International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.818

(05/2012)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

Web services-based management services

Recommendation ITU-T Q.818

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4, 5, 6, R1 AND R2	Q.120–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR THE NGN	Q.3000–Q.3999

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

Recommendation ITU-T Q.818

Web services-based management services

Summary

Recommendation ITU-T Q.818 defines a set of services required to support service-oriented web services-based interfaces and along with Recommendation ITU-T X.782 composes a framework for web services-based network management interfaces. It specifies protocol requirements, how some well-known web services should be used in network management interfaces and defines some network management-specific support services. The WSDL interface definitions for the network management-specific support services are also provided.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T Q.818	2012-05-14	2

Keywords

Distributed processing, extensible markup language (XML), managed objects, network management interfaces, service-oriented, universal description discovery and integration (UDDI), web services (WS), web services description language (WSDL), XML schema.

Table of Contents

			Page
1	Scope		1
2	References		1
3	Definitions		2
	3.1	Terms defined elsewhere	2
	3.2	Terms defined in this Recommendation	2
4	Abbreviations and acronyms		2
5	Conventions		3
6	Web services-based network management services goal and requirements		3
	6.1	Goals	3
	6.2	Information modelling dependencies	4
	6.3	Scoping	5
	6.4	Notifications	5
7	Framework overview and protocol requirements		6
	7.1	Framework overview	6
	7.2	Protocol, languages and services requirements	7
8	Usage of well-known web services		7
	8.1	Web services base notification	7
	8.2	UDDI service registration	16
9	Framework support services		21
	9.1	Heartbeat service	21
	9.2	Multiple object operation service	23
	9.3	Containment service	29
10	Compliance and conformance		31
	10.1	System conformance points	31
	10.2	Basic conformance profile	31
Annex A – WSDL definition of framework support services			33
	A.1	ITU notification service WSDL and XML schema definition	33
	A.2	ITU heartbeat service WSDL and XML schema definition	44
	A.3	ITU MOO service WSDL and XML schema definition	46
	A.4	ITU containment service WSDL and XML schema definition	51
Bibliography			55

Recommendation ITU-T Q.818

Web services-based management services

1 Scope

The network management architecture defined in [ITU-T M.3010] introduces the use of multiple management protocols. So far, the GDMO/CMIP and CORBA GIOP/IIOP are possible choices at the application layer. Based on the management interface specification methodology defined in [ITU-T M.3020], more technology-based paradigms can be introduced into network management interfaces, and web services/XML is now an additional paradigm for network management.

This Recommendation, together with [ITU-T X.782], sets out to define a framework for defining how interfaces supported by management systems and network elements should be modelled using web services/XML schema. It is the scope of this Recommendation to provide the following guidelines or instructions:

- protocol requirements for using web services in network management;
- how a web services-based notification service is used in network management interfaces;
- how a new service is registered and accessed using UDDI;
- how to monitor the availability of a web services notification forwarding mechanism;
- how to access multiple managed objects in one operation;
- compliance and conformance requirements.

2 References

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

[ITU-T X.701]	Recommendation ITU-T X.701 (1997), *Information technology – Open Systems Interconnection – Systems management overview*.
[ITU-T X.703]	Recommendation ITU-T X.703 (1997), *Information technology – Open Distributed Management Architecture*.
[ITU-T X.782]	Recommendation ITU-T X.782 (2012), *Guidelines for defining web services for managed objects and management interfaces*.
[ITU-T M.3010]	Recommendation ITU-T M.3010 (2000), *Principles for a telecommunications management network*.
[ITU-T M.3020]	Recommendation ITU-T M.3020 (2010), *Management interface specification methodology*.
[ITU-T M.3702]	Recommendation ITU-T M.3702 (2010), *Common management services – Notification management – Protocol neutral requirements and analysis*.
[OASIS UDDI]	OASIS Specification (2004), *Universal Description, Discovery and Integration (UDDI) v3.0.2*.
[OASIS WSN]	OASIS Specification (2006), *Web Services Base Notification v1.3*.

[W3C SOAP]	W3C Recommendation (2007), *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*.
[W3C XML]	W3C Recommendation (2000), *Extensible Markup Language (XML) 1.0 Second Edition*.
[W3C XS-P1]	W3C Recommendation (2004), *XML Schema Part 1: Structures Second Edition*.
[W3C XS-P2]	W3C Recommendation (2004), *XML Schema Part 2: Datatypes Second Edition*.
[W3C WSDL]	W3C Recommendation (2001), *Web Services Description Language (WSDL) 1.1*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 agent [ITU-T M.3020]

3.1.2 managed object class [ITU-T X.701]

3.1.3 manager [ITU-T M.3020]

3.1.4 notification [ITU-T X.703]

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture
GDMO	Guidelines for the Definition of Managed Objects
GIOP	General Inter-ORB Protocol
IIOP	Internet Inter-ORB Protocol
MO	Managed Object
MOC	Managed Object Class
MOO	Multiple Object Operation
SOAP	Simple Object Access Protocol
TMN	Telecommunications Management Network
UDDI	Universal Description Discovery and Integration
WS	Web Services
WSDL	Web Services Description Language

WSN	Web Services Notification
XML	extensible Markup Language
XSD	XML Schema Definition

5 Conventions

A few conventions are followed in this Recommendation to make the reader aware of the purpose of the text. While most of the Recommendation is normative, paragraphs succinctly stating mandatory requirements to be met by a management system (managing and/or managed) are preceded by a boldface "R" enclosed in parentheses, followed by a short name indicating the subject of the requirement, and a number. For example:

(R) EXAMPLE-1: An example mandatory requirement.

Requirements that may be optionally implemented by a management system are preceded by an "O" instead of an "R" For example:

(O) OPTION-1: An example optional requirement.

The requirement statements are used to create compliance and conformance profiles.

Many examples of WSDL and XML schema are included in this Recommendation, and WSDL specifying management specific services and supporting data types are included in a normative annex. The WSDL and XML schema are written in a 10 point courier typeface:

```
<!-- Example WSDL -->
<wsdl:message name="exampleMessage">
        <wsdl:part name="exampleRequest" type="exampleRequestType"/>
</wsdl:message>
```

6 Web services-based network management services goal and requirements

This clause describes the key goals of the services framework and the requirements that help the web services-based network management services support these goals. Clause 6.1 introduces the goals of the web services framework. Clause 6.2 then provides terminology and requirements. The requirements in this clause are requirements that the framework must satisfy and which are based on telecommunications management needs. Clauses 7, 8, 9 then describe a framework that meets these needs and defines how to achieve the requirements of this clause by using web services in a certain way.

6.1 Goals

This Recommendation sets out to define a framework for defining how interfaces supported by management systems and network elements should be modelled. Some key goals of the framework are identified here:

- application interoperability
- common usage of well-known web services.

This clause elaborates on these two goals.

6.1.1 Application interoperability

A key goal of the network management architecture and in particular the information architecture, is to promote a standard framework for providing interoperability and information exchange between systems from a diverse set of network management system suppliers. Interoperability between systems involves many aspects of development. At its lowest layer, a common communication mechanism must be in place to support a common syntax, the establishment of connectivity and the exchange of operation requests/replies between systems. This aspect of interoperability is inherently supported by the web services specifications.

For network management, there is the need to provide application interoperability. That is, management systems from diverse suppliers will be utilized within a single administration's TMN to support different functions necessary to support the management of its networks. To simplify integration of these various suppliers' systems, they must agree on the semantics of the information being exchanged. This is accomplished with the specification of an information model. Guidelines for the definition of web services-based information models are specified in [ITU-T X.782], but the services defined here should support those guidelines.

6.1.2 Common usage of well-known web services

A second aspect of this framework is the definition of common usage and profiling of the distributed processing environment of choice. This aspect of the framework should indicate the reasonable expectations network management system suppliers may have for one another. Rather than redefining the interface capabilities needed to support common network management functions such as notification filtering with each information model, the modelling guidelines in [ITU-T X.782] rely upon a set of support services. These support services enable the information models to be simpler, and also enhance interoperability.

In defining these services, special effort will be taken to make use of some of the well-known web services. Specifically, this Recommendation will address the use of the web services notification as defined in [OASIS WSN] and web services UDDI service registration as defined in [OASIS UDDI] which will impact system interoperability (i.e., issues involving the use of web services within a single system are outside the scope of this Recommendation). Where network management needs cannot be met by the above mentioned well known web services, additional services will be defined.

6.2 Information modelling dependencies

As described in the previous clause, the explicit modelling of resources that are manageable across an interface is central to application interoperability. The guidelines for defining web services-based managed objects detailed in [ITU-T X.782] describe the rules for modelling manageable resources. They also embody several decisions that must be supported by the network management web services-based services framework. This clause summarizes those points.

6.2.1 Access granularity

Web services interface granularity refers to the relationship between the resources that are modelled on an interface and the means by which they are accessed using web services. [ITU-T X.782] uses a service-grain modelling approach which means each modelled resource is only accessible through a specific web service. The objects that represent manageable resources are called managed objects.

6.2.2 Representation of containment and naming

Containment is a logical representation of how modelled resources contain other modelled resources. Containment has traditionally been a very important relationship in network management applications because it is a convenient means of identifying the large number of resources that typically must be managed. [ITU-T X.782] guidelines require that a unique name be assigned to each managed object, based in part on the name of the object that contains it. The web services-based management services must provide the means to store these names (and hence the containment relationships they represent).

6.2.3 Object creation and deletion

The web services platforms do not provide clients with the means to create or delete objects on remote systems. Instead, these functionalities are provided by remote systems for clients to create or delete objects on the remote system. [ITU-T X.782] provides a common object accessing service, so object creation and deletion will be model-independent. When deleting an object that contains other objects, the containment relationship shall be kept, so that there will not be any orphan objects existing in the remote systems. The containment information should be stored in the remote system and be maintained when creation and deletion happen.

6.3 Scoping

The ability to perform complex queries (i.e., GET operations), updates (i.e., SET operations) and delete operations on a group of entities with a single operation request is a valuable component of TMN. Management systems may have to manage up to 10^7 instances of managed objects. Due to the size of the management information base, a managing system cannot efficiently perform ad hoc queries on individual instances of managed objects (i.e., entities). Rather, the managing system expects a level of intelligence to be supported by the managed system.

The intelligence in the managed system allows the managing system to select a group of managed entities on which an operation will be performed. Managed entity selection involves two phases: scoping. This managed entity selection process is supported by a service defined later in this Recommendation. This service allows a managing system to select a scope of objects to act on (scope is defined through containment relationships; see clause 6.2.2). Once the scope of entities is determined, the operation (specified by the scope request) is performed only on those entities.

The use of scoping in this framework supports:

– Scoped get: Returns the values (for a list of attributes) from each of the entities that meet the scope.
– Scoped update: Replaces an attribute value or adds/removes values to/from set-valued attributes, in the group of entities meeting the scope, to the values specified in the scoped request. May be used to update one or multiple attributes in a single object or multiple objects.
– Scoped deletion: Deletes all entities that meet the scope.

Scoping entails the identification of the entities to which a filter is to be applied. Scoping is applied based on the containment hierarchy as defined in clause 6.2.2. The scope is applied from a base managed entity down to a particular depth in the containment tree.

The base entity for the scope is defined as the root of the containment tree from which the search is to commence. A scoped request must specify the base managed entity of the scope. The depth of the scoping level can then be specified in one of four manners within the scoped request:

1) the base entity
2) the n-th level subordinates of the base entity
3) the base entity and all of its subordinates down to and including the n-th level
4) the base entity and all of its subordinates (i.e., the whole subtree).

6.4 Notifications

The framework needs to support the ability to:

• deliver notifications
• subscribe for notification types
• forward notifications to multiple destinations

- filter notifications
- uniquely identify the resource that emits the notification.

