INTERNATIONAL TELECOMMUNICATION UNION

# ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

# M.3120

**Amendment 1**
(05/2002)

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 1: Protection switching

ITU-T Recommendation M.3120 (2001) – Amendment 1

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 1
## Protection switching


**Summary**

This amendment provides protection switching enhancements to the generic network information model. The model describes protection switching managed object classes and their properties that are generic and useful to describe information exchanged across all interfaces defined in M.3010 TMN architecture. These generic managed object classes are intended to be applicable across different technologies, architectures and services. The protection switching managed object classes in this amendment may be specialized to support the management of various telecommunications networks.



**Source**

Amendment 1 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 May 2002.

## 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 1
## Protection switching

### 1 Scope

This amendment provides protection switching model enhancements to the CORBA generic network and NE level information model. The IDL model describers protection switching managed object classes and their properties that are generic and useful to describe information exchanged across all interfaces defined in M.3010 TMN architecture. These generic protection switching object classes are intended to be applicable across different technologies, architectures and services. They can be extended by various telecommunication industries for managing specific network technologies, such as ATM, SONET/SDH and Ethernet.

### 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.

[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 interfaces*.

[5] ITU-T Recommendation M.3120 (2001), *CORBA generic network and network element level information model*.

[6] ITU-T Recommendation M.3100 (1995), *Generic network information model, plus* Amendment 2 (2000).

[7] ITU-T Recommendation G.774 (2001), *Synchronous digital hierarchy (SDH) – management information model for the network element view*.

[8] ITU-T Recommendation G.774.3 (2001), *Synchronous digital hierarchy (SDH) – management of multiplex-section protection for the network element view*.

### 3 Overview of the protection switching information model

Clause 4 defines a set of CORBA IDL interfaces for the protecting switching information model. These interfaces are translated manually from a set of ITU-T Recs. M.3100 Amendment 2 and G.774.3 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 4.5, a companion set of Façade interfaces are defined in 5.6. These façade interfaces are defined according to the coarse-grained framework and guidelines given in ITU-T Recs. Q.816.1 and X.780.1 for supporting coarse-grained CORBA interface. The name of these façade interfaces are the names of the corresponding fine-grained interfaces appended with "_F" (an underscore followed by a capital "F").

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.

Figures 1 and 2 show the inheritance, containment, and association relationship of the CORBA interfaces defined in this Recommendation. Note that façade interfaces follow the same inheritance hierarchy relationship as the corresponding fine-grained interfaces. It should also be noted that additional interfaces, although required, are not shown in the figures. Examples are the factory classes.

