedElement in ITU-T Rec. M.3120 is based on the M.3100 ManagedElementR1 object. The ManagedElementR2 object defined here is based on the M.3100 ManagedElementR2 object, which is defined in Amendment 6/M.3100. The interface name ManagedElementR1 is skipped in this amendment to keep the names of the M.3100 and M.3120 objects aligned.

ManagedElementR2 inherits all the attributes of ManagedElement and defines the following extra three:

8.1 Model code

This attribute stores the product model code of the Network Element. The product model code is the manufacturer's model identification information. It is vendor-provided information that the vendor uses to distinguish the network element among a family of products. This attribute is useful for OSSs performing equipment discovery and inventory processes.

The model code is a read-only attribute.

8.2 Network element aliases

This attribute is used to hold aliases given by the EMS to a certain Managed Element instance. Having such aliases available via the EMS/NMS interface is useful for relating Network Element names entered at the EMS, via the Graphical User Interface or otherwise, to those found on the NMS user interface. More importantly, these aliases may appear in alarms sent by certain EMS software outside the interface. Thus, it would be crucial for the NMS to recognize such aliases in order to perform alarm correlation or other fault and performance functions.

8.3 Network element type

Currently, the Managed Element class does not contain an attribute to specify the type of the network element it represents. This attribute holds a set of either textual strings or values from a predefined set (UIDs), that describes the generic type of the Network Element modelled by the ManagedElementR2 instance. Multiple managed element type values may be used to describe hybrid equipment. The network element type is a read-only attribute.

9 Expansion of characteristic information

In its current form, the Characteristic Information constants module leaves out a large number of widely used signal rates. The following is an expansion to the signal rates list so it can adequately describe as many signal rates and port types as possible.

The following additions to ITU-T Rec. M.3120 are necessary to expand the list of CharacteristicInfo type:

Clause 7.2

Inside the CharacteristicInfoConst module, add the following lines:

```
const short E5_565M                            = 24;
const short STS3c_and_VC4_1c                   = 25;
const short STS12c_and_VC4_4c                  = 26;
const short STS48c_and_VC4_16c                 = 27;
const short STS192c_and_VC4_64c                = 28;
const short Section_OC1_STS1_and_RS_STM0       = 29;
const short Section_OC192_STS192_and_RS_STM64  = 30;
const short Line_OC1_STS1_and_MS_STM0          = 31;
const short Line_OC192_STS192_and_MS_STM64     = 32;
const short FC_12_133M                         = 33;
   // Fiber Channel protocol,
const short FC_25_266M                         = 34;
   // Fiber Channel protocol,
const short FC_50_531M                         = 35;
   // Fiber Channel protocol,
const short FC_100_1063M                       = 36;
   // Fiber Channel protocol,
const short FDDI                               = 37;
const short Fast_Ethernet                      = 38;
const short Gigabit_Ethernet                   = 39;
const short ISDN_BRI                           = 40;
   // ISDN Basic Rate Interface PTP layer rate
```

```
       const short DSR_OC192_and_STM64                = 41;
       const short DSR_OC768_and_STM256               = 42;
       const short Section_OC24_STS24_and_RS_STM8     = 43;
       const short Line_OC24_STS24_and_MS_STM8        = 44;
       const short Section_OC768_STS768_and_RS_STM256 = 45;
       const short Line_OC768_STS768_and_MS_STM256    = 46;
       const short 10Gigabit_Ethernet                 = 47;
```

10 Information model

This amendment IDL is an integral part of ITU-T Rec. M.3120. This implies that all definitions (object classes, type, structure, etc.) defined in ITU-T Rec. M.3120 are in the same IDL module and can be referenced without the module identifier.