7 Framework overview and protocol requirements

7.1 Framework overview

This framework for web services-based management interfaces is a collection of capabilities. One part of the framework is a set of well-known web services. This framework defines their role in network management interfaces and defines conventions for their use. The framework also defines support services that have not been standardized in industry yet, but are expected to be standards on network management interfaces conforming to this framework. WSDL interfaces for these services are defined in Annex A.

To support the software objects representing manageable resources, the framework requires that they implement some common basic capabilities. Therefore, one base class is defined in [ITU-T X.782] for use in modelling network management resources. Managed object classes in information models must inherit and implement a basic set of capabilities from the base class in order to operate within this framework. Finally, some rules and conventions are defined for information modellers developing models for use with this framework. These consist of modelling guidelines and XML style idioms. All of these are depicted graphically in Figure 1.

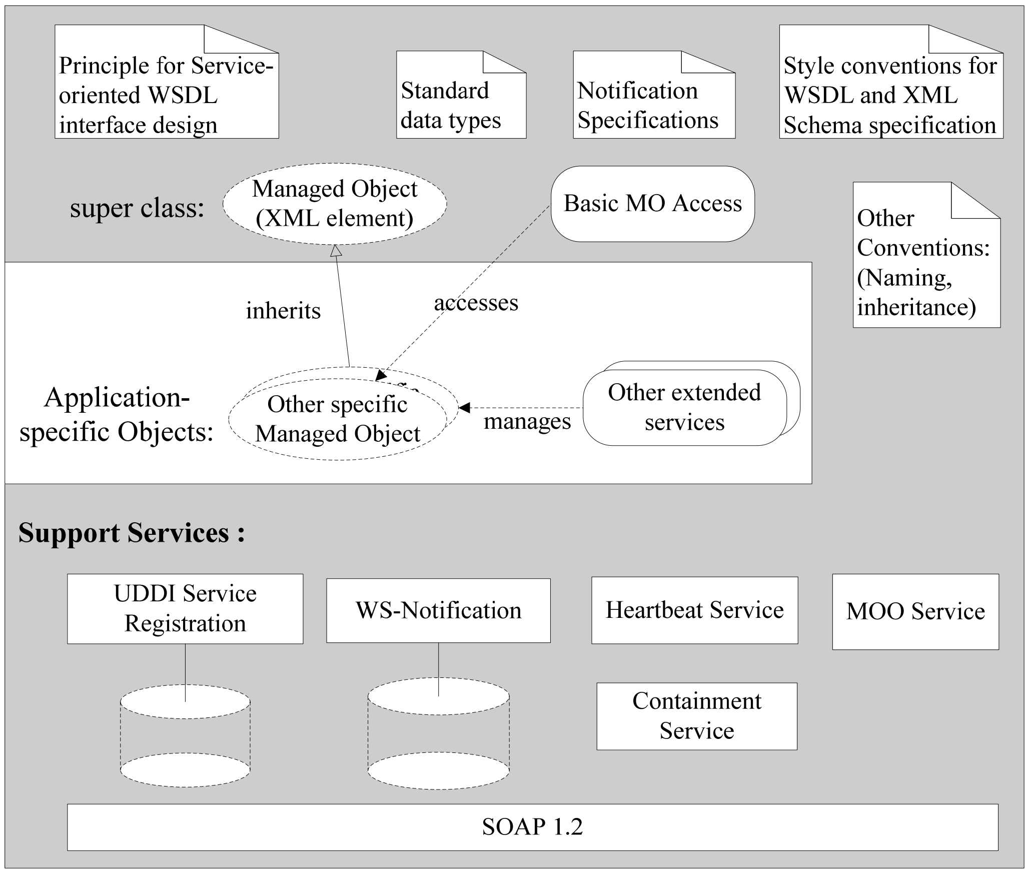

Principle for Service-oriented WSDL interface design
Standard data types
Notification Specifications
Style conventions for WSDL and XML Schema specification
super class:
Managed Object (XML element)
Basic MO Access
Other Conventions: (Naming, inheritance)
inherits
accesses
Application-specific Objects:
Other specific Managed Object
manages
Other extended services
Support Services :
UDDI Service Registration
WS-Notification
Heartbeat Service
MOO Service
Containment Service
SOAP 1.2

Figure 1 – Overview of a framework

Figure 1 shows the framework in grey. In the middle are the application-specific objects that are supported by the framework. Along the bottom is a box representing the communication protocol: SOAP. Above that are a number of boxes with names in them representing the services that compose the framework (some also have icons depicting the databases they would have to maintain to perform their functions). Along the top of the figure are icons representing the super class managed object, and the basic MO accessing service. Each managed object supported by this framework must ultimately inherit from the super class. Also shown on Figure 1 are icons of pages with up-turned corners representing standard object modelling conventions.

The framework services, represented as boxes with square corners, are defined in this Recommendation. The super class, notifications, and object modelling conventions are defined in [ITU-T X.782].

7.2 Protocol, languages and services requirements

This clause defines the versions of the languages, protocol, and services that are required to support this framework. Web services basic technology and protocol specifications are defined by W3C, and two well-known services used in this framework are defined by OASIS. Table 1 shows which version (including subsequent versions released up until the date of this Recommendation) of the applicable specifications must be supported to comply with this framework.

Table 1 – Web services language, protocol and services versions

Service	Version
WSDL Recommendation (W3C)	1.1 (March, 2001)
XML schema Recommendation (W3C)	1.1 (October, 2004)
Web services base notification ([OASIS WSN)	1.3 (October, 2006)
UDDI specification ([OASIS UDDI])	3.0.2 (October, 2004)
SOAP (W3C)	1.2 (April 2007)

8 Usage of well-known web services

8.1 Web services base notification

This framework uses the OASIS web services base notification specification for transferring notifications needed in network management interfaces.

8.1.1 Web services base notification overview

Web services base notification (WSN) is a specification that define a standard web services approach to notification using a topic-based publish/subscribe pattern. The contents of WSN include standard message exchanges to be implemented by notification providers, operational requirements expected of service providers and requestors that participate in notifications, and an XML model that describes topics.

The web services base notification (WSN) supports the asynchronous exchange of event messages between clients using a subscribe-and-publish paradigm, as described in [OASIS WSN]. It also supports the filtering of notifications which are also needed in network management interfaces.

8.1.2 Mapping of operations from Recommendation ITU-T M.3702 and to WSN

The mechanisms and WSDL interfaces of the [OASIS WSN] specification will be used in this framework, but it is used as an internal functionality, which is not disposed to notification consumers directly. [ITU-T M.3702] provides protocol neutral requirements and analysis for generic notification management services, which define the generic functions that are to be implemented by a protocol-specific management paradigm. In this framework, the generic notification management model is mapped to WSDL/XML definitions and some of the WSDL operation are then mapped to [OASIS WSN] operations.

The following table indicates the mapping from [ITU-T M.3702] to the web services notification.

Table 2 – Mappings from [ITU-T M.3702] to this Recommendation and [OASIS WSN]

No.	[ITU-T M.3702]		ITU-T Q.818		[OASIS WSN]	
	Operation name		Operation name		Operation	
	Request/ response	Parameters	Request/ response	Parameters	Request/ response	Parameters
1	subscribeNotification		subscribeNotification		Subscribe	
	Input parameter	– managerId – notificationTypes – filteringCriteria – destination	Request	– managerId: type="nts:IdType" – notificationTypes: type="nts:Notification TypeListType" – filteringCriteria type="nts:FilterType" – destination: type= "wsa:EndpointReference Type"	Request	– consumerReference, – filter, – subscriptionPolicy, – initialTerminationTime
	Output parameter	– subscriptionId – status	Response	– subscriptionId: type="nts:IdType" – status: type="nts:StatusType"	Response	subscriptionReference
2	unsubscribeNotification		unsubscribeNotification		Unsubscribe	
	Input parameter	– managerId – subscriptionId	Request	– managerId type="nts:IdType" – subscriptionId type="nts:IdType"	Request	– subscriptionReference
	Output parameter	– status	Response	– status: type="nts:StatusType"	Response	NULL
3	suspendSubscription		suspendSubscription		PauseSubscription	
	Input parameter	– managerId – subscriptionId	Request	– managerId: type="nts:IdType" – subscriptionId: type="nts:IdType"	Request	– subscriptionReference
	Output parameter	– status	Response	– status type="nts:StatusType"	Response	NULL
4	resumeSubscription		resumeSubscriptions		ResumeSubscription	
	Input parameter	– managerId – subscriptionId	Request	– managerId: type="nts:IdType" – subscriptionId: type="nts:IdType"	Request	– subscriptionReference
	Output parameter	– status	Response	– status: type="nts:StatusType"	Response	NULL
5	getNotificationTypes		getNotificationTypes		–-	
	Input parameter	– notificationIRPId	Request	– notificationIRPId: type="x782:NameType"		
	Output parameter	– status – notificationType List	Response	– notificationTypeList: type="nts:Notification TypeListType" – status type="nts:StatusType"		

Table 2 – Mappings from [ITU-T M.3702] to this Recommendation and [OASIS WSN]

No.	[ITU-T M.3702]		ITU-T Q.818		[OASIS WSN]	
	Operation name		Operation name		Operation	
	Request/ response	Parameters	Request/ response	Parameters	Request/ response	Parameters
6	querySubscription		querySubscription		–	
	Input parameter	– subscriptionId	Request	– subscriptionId: type="nts:IdType"		
	Output parameter	– subscribed NotificationType – subscriptionStatus – destination – filteringCriteria – status	Response	– notificationTypes: type="nts:Notification TypeListType" – subscriptionStatus: type="nts:Subscription StatusType" – destination: type="wsa: EndpointReference Type" – filteringCriteria: type= "nts:FilterType" – status type="nts:StatusType"		
7	modifySubscription		modifySubscription		--	
	Input parameter	– subscriptionId – filteringCriteria – destination – notificationTypes	Request	– subscriptionId: type="nts:IdType" – filteringCriteria: type="nts:FilterType" – destination: type= "wsa: EndpointReference Type" – notificationTypes: type="nts:Notification TypeListType"		
	Output parameter	– status	Response	– status type="nts:StatusType"		
8	listAllSubscriptionIds		listAllSubscriptionIds		--	
	Input parameter	– managerId	Request	– managerId: type="nts:IdType"		
	Output parameter	– subscriptionIdSet – status	Response	– subscriptionIdSet: type="nts:IdSetType" – status type="nts:StatusType"		
9	report notification:		– (Use WSN notify directly)		Notify:	
					notification Message	– SubscriptionReference – Topic – ProducerURL – Message
					Response	NULL

Table 3 gives additional descriptions for the mapping from the operations defined in this Recommendation to the ones defined in WS-Notification.

