INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2009-2012

STUDY GROUP 2

TD 317 Rev.1 (PLEN/2)-E

English only

Original: English

Question(s): 9/2 Geneva, 21-29 March 2012

TD

Source: Editors, Q.818

Title: Draft Q.818 (ex Q.ws-xml): Web-Services based Management Services (for consent)

Draft Recommendation ITU-T Q.818

Web-Services based Management Services

Summary

This draft Recommendation defines a set of services required to support service-oriented Web-services based interfaces, and along with ITU-T X.782 composes a framework for Web-services based network management interface. 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.

Keywords

Web Services (WS), Web Services Description Language (WSDL), eXtensible Markup Language (XML), XML Schema, Distributed Processing, Managed Objects, Network Management Interfaces, Service-Oriented, Universal Description Discovery and Integration (UDDI).

Contact:	WANG Ying MIIT/BUPT China	Tel: Fax: Email:	+86-10-62283119 ext. 8514 +86-10-62283412 wangy@bupt.edu.cn
Contact:	WANG Zhi-li MIIT/BUPT China	Tel: Fax: Email:	+86-10-62283119 ext. 8726 +86-10-62283412 zlwang@bupt.edu.cn
Contact:	Cheng Qiao-gang ZTE China	Tel: Fax: Email:	+86-755-26770000-3711 +86-755-26773583 chen.qiaogang@zte.com.cn

Contents

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 paradigm can be introduced into network management interface, 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 Web-services based notification service is used in network management interfaces;
- How new service is registered and accessed using UDDI;
- How to monitor the availability of 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 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 managemnet services – Notification services – Protocol neutral requirements and analysis.*

[ITU-T X.701] Recommendation ITU-T M.X.701(1997), Information technology – Open Systems Interconnection – Systems management overview.

[ITU-T X.703] Recommendation ITU-T M.X.703(1997), Information technology – Open Distributed Management Architecture.

[ITU-T X.782] Recommendation ITU-T X.782 (2012), Guidelines for the definition of Web-services based managed objects and management interfaces.

[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) Version 1.1.

[W3C SOAP] W3C Recommendation (2007), SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).

[OASIS WSN] OASIS Specification (2006), Web Services Base Notification v1.3.

[OASIS UDDI] OASIS Specification (2004), Universal Description, Discovery and Integration v3.0.2.

3 Definitions

3.1 Terms defined elsewhere:

This Recommendation uses the following terms defined elsewhere:

3.1.1 **<manager>** [ITU-T M.3020]

3.1.2 **<agent>** [ITU-T M.3020]

3.1.3 **<managed object class>** [ITU-T X.701]

3.1.4 **<notification>** [ITU-T X.703]

3.2 Terms defined in this Recommendation

This Recommendation does not define any new term.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CMIP Common Management Information Protocol

CORBA Common Object Request Broker Architecture

GDMO Guideline for the Definition of Managed Object

GIOP General Inter-ORB Protocol

IIOP Internet Inter-ORB Protocol

MO Managed Object

MOC Managed Object Class

OASIS Advancing Open Standards for the Information Society

SOAP Simple Object Access Protocol

TMN Telecommunications Management Network

UDDI Universal Description Discovery and Integration

W3C World Wide Web Consortium

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 andXML Schema are included in this Recommendation, and WSDL specifying the management specific services, and supporting data types, included in a normative annex. The WSDL and XML Schema are written in a 9-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. Sub-clause 4.1 introduces the goals of the Web-services framework. Sub-clause 6.2 then provides terminology and requirements. The requirements in this clause are requirements that the framework must satisfy, which are based on the telecommunications management needs. Clauses 7, 8, 9 then describe a framework that meets these needs and define 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 modeled. 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 specification.

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 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 modeling 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 the some 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 modeling dependencies

As described in the previous clause, the explicit modeling 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 modeling 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 modeled on an interface and the means by which they are accessed using Web-services. [ITU-T X.782] uses a *service-grain* modeling approach, which means each modeled resource is only accessible through a specific Web-services. 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 modeled resources contain other modeled 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 a 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 a 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 can not 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 some 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 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 6.2.2. The scope is applied from some base managed entity down to some 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 nth level subordinates of the base entity;

3) the base entity and all of its subordinates down to and including the nth 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 over view

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 modeling 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 modelers developing models for use with this framework. These consist of modeling guidelines, and XML style idioms. All of these are depicted graphically in Figure 1.

