INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.780

Corrigendum 1

(10/2001)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI management – Management functions and ODMA functions

TMN guidelines for defining CORBA managed objects

Corrigendum 1

ITU-T Recommendation X.780 (2001) – Corrigendum 1

(Formerly CCITT Recommendation)

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	**X.730–X.799**
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999

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

ITU-T Recommendation X.780

TMN guidelines for defining CORBA managed objects

CORRIGENDUM 1

Source

Corrigendum 1 to ITU-T Recommendation X.780 (2001) was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 7 October 2001.

CONTENTS

ITU-T Recommendation X.780

TMN guidelines for defining CORBA managed objects

CORRIGENDUM 1

1) *Subclause 2.1*

Add a new reference:

[8] ITU-T X.739 (1993) | ISO/IEC 10164-11 (1994), *Information technology – Open Systems Interconnection – Systems management: Metric objects and attributes*.

2) *Subclause 5.1.3*

After the second sentence in the first paragraph, add the following sentence:

String constants are defined for each conditional package string name.

3) *Subclause 5.5*

In the IDL, delete the following lines:

```
valuetype CreateErrorInfoType : ApplicationErrorInfoType {
        public MOSetType        relatedObjects;
        public AttributeSetType attributeList;
};

valuetype DeleteErrorInfoType : ApplicationErrorInfoType {
        public MOSetType        relatedObjects;
        public AttributeSetType attributeList;
};
```

and replace them with the following:

```
valuetype CreateErrorInfoType : ApplicationErrorInfoType {
        public MONameSetType    relatedObjects;
        public AttributeSetType attributeList;
};

valuetype DeleteErrorInfoType : ApplicationErrorInfoType {
        public MONameSetType    relatedObjects;
        public AttributeSetType attributeList;
};
```

4) *Subclause 6.3*

Delete the last line of the first paragraph:

Future versions of CORBA plan to allow user-defined exceptions on attribute access, and these guidelines may change to take advantage of this.

After the first paragraph, add the following paragraph:

The definition of attributes containing references to managed objects should be avoided. Instead, managed object names should be used.

5) *New subclause 6.3.6*

Add a new subclause 6.3.6 entitled "Common Types" as follows:

6.3.6 Common types

A module defining common types is found in Annex C. These types are used by more than one Recommendation. This module is used to provide a single place for the definition of these types. Additional types will be added to this module as needed.

6) *Subclause 6.4*

In the first paragraph, after the second sentence, add the following sentence:

The use of parameters containing references to managed objects, however, should be avoided. Instead, managed object names should be used

7) *Subclause 6.6*

After the third paragraph, add the following paragraph:

In addition, for each defined conditional package, there shall be defined a constant string that can be used in the packages attribute for managed object instances using that conditional package. To differentiate packages from different IDL files, this string must be fully qualified. For example:

```
const string administrativeStatePackage =
      "itut_m3120::administrativeStatePackage";
```

8) *Subclause 6.9.1*

Replace subclause 6.9.1 with the following:

6.9.1 Create Operations

Each factory interface shall define a single operation for clients to use to create objects. The name of this operation shall be "create" and it shall return a *ManagedObject* reference. This shall be used to return a reference to the newly created object, which may then be "narrowed" by the client to the specific interface type for that object, or used directly for base class operations. The first five parameters to every create operation are always the same. After these come parameters for each writeable or set-by-create attribute defined for the managed object. (A set-by-create attribute is one for which the object has no "set" operation, but for which a value is specified on the create operation.) The names of these parameters are the same as the name of the attribute. (This is the name of an attribute accessor operation minus the ending "Get" or "Set".) Each create operation also has to accept parameters to set the values of any writeable or set-by-create attributes of all super-classes of the object created by the factory. Here is an example of a create operation for an equipment factory:

```
ManagedObject create(
        in NameBindingType nameBinding,        // module name containing Name Binding
                                               // info.
        in MONameType superiorObject,          // 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 packages,             // List of packages requested.
        …                                      // Writeable and set-by-create values
                                               // for Equipment superclass attributes.
        …                                      // Writeable and set-by-create values
                                               // for Equipment attributes.
);
```

9) *Subclause 6.9.1.2*

In the first sentence of the first paragraph, replace the words "is a reference to" *with* "is the name of." *In the third sentence of the first paragraph, replace the words* "supplied reference" *with* "superior object." *In the last sentence of the second paragraph, replace the words* "reference would be null" *with* "name would be an empty string."

10) *New subclause 6.9.1.3*

Insert a new subclause 6.9.1.3 entitled "Requested ID" and renumber the existing subclause 6.9.1.3 and subsequent subclauses.