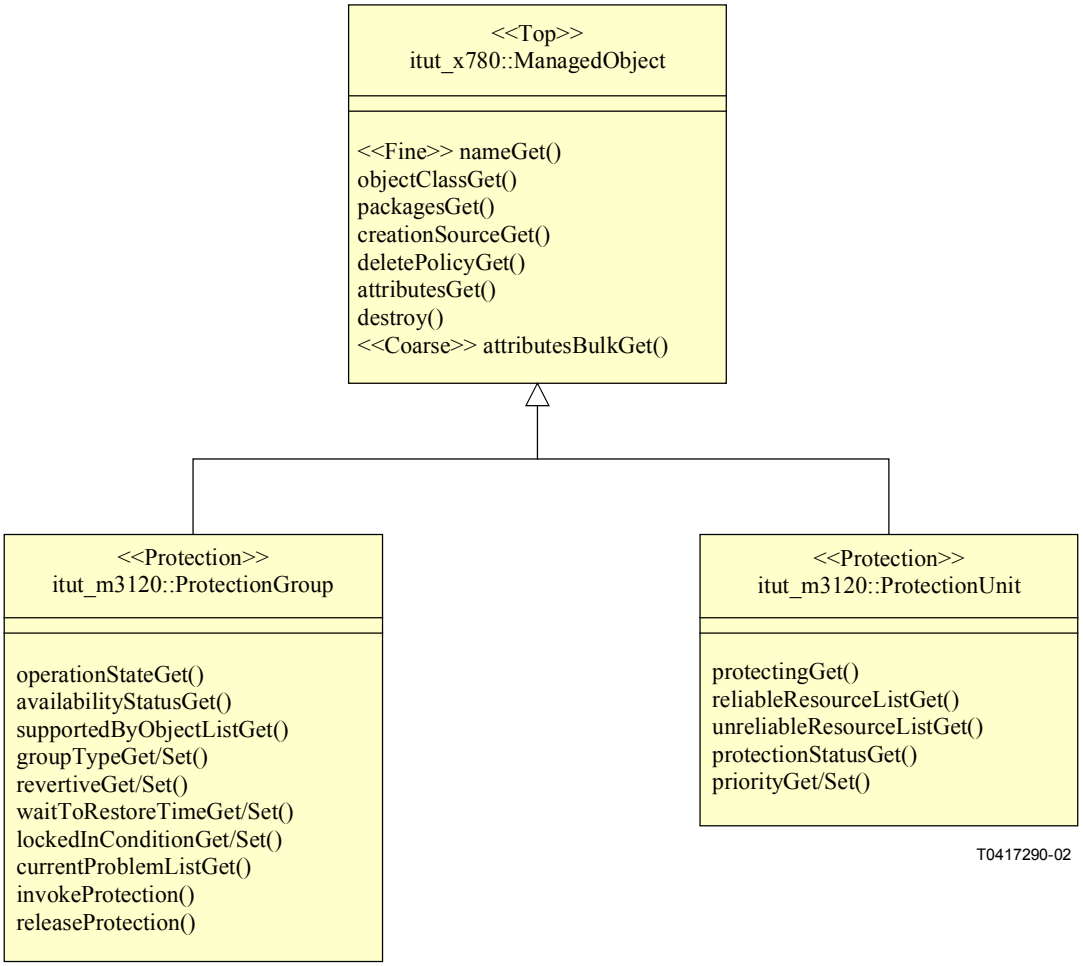


**Figure 1/M.3120 – Inheritance relationship**

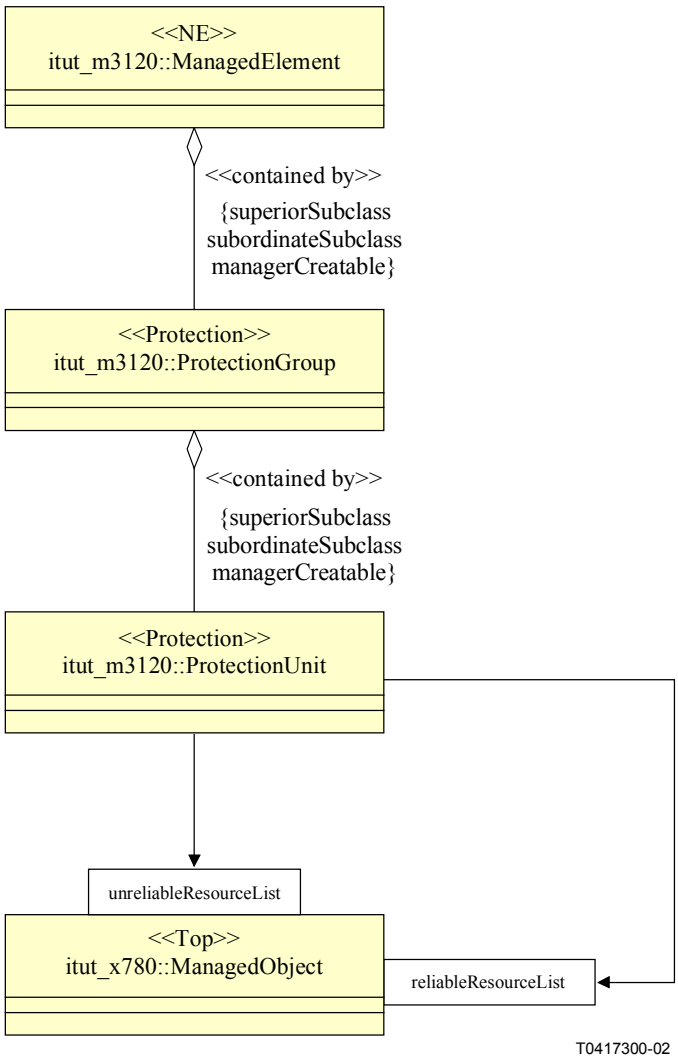


**Figure 2/M.3120 – Containment and association relationship**

## 4 Information Model IDL

```
#ifndef _itut_m3120_amd1_idl_
#define _itut_m3120_amd1_idl_


#include <itut_m3120.idl>


#pragma prefix "itu.int"


/**
This IDL code is intended to be stored in a file named "itut_m3120_amd1.idl"
located in the search path used by IDL compilers on your system. The M.3120
main module (defined in 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 M.3100 and G.855.1.
*/
module itut_m3120
{



/**
```

## 4.1 Imports

```
*/

/**
Types imported from itut_x780
*/
typedef itut_x780::AdditionalTextType AdditionalTextType;
typedef itut_x780::AdditionalInformationSetType AdditionalInformationSetType;
typedef itut_x780::CorrelatedNotificationSetType CorrelatedNotificationSetType;
typedef itut_x780::NotifIDType NotifIDType;
typedef itut_x780::NullType NullType;




/**
```

## 4.2 Forward Declarations

```
*/

/**
Interface forward declarations
*/
     interface ProtectionGroup;
     interface ProtectionUnit;



/**
Valuetype forward declarations
*/
     valuetype ProtectionGroupValueType;
     valuetype ProtectionUnitValueType;


/**
Typedefs forward declarations
*/
     typedef MONameType ProtectionGroupNameType;
     typedef MONameType ProtectionUnitNameType;




/**
```

## 4.3 Structures and Typedefs

```
Structures and typedefs defined in the following are arranged in alphanumeric
order.
*/

     typedef sequence<ProtectionGroupNameType> ProtectionGroupNameSetType;

     typedef sequence<ProtectionGroupNameType> ProtectionGroupNameSeqType;

     typedef sequence<ProtectionUnitNameType> ProtectionUnitNameSetType;

     typedef sequence<ProtectionUnitNameType> ProtectionUnitNameSeqType;

/**
This attribute specifies the criteria of the locked-in condition. The criteria
includes the automatic protection switching (APS) rate and the associated
setting and releasing time windows. If the number of APS of a Protection Unit
reaches the value specified in the hitsCount field within a moving time window
of specified length, the Protection Unit will enter the locked-in condition.
Each switch to protection and its subsequent release is considered as one hit.
The length of the time window for entering the locked-in condition is specified
in the settingWindowTime field. Once a Protection Unit is in the locked-in
condition, future request of APS will be denied until the locked-in condition is
released. The release criterion is no APS request within another moving time
window. The length of this time window is specified in the releasingWindowTime
field.
*/
     struct LockedInConditionType
     {
          unsigned short settingWindowTime;
          unsigned short releaseingWindowTime;
          unsigned short hitsCount;
     };

     enum ProtectionDirectionType
     {
          protectionDirectionTransmit,
          protectionDirectionReceive,
          protectionDirectionBidirectional
     };

     enum ProtectionGroupType
     {
          protectionGroupPlus, // 1+1 or hot-standby
          protectionGroupColon // M:N
     };

/**
The Protection Status attribute indicates the status of the protection switch in
an Protection Unit object. The following behaviour applies to this attribute:

-    This attribute must be capable of indicating pending as well as active
     switching requests relative to the protection unit. However, only one of
     the values lockout, forced switch, or manual switch can be present at the
     same time.

-    A protection system may support only a subset of the allowable values of
     this attribute. The subset of values to be supported by a system is
     implementation-specific.

-    The syntax of this attribute includes a sub-field "relatedUnit" which is of
     union of "fromProtectionUnitNumber" and "toProtectionUnitNumber". This sub-
     field is used to indicate on which unit the service is carried.
```