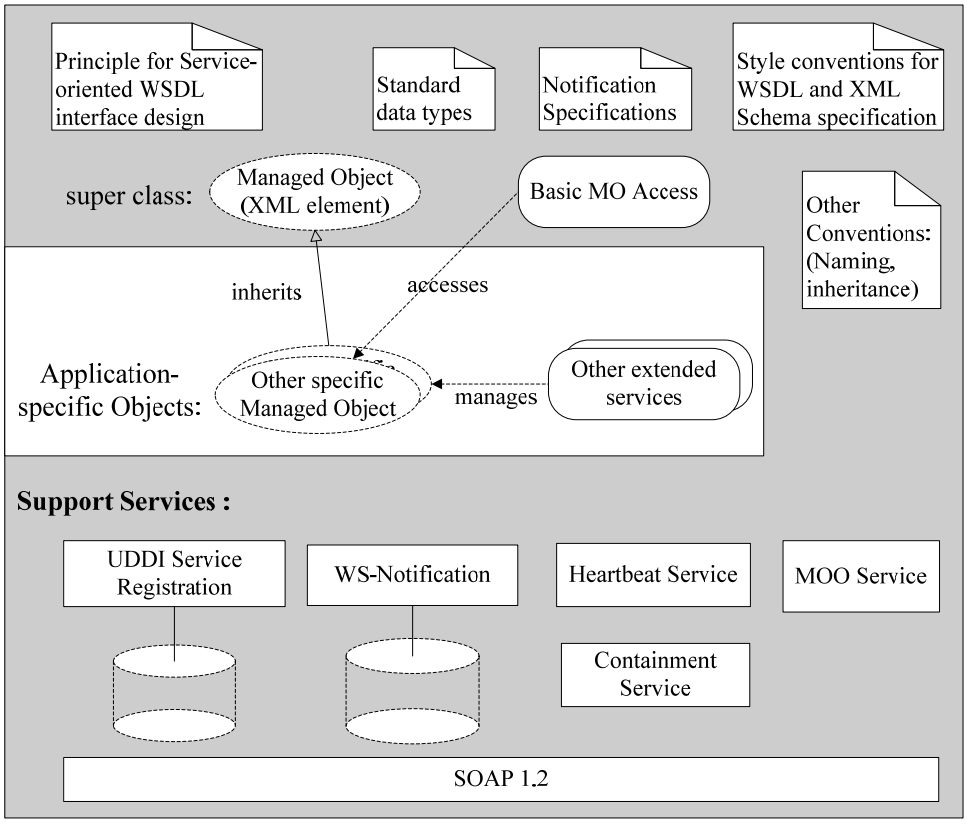


Figure 1 Overview of Framework

Figure 1 shows the framework in gray. 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 modeling conventions.

The framework services, represented as boxes with square corners, are defined in this Recommendation. The super class, notifications, and object modeling 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 techonology 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 of the applicable specifications must be supported to comply with this framework. A later version of a protocol, language or service that includes all the required capabilities of the stated version complies with this framework.

Table 1/Q.818 – Web-services language, prtocol and services versions

Service	Version
WSDL Recommendation (W3C)	1.1 (March, 2001)
XML Schema Recommedation (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 M.3702 and to WSN

The mechanisms and WSDL interfaces of [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 a protocol neutral requirements and analysis for generic notification management service, which defines the generic functions that are to be implemented by 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/Q.818 – Mappings from [ITU-T M.3702] to this Recommendation and [OASIS WSN]

No.	**[ITU-T M.3702]**		**[Q.818]**		**[OASIS WSN]**	
	Operation Name		**Operation Name**		**Operation**	
	Request	**Parameters**	**Request**	**Parameters**	**Request**	**Parameters**
1	subscribeNotification		subscribeNotification		Subscribe	
	Input parameter	- managerId - notificationTypes - filteringCriteria - destination	Request	- managerId: type="string" - notificationTypes: type="string" - filteringCriteria type="impl:FilterType" - destination: type= "wsa:EndpointReferenceType"	Request	-consumerReference, - filter, -subscriptionPolicy, -initialTerminationTime
	Output parameter	- subscriptionId - status	Response	- subscriptionId: type="string"	Response	subscriptionReference
2	unsubscribeNotification		unsubscribeNotification		Unsubscribe	
	Input parameter	- managerId - subscriptionId	Request	- managerId " type="string" - subscriptionId type="string"	Request	- subscriptionReference
	Output parameter	- status	Response	NULL	Response	NULL
3	suspendSubscription		suspendSubscription		PauseSubscription	
	Input parameter	- managerId - subscriptionId	Request	- managerId: type="string" - subscriptionId: type="string"	Request	- subscriptionReference
	Output parameter	- status	Response	NULL	Response	NULL
4	resumeSubscription		resumeSubscriptions		ResumeSubscription	