The IDL in this amendment has been compiled successfully without syntax error. The compiler used claims CORBA 2.3 compliance, which includes value type and M4 macro capabilities.

```
#ifndef _itut_m3120_amd2_idl_
#define _itut_m3120_amd2_idl_


#include <itut_m3120.idl>


#pragma prefix "itu.int"

/**
This IDL code (beginning with the line "#ifndef … " through the end of this
clause) is intended to be stored in a file named "itut_m3120_amd2.idl" located
in the search path used by the IDL compiler on your system. A compiler
supporting the CORBA version specified in ITU-T Rec. Q.816 must be used. The
M.3120 main module (defined in ITU-T Rec. M.3120) is contained in a separate
file named "itut_m3120.idl"
*/




/**
This fragment is added to the module, itut_m3120, which contains IDL definition
based on objects defined in ITU-T Rec. M.3100
*/

module itut_m3120
{

/**
```

10.1 Structures and TypeDefs

```
*/
     enum AttributeChoiceType
     {
       attributeChoiceLong,
       attributeChoiceLongLong,
       attributeChoiceUnsignedLong,
       attributeChoiceShort,
       attributeChoiceFloat
     };
```

```
union AttributeRangeType switch (AttributeChoiceType)
{
  case attributeChoiceLong:
     Istring  attributeName;
     long     minimumValue;
     long     maximumValue;
     long     granularity;
         // 0 indicates this attribute is not being used

  case attributeChoiceLongLong:
     Istring  attributeName;
     long long          minimumValue;
     long long          maximumValue;
     long long          granularity;
         // 0 indicates this attribute is not being used

  case attributeChoiceUnsignedLong:
     Istring  attributeName;
     unsigned long           minimumValue;
     unsigned long           maximumValue;
     unsigned long           granularity;
         // 0 indicates this attribute is not being used


  case attributeChoiceShort:
     Istring  attributeName;
     short         minimumValue;
     short         maximumValue;
     short         granularity;
         // 0 indicates this attribute is not being used

  case attributeChoiceFloat:
     Istring       attributeName;
     float         minimumValue;
     float         maximumValue;
};

typedef sequence<AttributeRangeType> AttributeRangeSetType;

enum MeTypeChoiceType
{
  MeTypeChoiceIstring,
  MeTypeChoiceUID
};

union MeType switch (MeTypeChoiceType)
{
  case MeTypeChoiceIstring:
     Istring  MeTypeString;

         // 0 indicates this attribute is not being used
     DefaultLongTypeOpt defaultValue;

  case MeTypeChoiceUID:
     UIDType  METypeUID;

};

typedef sequence<MeType> MeTypeSetType;
```

```
/**
Port ID structure, managedElement and port are required, other elements are
optional
*/
     struct PortIDType
     {
       Istring managedElement;
       Istring bay;
       Istring shelf;
       Istring drawer;
       Istring slot;
       Istring port;
     };

/**
Interface forward declarations
*/
     interface AttributeRanges;
     interface GenericTransportTTP;
     interface ManagedElementR2;

/**
Valuetype forward declarations
*/
     valuetype AttributeRangesValueType;
     valuetype GenericTransportTTPValueType;
     valuetype ManagedElementR2ValueType;

/**
Typedefs forward declarations
*/
     typedef MONameType AttributeRangesNameType;
     typedef MONameType GenericTransportTTPNameType;
     typedef MONameType ManagedElementR2NameType;



/**
     Exceptions for Conditional Package
*/
     exception NONeAliasPackage {};
     exception NOPortIdPackage {};



/**
```