Table 3 – Additional descriptions for the mappings from the operations in [ITU-T M.3702]

Operation name	Request/ response	ITU-T Q.818	[OASIS WSN]
subscribe	Request	*managerId*	There is no such managerId parameter in WS-Notification and this parameter will only be processed and stored by an agent, not by WS-Notification.
		filterCriteria	*Filter*
		destination	*ConsumerReference*
		NotificationTypes	There is no direct mapping in WSN corresponding to this NotificationTypes, but the filter parameter in WSN can support the function of filtering supported notification types.
		–	*subscriptionPolicy* This parameter is optional in WSN, which can be raw or any value in Notify message. Notify message can provide more information in a message according to the WSN definition. In the mapping of such an operation, the value of subscriptionPolicy should always use the Notify message in this Recommendation.
		–	*initialTerminationTime* initialTerminationTime indicates the termination time of a subscription. When the value of attribute xsi:nil is true, there is no limit for the subscription. In this Recommendation, the default value for this parameter in WSN will be xsi:nil= true.
	Response	subscriptionId	subscriptionReference
unsubscribe	Request	managerId	There is no such concept as managerId in WSN which will be processed and stored in the agent itself. The type is a string at this moment.
		subscriptionId	subscriptionReference
suspendSubscription	Request	managerId	There is no such concept as managerId in WSN which will be processed and stored in the agent itself. The type is a string at this moment.
		subscriptionId	subscriptionReference
resumeSubscription	Request	managerId	There is no such concept as managerId in WSN which will be processed and stored in the agent itself. The type is a string at this moment.
		subscriptionId	subscriptionReference

Table 3 – Additional descriptions for the mappings from the operations in [ITU-T M.3702]

Operation name	Request/ response	ITU-T Q.818	[OASIS WSN]
getNotificationTypes	There are no corresponding operations defined in WSN, and they should be implemented by an agent itself.		
querySubscription			
modifySubscription			
listAllSubscriptionIds			
notify	In this Recommendation, the notify operation from WSN is used directly in an agent.		

A detailed interface definition can be found in clause A.1.

8.1.3 Notification format definitions

8.1.3.1 Notification format to be used in this Recommendation

The following table provides the notify operations defined in [OASIS WSN] which will be used in this Recommendation when sending notifications.

Table 4 – Notify operations from [OASIS WSN]

```
<wsnt:Notify>
  <wsnt:NotificationMessage>
    <wsnt:SubscriptionReference>
      wsa:EndpointReferenceType
    </wsnt:SubscriptionReference> ?
    <wsnt:Topic Dialect="xsd:anyURI">
      {any} ?
    </wsnt:Topic>?
    <wsnt:ProducerReference>
      wsa:EndpointReferenceType
    </wsnt:ProducerReference> ?
    <wsnt:Message>
      {any}
    </wsnt:Message>
  </wsnt:NotificationMessage> +
  {any} *
</wsnt:Notify>
```

In this Recommendation not all the above components will be used. More detailed descriptions of the above components which can or cannot be used in this Recommendation are shown below:

– The WS-Addressing [action] Message Addressing Property MUST contain the URI http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify.

The components of the notify message are described as follows:

```
/wsnt:Notify
```

contains a collection of one or more NotificationMessages:

```
/wsnt:Notify/wsnt:NotificationMessage
```

This contains:

— a notification payload:

    /wsnt:Notify/wsnt:NotificationMessage/wsnt:SubscriptionReference

  This is an EndpointReference to the subscription that is associated with the notify message. This component is optional.

    /wsnt:Notify/wsnt:NotificationMessage/wsnt: ProducerReference

  This is an EndpointReference to the NotificationProducer that produced the notification artefact. This component is optional:

    /wsnt:Notify/wsnt:NotificationMessage/wsnt:Topic

  This is a TopicExpression describing exactly one topic, which MUST be the topic that is associated with the notification. This element describes the topic that matched to a subscription, causing the NotificationProducer to send the notify message to the NotificationConsumer. This component is also optional.

    /wsnt:Notify/wsnt:NotificationMessage/wsnt:Topic/@Dialect

  This is the dialect used in the TopicExpression. This MUST be the same dialect used by the subscriber when it created the subscription that yielded this notify message.

    /wsnt:Notify/wsnt:NotificationMessage/wsnt:Message

  This is a copy of the actual notification payload. This component is of any type, and will be replaced by predefined notification types, which are defined in clause A.5 and will be explained later.

  and any metadata components:

    /wsnt:Notify/{any}

The raw data shall not be used in this Recommendation.

The most simplified notification message format that can be used in this Recommendation is shown in Table 5.

Table 5 – Most simplified notification format in this Recommendation

```
<wsnt:Notify>
  <wsnt:NotificationMessage>
    <wsnt:Message>
      {any}
    </wsnt:Message>
  </wsnt:NotificationMessage> +
</wsnt:Notify>
```

8.1.3.2 Common notification header definition

This clause provides the detailed common notification header described in [ITU-T M.3702].

The following table provides the definition of the parameters of the common notification header.

Table 6 – Common notification header definition

Parameter name	Qualifiers	XSD data type	Descriptions
objectClass	M	`xsd:string`	It is the class name of the MO which sends out this notification. See clause 7.3.5 of [ITU-T M.3702]
objectInstance	M	`x782:NameType`	It is the DN of the MO which sends out this notification. See clause 7.3.5 of [ITU-T M.3702]
notificationId	M	`782:NotificationIdType`	It is the Identifier of the notification for the purpose of notification correlation. See clause 7.3.5 of [ITU-T M.3702]
eventTime	M	`xsd:dateTime`	It is the time when this event happened. See clause 7.3.5 of [ITU-T M.3702]
systemDN	M	`x782:NameType`	It is the DN of the system which sends out this notification. See clause 7.3.5 of [ITU-T M.3702]
notificationType	M	`nts:NotificationTypeType`	It indicates the type of notification. See clause 7.3.5 of [ITU-T M.3702]. The possible common notification types defined in this Recommendation can be the following: – attributeValueChange – objectCreation – objectDeletion – stateChange – communicationsAlarm – environmentalAlarm – equipmentAlarm – processingErrorAlarm – qualityOfServiceAlarm – integrityViolation – operationalViolation – physicalViolation – securityViolation – timeDomainViolation – relationshipChange – heartbeat This list can be extended to include new notification types as and when they are defined.

8.1.3.3 Common notification contents definition

1) Notification contents for the objectCreation and objectDeletion notifications.

Table 7 – Notification contents for objectCreation and objectDeletion

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`x782:CorrelatedNotificationType`	List of correlated notifications.
additionalText	O	`xsd:string`	Text message.
additionalInfo	O	`x782:AdditionalInformationSetType`	Additional information not in text format.
sourceIndicator	O	`x782:SourceIndicatorType`	Cause of event.
attributeList	O	`x782:AttributeNameAndValueType`	Attribute values.

2) Notification contents for the attributeValueChange notification.

Table 8 – Notification contents for attributeValueChange

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`x782:CorrelatedNotificationType`	List of correlated notifications.
additionalText	O	`xsd:string`	Text message.
additionalInfo	O	`x782:AdditionalInformationSetType`	Additional information not in text format.
sourceIndicator	O	`x782:SourceIndicatorType`	Cause of event.
attributeChanges	M	`x782:AttributeValueChangeSetType`	Changed attributes.

3) Notification contents for the stateChange notification.

Table 9 – Notification contents for stateChange

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`x782:CorrelatedNotificationType`	List of correlated notifications.
additionalText	O	`xsd:string`	Text message.
additionalInfo	O	`x782:AdditionalInformationSetType`	Additional information not in text format.
sourceIndicator	O	`x782:SourceIndicatorType`	Cause of event.
stateChanges	M	`x782:AttributeValueChangeSetType`	Changed states.

4) Notification contents for the communicationAlarm, environmentalAlarm, equipmentAlarm, processingErrorAlarm, qualityOfServiceAlarm notifications.

Table 10 – Notification contents for alarms

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`x782:CorrelatedNotificationSetType`	List of correlated notifications.
additionalText	O	`xsd:string`	Text message.
additionalInfo	O	`x782:AdditionalInformationSetType`	Additional information not in text format.
probableCause	M	`x782:ProbableCauseType`	The probable cause of the alarm.
specificProblems	O	`x782:SpecificProblemSetType`	Non standarized problems.
perceivedSeverity	M	`x782:PerceivedSeverityType`	It indicates the perceived severity of the alarm. See type definition for details.
backedUpStatus	O	`xsd:boolean`	"True" if backed up.
backUpObject	O	`X782:NameType`	It indicates the DN of the backup object if the backUpStatus is "false".
trendIndication	O	`x782:TrendIndicationType`	See type for details.
thresholdInfo	O	`x782:ThresholdInfoType`	See type for details.
stateChangeDefinition	O	`x782:AttributeChangeSetType`	It indicates the state changes in this alarm.
monitoredAttributes	O	`x782:AttributeNameAndValueType`	See type for details.
proposedRepairActions	O	`x782:ProposedRepairActionSetType`	It indicates the proposed actions to repair this fault.
alarmEffectOnService	O	`xsd:boolean`	True if alarm is service effecting.
alarmingResumed	O	`xsd:boolean`	True if alarming was just resumed, possibly resulting in delayed reporting.
suspectObjectList	O	`x782:SuspectObjectSetType`	Objects possibly involved in failure.

5) Notification contents for the integrityViolation, operationalViolation, physicalViolation, securityViolation, timeDomainViolation notifications.

Table 11 – Notification contents for violations

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`x782:CorrelatedNotificationSetType`	List of correlated notifications.
additionalText	O	`xsd:string`	Text message.
additionalInfo	O	`x782:AdditionalInformationSetType`	Additional information not in text format.
securityAlarmCause	M	`x782:SecurityAlarmCauseType`	The cause of the security alarm.
securityAlarmSeverity	M	`x782:PerceivedSeverityType`	Clears allowed? ITU-T X.721 appears to restrict the cleared value on this alarm, but clears should be allowed.
securityAlarmDetector	M	`x782:SecurityAlarmDetectorType`	See type for details.
serviceUser	M	`x782:ServiceUserType`	The user of the service which is violated.
serviceProvider	M	`x782:ServiceProviderType`	The service provider of the service which is violated.

6) Notification contents for the relationshipChange notification.

Table 12 – Notification contents for relationshipChange

Notification parameter	Qualifiers	Data type	Descriptions
correlatedNotifications	O	`x782:CorrelatedNotificationType`	List of correlated notifications.
additionalText	O	`xsd:string`	Text message.
additionalInfo	O	`x782:AdditionalInformationSetType`	Additional information not in text format.
sourceIndicator	O	`x782:SourceIndicatorType`	Cause of event.
relationshipChanges	M	`x782:AttributeChangeSetType`	Changed relationship attributes.

The detailed data type definitions for notification contents can be found in clause A.1.

8.2 UDDI service registration

The OASIS UDDI registry is a web services-based directory service.

The focus of universal description discovery and integration (UDDI) is the definition of a set of services supporting the description and discovery of the following:

– businesses, organizations, and other web services providers

– the web services they make available and

– the technical interfaces which may be used to access those services.

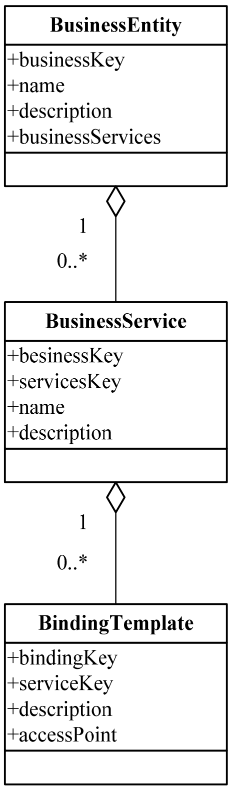
UDDI provides an interoperable, foundational infrastructure for a web services-based software environment for both publicly available services and services that are exposed only internally within an organization.

UDDI registry allows a client to find web services registered within it. This framework uses the well-known service UDDI for service registration and it provide the means for a managing system to discover the services provided by a managed system.

As UDDI is not designed specifically for network management there are many properties which are not needed when used in network management.

The following clauses provide a simplified UDDI information model and listed are some of the related operations which are expected to be used in a network management domain.