For a protected PU, both the fromProtectionUnit and the toProtectionUnit hold the ID of the related protecting PU. When switching to the protecting PU (i.e. service will be carried by the protecting PU), the toProtectionUnit choice is used. When switching back to the protected PU (service will be carried by the protected PU), the fromProtectionUnit choice is used.

For a protecting PU, both the fromProtectionUnit and the toProtectionUnit hold the ID of the related protected PU. When switching to the protected PU (i.e. service will be carried by the protected PU), the toProtectionUnit choice is used. When switching to the protecting PU (service will be carried by the protecting PU), the fromProtectionUnit choice is used.

If a system can support protection switching due to Resource Degrade (RD) besides Resource Fail (RF), protection switching of RD is similar to that in the subsequent description for RF.

The following allowable Protection Status values are associated with each protected Protection Unit (PU).

- No Request: No switch request is present on the unit. In this case, service is on the protected PU, status syntax is noRequest. For non-revertive system, the status syntax of the related protecting PU is also noRequest.

- Manual Switch to Protecting Unit Complete: The unit has completed a Manual Switch. In this case, service is on the related protectingPU, status syntax of the protected PU is manualSwitch (switchStatus: completed; relatedUnit: toPU). Status syntax of the related protecting PU is manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Release Failed: A time-out occurs while waiting for a release. In this case, service is still on the protectingPU, status syntax is releaseFailed plus the previous status, such as manualSwitch (switchStatus: completed; relatedUnit: toPU). Status syntax of the related protecting PU is still the previous status, such as manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Automatic Switch (RF) Pending: The unit has a Fail condition present and the protecting unit is unavailable. In this case, service is still on the protectedPU, status syntax is autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). Status syntax of the related protecting PU is autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) plus its previous status.

- Automatic Switch (RF) Complete: The unit has completed an Automatic Switch to the protecting unit due to an Equipment Fail condition. In this case, service is on the related protectingPU, status syntax of the protected PU is autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF). Status syntax of the related protecting PU is autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF).

- Automatic Switch (RF) Present, Operate failed: An automatic switch (RF) request is in progress and a time-out occurs while waiting for completion. In this case, service is still on the protectedPU, status syntax is autoSwitch (switchStatus: failed; relatedUnit: toPU; reason: RF). Status syntax of the related protecting PU is autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) plus its previous status.

- Force Switch Complete, Automatic Switch (RF) Pending: The unit has completed a Force Switch. Additionally, the unit has an automatic switch (RF) pending. In this case, service is on the related protectingPU, status syntax of the protected PU is forceSwitch (switchStatus: completed; relatedUnit: toPU) plus autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). Status syntax of the related protecting PU is

forceSwitch (switchStatus: completed; relatedUnit: fromPU) plus autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF).

- Automatic Switch Complete, Wait-To-Restore (revertive only): The unit has completed an Automatic Switch to the protecting unit. In this case, service is on the related protectingPU, status syntax of the protected PU is autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: WTR). Status syntax of the related protecting PU is autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: WTR).

- Force Switch Complete: The unit has completed a Force Switch to the protecting unit. In this case, service is on the related protectingPU, status syntax of the protected PU is forceSwitch (switchStatus: completed; relatedUnit: toPU). Status syntax of the related protecting PU is forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Protected Unit Lockout Completed: The unit has been locked out from the protecting unit. In this case, service is on the protectedPU, status syntax is lockout (switchStatus: completed).

- Protected Unit Lockout, Operate failed: The unit has been locked out from the protecting unit, and, the previously completed switch could not be released within the expected time-out. When the switch is released, the Operate failed status is removed. In this case, service is still on the related protectingPU, status syntax of the protected PU is lockout (switchStatus: completed) plus releaseFailed. Status syntax of the related protecting PU is still the previous status, such as manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Locked In: The unit is in the locked-in condition. This is caused by excessive protection switching events. In this case, service is on the protectedPU, status syntax is locked-in.

A non-revertive protected Protection Unit has the following additional status values:

- Do Not Revert: The protected unit has been switched to the protecting unit and the request to do so has been released. The switch to the protecting unit is maintained. In this case, service is on the related protectingPU, status syntax of the protected PU is doNotRevert. Status syntax of the related protecting PU is doNotRevert.

- Manual Switch to Protected Unit Complete: The unit has completed a Manual Switch from the protecting unit to the protected unit. In this case, service is on the protectedPU, status syntax is manualSwitch (switchStatus: completed; relatedUnit: fromPU). Status syntax of the related protecting PU is manualSwitch (switchStatus: completed; relatedUnit: toPU).

- Force Switch to Protected Unit Complete: The unit has completed a Force Switch from the protecting unit to the protected unit. In this case, service is on the protectedPU, status syntax is forceSwitch (switchStatus: completed; relatedUnit: fromPU). Status syntax of the related protecting PU is forceSwitch (switchStatus: completed; relatedUnit: toPU).