10.2 Interfaces – Fine-grained

```
*/


/**
```

10.2.1 AttributeRanges

The AttributeRanges class allows the managed system to report the minimum and maximum values a certain attribute accepts, as well as the granularity, or step increments, of the range. Each AttributeRanges instance contains ranges for attributes belonging to one object class. The "kind" attribute in AttributeRanges denotes the object class for which ranges are being defined. "attributeName" specifies the name of the attribute for which a range is being defined. The range is then defined using the "minimum", "maximum", and

```
"granularity" attributes ("granularity" is not available for float types since
it is not needed).

For each ManagedElement instance representing a network element, one or more
AttributeRanges instances may be created. AttributeRanges instances are bound to
the ManagedElement instance via a containment relationship.

Ranges are defined per ManagedElement instance. This allows for an attribute to
have different ranges when it belongs to different network elements. In other
words, the scope of each AttributeRanges instance is the relevant objects
associated with the ManagedElement which contains the AttributeRanges instance.
*/

     valuetype AttributeRangesValueType: truncatable
itut_x780::ManagedObjectValueType
     {
       public Istring            kind;
          // GET
       public attributeRangeSetType   ranges;
          // GET

     }; // valuetype AttributeRangesValueType



     interface AttributeRanges: itut_x780::ManagedObject
     {
/**
This operation is used to get the object class for which the AttributeRanges
instance is defining attribute ranges. The returned value is a string containing
an object class name.
*/
       Istring kindGet()
          raises (itut_x780::ApplicationError);
/**
This operation is used to get the set of attribute ranges for the class defined
in the kind attribute. The returned value is a set of AttributeRangesType
structs, each containing the attribute name, the minimum and maximum bounds, as
well as the allowable granularity, or step increments within the bounds.
*/

       AttributeRangesSetType rangesGet()
          raises (itut_x780::ApplicationError);



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

     }; // interface AttributeRanges


/**
AttributeRanges Factory
It is expected that this object be created upon initialization by the Managed
System.
*/
```

```
     interface AttributeRangesFactory: itut_x780::ManagedObjectFactory
     {
       AttributeRanges create
          (in NameBindingType nameBinding,
          in MONameType superior,
          inout Istring name, // auto naming if empty string
          in StringSetType packageNameList)
          raises (itut_x780::ApplicationError,
          itut_x780::CreateError);

     }; // interface AttributeRangesFactory

/**
```

10.2.2 Generic transport TTP

```
*/
/**
This object is used to represent physical port inventory.
*/

     valuetype GenericTransportTTPValueType: truncatable
     itut_m3120::NetworkTPValueType
     {
/**  GenericTransportTTPValueType uses the following inherited form
itut_m3120:: TPValueType:
       public MONameSetType           supportedByObjectList;
          // points to the supporting circuit pack
          // GET
       public OperationalStateType        operationalState;
          // conditional, present if an instance supports it.
          // GET
       public AlarmStatusType             alarmStatus;
// conditional, present if the TP supports communications
// alarm notification.
          // GET
       public CurrentProblemSetType       currentProblemList;
          // conditional, present if the TP supports communications
          // alarm notification.
          // GET
       public AlarmSeverityAssignmentProfileNameType
          alarmSeverityAssignmentProfilePointer;
          // conditional, present if an instance supports
          // configuration of alarm severities.
          // GET-REPLACE


     GenericTransportTTPValueType uses the following inherited form
itut_m3120:: NetworkTPValueType:
       public PointDirectionalityType pointDirectionality;
          // GET
       public SignalIdType       signalId;
          // GET, SET-BY-CREATE


*/

       public PortIDType PortID;
          // conditional
          // PortIdPackage
          // present if the server TTP port is represented
          // GET
```

```
        public MONameSetType clientLinkEndPointerList;
           // GET-REPLACE

        public PointCapacityType  potentialCapacity;
           // conditional
           // present if the TTP is a rate adaptive technology
           // GET

     }; // valuetype GenericTransportTTPValueType




/**
Fine-Grained Interface Definition
*/

     interface GenericTransportTTP: itut_m3120::NetworkTP
     {


        PortIDType portIDGet()
           raises (itut_x780::ApplicationError,
           NOPortIdPackage);


        MONameSetType clientLinkEndPointerListGet()
           raises (itut_x780::ApplicationError);
        void clientLinkEndPointerListSet
           (in MONameSetType clientLinkEndPointerList)
           raises (itut_x780::ApplicationError);

        PointCapacityType potentialLinkEndCapacityPackageGet()
           raises (itut_x780::ApplicationError);



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

     }; // interface GenericTransportTTP


     interface GenericTransportTTPFactory:
        itut_x780::ManagedObjectFactory
     {
        itut_x780::ManagedObject create
           (in NameBindingType nameBinding,
               // module name containing Name Binding info.
           in MONameType superior,
               // Name of containing object.
           in string reqID,
               // Requested ID value for name, will be
               // empty if auto-naming is to be used.
           out MONameType name,
               // Entire name of newly created object.
```

```
          in StringSetType packageNameList,
              // List of packages requested.

          in MONameSetType supportedByObjectList,
              // may be nil
          in AlarmSeverityAssignmentProfileNameType
          alarmSeverityAssignmentProfilePointer,
in PointDirectionalityType  pointDirectionality,

          in MONameSetType clientLinkEndPointerList)

          raises (itut_x780::ApplicationError,
              itut_x780::CreateError);

     }; // interface GenericTransportTTPFactory
```