8.2.1 UDDI information model used in network management



The simplified information model of UDDI

A UDDI information model which is used in network management is composed of instances of the following entity types:

– **businessEntity**: Describes a system entity which provides some management services. It is usually a managed system (e.g., EMS or agent) provided by a vendor and sometimes it can also be a third party system which provides standalone services to be shared by multiple managed systems (e.g., WSN services). Within a network management domain it is usually a managing system that uses services provided by multiple managed systems. A given instance of the businessEntity structure is uniquely identified by its attribute businessKey in the UDDI registry. It is assigned by the UDDI registry at the time of registration. The name attribute gives a nickname to the managed system. Both a managed system and a third party can be registered as an instance of businessEntity. Simple textual information about the managed system could be provided in the optional attribute description. The list of services provided by this managed system is described in the attribute businessServices. Other properties of the businessEntity in the original UDDI model are not used in this framework.

– **businessService**: Represents a logical web service which provides a management functionality (e.g., configuration management, performance management) by the managed system. businessService is the unit for collecting management operations which are closely related (e.g., retrieve or modify attribute values of managed objects). A given businessService instance is uniquely identified by its attribute serviceKey in the UDDI registry. The businessKey attribute uniquely identifies the containing businessEntity which is the provider of this service. The name attribute gives the nickname of a service. Simple textual information about the management service could be provided in the optional attribute description, indicating the functionalities, usage and other descriptive information of the web services. Other properties of businessService in the original UDDI model are not used in this framework.

– **bindingTemplate**: Describes the technical information necessary for a managing system to use a particular web service. Usually a businessService in a network management domain only has one accessPoint, thus there should be a one-to-one relationship between businessService and its bindingTemplate, except when load balance or backup is turned on, where more than one service implementations are provided. In this case, the information of each accessPoint is contained in a bindingTemplate. A given bindingTemplate entity is uniquely identified by its attribute bindingKey. It is not allowed for more than one implementation to perform different functionalities for a network management service as this confuses the managing system when it attempts to select the appropriate service accessPoint. The serviceKey attribute uniquely identifies the containing businessService. Simple textual information about the bindingTemplate could be given in the optional attribute description. The accessPoint attribute is a string used to convey the network address that suitable for invoking the web services.

8.2.2 Register a new service and access it using UDDI

[OASIS UDDI] provide several APIs for both web services providers and web services clients to use, either to register a service or find a service. The following subclauses describe some related APIs that are expected to be used in a network domain.

8.2.2.1 Using publication APIs to publish a new web service in UDDI registry

There are three operation APIs for a managed system (or a third party) to register its provided web services into the UDDI registry which are listed in the following table.

Table 13 – Operation APIs in the UDDI to register a web service

Operation APIs	Parameter direction	Parameter name	Descriptions
get_authToken	arguments	userId	This required attribute argument is the user identifier that an individual authorized user was assigned by a UDDI node. Nodes should provide a means for individuals to obtain a userID and password credentials that will be valid at the given node.
		cred	This required attribute argument is the password or credential that is associated with the user.
	return	authToken	Upon successful completion this API call returns an authToken structure that contains a valid authInfo element that can be used in subsequent calls to API calls that require an authInfo value.
save_business:	arguments	authInfo	This argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call. Registries that serve multiple publishers and registries that restrict who can publish in them typically require authInfo for this call.
		businessEntity	Required repeating element containing one or more businessEntity structures.
	return	businessKey	This is assigned as a result of processing the save_business API.
save_service	arguments	authInfo	This argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call. Registries that serve multiple publishers and registries that restrict who can publish in them typically require authInfo for this call.
		businessService	Required repeating element containing one or more complete businessService elements.
	return	serviceKey	The serviceKey and the bindingKey values that were assigned as a result of processing the save_service API are included in the businessService data.
		bindingKey	

8.2.2.2 Using inquiry APIs to find web services in the UDDI registry

Table 14 – Operation APIs in the UDDI to find a web service

Operation APIs	Parameter direction	Parameter name	Descriptions
find_business	arguments	name	This optional collection of string values represents one or more names potentially qualified with xml:lang attributes. Since exactMatch is the default behaviour, the value supplied for the name argument must be an exact match.
		findQualifiers	This collection of findQualifier elements can be used to alter the default behaviour exactMatch of the search functionality.
		maxRows	This optional integer value allows the requesting program to limit the number of results returned. This argument can be used in conjunction with the listHead argument.
	return	businessList	Returns a businessList that matches the criteria specified in the arguments. BusinessList: this structure is a sequence of businesses.
find_service	arguments	name	This optional collection of string values represents one or more names potentially qualified with xml:lang attributes. Since exactMatch is the default behaviour, the value supplied for the name argument must be an exact match.
		findQualifiers	This collection of findQualifier elements can be used to alter the default behaviour exactMatch of the search functionality.
		businessKey	This uddi_key is used to specify a particular businessEntity instance to search for. This argument is used to specify an existing businessEntity within which services should be found.
		maxRows	This optional integer value allows the requesting program to limit the number of results returned. This argument can be used in conjunction with the listHead argument.
	return	serviceList	Returns a serviceList that matches the criteria specified in the arguments. ServiceList : this structure is a sequence of businesses.

Table 14 – Operation APIs in the UDDI to find a web service

Operation APIs	Parameter direction	Parameter name	Descriptions
find_binding	arguments	serviceKey	This optional uddi_key is used to specify a particular instance of a businessService element in the registered data. Only bindings in the specific businessService data identified by the serviceKey are searched.
		findQualifier	This collection of findQualifier elements can be used to alter the default behaviour exactMatch of the search functionality.
		maxRows	This optional integer value allows the requesting program to limit the number of results returned. This argument can be used in conjunction with the listHead argument.
	return	bindingDetail	Returns a bindingDetail that contains zero or more bindingTemplate structures matching the criteria specified in the argument list. Each bindingTemplate structure contains the description and accessPoint.

9 Framework support services

This clause defines common support services included in the framework which are not well known web services. This clause defines services that will be broadly used by network management applications. These services also provide functionality required to enable the reuse of existing information models without significant changes in semantics. The WSDL describing the interfaces to these services can be found in Annex A.

9.1 Heartbeat service

The heartbeat service is used to verify the operation of the notification forwarding mechanism (i.e., WSN in this framework) in a managed system, as well as the communications network between the managed system and managing system.

It periodically sends a small notification to a managing system interested in receiving it and that notification identifies the system that emitted the heartbeat. After configuring this service a managing system can ensure the WSN is functioning. Since these notifications flow through the same software and networks as notifications from other resources, they periodically verify the operation of these resources.

The heartbeat service has two internal attributes, each of which has a pair of accessing methods, one for value get and one for value set, as presented in the following table:

Table 15 – Attributes and accessing operations in a heartbeat service

Internal attributes of heartbeat services	Operations	Message for requests and responses

systemLabel	systemLabelGet	`Request: NULL`
		`Response:`
		`  name="systemLabel" type="hs:SystemLabelType"`

	systemLabelSet	`Request:`
		`  name="systemLabel" type="hs:SystemLabelType"`
		`Response: NULL`

period	periodGet	`Request: NULL`
		`Response:`
		`  name="period" type="hs:HeartbeatPeriodType"`

	periodSet	`Request:`
		`  name="period" type="hs:HeartbeatPeriodType"`
		`Response: NULL`

The following table contains the contents to be included in a heartbeat notification.

Table 16 – Notification contents for heartbeat

Heartbeat notification contents	Data type	Descriptions
systemLabel	`xsd:string`	Identifies the managed system which sent out the heartbeat notification.
period	`xsd:unsignedLong`	Indicates the time period between two heartbeats.
timeStamp	`xsd:dateTime`	Indicates the time stamp when the heartbeat notification is generated.

The attribute systemLabel and period are two attributes of a heartbeat service. systemLabel is a user-supplied identifier. The intended use is to allow a managing system to insert a label to identify the system providing the heartbeat. The value of systemLabel can be modified using the systemLabelSet operation. period is the period between heartbeats. The value of period can be controlled using the periodSet operation. The value submitted to this operation is the period, in seconds, that the heartbeat service waits between emitting notifications.

Each notification includes the value of the systemLabel, the current value for the period, and a timestamp.

(R) HEARTBEAT-1: If the heartbeat service is supported by a managed system it shall support the heartbeat interface described above and defined in the WSDL in clause A.2. The functionality described above shall be supported.

(R) HEARTBEAT-2: Updating of the period shall cause the service to deliver a notification to the WSN with the new period value and then begin a new period. Setting the period to zero shall cause the service to emit one final notification with a period value of zero, then no more (until the period is reset).

(R) HEARTBEAT-3: Until the period is changed, the heartbeat notifications shall be sent to the WSN service once within each period. The time between heartbeat notifications being sent to a WSN service shall never be greater than twice the period.

9.2 Multiple object operation service

The multiple object operation (MOO) service's interface is defined in clause A.3 and in programming terms it is weakly typed. It provides a set of generic capabilities that may be invoked on any kinds of sets of managed objects (of any kind). The operations supported are listed below.

- Scoped get: Returns the values from each of the objects for a list of attributes.
- Scoped update: Used to replace an attribute value or to add or remove values to/from set-valued attributes. May be used to update one or multiple attributes in a single object or multiple objects.
- Scoped delete: Deletes multiple objects.

A basic service only needs to implement the scoped get operation. The other two operations are optional.

9.2.1 Common parameters in MOO service operations

Each of the scoped operations requires four parameters to define the set of objects on which the operation will be performed.

- Base object name: The name of the object at the root of a tree of objects on which the operation will potentially be performed.
- Scope: A complexType identifying the objects contained under the base object on which the operation will potentially be performed. The complexType contains two elements, one element scope indicating four scope cases. Two of the cases include an integer specifying a level of objects contained below the base object, which is presented by the other element level in the complexType. The four cases are:
  - Base object only. If the scope is baseObjectOnly, then only the named target (base) object is included in the scope. In this case, the level is not used.
  - Whole subtree. If the scope is wholeSubtree, the scope is all of the objects contained below the base object, along with the base object.
  - Individual level. If the scope is individualLevel, the positive integer-valued level will also be used. All of the objects contained at a level below the base object equal to this value are in the scope. The objects directly contained by the base object are level one.
  - Base to level. If the scope is baseToLevel, the positive integer-valued level will also be used. The scope will be all of the objects down to the given level, including the base object and the object at the given level.

The XSD signature for the scopeType is the following:

Table 17 – ScopeType definition

ScopeEnumType	`<xsd:simpleType name="ScopeEnumType">`
	`   <xsd:restriction base="xsd:string">`
	`      <xsd:enumeration value="BasicObjectOnly"/>`
	`      <xsd:enumeration value="WholeSubtree"/>`
	`      <xsd:enumeration value="IndividualLevel"/>`
	`      <xsd:enumeration value="BaseToLevel"/>`
	`   </xsd:restriction>`
	`</xsd:simpleType>`

ScopeType	`<xsd:complexType name="ScopeType">`
	`   <xsd:sequence>`
	`      <xsd:element name="scopeInd" type="moos:ScopeEnumType"/>`
	`      <xsd:element name="level" type="xsd:short"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

The object names are the NameType (DN) format as describe in the [ITU-T X.782]. The scope is a complexType with values as described above.

The following clauses give additional details on each of the scoped operations.