- Automatic Switch (RF) to Protected Unit Complete: The protecting unit has an Equipment Fail condition present and the protected unit is now being utilized. In this case, service is on the protectedPU, status syntax is autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF). Status syntax of the related protecting PU is autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF).

- Force Switch from Protecting Unit Complete, Automatic Switch (RF) Pending: The unit has completed a Force Switch from the protecting unit to the protected unit. Additionally, the protected unit has an automatic switch (RF) condition present. In this case, service is on the protectedPU, status syntax is forceSwitch (switchStatus: completed; relatedUnit: fromPU) plus autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). Status syntax of the related protecting PU is forceSwitch (switchStatus: completed; relatedUnit: toPU) plus autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF).

The following allowable Protection Status values are associated with each protecting Protection Unit:

- No Request: No switch request is present on the protecting unit. In this case, service is not on the protecting PU, status syntax is noRequest. For non-revertive system, the status syntax of the related protected PU is noRequest.

- Manual Switch to Protecting Unit Complete: A protected unit has completed a Manual Switch. In this case, service is on the protectingPU, status syntax is manualSwitch (switchStatus: completed; relatedUnit: fromPU). Status syntax of the related protected PU is manualSwitch (switchStatus: completed; relatedUnit: toPU).

- Automatic Switch (RF) Pending: A protected unit has an Equipment Fail condition present and the protecting unit is unavailable for this request. In this case, service is still on the protectedPU. Status syntax of the protecting PU is autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) plus its previous status, which causes its unavailability. Status syntax of the related protected PU is autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF).

- Automatic Switch Complete (RF) to Protecting Unit: A protected unit has completed an automatic switch (RF) to the protecting unit. In this case, service is on the protectingPU, status syntax is autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF). Status syntax of the related protected PU is autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF).

- Automatic Switch (RF) to Protecting Complete, Wait-To-Restore (revertive only): The unit has completed an Automatic Switch from the protected unit. In this case, service is on the protectingPU, status syntax is autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: WTR). Status syntax of the related protected PU is autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: WTR).

- Protecting Unit RF Present: The protecting unit has an Equipment Fail condition present. Status syntax of the protecting PU is resourceFailed.

- Force Switch Complete to Protecting Unit: The unit has completed a Force Switch from a protected unit to the protecting unit. In this case, service is on the protectingPU, status syntax is forceSwitch (switchStatus: completed; relatedUnit: fromPU). Status syntax of the related protected PU is forceSwitch (switchStatus: completed; relatedUnit: toPU).

- Protecting Unit Locked Out: The protecting unit has been locked out. In this case, service is not on the protectingPU, status syntax is lockout (switchStatus: completed).

- Protecting Unit Release Lock Out Failed: A release of a lockout is in progress and a time-out occurs waiting for the lockout condition to clear. In this case, service is not on the protectingPU, status syntax is lockout (switchStatus: failed).

A non-revertive protecting Protection Unit has the following additional status values:

- Do Not Revert: A protected unit has been switched to the protecting unit and the request to do so has been released. The switch to the protecting unit is maintained. In this case, service is on the protectingPU, status syntax is doNotRevert. Status syntax of the related protected PU is doNotRevert.

- Manual Switch to Protected Unit Complete: The unit has completed a Manual Switch from the protecting unit to the protected unit. In this case, service is on the protectedPU. Status syntax of the protecting PU is manualSwitch (switchStatus: completed; relatedUnit: toPU). Status syntax of the related protected PU is manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Force Switch to Protected Unit Complete: The protecting unit has completed a forced switch to the protected unit. In this case, service is on the protectedPU. Status syntax of the protecting PU is forceSwitch (switchStatus: completed; relatedUnit: toPU). Status syntax of the related protected PU is forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Force Switch to Protected Unit Complete, Protecting Unit Equipment Failed: The protecting unit has completed a forced switch to the protected unit. Additionally, there is an Equipment Fail condition on the protecting unit. In this case, service is on the protectedPU. Status syntax of the protecting PU is forceSwitch (switchStatus: completed; relatedUnit: toPU) plus equipmentFailed. Status syntax of the related protected PU is forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Automatic Switch (RF) to Protected Unit Complete: The protecting unit has an Equipment Fail condition present and the protected unit is now being utilized. In this case, service is on the protectedPU. Status syntax of the protecting PU is autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF). Status syntax of the related protected PU is autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF).