	Input parameter	- managerId - subscriptionId	Request	- managerId: type="string" - subscriptionId: type="string"	Request	- subscriptionReference
	Output parameter	- status	Response	NULL	Response	NULL
5	getNotificationTypes		getNotificationTypes		--	
	Input parameter	- notificationIRPId	Request	- notificationIRPId: type="string"		
	Output parameter	- status -notificationTypeList	Response	- notificationTypeList: type="impl:typeList "		
6	querySubscription		querySubscription		--	
	Input parameter	- subscriptionId	Request	- subscriptionId: type="string"		
	Output parameter	- subscribedNotificationType - subscriptionStatus - destination -filteringCriteria - status	Response	- notificationTypes: type="string" - subscriptionStatus: type="impl:status" - destination: type="wsa: EndpointReferenceType" - filteringCriteria: type= "impl:FilterType"		
7	modifySubscription		modifySubscription		--	
	Input parameter	- subscriptionId - filteringCriteria - destination - notificationTypes	Request	- subscriptionId: type="string" - filteringCriteria: type="impl:FilterType" - destination: type= "wsa: EndpointReferenceType" - notificationTypes: type="string"		
	Output parameter	- status	Response	NULL		
8	listAllSubscriptionIds		listAllSubscriptionIds		--	
	Input parameter	- managerId	Request	- managerId: type="string"		
	Output parameter	- subscriptionIdSet	Response	- subscriptionIdSet: type="impl:tIdSet "		

		- status				
9	report notification:		-- (Use WSN notify directly)		Notify:	
					notificationMessage	- SubscriptionReference - Topic - ProducerURL - Message
					Response	NULL

Additional descriptions for the mapping from the operations defined in this Recommendation to the ones defined in WS-Notification.

Table 3/Q.818 – Additional descriptions for the mappings from the operations in M.3702

Operation name	**Request/ Response**	**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 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 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 WSN definition. In the mapping of such an operation, the value of "subscriptionPolicy" should always use the "Notify message" in this Recommendation.
		--	*initialTerminationTime* initialTerminationTime indicate 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 a concept "managerId" in WSN, which will be processed and stored in Agent itself. and the type is string at this moment.
		subscriptionId	subscriptionReference
suspendSubscription	Request	managerId	There is no such a concept "managerId" in WSN, which will be processed and stored in Agent itself. and the type is string at this moment.

		subscriptionId	subscriptionReference
resumeSubscription	Request	managerId	There is no such a concept "managerId" in WSN, which will be processed and stored in Agent itself. and the type is string at this moment.
		subscriptionId	subscriptionReference
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, "notify"operation from WSN is used directly in an agent.		

The detailed interface definition can be found in Annex 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" operation which are defined in [OASIS WSN], which will be used in this Recommendation when sending notifications.

Table 4/Q.818 – notify operation 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 for the above components to be used or not 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 conponents of the Notify message are further described as follows:

`/wsnt:Notify`

Contains a collection of one or more NotificationMessages.

`/wsnt:Notify/wsnt:NotificationMessage`

Contains a Notification payload along with any metadata components, it will be explained in detail in the following:

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

An EndpointReference to the Subscription that is associated with the Notify message. This component is optional.

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

An EndpointReference to the NotificationProducer that produced the Notification artifact. This component is optional.

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

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`

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`

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

`/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 Table5.

Table 5/Q.818 – Most simplied 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/Q.818 – Common notificaiton header defintion

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

Parameter name	Qualifiers	XSD Data type	Descriptions
			notification. See Clause 7.3.5 of [ITU-T M.3702]
notificationId	M	xsd:string	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 DNof the System which sends out this notification. See Clause 7.3.5 of [ITU-T M.3702]
notificationType	M	xsd:string	It indicates the type of the 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” The notificaiton type can be extended when there are new types of notification defined.

8.1.3.3 Common Notification contents definition

(1) Notification contents for “objectCreation” and “objectDeletion” notifications

Table 7/Q.818 – Notificaiton contents for “objectCreation” and “objectDeletion”

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 inforamtion not in text format.
sourceIndicator	O	x782:SourceIndicatorType	Cause of event.
attributeList	O	x782:AttributeNam	Attribute values.

		eAndValueType	

(2) Notification contents for “attributeValueChange” notification

Table 8/Q.818 – Notificaiton contents for “attributeValueChange”

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 inforamtion not in text format.
sourceIndicator	O	x782:SourceIndicatorType	Cause of event.
attributeChanges	M	x782:AttributeValueChangeSetType	Changed attributes.

(3) Notification contents for “stateChange” notification

Table 9/Q.818 – Notificaiton contents for “stateChange”

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 inforamtion not in text format.
sourceIndicator	O	x782:SourceIndicatorType	Cause of event.
stateChanges	M	x782:AttributeValueChangeSetType	Changed states.

