INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.780

Amendment 1

(05/2002)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI management – Management functions and ODMA functions

TMN guidelines for defining CORBA managed objects

Amendment 1: System objects and user guide for bulk attribute retrieval

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

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

Amendment 1
System objects and user guide for bulk attribute retrieval

Summary

This amendment to ITU-T Rec. X.780 (2001) adds the definition of the System and Subsystem managed objects, and a non-normative appendix containing a user guide.

Source

Amendment 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 29 May 2002.

CONTENTS

ITU-T Recommendation X.780

TMN guidelines for defining CORBA managed objects

Amendment 1
System Objects and user guide for bulk attribute retrieval

1) Clause 2.1

Add the following new reference to clause 2.1:

[8] ITU-T Recommendation X.720 (1992) | ISO/IEC 10165:1993, *Information technology – Open Systems Interconnection – Structure of management information: Management information model.*

2) New clauses 5.8 and 5.9

Add the following new clauses after clause 5.7:

5.8 System managed object

In addition to the top-most *ManagedObject* class, the IDL also contains a couple of managed object definitions that specialize *ManagedObject*. The *System* managed object class is used to represent a set of hardware and software that forms an autonomous whole capable of performing information processing and/or information transfer. The entire specification of the sequence of name bindings to be used in constructing the distinguished name for a *System* managed object is outside the scope of this Recommendation. Examples of names for systems are specified in ITU-T Rec. X.720 [8].

An instance of this managed object class may be used as the superior in naming managed objects representing either information processing and or information transfer resources contained within this instance.

Note that the managed object defined here does not have the ITU-T Rec. X.721 [6] Supported Features attribute since the CORBA interface does not define negotiable Functional Units.

5.8.1 Notifications on the system managed object

The *System* managed object has the following notifications:

Table n/X.780 – System notifications

Notification	Conditional Package (if Conditional)
Object Creation	"itut_x780::createDeleteNotificationsPackage"
Object Deletion	"itut_x780::createDeleteNotificationsPackage"
State Change	"itut_x780::stateChangeNotificationPackage"

Changes in the following states (when defined) will cause State Change notifications (when supported) to be emitted:

– Administrative State.

– Operational State.

– Usage State.

5.9 Subsystem managed Object

The *Subsystem* managed object is a subclass of the *System* managed object class (see 5.8) and is contained by a *System* or another *Subsystem* managed object class instance. The *Subsystem* managed object class may be used as a common containment point for managed objects in a system that relate to the operation of a given layer. The choice of structuring within a system is dependent on what structure the system designer wishes to present externally for management purposes.

This managed object class represents a portion of a system where components are named independently of the components of other subsystems.

3) Annex A

Add the following to the IDL in Annex A.

After the last exception in the **"EXCEPTIONS"** *portion of the IDL, which is shown in this line:*

```
     exception InvalidString {};
```

Add these lines:

```
     const string administrativeStatePackage =
          "itut_x744d1::administrativeStatePackage";
     const string createDeleteNotificationsPackage =
          "itut_x744d1::createDeleteNotificationsPackage";
     const string stateChangeNotificationPackage =
          "itut_x744d1::stateChangeNotificationPackage";
     exception NOadministrativeStatePackage {};
```

After the **MANAGED OBJECT FACTORY INTERFACE** *definition, which ends with this line:*

```
     }; // end of ManagedObjectFactory interface
```

Add the following lines:

```
// SYSTEM INTERFACE

     /** This valuetype is used to retrieve multiple attributes.  */

     valuetype SystemValueType : truncatable ManagedObjectValueType {
          public OperationalStateType operationalState;
              // GET
          public UsageStateType usageState;
              // GET
          public AdministrativeStateType administrativeState;
              // GET-REPLACE
              // administrativeStatePackage

     }; // valuetype SystemValueType

     /**  The System managed object class is used to represent a set of hardware
     and software that forms an autonomous whole capable of performing
     information processing and/or information transfer.

     The entire specification of the sequence of name bindings to be used in
     constructing the distinguished name for a System managed object is outside
     the scope of this Recommendation. Name bindings to the Recommendation
     M.3120 Managed Element managed object class is supplied. Examples of names
     for systems are specified in Recommendation X.720.

     NOTE - This definition does not correspond to real open system but
     corresponds to real systems in Recommendation X.200.
```

```
    An instance of this managed object class may be used as the superior in
    naming managed objects representing either information processing and or
    information transfer resources contained within this instance.

    Note that this does not have the Supported Features attribute defined in
    Recommendation X.721, since the CORBA interface does not define negotiable
    Functional Units.
    */

    interface System : ManagedObject
    {
      /**
      Operational State, Usage State and Administrative State are described in
      Recommendation X.731
      */

      OperationalStateType operationalStateGet ()
             raises (ApplicationError);

      UsageStateType usageStateGet ()
             raises (ApplicationError);

      /**
      PRESENT IF an instance supports it.
      */

      AdministrativeStateType administrativeStateGet ()
             raises (ApplicationError,
                   NOadministrativeStatePackage);

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

      CONDITIONAL_NOTIFICATION(
             Notifications, objectCreation,
             createDeleteNotificationsPackage)

      CONDITIONAL_NOTIFICATION(
             Notifications, objectDeletion,
             createDeleteNotificationsPackage)

      CONDITIONAL_NOTIFICATION(
             Notifications, stateChange,
             stateChangeNotificationPackage)

    }; // interface System
```