9.2.2 Scoped get

The WSDL signature for the scoped get operation on the basic MOO service is:

Table 18 – Request and response for ScopedGet operation

ScopedGet Request	`<xsd:complexType name="ScopedGetRequestType">`
	`   <xsd:sequence>`
	`      <xsd:element name="baseName" type="x782:NameType"/>`
	`      <xsd:element name="scope" type="moos:ScopeType"/>`
	`      <xsd:element name="moClassList"`
	`type="x782:MOClassListType" minOccur="0" maxOccur="1"/>`
	`      <xsd:element name="attributes" type="x782:StringSetType"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

ScopedGet Response	`<xsd:complexType name="ScopedGetResponseType">`
	`   <xsd:sequence>`
	`      <xsd:element name="moInfo" type="moos:GetResultsType"`
	`minOccur="0" maxOccur="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

As described above, the first two parameters baseName and scope in the scopedGet request are used to select a set of objects on which to perform the get operation. The optional parameter moClassList in the scopedGet request can be used to select the objects of a certain MOC(s). For each of the objects, the MOO service will try to return a value for each of the attributes named in the attributes parameter, which is just a list of strings. A submitted null attribute list, however, has the special meaning that all attribute values for the selected objects should be returned. The data types involved in the return value are listed in the following table:

Table 19 – Data types for ScopedGet

X782:Attribute NameAndValue Type	`<xsd:complexType name="AttributeNameAndValueType">`
	`   <xsd:sequence>`
	`      <xsd:element name="attributeName" type="xsd:string"/>`
	`      <xsd:element name="attributeType" type="xsd:string"/>`
	`      <xsd:element name="attributeValue"`
	`type="x782:AttributeValueType"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

X782:Attribute NameAndValueSet Type	`<xsd:complexType name="AttributeNameAndValueSetType">`
	`   <xsd:sequence>`
	`      <xsd:element name="attributeNameAndValue"`
	`type="x782:AttributeNameAndValueType" minOccurs="0"`
	`maxOccurs="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

GetResultsType	`<xsd:complexType name="GetResultsType">`
	`   <xsd:sequence>`
	`      <xsd:element name="name" type="x782:NameType"/>`
	`      <xsd:element name="attributes"`
	`type="x782:AttributeNameAndValueSetType"/>`
	`      <xsd:element name="failedAttributes"`
	`type="x782:StringSetType"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

X782:StringSet Type	`<xsd:complexType name="StringSetType">`
	`   <xsd:sequence>`
	`      <xsd:element name=”value” type=”xsd:string”`
	`minOccurs="0" maxOccurs="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

The first two types *x782:AttributeNameAndValueType* and *x782:AttributeNameAndValue-SetType* form a name-value pair list. The return type is a complexType which contains a sequence of structures, one for each selected managed object. In each structure, there is an MO's name, the list of attribute values from that object, and the names of any attributes that could not be retrieved from that object. If an attribute value of an MO could not be retrieved either because the object did not have a matching attribute or an exception was raised on access, that attribute's name should be put on the failed attribute list for that object.

9.2.3 Scoped update

The WSDL signature for the scoped update operation on the MOO service is:

Table 20 – Request and response for ScopedUpdate operation

ScopedUpdate Request	`<wsdl:message name="scopedUpdateRequest">`
	`   <wsdl:part name="scopedUpdateInput" type="moos:ScopedUpdateRequestType"/>`
	`</wsdl:message>`
	`<xsd:complexType name="ScopedUpdateRequestType">`
	`   <xsd:sequence>`
	`      <xsd:element name="baseName" type="x782:NameType"/>`
	`      <xsd:element name="scope" type="moos:ScopeType"/>`
	`      <xsd:element name="moClassList" type="x782:MOClassListType" minOccur="0" maxOccur="1"/>`
	`      <xsd:element name="modifications" type="moas:AttributeNVMListType"/>`
	`      <xsd:element name="failuresOnly" type="xsd:boolean"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

ScopedUpdate Response	`<wsdl:message name="scopedUpdateResponse">`
	`   <wsdl:part name="ScopedUpdateOutput" type="moos:ScopedUpdateResponseType"/>`
	`</wsdl:message>`
	`<xsd:complexType name="ScopedUpdateResponseType">`
	`   <xsd:sequence>`
	`      <xsd:element name="updateResult" type="moos:UpdateResultsType" minOccurs="0" maxOccurs="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

Again, the first two parameters are used to select the set of objects on which the update is performed. The optional parameter moClassList can be used to select the objects of a certain MOC(s). The modifications is a sequence of structures, each with the name of an attribute, the type of the attribute, a value for that attribute, and an enumerated value indicating if the value should replace the attribute's current value, be added to the attribute's current value, or be removed from it. The ADDValues and REMOVEValues options are valid only if the attribute's type is an XSD complex type of a sequence, and the values of the attribute can be added or removed. The values in the modifications sequence of structures are passed across as XML anyTypes. The XSD signature for the modification related data types are shown as the following:

Table 21 – Modification related data types for ScopedUpdate operation

`moas:Modify OptionType`	`<xsd:simpleType name="ModifyOptionType">`
	`   <xsd:restriction base="xsd:string">`
	`      <xsd:enumeration value="REPLACE"/>`
	`      <xsd:enumeration value="ADDValues"/>`
	`      <xsd:enumeration value="REMOVEValues"/>`
	`      <xsd:enumeration value="SETToefault"/>`
	`   </xsd:restriction>`
	`</xsd:simpleType>`

`moas:Attribute NVMType`	`<xsd:complexType name="AttributeNVMType">`
	`   <xsd:sequence>`
	`      <xsd:element name="attributeName" type="xsd:string"/>`
	`      <xsd:element name="attributeType" type="xsd:string"/>`
	`      <xsd:element name="attributeValue"`
	`type="x782:AttributeValueType"/>`
	`     <xsd:element name="modifyOption"`
	`type="moas:ModificationOpType"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

`moas:Attribute NVMListType`	`<xsd:complexType name="AttributeNVMListType">`
	`   <xsd:sequence>`
	`      <xsd:element name="attributeNVM" type="moas:`
	`AttributeNVMType" minOccurs="1" maxOccurs="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

The *failuresOnly* flag is used to indicate if the client wants the service to return results for all objects meeting the scope, or just those objects for which at least one of the modifications could not be performed even though the scope has been satisfied.

The return value is a sequence of structures, each containing an object's name and a list of any attributes that could not be modified. The service will try to perform all the modifications in the list, in order, continuing to try the rest even if one modification fails. If any operation fails on an attribute, that attribute's name is added to the list of failures. If the failedAttributes data member is empty, the client will know all updates were performed on that object. The new types involved in the return value are shown in the following table:

Table 22 – Return data types for ScopedUpdate operation

`UpdateResults Type`	`<xsd:complexType name="UpdateResultsType">`
	`   <xsd:sequence>`
	`      <xsd:element name="name" type="x782:NameType"`
	`minOccurs="1" maxOccurs="unbounded"/>`
	`      <xsd:element name="failedAttributes"`
	`type="x782:StringSetType" minOccurs="0"`
	`maxOccurs="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

9.2.4 Scope delete

The WSDL signature for the scoped delete operation on the advanced MOO service is:

Table 23 – Request and response for ScopedDelete operation

`ScopedDelete Request`	`<wsdl:message name="scopedDeleteRequest">`
	`   <wsdl:part name="scopedDeleteInput" type="nts:ScopedDeleteRequestType"/>`
	`</wsdl:message>`
	`<xsd:complexType name="ScopedDeleteRequestType">`
	`   <xsd:sequence>`
	`      <xsd:element name="baseName" type="x782:NameType"/>`
	`      <xsd:element name="scope" type="moos:ScopeType"/>`
	`      <xsd:element name="moClassList" type="x782:MOClassListType" minOccur="0" maxOccur="1"/>`
	`      <xsd:element name="failuresOnly" type="xsd:boolean"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

`ScopedDelete Response`	`<wsdl:message name="scopedDeleteResponse">`
	`   <wsdl:part name="scopedDeleteOutput" type="moos:ScopedDeleteResponseType"/>`
	`</wsdl:message>`
	`<xsd:complexType name="ScopedDeleteResponseType">`
	`   <xsd:sequence>`
	`      <xsd:element name="deleteResult" type="moos: DeleteResultsType" minOccurs="0" maxOccurs="unbounded"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

This operation simply attempts to delete each object selected by the scope parameter. The optional parameter moClassList can be used to select the objects of a certain MOC(s).

The failuresOnly flag is used to indicate if the client wants the service to return results for all objects meeting the scope, or just those objects that could not be deleted. Because object deletion notifications are typically sent, clients may often want to choose to receive results for only those objects that could not be deleted.

The return value lists the name of each object along with a flag, the notDeletable flag shall be true if the selected object could not be deleted, either due to its delete policy, or because it raised an exception.

The data types involved in the return value are listed in the following table:

Table 24 – Data types for ScopedDelete operation

`DeleteResults Type`	`<xsd:complexType name="DeleteResultsType">`
	`   <xsd:sequence>`
	`      <xsd:element name="name" type="x782:NameType" minOccurs="1" maxOccurs="unbounded"/>`
	`      <xsd:element name="notDeletable" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>`
	`   </xsd:sequence>`
	`</xsd:complexType>`

Because many objects cannot be deleted if they contain other objects, for scopes based on containment relationships the service must begin deleting the leaf objects that are within scope and work towards the root object. When deleting objects, the MOO service must follow the rules for deleting an object based on the object's delete policy. Because the rules are being applied to each of the objects in the scope, starting from the bottom up, however, the effect will be different than simply trying to delete the object at the root of a subtree. Also, the MOO service is best-effort. Therefore, it is possible for some of the objects in a scoped subtree to be deleted while others are not. These are the rules that must be applied to scoped delete operations:

1) No objects may be orphaned. That is, an object may not be deleted without deleting all of its contained (child) objects.
2) Performing a scoped delete on an entire subtree results in all of the objects in that subtree being deleted unless an object is not deletable, or an object has a subordinate that is not deletable.
3) Performing a scoped delete on part of a subtree requires evaluating each of the objects at the lowest scoped layer. If an object at the lowest layer of the scope may be deleted, it and any subordinates are deleted. If a lowest layer object cannot be deleted, it is not deleted nor are any of its superior objects. Other objects in the scope may be deleted, however, if the delete rules allow it. The service then moves up to the next layer, and so on.

9.2.5 MOO service requirements

This clause summarizes the multiple object operation service requirements.

(R) MOO-1: An implementation of the MOO service shall support the scopedGet operation described above and whose WSDL is defined in clause A.3.

(O) MOO-2: Optionally, an implementation of the MOO service may support the scopedUpdate and ScopdeDelete Operation described above and whose WSDL is defined in clause A.3.

9.3 Containment service

In the network management field a function is needed to be able to report which objects are contained by a superior object, to verify that a superior object exists before a subordinate is created, to make sure two objects with the same name are not created, etc. The framework will be extended to support this function by adding a new service, the containment service.

9.3.1 Containment service description

The main function to be supported by the containment service is to enable a managing system to query a managed system with the name of an object, and receive back the names of the objects contained by that object. In addition, a means of getting names added to and removed from the service will be defined. These are not for use by managing systems but internally by managed objects, factories, and other parts of a managed system. They are provided to promote the development of reusable components, possibly by third parties, and are defined on an interface separate from that used by managing systems.

9.3.2 The containment service definition

The containment service provides three operations to retrieve containment information. The WSDL describing the containment service interface and the corresponding XML schema can be found in clause A.4.

A short description of the semantics of the containment service is illustrated in the following table:

Table 25 – Operations in containment service

Operation	Parameter direction	Parameter name	Descriptions
exists	request	name: x782:NameType	This is the DN of a managed object to be checked whether it exists in the containment service or not.
	response	existsOutput : xsd:boolean	Returns true if the specified MO exists in the containment service, otherwise the result will be false.
getContained	request	base: x782:NameType	It indicates the base MO instance for a specified tree for retrieving containment information.
		scope: ScopeType	It indicates the scope information for retrieving containment information. See clause 9.2.1 for the semantics of the scope.
	response	moList: x782:NameSet Type	Returns a list of MO names that is specified by the base and scope parameters.
getContainedBy Class	request	base : x782:NameType	It indicates the base MO instance for a specified tree for retrieving containment information.
		moClass: xsd:string	It indicates the managed object class for retrieving containment information.
		scope: moos:ScopeType	It indicates the scope information for retrieving containment information. See clause 9.2.1 for the semantics of the scope.
	response	moList: x782:NameSet Type	Returns a list of MO names that is specified by the base and scope parameters and they are all instances of class specified by the moClass parameter.