(4) Notification contents for “communicationAlarm”, “environmentalAlarm”, “equipmentAlarm”, “processingErrorAlarm”, “qualityOfServiceAlarm” notifications

Table 10/Q.818 – Notificaiton 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:AdditionalInfo	Additional inforamtion not in text format.

		rmationSetType	
probableCause	M	x782:ProbableCause Type	The probable cause of the alarm
specificProblems	O	x782:SpecificProble mSetType	Non standarized problems.
perceivedSeverity	M	x782:PerceivedSeve rityType	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:TrendIndicatio nType	See type for details
thresholdInfo	O	x782:ThresholdInfo Type	See type for details
stateChangeDefinition	O	x782:AttributeChan geSetType	It indicates the state changes in this alarm.
monitoredAttributes	O	x782:AttributeName AndValueType	See type for details
proposedRepairActions	O	x782:ProposedRepai rActionSetType	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:SuspectObject SetType	Objects possibly involved in failure.

(5) Notification contents for “integrityViolation”, “operationalViolation”, “physicalViolation”, “securityViolation”, “timeDomainViolation” notifications

Table 11/Q.818 – Notificaiton contents for Violations

Notification parameter	Qualifiers	**Data Type**	**Descriptions**
correlatedNotifications	O	x782:CorrelatedNoti ficationSetType	List of correlated notifications.
additionalText	O	xsd:string	Text message.
additionalInfo	O	x782:AdditionalInfo rmationSetType	Additional inforamtion not in text format.
securityAlarmCause	M	x782:SecurityAlarm CauseType	The cause of the security alarm
securityAlarmSeverity	M	x782:PerceivedSeve rityType	Clears allowed? X.721 appears to restrict the "cleared" value on this alarm, but clears should be allowed.
securityAlarmDetector	M	x782:SecurityAlarm DetectorType	See type for details
serviceUser	M	x782:ServiceUserTy pe	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 “relationshipChange” notification

Table 12/Q.818 – Notificaiton contents for “relationshipChange”

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 inforamtion 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 Annex A.5.

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.

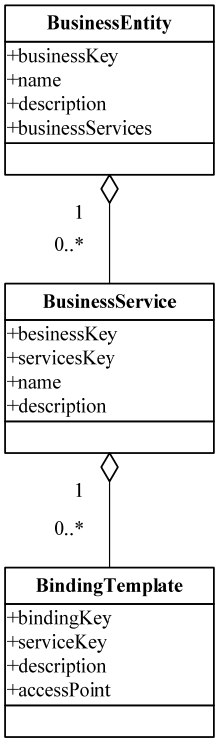
UDDI provides an interoperable, foundational infrastructure for a Web services-based software environment for both publicly available services and services only exposed internally within an organization.

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

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

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

8.2.1 UDDI information model used in network management



The simplifyied 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 network management domain, usually a managing system 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 nick name of the managed system. Both a managed system and a third party can be sregistered 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 management functionality (e.g., Configuration management, Performance management) by the managed system. businessService is the unit to collect 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 nick name 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 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 that case, the information of each accessPoint is contained in a bindingTemplate. A given bindingTemplate entity is uniquely identified by its attribute "bindingKey", and it is not allowed to provide more than one implementations that perform different functionalities for a network management service, which confuses a managing system for selecting 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 client to use, either to register a service or find a service. The following sub-clauses describe some related APIs that are expected to be used in network domain.

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

There are 3 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/Q.818 – Operation APIs in UDDI to register a Web service

Operation APIs	Parameter direction	Parameter name	Descriptions
get_authToken	arguements	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:	arguements	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 are included in the returned data
save_service	arguements	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 UDDI registry

Table 14/Q.818 – Operation APIs in UDDI to find a Web service

Operation APIs	Parameter direction	Parameter name	Descriptions
find_business	arguements	name	This optional collection of string values represents one or more names potentially qualified with xml:lang attributes. Since "exactMatch" is the default behavior, 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 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	arguements	name	This optional collection of string values represents one or more names potentially qualified with xml:lang attributes. Since "exactMatch" is the default behavior, 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 search functionality.
		businessKey	This uddi_key is used to specify a particular businessEntity instance to search. 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.
find_binding	arguements	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 passed are searched.
		findQualifier	This collection of findQualifier elements can be used to alter the default behaviour "exactMatch" of 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 descrption 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 that 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, and each of which has a pair of accessing methods, one for value get one for value set, as presented in the following table:

Table 15/Q.818 – Attributes and accessing operations in Heartbeat service

Internal attributes of Heartbeat services	**Operations**	**Message for Requests and Responses**

systemLabel	systemLabelGet	`Request: NULL`
		`Response:`
		`  name="systemLabelValue" type="xsd:string"`

	systemLabelSet	`Request:`
		`  name="systemLabelValue" type="xsd:string"`
		`Response: NULL`

period	periodGet	`Request: NULL`
		`Response:`
		`  name="periodValue" type="xsd:long"`

	periodSet	`Request:`
		`  name="periodValue" type="xsd:time"`
		`Response: NULL`

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

Table 16/Q.818 – Notificaiton contents for "heartbeat"