// SYSTEM FACTORY INTERFACE

```
    /**
    Factory for System
    */

    interface SystemFactory : ManagedObjectFactory
    {
         ManagedObject create
             (in NameBindingType nameBinding,
             in MONameType superior,
             in string reqID,    // auto naming if empty string
             out MONameType name,
             in AdministrativeStateType administrativeState
                   // GET-REPLACE
```

```
                // administrativeStatePackage
            )
            raises (ApplicationError,
                CreateError);

    }; // interface SystemFactory

// SUBSYSTEM INTERFACE

    /**  This valuetype is used to retrieve multiple attributes. */

    valuetype SubsystemValueType : truncatable SystemValueType {

    }; // valuetype SubsystemValueType

    /**
    The Subsystem managed object class may be used as a common containment
    point for managed objects in a system that relate to the operation of a
    given layer. The choice of structuring within a system is dependent on what
    structure the system designer wishes to present externally for management
    purposes.

    This managed object class represents a portion of a system where components
    are named independently of the components of other subsystems.
    */

    interface Subsystem : System
    {

    }; // interface Subsystem


// SUBSYSTEM FACTORY INTERFACE

    /**
    Factory for Subsystem
    */

    interface SubsystemFactory : ManagedObjectFactory
    {
        ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID,    // auto naming if empty string
            out MONameType name,
            in AdministrativeStateType administrativeState
                // GET-REPLACE
                // administrativeStatePackage
            )
            raises (ApplicationError,
                CreateError);

    }; // interface SubsystemFactory
```

After the definition of the **NOTIFICATIONS INTERFACE***, which ends with this line:*

```
  }; // end of Notifications interface
```

Add the following lines:

```
// NAME BINDINGS

     /**
     This name binding is used to name the System object
     relative to the local root. That is, enable it to be
     the top-most managed object on a system.
     */

     module System
     {
          const string superiorClass = "";
          const boolean superiorSubclassesAllowed = FALSE;
          const string subordinateClass = "itut_x780::System";
          const boolean subordinateSubclassesAllowed = TRUE;
          const boolean managerCreatesAllowed = FALSE;
          const DeletePolicyType deletePolicy =
              itut_x780::deleteOnlyIfNoContainedObjects;
          const string kind = "System";

     }; // module System

     /**
     This name binding is used to name the Subsystem object
     relative to a Subsystem object.
     */

     module Subsystem_Subsystem
     {
          const string superiorClass = "itut_x780::Subsystem";
          const boolean superiorSubclassesAllowed = TRUE;
          const string subordinateClass = "itut_x780::Subsystem";
          const boolean subordinateSubclassesAllowed = TRUE;
          const boolean managerCreatesAllowed = TRUE;
          const DeletePolicyType deletePolicy =
              itut_x780::deleteOnlyIfNoContainedObjects;
          const string kind = "Subsystem";

     }; // module Subsystem_System

     /**
     This name binding is used to name the Subsystem object
     relative to a System object.
     */

     module Subsystem_System
     {
          const string superiorClass = "itut_x780::System";
          const boolean superiorSubclassesAllowed = TRUE;
          const string subordinateClass = "itut_x780::Subsystem";
          const boolean subordinateSubclassesAllowed = TRUE;
          const boolean managerCreatesAllowed = TRUE;
          const DeletePolicyType deletePolicy =
              itut_x780::deleteOnlyIfNoContainedObjects;
          const string kind = "Subsystem";

     }; // module Subsystem_System
```

4) New Appendix II

Add the following new non-normative appendix:

Appendix II
User guide for bulk attribute retrieval

This appendix provides additional information about the TMN CORBA framework intended to help those implementing systems that conform to the framework Recommendations.

Bulk retrieval of attributes

The top-most managed object interface, *ManagedObject*, defines an operation that enables a managing system to retrieve multiple attributes from a managed object in one operation. The signature of this operation, *AttributesGet*, is shown below:

```
ManagedObjectValueType attributesGet (
     in NameType name,
     inout  StringSetType attributeNames)
raises (ApplicationError);
```

Note that the names of the attributes requested by the managing system are submitted in the *attributeNames* parameter, and the names of the attributes actually returned by the managed object are also returned in the *attributeNames* parameter. Because the list of attributes returned by the managed object may differ from the requested list, the in/out parameter *attributeNames* may be changed by the managed object. Managing system implementations that wish to repeatedly use the same list of attributes will not want that list modified by the managed object. One solution to this problem is made possible by the way in which CORBA ORBs exchange messages to remotely invoke methods on objects and return results. Operation parameters are passed by order, not by name. Thus, a single *inout* parameter can be replaced with separate in and out parameters without impacting the inter-operability of systems. This is because the *in* parameter will take the same place in the invocation message as would the *inout* parameter, and the *out* parameter will take the same place in the results message as would the *inout* parameter. As long as the order of the parameters is not changed, the ORBs will correctly match the parameters in the messages to the parameters on the method invocations. So, the developer of an implementation using the *ManagedObject* interface may modify the IDL by replacing the *attributesGet* signature above with the one below.

```
ManagedObjectValueType attributesGet (
     in NameType name,
     in StringSetType requestedAttributeNames,
     out StringSetType returnedAttributeNames)
raises (ApplicationError);
```

This will prevent the client system's list of requested attribute names from being overwritten and still be interoperable with managed object implementations using the first operation signature, above.

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

Geneva, 2002