*/

```
enum PAutoSwitchReasonType
{
     pAutoSwitchReasonWaitToRestore,
     pAutoSwitchReasonSignalDegrade,
     pAutoSwitchReasonSignalFail
};

enum PRelatedUnitChoiceType
{
     pFrom,
     pTo
};

enum PSwitchStatusType
{
     pSwitchStatusPending,
     pSwitchStatusCompleted,
     pSwitchStatusFailed
};

enum PLockoutChoiceType
{
     pSwitchStatus,
     pReleaseFailed
};
```

```
union PRelatedUnitType switch (PRelatedUnitChoiceType)
{
     case pFrom:
        ProtectionUnitNameType fromUnit;
     case pTo:
        ProtectionUnitNameType toUnit;
};

struct PManualOrForcedSwitchType
{
     PSwitchStatusType      switchStatus;
     PRelatedUnitType       relatedUnit;
};

struct PAutoSwitchType
{
     PSwitchStatusType      switchStatus;
     PRelatedUnitType       relatedUnit;
     PAutoSwitchReasonType  autoSwitchReason;
};

union PLockoutType switch (PLockoutChoiceType)
{
     case pSwitchStatus:
        PSwitchStatusType   switchStatus;
     case pReleaseFailed:
        NullType            releaseFailed;

};

enum ProtectionStatusChoiceType
{
     protectionStatusNoRequest,
     protectionStatusDoNotRevert,
     protectionStatusManualSwitch,
     protectionStatusAutoSwitch,
     protectionStatusForcedSwitch,
     protectionStatusLockout,
     protectionStatusReleaseFailed,
     protectionStatusResourceFailed,
     protectionStatusLockedIn
};

union ProtectionStatusType switch (ProtectionStatusChoiceType)
{
     case protectionStatusNoRequest:
        NullType       noRequest;
     case protectionStatusDoNotRevert:
        NullType       doNotRevert;
     case protectionStatusManualSwitch:
        PManualOrForcedSwitchType    manualSwitch;
     case protectionStatusAutoSwitch:
        PAutoSwitchType              autoSwitch;
     case protectionStatusForcedSwitch:
        PManualOrForcedSwitchType    forcedSwitch;
     case protectionStatusLockout:
        PLockoutType                 lockout;
     case protectionStatusReleaseFailed:
        NullType       releaseFailed;
     case protectionStatusResourceFailed:
        NullType       resourceFailed;
     case protectionStatusLockedIn:
        NullType       lockedIn;
};
```

```
struct ProtectionStatusType
{
     ProtectionStatusChoiceType       choice;
     PSwitchStatusType                switchStatus;
        // may be NULL
     PRelatedUnitType                 relatedUnit;
        // may be NULL
     PAutoSwitchReasonType            autoSwitchReason;
        // may be NULL
};

typedef sequence<ProtectionStatusType> ProtectionStatusSetType;

enum PSwitchActionResultType
{
     pSwitchActionResultPreempted,
     pSwitchActionResultFailure,
     pSwitchActionResultTimeout
};

enum PSwitchType
{
     pSwitchManual,
     pSwitchForced,
     pSwitchLocked
};
```

```
/**
```

## 4.4 Exceptions

```
*/

/**
```

### 4.4.1 Exceptions and Constants for Conditional Package

```
*/

     exception NOpriorityPackage {};

     exception NOprotectionAlarmPackage {};

/**
This collection of constants represent the conditional packages defined in this
Recommendation. When a conditional package is supported in a particular managed
object instance, one of these strings will be contained in the packages
attribute
*/
       const string priorityPackage =
          "itut_m3120::priorityPackage";
       const string protectionAlarmPackage =
          "itut_m3120::protectionAlarmPackage";



/**
```

## 4.5 Interfaces – Fine-grained

```
*/

/**
```

### 4.5.1 ProtectionGroup

```
This object class is used for representing a protection system within a Network
Element (NE). Notifications of protection switch events and management system
control of lockouts, forced switches, and manual switches are the primary
management functions supported by this entity.

Instances of this object class may be automatically created in an agent, e.g.
immediately following the initialization of the NE resources involved in the
protection system, according to the make-up and mode of the NE. Instances of
this object class may be automatically deleted in the agent.

Multiple instances of the protectionGroupR2 object may exist in an NE (one for
each protection system supported by the NE). An instance of the ProtectionGroup
object would contain two or more instances of the ProtectionUnit object.

Either all or none of the Protection Unit instances within a Protection Group
object shall have the priorityPkg package. It is to be noted that, before the
creation of the ProtectionGroup object, the supported by object list (sbol)
attribute of a reliable resource such as termination point object may point to
an unreliable resource object such as circuit pack. But once the protection
group object is created, the sbol attribute would start pointing at the
protection group object.

This object class has the following attributes:

-   Operational State: This read-only attribute identifies whether or not the
    protection mechanism represented by this instance is capable of performing
    its normal functions.

-   Protection Group Type: This read-write attribute identifies whether the
    protection scheme used is 1+1 or M:N.

-   Revertive: This read-write attribute identifies whether or not the
    protection scheme used is revertive. The default value for this attribute
    shall indicate revertive operation, but this attribute should be able to be
    set to indicate non-revertive operation by a command from the manager.

-   Wait-To-Restore Time: This read-write attribute identifies the amount of
    time, in seconds, that the protection system should wait after a fault
    clears before switching back to the protected resource. This attribute is
    only relevant for revertive system operation.

-   LockedInCondition: This read-write attribute specifies the criteria of the
    locked-in condition.

This interface contains the following CONDITIONAL PACKAGES.

-   protectionAlarmPackage: Present if the system is capable of reporting
    failure of protection mechanism or failure of the protecting resource.

-   createDeleteNotificationsPackage: Present if an instance supports it.

-   attributeValueChangeNotificationPackage: Present if an instance supports
    it.

*/
    valuetype ProtectionGroupValueType: itut_x780::ManagedObjectValueType
    {
```

```
      public OperationalStateType      operationalState;
         // GET
      public AvailabilityStatusSetType availabilityStatus;
         // GET
      public MONameSetType             supportedByObjectList;
         // GET-REPLACE, ADD-REMOVE
      public ProtectionGroupType       groupType;
         // GET-REPLACE
      public boolean                   revertive;
         // GET-REPLACE-WITH-DEFAULT
      public unsigned short            waitToRestoreTime;
         // GET-REPLACE
      public LockedInConditionType     lockedInCondition;
         // GET-REPLACE
      public CurrentProblemSetType     currentProblemList;
         // conditional
         // protectionAlarmPackage
         // GET
}; // valuetype ProtectionGroupValueType


interface ProtectionGroup: itut_x780::ManagedObject
{
      const boolean revertiveDefault = TRUE;

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

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


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

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

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

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


      ProtectionGroupType groupTypeGet ()
         raises (itut_x780::ApplicationError);


      boolean revertiveGet ()
         raises (itut_x780::ApplicationError);

      void revertiveSet
         (in boolean revertive)
         raises (itut_x780::ApplicationError);


      unsigned short waitToRestoreTimeGet ()
         raises (itut_x780::ApplicationError);

      void waitToRestoreTimeSet
```

```
            (in unsigned short waitTime)
            raises (itut_x780::ApplicationError);


         LockedInConditionType lockedInConditionGet ()
            raises (itut_x780::ApplicationError);

         void lockedInConditionSet
            (in LockedInConditionType condition)
            raises (itut_x780::ApplicationError);


         CurrentProblemSetType currentProblemListGet ()
            raises (itut_x780::ApplicationError,
                      NOprotectionAlarmPackage);

/**
Invoke Protection: This action is used to request a lockout, a forced switch, or
a manual switch on one or more ProtectionUnit instances contained in the
ProtectionGroup object. The following input parameters are included in the
Invoke Protection action:

-    Switch Type (Manual, Forced or Lockout).

-    Protection Entity (Optional): ID(s) of the protected and/or protecting.

Protection Unit entity to which the request applies. If not present, the request
is meant to apply to all such entities in the Protection Group.

If a protecting unit is identified in the protectedUnits field, or if a
protected unit is identified in the protectingUnits field, the action fails.

If the request is Forced Switch or Manual Switch, the protectedUnits field shall
identify one or more protection units. If only one unit is identified in the
protectedUnits field, and there is only one protecting unit in the protection
group, the protectingUnits field may be omitted. If the protectingUnits field is
present, it shall identify the same number of units as the protectedUnits field.

If the request is Lockout, the protection entity field may be absent, indicating
that the request applies to all contained protection units. If the protection
entity field is present, any number of protection units may be identified in the
protectedUnits and/or protectingUnits field, and either field may be absent.

For a Lockout request, the specified protected units and/or protecting units are
locked out.

For requests which cannot be completed, either because the request is the
protecting unit is serving a request of higher priority, or failure occurs
(failure), or timeout occurs (timeout), the reply shall indicate why the request
could not be completed, and the request shall not be made pending.

*/
      void invokeProtection
            (in PSwitchType switchType,
            in ProtectionUnitNameSeqType protectedUnitList,
            in ProtectionUnitNameSeqType protectingUnitList,
            out PSwitchActionResultType actionResult)
            raises (itut_x780::ApplicationError);

/**
Release Protection: This action is used to release a lockout, a forced switch,
or a manual switch on one or more of the protectionUnit instances contained in
the protectionGroup object. The following input parameters are included in the
Release Protection action:
```