Heartbeat notification contents	Data Type	Descriptions
systemLabel	`xsd:string`	Identifies the managed system which sent out the heartbeat notification.
period	`xsd:long`	Indicates the time period of 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 Annex 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 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, defined in Annex A.3, is weakly-typed. It provides a set of generic capabilities that may be invoked on sets of any kinds of managed objects of any kinds. The operations supported are:

- 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 need only implements the scoped get operation. The rest 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 potentially will 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 complextType. 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/Q.818 – 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="q818: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/Q.818 – Request and response for ScopedGet operation

ScopedGetRequest	<xsd:complexType name="ScopedGetRequestType"> <xsd:sequence> <xsd:element name="baseName" type="x782:NameType"/> <xsd:element name="scope" type="q818:ScopeType"/> <xsd:element name="moClassList" type="x782:MOClassListType" minOccur=”0” maxOccur=”1”/> <xsd:element name="attributes" type="q818:StringSetType"/> </xsd:sequence> </xsd:complexType>
ScopedGetResponse	<xsd:complexType name="ScopedGetResponseType"> <xsd:sequence> <xsd:element name="results" type="q818:GetResultsSetType"/> </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/Q.818 – Data types for ScopedGet operationAttributeValueType	<xsd:complexType name="AttributeValueType"> <xsd:sequence> <xsd:element name="attributeName" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/> <xsd:element name="value" type="xsd:anyType" minOccurs="1" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType>
AttributeValueSetType	<xsd:complexType name="x782:AttributeNameAndValueType "> <xsd:sequence> <xsd:element name="element" type="x782:AttributeValueType" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType>
GetResultsType	<xsd:complexType name="GetResultsType"> <xsd:sequence> <xsd:element name="name" type="x782:NameType" minOccurs="1" maxOccurs="unbounded"/> <xsd:element name="attributes" type="x782:AttributeNameAndValueType" minOccurs="0" maxOccurs="unbounded"/> <xsd:element name="failedAttributes" type="impl:StringSetType" minOccurs="0" maxOccurs="unbounded"/>

	</xsd:sequence> </xsd:complexType> <xsd:complexType name="GetResultsSetType"> <xsd:sequence> <xsd:element name="element" type="impl:GetResultsType" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType>
StringSetType	<xsd:complexType name="StringSetType"> <xsd:sequence> <xsd:any minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType>

The first two types *AttributeValueType* and *AttributeValueSetType* 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 that structure, there are 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 a MO could not be retrieved either because the object did not have a matching attribute or some 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/Q.818 – Request and response for ScopedUpdate operation

ScopedUpdateRequest	<wsdl:message name="ScopedUpdateRequest"> <wsdl:part name="ScopedDeleteRequest" type="wst:ScopedDeleteRequestType"/> </wsdl:message> <xsd:complexType name="ScopedUpdateRequestType"> <xsd:sequence> <xsd:element name="baseName" type="x782:NameType"/> <xsd:element name="scope" type="q818:ScopeType"/> <xsd:element name="moClassList" type="x782:MOClassListType" minOccur="0" maxOccur="1"/> <xsd:element name="modifications" type="q818:ModificationSeqType"/> <xsd:element name="failuresOnly" type="xsd:boolean"/> </xsd:sequence> </xsd:complexType>
ScopedUpdateResponse	<wsdl:message name="ScopedUpdateResponse"> <wsdl:part name="ScopedUpdateResponse" type="wst:ScopedUpdateResponseType"/>

	</wsdl:message> <xsd:complexType name="ScopedUpdateResponseType"> <xsd:sequence> <xsd:element name="results" type="q818:UpdateResultsSetType"/> </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, 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 removed from it. The “add” and “remove” options are valid only if the attribute's type is a XML sequence type, 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/Q.818 – Modification related Data types for ScopedUpdate operation

ModificationOpType	<xsd:simpleType name="ModificationOpType"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="set"/> <xsd:enumeration value="add"/> <xsd:enumeration value="remove"/> </xsd:restriction> </xsd:simpleType>
ModificationType	<xsd:complexType name="ModificationType"> <xsd:sequence> <xsd:element name="attribute" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/> <xsd:element name="op" type="q818:ModificationOpType" minOccurs="1" maxOccurs="unbounded"/> <xsd:element name="value" type="xsd:anyType" minOccurs="1" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType>
ModificationSeqType	<xsd:complexType name="ModificationSeqType"> <xsd:sequence> <xsd:element name="element" type="q818:ModificationType" 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 is 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/Q.818 – Return data types for ScopedUpdate operation