The 'exists' operation takes a name and returns true if it is registered with the containment service. The other two operations return the names of objects contained by the object named in the base parameter. The scope parameter on both of these operations can be used to specify which part of the tree of objects contained below the base object is to be retrieved. The third operation, getContainedByClass, takes an moClass parameter to instruct the containment service to return the names for objects of a certain MOC.

(R) CONTAINMENT-1: The interface supported by the containment service shall be the containment interface described above and defined in clause A.4.

(R) CONTAINMENT-2: In response to an invocation of the 'exists' operation, the containment service shall return true if the name is currently registered with the service and false otherwise. If some error on the server prevents this determination an appropriate application error exception response shall be returned to the managing system.

(R) CONTAINMENT-3: In response to an invocation of the getContained operation, the containment service shall return a list of the names of the objects contained by the object named in the base parameter. The list of contained objects shall be determined according to the scope parameter. If an empty base name is submitted, the first level of contained names shall be the names of the registered root MO. If some error prevents the list from being returned, an appropriate application error exception response shall be returned, for example, the base name is not registered.

(R) CONTAINMENT-4: The containment service shall respond to the invocation of the *getContainedByClass* operation as described in requirement CONTAINMENT-3, except only those names that match the *moClass* parameter are returned.

The registration of MO names in the containment service is a function that should be implemented by managed systems which is outside the scope of this Recommendation.

10 Compliance and conformance

This clause defines the criteria that must be met by other standards documents claiming compliance to this framework and the functions that must be implemented by systems claiming conformance to this Recommendation.

10.1 System conformance points

This clause summarizes the individual functions described earlier in this Recommendation. These conformance points are then combined in profiles that must be supported by systems claiming conformance to this Recommendation.

1) An implementation claiming conformance to the notifications requirements must:
   - support the OASIS web services base Notification [OASIS WSN] version specified in clause 7.2;
   - support the notification interface described in clause 8.1 and defined in the WSDL and XML schema in clause A.1;
   - support the notification format described in clause 8.1.3 and defined in the XML schema in clause A.5.
2) An implementation claiming conformance to the services registry requirements must:
   - support the OASIS UDDI service registration [OASIS UDDI] version specified in clause 7.2;
   - support the usage of UDDI service registration as specified in clause 8.2.
3) An implementation claiming conformance to the heartbeat service must:
   - support the heartbeat service interface described in clause 9.1 and defined in the WSDL and XML schema in clause A.2;
   - support the heartbeat service requirements specified in clause 9.1.
4) An implementation claiming conformance to the multiple object operations (MOO) service must:
   - support the MOO service interface described in clause 9.2 and defined in the WSDL and XML schema in clause A.3;
   - support all of the mandatory MOO service requirements specified in clause 9.2.5.
5) An implementation claiming conformance to the containment service must:
   - support the containment service interface described in clause 9.3 and defined in the WSDL and XML schema in clause A.4;
   - support the containment service requirements specified in clause 9.2.

10.2 Basic conformance profile

A system claiming conformance to the ITU-T.Q.818 basic profile shall support:

1) the version of WSDL, XML schema and SOAP as specified in clause 7.2
2) the WSN (see conformance point 1)
3) the UDDI service registration (see conformance point 2)

4) the heartbeat service (see conformance point 3)
5) the MOO service (see conformance point 5)
6) the containment service (see conformance point 4).

Annex A

WSDL definition of framework support services

(This annex forms an integral part of this Recommendation.)

A.1 ITU notification service WSDL and XML schema definition

1) ITU notification service XML schema definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for data types to be used in Notification Service
specified in this Recommendation.
   Filename : q818_NotificationService.xsd -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
   xmlns:nts="http://www.itu.int/xml-namespace/itu-t/q.818/NotificationService"
   targetNamespace="http://www.itu.int/xml-namespace/itu-
t/q.818/NotificationService"
   elementFormDefault="qualified" attributeFormDefault="unqualified"
version="1.0">

<xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
schemaLocation="x782.xsd"/>

   <xsd:complexType name="FilterType">
      <xsd:sequence>
         <xsd:element name="language" type="xsd:string"/>
         <xsd:any minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="EndpointReferenceType" mixed="false">
      <xsd:sequence>
         <xsd:element name="address" type="nts:AttributedURIType"/>
         <xsd:element name="referenceParameters"
type="nts:ReferenceParametersType" minOccurs="0"/>
         <xsd:element name="metadata" type="nts:MetadataType" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <xsd:complexType name="AttributedURIType" mixed="false">
      <xsd:simpleContent>
         <xsd:extension base="xsd:anyURI">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <xsd:complexType name="ReferenceParametersType" mixed="false">
      <xsd:sequence>
```

```
         <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <xsd:complexType name="MetadataType" mixed="false">
      <xsd:sequence>
         <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>

   <xsd:simpleType name="StatusType">
      <xsd:restriction base="xsd:boolean"/>
   </xsd:simpleType>
   <xsd:simpleType name="IdType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:complexType name="IdSetType">
      <xsd:sequence>
         <xsd:element name="id" type="nts:IdType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="SubscriptionStatusType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="locked"/>
         <xsd:enumeration value="unlocked"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="SubscribeRequestType">
      <xsd:sequence>
         <xsd:element name="managerId" type="nts:IdType"/>
         <xsd:element name="notificationTypes"
type="nts:NotificationTypeListType"/>
         <xsd:element name="filteringCriteria" type="nts:FilterType"
minOccurs="0"/>
         <xsd:element name="destination" type="nts:EndpointReferenceType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SubscribeResponseType">
      <xsd:sequence>
         <xsd:element name="subscriptionId" type="nts:IdType"/>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="UnsubscribeRequestType">
      <xsd:sequence>
         <xsd:element name="managerId" type="nts:IdType"/>
         <xsd:element name="subscriptionId" type="nts:IdType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="UnsubscribeResponseType">
      <xsd:sequence>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SuspendSubscriptionRequestType">
      <xsd:sequence>
         <xsd:element name="managerId" type="nts:IdType"/>
         <xsd:element name="subscriptionId" type="nts:IdType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SuspendSubscriptionResponseType">
      <xsd:sequence>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ResumeSubscriptionRequestType">
      <xsd:sequence>
         <xsd:element name="managerId" type="nts:IdType"/>
         <xsd:element name="subscriptionId" type="nts:IdType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ResumeSubscriptionResponseType">
      <xsd:sequence>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="GetNotificationTypesRequestType">
      <xsd:sequence>
         <xsd:element name="notificationIRPId" type="x782:NameType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="GetNotificationTypesResponseType">
      <xsd:sequence>
         <xsd:element name="notificationTypeList"
type="nts:NotificationTypeListType"/>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="QuerySubscriptionRequestType">
      <xsd:sequence>
         <xsd:element name="subscriptionId" type="nts:IdType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="QuerySubscriptionResponseType">
      <xsd:sequence>
         <xsd:element name="notificationTypeList"
type="nts:NotificationTypeListType"/>
         <xsd:element name="subscriptionStatus"
type="nts:SubscriptionStatusType"/>
```

```
         <xsd:element name="filteringCriteria" type="nts:FilterType"/>
         <xsd:element name="destination" type="nts:EndpointReferenceType"/>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ModifySubscriptionRequestType">
      <xsd:sequence>
         <xsd:element name="subscriptionId" type="xsd:string"/>
         <xsd:element name="filteringCriteria" type="nts:FilterType"
minOccurs="0"/>
         <xsd:element name="destination" type="nts:EndpointReferenceType"
minOccurs="0"/>
         <xsd:element name="notificationTypes" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ModifySubscriptionResponseType">
      <xsd:sequence>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ListAllSubscriptionIdsRequestType">
      <xsd:sequence>
         <xsd:element name="managerId" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ListAllSubscriptionIdsResponseType">
      <xsd:sequence>
         <xsd:element name="subscriptionIdSet" type="nts:IdSetType"/>
         <xsd:element name="status" type="nts:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>


<!-- The following provides the XML Schema definitions for the common
notifications contents defined in clause 8.3.1.2 and 8.3.1.3 of

this Recommendation. The common data types referenced in this Annex is from
[ITU-T X.782]. -->

   <xsd:simpleType name="NotificationTypeType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="objectCreation"/>
         <xsd:enumeration value="objectDeletion"/>
         <xsd:enumeration value="attributeValueChange"/>
         <xsd:enumeration value="stateChange"/>
         <xsd:enumeration value="communicationsAlarm"/>
         <xsd:enumeration value="environmentalAlarm"/>
         <xsd:enumeration value="equipmentAlarm"/>
         <xsd:enumeration value="processingErrorAlarm"/>
         <xsd:enumeration value="qualityOfServiceAlarm"/>
         <xsd:enumeration value="integrityViolation"/>
         <xsd:enumeration value="operationalViolation"/>
         <xsd:enumeration value="physicalViolation"/>
         <xsd:enumeration value="securityViolation"/>
         <xsd:enumeration value="timeDomainViolation"/>
         <xsd:enumeration value="relationshipChange"/>
         <xsd:enumeration value="heartbeat"/>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="NotificationTypeListType">
      <xsd:sequence>
         <xsd:element name="notificationType" type="nts:NotificationTypeType"
minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the common notification header content definition -->
   <xsd:complexType name="CommonNotificationHeaderType">
      <xsd:sequence>
         <xsd:element name="objectClass" type="xsd:string"/>
         <xsd:element name="objectInstance" type="x782:NameType"/>
         <xsd:element name="notificationID" type="x782:NotificationIDType"/>
         <xsd:element name="eventTime" type="xsd:dateTime"/>
         <xsd:element name="systemDN" type="x782:NameType"/>
         <xsd:element name="notificationType" type="nts:NotificationTypeType"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for objectCreation
and objectDeletion-->
   <xsd:complexType name="ObjectCreationDeletionNotificationType">
      <xsd:sequence>
         <xsd:element name="notificationHeader"
type="nts:CommonNotificationHeaderType"/>
         <xsd:element name="correlatedNotifications"
type="x782:CorrelatedNotificationType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalText" type="x782:AdditionalTextType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalInfo"
type="x782:AdditionalInformationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="sourceIndicator" type="x782:SourceIndicatorType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="attributeList" type="x782:AttributeNameAndValueType"
minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for stateChange -->
   <xsd:complexType name="StateChangeNotificationType">
      <xsd:sequence>
         <xsd:element name="notificationHeader"
type="nts:CommonNotificationHeaderType"/>
         <xsd:element name="correlatedNotifications"
type="x782:CorrelatedNotificationSetType" minOccurs="0" maxOccurs="1"/>
```

```
         <xsd:element name="additionalText" type="x782:AdditionalTextType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalInfo"
type="x782:AdditionalInformationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="sourceIndicator" type="x782:SourceIndicatorType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="stateChanges" type="x782:AttributeChangeSetType"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for
attributeValueChange -->
   <xsd:complexType name="AttributeValueChangeNotificationType">
      <xsd:sequence>
         <xsd:element name="notificationHeader"
type="nts:CommonNotificationHeaderType"/>
         <xsd:element name="correlatedNotifications"
type="x782:CorrelatedNotificationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalText" type="x782:AdditionalTextType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalInfo"
type="x782:AdditionalInformationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="sourceIndicator" type="x782:SourceIndicatorType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="attributeChanges"
type="x782:AttributeChangeSetType"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for
"communicationAlarm", "environmentalAlarm", "equipmentAlarm",