- Switch Type (Manual, Forced or Lockout).

- Protection Entity (Optional): ID(s) of the protected and/or protecting.

Protection Unit entity to which the request applies. If not present, the request is meant to apply to all such entities in the Protection Group.

If a protecting unit is identified in the protectedUnits field, or if a protected unit is identified in the protectingUnits field, the action fails.

If the request is Forced Switch or Manual Switch, the protectedUnits field shall identify one or more protection units, and the protectingUnits field shall be omitted. For each identified protected unit, if it is not switched to a protecting unit, the action fails.

If the request is Lockout, the protectionEntity field may be absent, indicating that the request applies to all contained protection units. If the protection entity field is present, any number of protection units may be identified in the protectedUnits and/or protectingUnits field, and either field may be absent.

For a Lockout request, the specified protected units and/or protecting units are no longer locked out. That is, the protected units are now under protection and the protecting units are now capable of providing protection.

For release requests which cannot be completed, the reply shall indicate why the request could not be completed.

```
*/
       void releaseProtection
          (in PSwitchType switchType,
          in ProtectionUnitNameSeqType protectedUnitList,
          in ProtectionUnitNameSeqType protectingUnitList,
          out PSwitchActionResultType actionResult)
          raises (itut_x780::ApplicationError);


       MANDATORY_NOTIFICATION(
              Notifications, protectionSwitchReporting)
       MANDATORY_NOTIFICATION(
              itut_x780::Notifications, stateChange)

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

     };  // interface ProtectionGroup


     interface ProtectionGroupFactory: 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,
        // GET-REPLACE, ADD-REMOVE
        in ProtectionGroupType groupType,
        // GET-REPLACE
        in boolean revertive,
        // GET-REPLACE-WITH-DEFAULT
        in unsigned short waitToRestoreTime,
        // GET-REPLACE
        in LockedInConditionType lockedInCondition)
        // GET-REPLACE
        raises (itut_x780::ApplicationError,
        itut_x780::CreateError);

    }; // interface ProtectionGroupFactory



/**
```

### 4.5.2 ProtectionUnit

The ProtectionUnit managed object class is used to represent the protected (i.e. working, regular or preferred) or protecting (i.e. backup or standby) resource in a protection system. It relates the resources (e.g. circuit packs) involved in the protection system and keeps track of the protection switching status of the resources.

Instances of this object class are instantiated by the agent according to the protection schemes adopted by the NE. A Protection Unit instance is deleted when the resource object instance pointed to by the Unreliable Resource List attribute is deleted, and may be created automatically when the associated resource object is created. The agent may also create and delete instances of this object class in order to reflect local modifications in the protection schemes. The attributeValueChange notification is used to notify changes of the Reliable Resource List, Protection Status, and Priority attributes.

Two or more instances of the ProtectionUnit object may exist within an instance of the ProtectionGroup object.

An instance of the ProtectionUnit object could contain an instance of the ProtectionCurrentData object.

A ProtectionUnit instance is related to instances of resource entities (e.g. Circuit Pack) via the Unreliable Resource List attribute. If the function of the resource entities (e.g. timing function, transport termination point function, etc.) is explicitly modelled as object instances in the NE, then a ProtectionUnit instance is also related to instances of the modelled function entity via the Reliable Resource List attribute.