UpdateResultsType	<xsd:complexType name="UpdateResultsType"> <xsd:sequence> <xsd:element name="name" type="x782:NameType" minOccurs="1" maxOccurs="unbounded"/> <xsd:element name="failedAttributes" type="q818:StringSetType" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType>
UpdateResultsSetType	<xsd:complexType name="UpdateResultsSetType"> <xsd:sequence> <xsd:element name="element" type="q818:UpdateResultsType" 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/Q.818 – Request and response for ScopedDelete operation

ScopedDeleteRequest	<wsdl:message name="ScopedDeleteRequest"> <wsdl:part name="ScopedDeleteRequest" type="wst:ScopedDeleteRequestType"/> </wsdl:message> <xsd:complexType name="ScopedDeleteRequestType"> <xsd:sequence> <xsd:element name="baseName" type="x782:NameType"/> <xsd:element name="scope" type="q818:ScopeType"/> <xsd:element name="moClassList" type="x782:MOClassListType" minOccur=”0” maxOccur=”1”/> </xsd:sequence> </xsd:complexType>
ScopedDeleteResponse	<wsdl:message name="ScopedDeleteResponse"> <wsdl:part name="ScopedDeleteResponse" type="wst:ScopedDeleteResponseType"/> </wsdl:message> <xsd:complexType name="ScopedDeleteResponseType">

	<xsd:sequence> <xsd:element name="results" type="q818:DeleteResultsSetType"/> </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/Q.818 – Data types for ScopedDelete operation

DeleteResultsType	<xsd:complexType name="DeleteResultsType"> <xsd:sequence> <xsd:element name="name" type="x782:NameType" minOccurs="1" maxOccurs="unbounded"/> <xsd:element name="noDeletable" type="xsd:boolean" minOccurs="0" maxOccurs="1"/> </xsd:sequence> </xsd:complexType>
DeleteResultsSetType	<xsd:complexType name="DeleteResultsSetType"> <xsd:sequence> <xsd:element name="element" type="q818:DeleteResultsType" minOccurs="0" maxOccurs="unbounded"/> </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 toward 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 aren't. 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, whose WSDL is defined in Annex A.3.

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

9.3 Containment service

In 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 Annex A.4.

A short description for the semantics of the Containment Service is illustrated in the following Table:

Table 25/Q.818 – Operations in Containment service

Operation	Parameter direction	Parameter name	Descriptions
exists	arguements	name: NameType	This is the DN of a managed object to be checked whether it exists in the Containment Services or not.
	return	result : boolean	Returns true if the specified MO exists in the Containment Service, otherwise the result will be false.
getContained	arguements	base: NameType	It indicates the base MO instance for a specified true for retrieving containment information.
		Scope: ScopeType	It indicates the scope information for retrieving

			containment information. See clause 9.2.1 for the semantics of scope.
	return	moList: NameSetType	Returns a list of MO names that is specified by the base and scope parameters.
getContainedByClass	arguements	base : NameType	It indicates the base MO instance for a specified true for retrieving containment information.
		moClass: string	It indicates the managed object class for retrieving containment information
		scope: ScopeType	It indicates the scope information for retrieving containment information. See clause 9.2.1 for the semantics of scope.
	return	moList: NameSetType	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 a *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 the Annex 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 servise is a function that should be implemented by managed systems, which is out side 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 7.2.
   - support the Notification interface described in clause 8.1 and defined in the WSDL and XML Schema in Annex A.1.
   - support the Notification format described in clause 8.1.3 and defined in the XML Schema in Annex A.5.
2) An implementation claiming conformance to the Services Registry requirements must:
   - support the OASIS UDDI Service Registration [OASIS UDDI] version specified in 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 servicer interface described in 9.1 and defined in the WSDL and XML Schema in Annex 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 9.2 and defined in the WSDL and XML Schema in Annex A.3.
   - support all of the mandatory MOO Service requirements specified in 9.2.5.
5) An implementation claiming conformance to the Containment service must:
   - support the Containment service interface described in 9.3 and defined in the WSDL and XML Schema in Annex A.4.
   - support the Containment Sevice 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 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) ITUNotificationService XML Schema definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:wsnt="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace=""http://www.itu.int/xml-
namespace/itu-t/q.818" elementFormDefault="qualified" attributeFormDefault="unqualified">
       <xsd:complexType name="FilterType">
           <xsd:sequence>
               <xsd:any minOccurs="0" maxOccurs="unbounded"/>
           </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="EndpointReferenceType" mixed="false">
           <xsd:sequence>
               <xsd:element name="Address" type="wsnt:AttributedURIType"/>
               <xsd:element name="ReferenceParameters"
type="wsnt:ReferenceParametersType" minOccurs="0"/>
               <xsd:element name="Metadata" type="wsnt: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:complexType name="typeList">
           <xsd:sequence>
```