"processingErrorAlarm", "qualityOfServiceAlarm" -->
   <xsd:complexType name="AlarmNotificationType">
      <xsd:sequence>
         <xsd:element name="notificationHeader"
type="nts:CommonNotificationHeaderType"/>
         <xsd:element name="correlatedNotifications"
type="x782:CorrelatedNotificationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalText" type="x782:AdditionalTextType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalInfo"
type="x782:AdditionalInformationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="probableCause" type="x782:ProbableCauseType"/>
         <xsd:element name="specificProblems" type="x782:SpecificProblemSetType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="perceivedSeverity"
type="x782:PerceivedSeverityType"/>
         <xsd:element name="backedUpStatus" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
         <xsd:element name="backedUpObject" type="x782:NameType" minOccurs="0"
maxOccurs="1"/>
         <xsd:element name="trendIndication" type="x782:TrendIndicationType"
minOccurs="0" maxOccurs="1"/>
```

```
         <xsd:element name="thresholdInfo" type="x782:ThresholdInfoType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="stateChangeDefinition"
type="x782:AttributeChangeSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="monitoredAttributes"
type="x782:AttributeNameAndValueType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="proposedRepairActions"
type="x782:ProposedRepairActionSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="alarmEffectOnService" type="xsd:boolean"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="alarmingResumed" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
         <xsd:element name="suspectObjectList" type="x782:SuspectObjectSetType"
minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for
"integrityViolation", "operationalViolation", "physicalViolation",

"securityViolation", "timeDomainViolation" -->
   <xsd:complexType name="ViolationNotificationType">
      <xsd:sequence>
         <xsd:element name="notificationHeader"
type="nts:CommonNotificationHeaderType"/>
         <xsd:element name="correlatedNotifications"
type="x782:CorrelatedNotificationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalText" type="x782:AdditionalTextType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalInfo"
type="x782:AdditionalInformationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="securityAlarmCause"
type="x782:SecurityAlarmCauseType"/>
         <xsd:element name="securityAlarmSeverity"
type="x782:PerceivedSeverityType"/>
         <xsd:element name="securityAlarmDetector"
type="x782:SecurityAlarmDetectorType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="serviceUser" type="x782:ServiceUserType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="serviceProvider" type="x782:ServiceProviderType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="thresholdInfo" type="x782:ThresholdInfoType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="stateChangeDefinition"
type="x782:AttributeChangeSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="monitoredAttributes"
type="x782:AttributeNameAndValueType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="proposedRepairActions"
type="x782:ProposedRepairActionSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="alarmEffectOnService" type="xsd:boolean"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="alarmingResumed" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
         <xsd:element name="suspectObjectList" type="x782:SuspectObjectSetType"
minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for
"relationshipChange" -->
   <xsd:complexType name="RelationshipChangeNotificationType">
      <xsd:sequence>
         <xsd:element name="notificationHeader"
type="nts:CommonNotificationHeaderType"/>
         <xsd:element name="correlatedNotifications"
type="x782:CorrelatedNotificationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalText" type="x782:AdditionalTextType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="additionalInfo"
type="x782:AdditionalInformationSetType" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="sourceIndicator" type="x782:SourceIndicatorType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="RelationshipChanges"
type="x782:AttributeChangeSetType"/>
      </xsd:sequence>
   </xsd:complexType>

   <!-- The following is the notification content definition for "heartbeat" -->
   <xsd:complexType name="HeartbeatNotificationType">
      <xsd:sequence>
         <xsd:element name="systemLabel" type="xsd:string"/>
         <xsd:element name="period" type="xsd:unsignedLong"/>
         <xsd:element name="timeStamp" type="xsd:dateTime"/>
      </xsd:sequence>
   </xsd:complexType>

</xsd:schema>
```

2) ITU notification service WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WSDL Operation Definition for Notification Service specified in this
Recommendation.
   Filename : q818_NotificationService.wsdl -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:nts="http://www.itu.int/xml-namespace/itu-t/q.818/NotificationService"
name="NotificationService" targetNamespace="http://www.itu.int/xml-
namespace/itu-t/q.818/NotificationService">
   <import namespace="http://www.itu.int/xml-namespace/itu-
t/q.818/NotificationService" location="q818_NotificationService.xsd"/>
   <wsdl:message name="subscribeRequest">
      <wsdl:part name="subscribeInput" type="nts:SubscribeRequestType"/>
   </wsdl:message>
   <wsdl:message name="subscribeResponse">
      <wsdl:part name="subscribeOutput" type="nts:SubscribeResponseType"/>
   </wsdl:message>
   <wsdl:message name="unsubscribeRequest">
      <wsdl:part name="unsubscribeInput" type="nts:UnsubscribeRequestType"/>
   </wsdl:message>
   <wsdl:message name="unsubscribeResponse">
      <wsdl:part name="unsubscribeOutput" type="nts:UnsubscribeResponseType"/>
   </wsdl:message>
   <wsdl:message name="suspendSubscriptionRequest">
      <wsdl:part name="suspendSubscriptionInput"
type="nts:SuspendSubscriptionRequestType"/>
   </wsdl:message>
   <wsdl:message name="suspendSubscriptionResponse">
      <wsdl:part name="suspendSubscriptionOutput"
type="nts:SuspendSubscriptionResponseType"/>
   </wsdl:message>
   <wsdl:message name="resumeSubscriptionRequest">
      <wsdl:part name="resumeSubscriptionInput"
type="nts:ResumeSubscriptionRequestType"/>
   </wsdl:message>
   <wsdl:message name="resumeSubscriptionResponse">
      <wsdl:part name="resumeSubscriptionOutput"
type="nts:ResumeSubscriptionResponseType"/>
   </wsdl:message>
   <wsdl:message name="getNotificationTypesRequest">
      <wsdl:part name="getNotificationTypesInput"
type="nts:GetNotificationTypesRequestType"/>
   </wsdl:message>
   <wsdl:message name="getNotificationTypesResponse">
      <wsdl:part name="getNotificationTypesOutput"
type="nts:GetNotificationTypesResponseType"/>
   </wsdl:message>
   <wsdl:message name="querySubscriptionRequest">
      <wsdl:part name="querySubscriptionInput"
type="nts:QuerySubscriptionRequestType"/>
   </wsdl:message>
   <wsdl:message name="querySubscriptionResponse">
      <wsdl:part name="querySubscriptionOutput"
type="nts:QuerySubscriptionResponseType"/>
   </wsdl:message>
   <wsdl:message name="modifySubscriptionRequest">
      <wsdl:part name="modifySubscriptionInput"
type="nts:ModifySubscriptionRequestType"/>
   </wsdl:message>
   <wsdl:message name="modifySubscriptionResponse">
      <wsdl:part name="modifySubscriptionOutput"
type="nts:ModifySubscriptionResponseType"/>
   </wsdl:message>
   <wsdl:message name="listAllSubscriptionIdsRequest">
      <wsdl:part name="listAllSubscriptionIdsInput"
type="nts:ListAllSubscriptionIdsRequestType"/>
   </wsdl:message>
   <wsdl:message name="listAllSubscriptionIdsResponse">
      <wsdl:part name="listAllSubscriptionIdsOutput"
type="nts:ListAllSubscriptionIdsResponseType"/>
   </wsdl:message>
   <wsdl:portType name="NotificationService">
```

```
      <wsdl:operation name="subscribe">
         <wsdl:input message="nts:subscribeRequest"/>
         <wsdl:output message="nts:subscribeResponse"/>
      </wsdl:operation>
      <wsdl:operation name="unsubscribe">
         <wsdl:input message="nts:unsubscribeRequest"/>
         <wsdl:output message="nts:unsubscribeResponse"/>
      </wsdl:operation>
      <wsdl:operation name="suspendSubscription">
         <wsdl:input message="nts:suspendSubscriptionRequest"/>
         <wsdl:output message="nts:suspendSubscriptionResponse"/>
      </wsdl:operation>
      <wsdl:operation name="resumeSubscription">
         <wsdl:input message="nts:resumeSubscriptionRequest"/>
         <wsdl:output message="nts:resumeSubscriptionResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getNotificationTypes">
         <wsdl:input message="nts:getNotificationTypesRequest"/>
         <wsdl:output message="nts:getNotificationTypesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="querySubscription">
         <wsdl:input message="nts:querySubscriptionRequest"/>
         <wsdl:output message="nts:querySubscriptionResponse"/>
      </wsdl:operation>
      <wsdl:operation name="modifySubscription">
         <wsdl:input message="nts:modifySubscriptionRequest"/>
         <wsdl:output message="nts:modifySubscriptionResponse"/>
      </wsdl:operation>
      <wsdl:operation name="listAllSubscriptionIds">
         <wsdl:input message="nts:listAllSubscriptionIdsRequest"/>
         <wsdl:output message="nts:listAllSubscriptionIdsResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="NotificationServiceBinding"
type="nts:NotificationService">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="subscribe">
         <soap:operation soapAction="subscribe"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="unsubscribe">
         <soap:operation soapAction=" unsubscribe "/>
         <wsdl:input>
```

```
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="suspendSubscription">
         <soap:operation soapAction="suspendSubscription"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="resumeSubscription">
         <soap:operation soapAction="resumeSubscription"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getNotificationTypes">
         <soap:operation soapAction="getNotificationTypes"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="querySubscription">
         <soap:operation soapAction="querySubscription"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="modifySubscription">
```

```
         <soap:operation soapAction="modifySubscription"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="listAllSubscriptionIds">
         <soap:operation soapAction="listAllSubscriptionIds"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="NotificationService">
      <wsdl:port name="NotificationService"
binding="nts:NotificationServiceBinding">
         <soap:address location="http://www.itu.int/xml-namespace/itu-
t/q.818/NotificationSerivce"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

A.2 ITU heartbeat service WSDL and XML schema definition

1) ITU heartbeat service XML schema definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for data types to be used in HeartbeatService
specified in this Recommendation.
   Filename : q818_HeartbeatService.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:hs="http://www.itu.int/xml-namespace/itu-t/q.818/HeartbeatService"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/q.818/HeartbeatService"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
   <xsd:simpleType name="SystemLabelType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="HeartbeatPeriodType">
      <xsd:restriction base="xsd:unsignedLong"/>
   </xsd:simpleType>
   <xsd:simpleType name="GeneralizedTimeType">
```

```
      <xsd:restriction base="xsd:dateTime"/>
   </xsd:simpleType>
</xsd:schema>
```