This object class has the following attributes:

- Protecting: This read-only attribute identifies whether or not the protection unit is associated with a resource providing a protecting ("true") or protected ("false") role in the protection system.

- Unreliable Resource List: This read-only attribute identifies the unreliable resource (e.g. circuit pack entity) associated with the Protection Unit object (e.g. the actual protected or protecting resource). The syntax of this attribute is set-valued and could point to multiple instances of unreliable resources when a set of resources forms an atomic unit in the protection system.

- Reliable Resource List: This read-only attribute identifies the reliable resource (i.e. the functional entity), if there is any, associated with the Protection Unit. The value of this attribute of a protection unit (PU) will change when the PU is involved in a protection switch or release. For a protected PU, when it is not switched, this attribute is pointing to the associated reliable resource (i.e. the functional object) and when it is switched, this attribute points to NULL. For a protecting PU, when it is not switched, this attribute is pointing to NULL, and when it is switched, this attribute is pointing to the associated reliable resource (i.e. the functional object). The syntax of this attribute is set-valued and could point to multiple instances of reliable resource when a set of functional objects form an atomic unit in the protection system.

- Priority: This read-write attribute specifies the priority of the service carried on the resource associated with the protection Unit instance. Valid values for this attribute are integers, where the value 1 indicates the highest priority, and a larger value indicates a lower priority.

- Protection Status R1: This read-only attribute indicates the status of the protection switch in a Protection Unit object. The following behaviour applies to this attribute:

  - This attribute must be capable of indicating pending as well as active switching requests relative to the protection unit. However, only one of the values lockout, forced switch, or manual switch can be present at the same time.

  - A protection system may support only a subset of the allowable values of this attribute. The subset of values to be supported by a system is implementation-specific.

This interface contains the following CONDITIONAL PACKAGES.

- priorityPackage: Present if an instance supports it.

- attributeValueChangeNotificationPackage: Present if an instance supports it.

```
*/
     valuetype ProtectionUnitValueType: itut_x780::ManagedObjectValueType
     {
          public boolean                   protecting;
             // GET
          public MONameSetType             reliableResourceList;
             // GET
          public MONameSetType             unreliableResourceList;
             // GET
          public ProtectionStatusSetType   protectionStatus;
             // GET
          public unsigned short            priority;
             // conditional
             // priorityPackage
             // GET-REPLACE
     }; // valuetype ProtectionUnitValueType


     interface ProtectionUnit : itut_x780::ManagedObject
     {
       boolean protectingGet ()
          raises (itut_x780::ApplicationError);

/**
```

```
The value of the reliableResourceList attribute points to the reliable
resource(s) (e.g. the functional objects) that is/are associated with the
Protection Unit instance.
*/
       MONameSetType reliableResourceListGet ()
          raises (itut_x780::ApplicationError);

/**
The value of the unreliableResourceList attribute points to the unreliable
resource(s) (e.g. circuit pack) that is/are associated with the Protection Unit
instance.
*/
       MONameSetType unreliableResourceListGet ()
          raises (itut_x780::ApplicationError);

/**
*/
       ProtectionStatusSetType protectionStatusGet ()
          raises (itut_x780::ApplicationError);

/**
This attribute specifies the priority of the service (e.g. traffic) carried on
the resource associated with the protected ProtectionUnit instance. Valid values
for this attribute are integers, where the value 1 indicates the highest
priority, and a larger value indicates a lower priority.

For a protecting ProtectionUnit, the value of this attribute indicates the
priority of choice of the protecting ProtectionUnit relative to other available
protecting ProtectionUnit(s) within the same ProtectionGroup. The lower the
value, the more preferred the ProtectionUnit is relative to other
ProtectionUnits.

*/
       unsigned short priorityGet ()
          raises (itut_x780::ApplicationError,
                 NOpriorityPackage);

       void prioritySet
          (in unsigned short priority)
          raises (itut_x780::ApplicationError,
                 NOpriorityPackage);

       CONDITIONAL_NOTIFICATION(
          itut_x780::Notifications, attributeValueChange,
          attributeValueChangeNotificationPackage)

     }; // interface ProtectionUnit


     interface ProtectionUnitFactory: 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 unsigned short priority)
              // conditional
              // priorityPackage
              // GET-REPLACE
          raises (itut_x780::ApplicationError,
              itut_x780::CreateError);
```

```
};  // interface ProtectionUnitFactory



/**
```

## 4.6 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 4.5 for the
behaviour of the façade interface.

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

/**
```

### 4.6.1 ProtectionGroup_F

```
*/

     interface ProtectionGroup_F: itut_x780::ManagedObject_F
     {
          const boolean revertiveDefault = TRUE;

          OperationalStateType operationalStateGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          AvailabilityStatusSetType availabilityStatusGet
              (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);


          ProtectionGroupType groupTypeGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);


          boolean revertiveGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);
```

```
void revertiveSet
    (in MONameType name,
    in boolean revertive)
    raises (itut_x780::ApplicationError);


unsigned short waitToRestoreTimeGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);

void waitToRestoreTimeSet
    (in MONameType name,
    in unsigned short waitTime)
    raises (itut_x780::ApplicationError);


LockedInConditionType lockedInConditionGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);

void lockedInConditionSet
    (in MONameType name,
    in LockedInConditionType condition)
    raises (itut_x780::ApplicationError);