```
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="IdSet">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="status">
        <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="xsd:string"/>
            <xsd:element name="notificationTypes" type="xsd:string"/>
            <xsd:element name="filteringCriteria" type="wsnt:FilterType"/>
            <xsd:element name="destination" type="wsnt:EndpointReferenceType"/>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="SubscribeResponseType">
        <xsd:sequence>
            <xsd:element name="subscriptionId" type="xsd:string"/>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="UnsubscribeRequestType">
        <xsd:sequence>
            <xsd:element name="managerId" type="xsd:string"/>
            <xsd:element name="subscriptionId" type="xsd:string"/>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="UnsubscribeResponseType">
        <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="suspendSubscriptionRequestType">
        <xsd:sequence>
            <xsd:element name="managerId" type="xsd:string"/>
            <xsd:element name="SubscriptionId" type="xsd:string"/>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="suspendSubscriptionResponseType">
```

```
            <xsd:sequence>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="resumeSubscriptionRequestType">
            <xsd:sequence>
                <xsd:element name="subscriptionId" type="xsd:string"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="resumeSubscriptionResponseType">
            <xsd:sequence>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="getNotificationTypesRequestType">
            <xsd:sequence>
                <xsd:element name="notificationIRPId" type="xsd:string"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="getNotificationTypesResponseType">
            <xsd:sequence>
                <xsd:element name="notificationTypeList" type="wsnt:typeList"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="querySubscriptionRequestType">
            <xsd:sequence>
                <xsd:element name="subscriptionId" type="xsd:string"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="querySubscriptionResponseType">
            <xsd:sequence>
                <xsd:element name="notificationTypeList" type="wsnt:typeList"/>
                <xsd:element name="subscriptionStatus" type="wsnt:status"/>
                <xsd:element name="filteringCriteria" type="wsnt:FilterType"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="modifySubscriptionRequestType">
            <xsd:sequence>
                <xsd:element name="subscriptionId" type="xsd:string"/>
                <xsd:element name="filteringCriteria" type="wsnt:FilterType"/>
                <xsd:element name="destination" type="wsnt:EndpointReferenceType"/>
                <xsd:element name="notificationTypes" type="xsd:string"/>
```

```
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="modifySubscriptionResponseType">
        <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="listAllSubscriptionIdsRequestType">
        <xsd:sequence>
            <xsd:element name="managerId" type="xsd:string"/>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="listAllSubscriptionIdsResponseType">
        <xsd:sequence>
            <xsd:element name="subscriptionIdSet" type="wsnt:IdSet"/>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

(2) ITUNotificationService WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsnt="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:wsdl-soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:impl="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://www.itu.int/xml-namecpace/itu-t/q.818">
    <wsdl:types>
        <xsd:schema>
            <xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/q.818"
schemaLocation="ITUNotificationService.xsd"/>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="SubscribeRequest">
        <wsdl:part name="SubscribeRequest" type="wsnt:SubscribeRequestType"/>
    </wsdl:message>
    <wsdl:message name="SubscribeResponse">
        <wsdl:part name="SubscribeResponse" type="wsnt:SubscribeResponseType"/>
    </wsdl:message>
    <wsdl:message name="UnsubscribeRequest">
        <wsdl:part name="UnsubscribeRequest" type="wsnt:UnsubscribeRequestType"/>
    </wsdl:message>
    <wsdl:message name="UnsubscribeResponse">
        <wsdl:part name="UnsubscribeResponse" type="wsnt:UnsubscribeResponseType"/>
```