/**

10.2.3 ManagedElementR2

The ManagedElementR2 objects are managed objects that represent telecommunications equipment or TMN entities (either groups or parts) within the telecommunications network that perform managed element functions, i.e., provides support and/or service to the subscriber. Managed elements may or may not additionally perform mediation/OS functions. A managed element communicates with the manager over standard CORBA interfaces for the purpose of being monitored and/or controlled. A managed element contains equipment that may or may not be geographically distributed.

When the Managed Element object supports attribute value change notifications, the attributeValueChange notification shall be emitted when the value of one of the following attributes changes: alarm status, user label, version, location name, current problem list and enable audible visual local alarm. For the above attributes that may not be supported, the behaviour for emitting the attribute value change notification applies only when the attribute is supported by the managed object. When the object supports state change notifications, the stateChangeNotification shall be emitted if the value of administrative state or operational state or usage state changes.

Deletion by management protocol is not allowed. (The object should throw a DeleteNotAllowed exception in response to a delete operation.)

This interface is based on the Amendment 6/M.3100 Managed Element R2 object. The interface name ManagedElementR1 is skipped in this amendment to keep the names of the M.3100 and M.3120 objects aligned.

This valuetype is used to retrieve all of the ManagedElementR2 attributes in one operation. Most unsupported attributes will be returned as an empty string or list if they are not supported. Receipt of a empty string value does not mean the attribute is not supported, though.
*/

```
     valuetype ManagedElementR2ValueType: truncatable
itut_m3120::ManagedElementValueType
     {
       public Istring             modelCode;
          // GET
       public IstringSetType          neAliases;
          // conditional
          // neAliasPackage
          // GET
       public MeTypeSetType           managedElementType;
          // GET

     }; // valuetype ManagedElementR2ValueType


     interface ManagedElementR2: itut_m3120::ManagedElement
     {
/**
The following method the product model code of the Network Element. The product
model code is the manufacturer’s model identification information. It is vendor-
provided information that the vendor uses to distinguish the network element
among a family of products. This attribute is used by OSSs performing equipment
discovery and inventory processes.
*/
       Istring modelCodeGet ()
          raises (itut_x780::ApplicationError);
/**
The following method returns a set of strings containing product aliases of the
managed element as defined by the EMS. These aliases are given by the EMS to a
certain Managed Element instance. Having such aliases available via the EMS/NMS
interface is useful for relating Network Element names entered at the EMS, via
the Graphical User Interface or otherwise, to those found on the NMS user
interface. More importantly, these aliases may appear in alarms sent by certain
EMS software via a non-CORBA interface.
*/
       Istring neAliasesGet ()
          raises (itut_x780::ApplicationError,
             NOneAliasPackage);
/**
The following method returns a set of textual strings and UIDs that describes
the generic type of the Network Element modeled by the ManagedElementR2
instance. Multiple managed element type values may be used to describe hybrid
equipment.
*/
       MeTypeSetType managedElementTypeGet ()
          raises (itut_x780::ApplicationError);


       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, stateChange,
          stateChangeNotificationPackage)

     }; // interface ManagedElementR2


     interface ManagedElementR2Factory: itut_x780::ManagedObjectFactory
     {
       ManagedElementR2 create
          (in NameBindingType nameBinding,
          in MONameType superior,
          inout Istring name,     // auto naming if empty string
```

```
        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 ManagedElementR2Factory


/**
```