6.9.1.3 Requested ID

The third parameter is the ID requested to be assigned to the new object. This string will become the *ID* field of the CORBA Name Binding created in the CORBA naming service for the new object. Thus, this ID string, along with the *kind* field contained in the name binding module, names the new object relative to the superior object. If the client supplies an empty string for this parameter, the factory shall automatically choose an ID value. The factory raises a *duplicateName CreateError* exception if the supplied ID is a duplicate. (This means both the *ID* and *kind* fields match an existing object contained by the superior object.)

11) *Subclause 6.9.1.4*

Replace the contents of subclause 6.9.1.4 (entitled "Name," and newly renumbered by the item above) with the following:

The fourth parameter shall be used by the factory to return the entire name of the newly created managed object to the client.

12) *Subclause 9.1*

After the 8th list item, add the following list item:

9) Adhere to the conditional package string constant conventions specified in 6.6.

The subsequent items should be renumbered.

13) *Annex A*

In the IDL, delete the following lines:

```
/** A SuspectObject identifies an object that may be the cause of a
failure. It is usually a component of a SuspectObjectList.
@member objectClass            Object class of the suspect object
@member suspectObjectInstance  Object instance of the suspect
                               object
@member failureProbability     Optional failure responsibility
                               probability from 1 to 100
*/

struct SuspectObjectType {
        ObjectClassType                objectClass;
        ManagedObject                  suspectObjectInstance;
        UnsignedShortTypeOpt           failureProbability;
};
```

and in their place insert the following lines:

```
/** A SuspectObject identifies an object that may be the cause of a
```

```
failure. It is usually a component of a SuspectObjectList.
@member objectClass           Object class of the suspect object
@member suspectObject         Name of the suspect object
@member failureProbability    Optional failure responsibility
                              probability from 1 to 100
*/

struct SuspectObjectType {
        ObjectClassType                 objectClass;
        MONameType                      suspectObject;
        UnsignedShortTypeOpt            failureProbability;
};
```

Delete the following lines:

```
valuetype CreateErrorInfoType : ApplicationErrorInfoType {
         public MOSetType        relatedObjects;
         public AttributeSetType attributeList;
};
```

and in their place insert the following lines:

```
valuetype CreateErrorInfoType : ApplicationErrorInfoType {
         public MONameSetType    relatedObjects;
         public AttributeSetType attributeList;
};
```

Delete the following lines:

```
valuetype DeleteErrorInfoType : ApplicationErrorInfoType {
         public MOSetType        relatedObjects;
         public AttributeSetType attributeList;
};
```

and in their place insert the following lines:

```
valuetype DeleteErrorInfoType : ApplicationErrorInfoType {
         public MONameSetType    relatedObjects;
         public AttributeSetType attributeList;
};
```

Delete the following lines:

```
/** Forward declaration. CORBA uses object references
of type "object" to identify objects. These are used instead of ASN.1
object instances. For network management interfaces, all objects will
inherit from the "ManagedObject" interface. */

interface ManagedObject;

/** MO Set is a set of ManagedObject references. */

typedef sequence <ManagedObject> MOSetType;

/** MO Seq is a sequence of ManagedObject references. */

typedef sequence <ManagedObject> MOSeqType;
```

and in their place insert the following lines:

```
/** A sequence of names is defined as just a sequence of names. */

        typedef sequence <NameType> NameSeqType;
```

After these lines:

```
/** A set of names is definded as a sequence of names. */

typedef sequence <NameType> NameSetType;
```

insert the following lines:

```
        /** A managed object name is just a name */

        typedef NameType MONameType;

        /** A sequence of managed object names is defined as just a sequence
        of names. */

        typedef NameSeqType MONameSeqType;

        /** A set of managed object names is defined as just a set
        of names. */

        typedef NameSetType MONameSetType;
```

14) *New Annex C*

Add the following new annex defining common types.

ANNEX C

The common types CORBA IDL module

```
#ifndef _itut_x780ct_idl_
#define _itut_x780ct_idl_

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

#pragma prefix "itu.int"

/**
This IDL code is intended to be stored in a file named "itut_x780ct.idl"
located in the search path used by IDL compilers on your system.
*/

/**
This module, itut_x780ct, contains common data type for ITU-T CORBA
framework based information model. This module can be extended by adding
new data type.
*/
module itut_x780ct
{
```

// Imports

```
/**
Types imported from itut_x780
*/
        typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
        typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
```