CurrentProblemSetType currentProblemListGet
    (in MONameType name)
    raises (itut_x780::ApplicationError,
              NOprotectionAlarmPackage);


void invokeProtection
    (in MONameType name,
    in PSwitchType switchType,
    in ProtectionUnitNameSeqType protectedUnitList,
    in ProtectionUnitNameSeqType protectingUnitList,
    out PSwitchActionResultType actionResult)
    raises (itut_x780::ApplicationError);

void releaseProtection
    (in MONameType name,
    in PSwitchType switchType,
    in ProtectionUnitNameSeqType protectedUnitList,
    in ProtectionUnitNameSeqType protectingUnitList,
    out PSwitchActionResultType actionResult)
    raises (itut_x780::ApplicationError);


MANDATORY_NOTIFICATION(
    Notifications, protectionSwitchReporting)
MANDATORY_NOTIFICATION(
    itut_x780::Notifications, stateChange)

CONDITIONAL_NOTIFICATION(
    Notifications, protectionAlarm,
    protectionAlarmPackage)
CONDITIONAL_NOTIFICATION(
    itut_x780::Notifications, objectCreation,
    createDeleteNotificationsPackage)
CONDITIONAL_NOTIFICATION(
    itut_x780::Notifications, objectDeletion,
    createDeleteNotificationsPackage)
```

```
          CONDITIONAL_NOTIFICATION(
              itut_x780::Notifications, attributeValueChange,
              attributeValueChangeNotificationPackage)

          };  // interface ProtectionGroup_F



/**
```

### 4.6.2 ProtectionUnit_F

```
*/

     interface ProtectionUnit_F : itut_x780::ManagedObject_F
     {
          boolean protectingGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          MONameSetType reliableResourceListGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          MONameSetType unreliableResourceListGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          ProtectionStatusType protectionStatusGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          unsigned short priorityGet
              (in MONameType name)
              raises (itut_x780::ApplicationError,
                        NOpriorityPackage);

          void prioritySet
              (in MONameType name,
              in unsigned short priority)
              raises (itut_x780::ApplicationError,
                        NOpriorityPackage);

     }; // interface ProtectionUnit_F



/**
```

## 4.7 Notifications

```
*/
/**
This interface contains the definitions of notifications emitted by managed
objects.

Notification users wishing to use typed notifications need to append the
operations below to M.3120 notification interface if there is any.

Notification publishers and subscribers wishing to use structured notifications
based on the operations defined below should follow rules for constructing and
reading the notification structure defined in ITU-T Rec. Q.816.

*/
          interface Notifications
```

```
    {

/**
The protectionSwitchReporting notification is emitted from the ProtectionGroup
object to report any protection switch events.

*/
      void protectionSwitchReporting
            (in ExternalTimeType              eventTime,
            in MONameType                     source,
                  // protection group
            in ObjectClassType                sourceClass,
                  // always ProtectionGroup
            in NotifIDType                    notificationIdentifier,
            in CorrelatedNotificationSetType  correlatedNotifications,
            in AdditionalTextType             additionalText,
            in AdditionalInformationSetType   additionalInfo,
            in MONameType                     reportedProtectionUnit,
            in ProtectionStatusSetType        oldProtectionStatus,
            in ProtectionStatusSetType        newProtectionStatus,
            in ProtectionDirectionType        protectionDirection);


/**
The protectionAlarm notification is emitted from the ProtectionGroup object to
report any protection mechanism failure or protecting resource failure.

*/
      void protectionAlarm
            (in ExternalTimeType              eventTime,
            in MONameType                     source,
               // protection group
            in ObjectClassType                sourceClass,
               // always ProtectionGroup
            in NotifIDType                    notificationIdentifier,
            in CorrelatedNotificationSetType  correlatedNotifications,
            in AdditionalTextType             additionalText,
            in AdditionalInformationSetType   additionalInfo,
            in ProbableCauseType              probableCause);

    }; // interface Notfications

/**
```

## 4.8 Name Binding

```
*/

/**

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

### 4.8.1 ProtectionGroup

```
*/
      module ProtectionGroup_ManagedElement
      {
            const string   superiorClass =
             "itut_m3120::ManagedElement";
            const boolean  superiorSubclassesAllowed = TRUE;
            const string   subordinateClass =
```

```
                "itut_m3120::ProtectionGroup";
            const boolean  subordinateSubclassesAllowed = TRUE;
            const boolean  managerCreatesAllowed = TRUE;
            const DeletePolicyType deletePolicy =
               itut_x780::deleteOnlyIfNoContainedObjects;
            const string   kind = "ProtectionGroup";
      }; // module ProtectionGroup_ManagedElement

/**
```

### 4.8.2 ProtectionUnit

```
*/
      module ProtectionUnit_ProtectionGroup
      {
            const string   superiorClass =
                  "itut_m3120::ProtectionGroup";
            const boolean  superiorSubclassesAllowed = TRUE;
            const string   subordinateClass =
                  "itut_m3120::ProtectionUnit";
            const boolean  subordinateSubclassesAllowed = TRUE;
            const boolean  managerCreatesAllowed = TRUE;
            const DeletePolicyType deletePolicy =
                  itut_x780::deleteOnlyIfNoContainedObjects;
            const string   kind = "ProtectionUnit";
      }; // module ProtectionUnit_ProtectionGroup


    }; // module NameBinding

/**
```

## 4.9 ProbableCauseConst

```
This module contains the constant values defined for the ProbableCause UID.
The values below need to append to M.3120 module ProbableCauseConst.

*/

    module ProbableCauseConst
    {

/**
Values 81-100 are reserved for equipment alarm related probable causes.
*/
      const short protectionMechanismFailure = 81;
      const short protectingResourceFailure = 82;

    }; // module ProbableCauseConst




}; // module itut_m3120



#endif // _itut_m3120_amd1_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 |

*22414*

Printed in Switzerland
Geneva, 2002