10.3 Interfaces – Façade

The behaviour of the façade interfaces are identical to the corresponding fine-grained interfaces. Therefore, comments are not included in the façade interfaces. Readers are referred to the fine-grained interface in clause 10.2 for the behaviour of the façade interface.

```
This clause can be omitted from IDL if a management system only supports
fine-grained interface.
*/




/**
```

10.3.1 AttributeRanges_F

```
*/

     interface AttributeRanges_F: itut_x780::ManagedObject_F
     {
       Istring kindGet(in MONameType name)
          raises (itut_x780::ApplicationError);

       attributeRangesSetType rangesGet(in MONameType name)
          raises (itut_x780::ApplicationError);



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

     }; // interface AttributeRanges_F

/**
```

10.3.2 GenericTransportTTP_F

```
*/

/**
Coarse-Grained Interface Definition
*/
     interface GenericTransportTTP_F: itut_m3120::NetworkTP_F
     {

/**
Instances of GenericTransportTTP are created using the
GenericTransportTTPFactory or automatically by the managed system.
*/

/**  GenericTransportTTP_F inherits the following methods from
     itut_m3120::TP_F:
     supportedByObjectListGet,
     operationalStateGet, alarmStatusGet, containedInSubnetworkListGet,
     currentProblemListGet, alarmSeverityAssignmentProfilePointerGet,
     alarmSeverityAssignmentProfilePointerSet

     GenericTransportTTP_F inherits the following methods from
     itut_m3120:: NetworkTP_F:
     pointDirectionalityGet, signalIdGet
*/
```

```
       PortIDType portIDGet
          (in MONameType name)
          raises (itut_x780::ApplicationError,
          NOPortIdPackage);

       MONameSetType clientLinkEndPointerListGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);
       void clientLinkEndPointerListSet
          (in MONameType name,
          in MONameSetType clientLinkEndPointerList)
          raises (itut_x780::ApplicationError);

/**

Provides potential bandwidth for rate adaptive server technology.
*/
       PointCapacityType potentialLinkEndCapacityPackageGet
       (in MONameType name)
          raises (itut_x780::ApplicationError);

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

     }; // interface GenericTransportTTP_F

/**
```

10.3.3 ManagedElementR2_F

```
*/

     interface ManagedElementR2_F: itut_m3120::ManagedElement_F
     {

       Istring modelCodeGet (in MONameType name)
          raises (itut_x780::ApplicationError);

       Istring neAliasesGet
          (in MONameType name)
          raises (itut_x780::ApplicationError,
          NOneAliasPackage);

       Istring managedElementTypeGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);


       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 ManagedElementR2_F

/**
```

10.4 Name binding

```
*/

/**
The following module contains name binding information.
*/
     module NameBinding
     {

/**
```

10.4.1 AttributeRanges

```
*/
       module AttributeRanges_ManagedElement
       {
          const string       superiorClass =
          "itut_m3120::ManagedElement";
          const boolean      superiorSubclassesAllowed = TRUE;
          const string       subordinateClass =
          "itut_m3120::AttributeRanges";
          const boolean      subordinateSubclassesAllowed = TRUE;
          const boolean      managerCreatesAllowed = FALSE;
          const DeletePolicyType deletePolicy =
          itut_x780::notDeletable;
          const string       kind = "AttributeRanges";
       }; // module AttributeRanges_ManagedElement

/**
```

10.4.2 GenericTransportTTP

```
*/
       module GenericTransportTTP_ManagedElement
       {
          const string  superiorClass =
             "itut_m3120::ManagedElement";
          const boolean superiorSubclassesAllowed = TRUE;
          const string  subordinateClass =
             "itut_m3120::GenericTransportTTP";
          const boolean subordinateSubclassesAllowed = TRUE;
          const boolean managerCreatesAllowed = FALSE;
          const DeletePolicyType deletePolicy =
             itut_x780::notDeletable;
          const string  kind = "GenericTransportTTP";
       }; // module GenericTransportTTP_ManagedElement

     }; // module NameBinding

}; // module itut_m3120

#endif // _itut_m3120_amd2_idl_
```

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	**TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits**
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2003