// Data Types (X.721)

```
        struct CounterThresholdType
        {
                long            comparisonLevel;
                long            offsetValue;
                boolean         notificationOnOff;
        };
        struct NotifyThresholdType
        {
                float           threshold;
                boolean         notifyOnOff;
        };

        struct GaugeThresholdType
        {
                NotifyThresholdType     notifyLow;
                NotifyThresholdType     notifyHigh;
        };

        typedef sequence<GaugeThresholdType> GaugeThresholdSetType;

        enum TideMarkChoice
        {
                tideMarkChoiceMax,
                tideMarkChoiceMin
        };

        union TideMarkType switch (TideMarkChoice)
        {
                case tideMarkChoiceMax:
                        float           maxTideMark;
                case tideMarkChoiceMin:
                        float           minTideMark;
        };

        struct TideMarkInfoType
        {
                TideMarkType            currentTideMark;
                TideMarkType            previousTideMark;
                GeneralizedTimeType     resetTime;
        };
```

// Data Types (X.739)

```
/**
PerceivedSeverityTypeOpt is an optional type. If the discriminator is true
the value is present, otherwise the value is null.
*/
        union PerceivedSeverityTypeOpt switch (boolean)
        {
                case TRUE:
                        PerceivedSeverityType   val;
        };

/**
The SeverityIndicatingThresholdType contains the threshold level, which is to
be applied to counter/gauge attribute. It shall be initialized when the
managed object in which it is included is created and may be modified. An
optional parameter is used to associate the threshold level to the severity
parameter of the emitted notification. The generation of the notification can
be switched off using the boolean parameter notifyOnOff. The severity parameter
is mandatory if notifyOnOff is true.

If a notify-high or notify-low switch is on (true) and the severity indication
value is not present, then the indeterminate value is sent in the notification
unless the managed object behaviour specifies another value.
*/
        struct SeverityIndicatingThresholdType
        {
                float                           threshold;
                boolean                         notifyOnOff;
                PerceivedSeverityTypeOpt        severityIndication;
        };

/**
The attribute of SeverityIndicatingGaugeThresholdType has similar behaviour to
the gauge-threshold attribute defined in X.721. The syntax has an added
parameter for indicating the associated severity, as defined in
```

```
SeverityIndicatingThresholdType, to the notification triggered by the crossing
of the corresponding threshold level. As an enhancement to the syntax of the
gauge-threshold attribute type it adds an optional severity indication
parameter to the syntax of both the notify-high and notify-low sub-members
within each threshold level member. This attribute type has additional
behaviour associated with these optional perceived severity indication
parameters, which is defined as follows:

- If the notify-high's switch is on (true), the notify-high's severity
indication value shall be reported in the perceived severity parameter of a
notification triggered by the gauge value crossing the notify-high's
gauge-threshold value in the positive going direction.

- If the notify-low's switch is on (true), the notify-low's severity
indication value shall be reported in the perceived severity parameter of a
notification triggered by the gauge value crossing the notify-low's
gauge-threshold value in the negative going direction.

- If both switches are on (true) for a single threshold level, one of the
severity indication values shall be "clear". The severity indicating
gauge-threshold shall only emit a clear event notification if the
corresponding threshold level (either notify-high or notify-low) notification
has been emitted and no other clear notification for this threshold level pair
has been emitted since the previous corresponding threshold level notification
has been emitted.
*/
        struct SeverityIndicatingGaugeThresholdType
        {
                SeverityIndicatingThresholdType notifyLow;
                SeverityIndicatingThresholdType notifyHigh;
        };

/**
The order is insignificant.
*/
        typedef sequence<SeverityIndicatingGaugeThresholdType>
                SeverityIndicatingGaugeThresholdSetType;

        struct TimeIntervalType
        {
                unsigned short day;
                unsigned short hour;
                unsigned short minute;
                unsigned short second;
                unsigned short ms;
        };

/**
The following definitions are translated from X.739 ASN.1 definitions.
*/
        enum TimePeriodChoice
        {
                timePeriodChoiceDays,
                timePeriodChoiceHours,
                timePeriodChoiceMinutes,
                timePeriodChoiceSeconds,
                timePeriodChoiceMilliSeconds,
                timePeriodChoiceMicroSeconds,
                timePeriodChoiceNanoSeconds,
                timePeriodChoicePicoSeconds
        };

        union TimePeriodType switch (TimePeriodChoice)
        {
                case timePeriodChoiceDays:
                        long day;
                case timePeriodChoiceHours:
                        long hour;
                case timePeriodChoiceMinutes:
                        long minute;
                case timePeriodChoiceSeconds:
                        long second;
                case timePeriodChoiceMilliSeconds:
                        long ms;
                case timePeriodChoiceMicroSeconds:
                        long us;
                case timePeriodChoiceNanoSeconds:
                        long ns;
                case timePeriodChoicePicoSeconds:
                        long ps;
        };

}; // module itut_x780ct

#endif // _itut_x780ct_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, 2002