2) ITU heartbeat service WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WSDL Operation Definition for Heartbeat Service specified in this
Recommendation.
   Filename : q818_HeartbeatService.wsdl -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:hs="http://www.itu.int/xml-namespace/itu-t/q.818/HeartbeatService"
name="HeartbeatService" targetNamespace="http://www.itu.int/xml-namespace/itu-
t/q.818/HeartbeatService">
   <import namespace="http://www.itu.int/xml-namespace/itu-
t/q.818/HeartbeatService" location="q818_HeartbeatService.xsd"/>
   <wsdl:message name="periodMessage">
      <wsdl:part name="period" type="hs:HeartbeatPeriodType"/>
   </wsdl:message>
   <wsdl:message name="systemLabelMessage">
      <wsdl:part name="systemLabel" type="hs:SystemLabelType"/>
   </wsdl:message>
   <wsdl:portType name="HeartbeatServicePort">
      <wsdl:operation name="periodGet">
         <wsdl:output message="hs:periodMessage"/>
      </wsdl:operation>
      <wsdl:operation name="periodSet">
         <wsdl:input message="hs:periodMessage"/>
      </wsdl:operation>
      <wsdl:operation name="systemLabelGet">
         <wsdl:output message="hs:systemLabelMessage"/>
      </wsdl:operation>
      <wsdl:operation name="systemLabelSet">
         <wsdl:input message="hs:systemLabelMessage"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="HeartbeatServiceBinding" type="hs:HeartbeatServicePort">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="periodGet">
         <soap:operation soapAction="periodGet"/>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="periodSet">
         <soap:operation soapAction="periodSet"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
      </wsdl:operation>
      <wsdl:operation name="systemLabelGet">
         <soap:operation soapAction="systemLabelGet"/>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="systemLabelSet">
         <soap:operation soapAction="systemLabelSet"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="HeartbeatService">
      <wsdl:port name="HeartbeatServicePort"
binding="hs:HeartbeatServiceBinding">
         <soap:address location="http://www.itu.int/xml-namespace/itu-
t/q.818/HeartbeatSerivce"/>
      </wsdl:port>
   </wsdl:service>
   <!-- The contents of heartbeat notification is defined in Annex A.1, and it
is to be sent by OASIS WSN Service, not by HeartbeatService itself. -->
</wsdl:definitions>
```

A.3 ITU MOO service WSDL and XML schema definition

1) ITU MOO service XML schema definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for data types to be used in Multiple Object
Operation(MOO) Service specified in this Recommendation.
   Filename : q818_MOOService.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
xmlns:moos="http://www.itu.int/xml-namespace/itu-
```

```
t/q.818/MultipleObjectOperationService" targetNamespace="http://www.itu.int/xml-
namespace/itu-t/q.818/MultipleObjectOperationService"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">

   <xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
schemaLocation="x782.xsd"/>

   <xsd:import namespace="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService" schemaLocation="x782_MOAccessService.xsd"/>

   <xsd:simpleType name="ScopeEnumType">

      <xsd:restriction base="xsd:string">

         <xsd:enumeration value="BasicObjectOnly"/>

         <xsd:enumeration value="WholeSubtree"/>

         <xsd:enumeration value="IndividualLevel"/>

         <xsd:enumeration value="BaseToLevel"/>

      </xsd:restriction>

   </xsd:simpleType>

   <xsd:complexType name="ScopeType">

      <xsd:sequence>

         <xsd:element name="scopeInd" type="moos:ScopeEnumType"/>

         <xsd:element name="level" type="xsd:short" minOccurs="0"/>

      </xsd:sequence>

   </xsd:complexType>

   <xsd:complexType name="GetResultsType">

      <xsd:sequence>

         <xsd:element name="name" type="x782:NameType"/>

         <xsd:element name="attributes"
type="x782:AttributeNameAndValueSetType"/>

         <xsd:element name="failedAttributes" type="x782:StringSetType"/>

      </xsd:sequence>

   </xsd:complexType>


   <xsd:complexType name="UpdateResultsType">

      <xsd:sequence>

         <xsd:element name="name" type="x782:NameType" minOccurs="1"
maxOccurs="unbounded"/>

         <xsd:element name="failedAttributes" type="x782:StringSetType"
minOccurs="0" maxOccurs="unbounded"/>

      </xsd:sequence>

   </xsd:complexType>

   <xsd:complexType name="DeleteResultsType">

      <xsd:sequence>

         <xsd:element name="name" type="x782:NameType" minOccurs="1"
maxOccurs="unbounded"/>

         <xsd:element name="notDeletable" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>

      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedGetRequestType">
      <xsd:sequence>
         <xsd:element name="baseName" type="x782:NameType"/>
         <xsd:element name="scope" type="moos:ScopeType"/>
         <xsd:element name="moClassList" type="x782:MOClassListType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="attributes" type="x782:StringSetType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedGetResponseType">
      <xsd:sequence>
         <xsd:element name="moInfo" type="moos:GetResultsType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedUpdateRequestType">
      <xsd:sequence>
         <xsd:element name="baseName" type="x782:NameType"/>
         <xsd:element name="scope" type="moos:ScopeType"/>
         <xsd:element name="moClassList" type="x782:MOClassListType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="modifications" type="moas:AttributeNVMListType"/>
         <xsd:element name="failuresOnly" type="xsd:boolean"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedUpdateResponseType">
      <xsd:sequence>
         <xsd:element name="updateResult" type="moos:UpdateResultsType"
minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedDeleteRequestType">
      <xsd:sequence>
         <xsd:element name="baseName" type="x782:NameType"/>
         <xsd:element name="scope" type="moos:ScopeType"/>
         <xsd:element name="moClassList" type="x782:MOClassListType"
minOccurs="0" maxOccurs="1"/>
         <xsd:element name="failuresOnly" type="xsd:boolean"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedDeleteResponseType">
      <xsd:sequence>
```

```
         <xsd:element name="deleteResult" type="moos:DeleteResultsType"
minOccurs="0" maxOccurs="unbounded"/>

      </xsd:sequence>

   </xsd:complexType>

</xsd:schema>
```

2) ITU MOO service WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- WSDL Operation Definition for Multiple Object Operation(MOO) Service
specified in this Recommendation.

   Filename : q818_MOOService.wsdl -->

<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:moos="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationService" name="MOOService"
targetNamespace="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationService">

   <import namespace="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationService" location="q818_MOOService.xsd"/>

   <wsdl:message name="scopedGetRequest">

      <wsdl:part name="scopedGetInput" type="moos:ScopedGetRequestType"/>

   </wsdl:message>

   <wsdl:message name="scopedGetResponse">

      <wsdl:part name="scopedGetOutput" type="moos:ScopedGetResponseType"/>

   </wsdl:message>

   <wsdl:message name="scopedUpdateRequest">

      <wsdl:part name="scopedUpdateInput" type="moos:ScopedUpdateRequestType"/>

   </wsdl:message>

   <wsdl:message name="scopedUpdateResponse">

      <wsdl:part name="scopedUpdateOutput"
type="moos:ScopedUpdateResponseType"/>

   </wsdl:message>

   <wsdl:message name="scopedDeleteRequest">

      <wsdl:part name="scopedDeleteInput" type="moos:ScopedDeleteRequestType"/>

   </wsdl:message>

   <wsdl:message name="scopedDeleteResponse">

      <wsdl:part name="scopedDeleteOutput"
type="moos:ScopedDeleteResponseType"/>

   </wsdl:message>

   <wsdl:portType name="MOOServicePort">

      <wsdl:operation name="scopedGet">

         <wsdl:input message="moos:scopedGetRequest"/>

         <wsdl:output message="moos:scopedGetResponse"/>
      </wsdl:operation>
      <wsdl:operation name="scopedUpdate">
         <wsdl:input message="moos:scopedUpdateRequest"/>
         <wsdl:output message="moos:scopedUpdateResponse"/>
      </wsdl:operation>
      <wsdl:operation name="scopedDelete">
         <wsdl:input message="moos:scopedDeleteRequest"/>
         <wsdl:output message="moos:scopedDeleteResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MOOServiceBinding" type="moos:MOOServicePort">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="scopedGet">
         <soap:operation soapAction="scopedGet"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="scopedUpdate">
         <soap:operation soapAction="scopedUpdate"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="scopedDelete">
         <soap:operation soapAction="scopedDelete"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MOOService">
    <wsdl:port name="MOOServicePort" binding="moos:MOOServiceBinding">
      <soap:address location="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationSerivce"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A.4 ITU containment service WSDL and XML schema definition

1) ITU containment service XML schema definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for Containment Service to be used in this
Recommendation.
  Filename : q818_ContainmentService.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:cs="http://www.itu.int/xml-namespace/itu-t/q.818/ContainmentService"
  xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
  xmlns:moos="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationService"
  targetNamespace="http://www.itu.int/xml-namespace/itu-
t/q.818/ContainmentService"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
version="1.0">

  <xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
schemaLocation="x782.xsd"/>
  <xsd:import namespace="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationService" schemaLocation="q818_MOOService.xsd"/>

  <xsd:complexType name="GetContainedRequestType">
    <xsd:sequence>
      <xsd:element name="base" type="x782:NameType"/>
      <xsd:element name="scope" type="moos:ScopeType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="GetContainedByClassRequestType">
    <xsd:sequence>
      <xsd:element name="base" type="x782:NameType"/>
      <xsd:element name="class" type="xsd:string"/>
      <xsd:element name="scope" type="moos:ScopeType"/>
    </xsd:sequence>
```

```
   </xsd:complexType>
</xsd:schema>
```

2) ITU containment service WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WSDL Operation Definition for Containment Service specified in this
Recommendation.
   Filename : q818_ContainmentService.wsdl -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
xmlns:cs="http://www.itu.int/xml-namespace/itu-t/q.818/ContainmentService"
xmlns:ns="http://www.itu.int/xml-namespace/itu-
t/q.818/MultipleObjectOperationService" xmlns:ns1="http://www.itu.int/xml-
namespace/itu-t/x.782/MOAccessService" name="ContainmentService"
targetNamespace="http://www.itu.int/xml-namespace/itu-
t/q.818/ContainmentService">
   <import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
location="x782.xsd"/>
   <import namespace="http://www.itu.int/xml-namespace/itu-
t/q.818/ContainmentService" location="q818_ContainmentService.xsd"/>

   <wsdl:message name="existsRequest">
      <wsdl:part name="name" type="x782:NameType"/>
   </wsdl:message>
   <wsdl:message name="existsResponse">
      <wsdl:part name="existsOutput" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="getContainedRequest">
      <wsdl:part name="getContainedInput" type="cs:GetContainedRequestType"/>
   </wsdl:message>
   <wsdl:message name="getContainedResponse">
      <wsdl:part name="moList" type="x782:NameSetType"/>
   </wsdl:message>
   <wsdl:message name="getContainedByClassRequest">
      <wsdl:part name="getContainedByClassInput"
type="cs:GetContainedByClassRequestType"/>
   </wsdl:message>
   <wsdl:message name="getContainedByClassResponse">
      <wsdl:part name="moList" type="x782:NameSetType"/>
   </wsdl:message>
   <wsdl:portType name="Containment">
      <wsdl:operation name="exists">
         <wsdl:input message="cs:existsRequest"/>
         <wsdl:output message="cs:existsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getContained">
         <wsdl:input message="cs:getContainedRequest"/>
         <wsdl:output message="cs:getContainedResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getContainedByClass">
         <wsdl:input message="cs:getContainedByClassRequest"/>
         <wsdl:output message="cs:getContainedByClassResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="ContainmentServiceSoapBinding" type="cs:Containment">
      <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="exists">
         <soap:operation soapAction="exists"/>
         <wsdl:input name="existRequest">
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs"/>
         </wsdl:input>
         <wsdl:output name="existResponse">
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getContained">
         <soap:operation soapAction="getContained"/>
         <wsdl:input name="getContainedRequest">
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs"/>
         </wsdl:input>
         <wsdl:output name="getContainedResponse">
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getContainedByClass">
         <soap:operation soapAction="getContainedByClass"/>
         <wsdl:input name="getContainedByClassRequest">
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs"/>
         </wsdl:input>
         <wsdl:output name="getContainedByClassResponse">
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="ContainmentService">
      <wsdl:port name="ContainmentService"
binding="cs:ContainmentServiceSoapBinding">
         <soap:address location="http://www.itu.int/xml-namespace/itu-
t/q.818/ContainmentSerivce"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Bibliography

[b-ITU-T Q.816]	Recommendation ITU-T Q.816 (2001), *CORBA-based TMN services*.
[b-ITU-T Q.816.1]	Recommendation ITU-T Q.816.1 (2001), *CORBA-based TMN services: Extensions to support coarse-grained interfaces*.
[b-ITU-T Q.816.2]	Recommendation ITU-T Q.816.2 (2007), *CORBA-based TMN services: Extensions to support service-oriented interfaces*.

Printed in Switzerland
Geneva, 2012