```
        </wsdl:message>
        <wsdl:message name="suspendSubscriptionRequest">
            <wsdl:part name="suspendSubscriptionRequest"
type="wsnt:suspendSubscriptionRequestType"/>
        </wsdl:message>
        <wsdl:message name="suspendSubscriptionResponse">
            <wsdl:part name="suspendSubscriptionResponse"
type="wsnt:suspendSubscriptionResponseType"/>
        </wsdl:message>
        <wsdl:message name="resumeSubscriptionRequest">
            <wsdl:part name="resumeSubscriptionRequest"
type="wsnt:resumeSubscriptionRequestType"/>
        </wsdl:message>
        <wsdl:message name="resumeSubscriptionResponse">
            <wsdl:part name="resumeSubscriptionResponse"
type="wsnt:resumeSubscriptionResponseType"/>
        </wsdl:message>
        <wsdl:message name="getNotificationTypesRequest">
            <wsdl:part name="getNotificationTypesRequest"
type="wsnt:getNotificationTypesRequestType"/>
        </wsdl:message>
        <wsdl:message name="getNotificationTypesResponse">
            <wsdl:part name="getNotificationTypesResponse"
type="wsnt:getNotificationTypesResponseType"/>
        </wsdl:message>
        <wsdl:message name="querySubscriptionRequest">
            <wsdl:part name="querySubscriptionRequest"
type="wsnt:querySubscriptionRequestType"/>
        </wsdl:message>
        <wsdl:message name="querySubscriptionResponse">
            <wsdl:part name="querySubscriptionResponse"
type="wsnt:querySubscriptionResponseType"/>
        </wsdl:message>
        <wsdl:message name="modifySubscriptionRequest">
            <wsdl:part name="modifySubscriptionRequest"
type="wsnt:modifySubscriptionRequestType"/>
        </wsdl:message>
        <wsdl:message name="modifySubscriptionResponse">
            <wsdl:part name="modifySubscriptionResponse"
type="wsnt:modifySubscriptionResponseType"/>
        </wsdl:message>
        <wsdl:message name="listAllSubscriptionIdsRequest">
            <wsdl:part name="listAllSubscriptionIdsRequest"
type="wsnt:listAllSubscriptionIdsRequestType"/>
        </wsdl:message>
        <wsdl:message name="listAllSubscriptionIdsResponse">
            <wsdl:part name="listAllSubscriptionIdsResponse"
type="wsnt:listAllSubscriptionIdsResponseType"/>
        </wsdl:message>
        <wsdl:portType name="NotificationService">
            <wsdl:operation name="Subscribe">
                <wsdl:input name="SubscribeRequest" message="impl:SubscribeRequest"/>
                <wsdl:output name="SubscribeRequest" message="impl:SubscribeResponse"/>
            </wsdl:operation>
            <wsdl:operation name="Unsubscribe">
                <wsdl:input name="UnsubscribeRequest" message="impl:UnsubscribeRequest"/>
```

```
            <wsdl:output name="UnsubscribeResponse"
message="impl:UnsubscribeResponse"/>
        </wsdl:operation>
        <wsdl:operation name="suspendSubscription">
            <wsdl:input name="suspendSubscriptionRequest"
message="impl:suspendSubscriptionRequest"/>
            <wsdl:output name="suspendSubscriptioneResponse"
message="impl:suspendSubscriptionResponse"/>
        </wsdl:operation>
        <wsdl:operation name="resumeSubscription">
            <wsdl:input name="resumeSubscriptionRequest"
message="impl:resumeSubscriptionRequest"/>
            <wsdl:output name="resumeSubscriptioneResponse"
message="impl:resumeSubscriptionResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getNotificationTypes">
            <wsdl:input name="getNotificationTypesRequest"
message="impl:getNotificationTypesRequest"/>
            <wsdl:output name="getNotificationTypesResponse"
message="impl:getNotificationTypesResponse"/>
        </wsdl:operation>
        <wsdl:operation name="querySubscription">
            <wsdl:input name="querySubscriptionRequest"
message="impl:querySubscriptionRequest"/>
            <wsdl:output name="querySubscriptionResponse"
message="impl:querySubscriptionResponse"/>
        </wsdl:operation>
        <wsdl:operation name="modifySubscription">
            <wsdl:input name="modifySubscriptionRequest"
message="impl:modifySubscriptionRequest"/>
            <wsdl:output name="modifySubscriptionResponse"
message="impl:modifySubscriptionResponse"/>
        </wsdl:operation>
        <wsdl:operation name="listAllSubscriptionIds">
            <wsdl:input name="listAllSubscriptionIdsRequest"
message="impl:listAllSubscriptionIdsRequest"/>
            <wsdl:output name="listAllSubscriptionIdsResponse"
message="impl:listAllSubscriptionIdsResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="NotificationServiceBinding" type="impl:NotificationService">
        <wsdl-soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="Subscribe">
            <wsdl-soap:operation soapAction="Subscribe"/>
            <wsdl:input>
                <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:input>
            <wsdl:output>
                <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="Unsubscribe">
            <wsdl-soap:operation soapAction=" Unsubscribe "/>
```

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

```
            <wsdl:input>
                <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:input>
            <wsdl:output>
                <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="listAllSubscriptionIds">
            <wsdl-soap:operation soapAction="listAllSubscriptionIds"/>
            <wsdl:input>
                <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:input>
            <wsdl:output>
                <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ITUNotificationService">
        <wsdl:port name="NotificationService"
binding="impl:NotificationServiceBinding">
        </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"?>
<xsd:schema xmlns="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:impl="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace=
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:simpleType name="SystemLabelType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="HeartbeatPeriodType">
      <xsd:restriction base="xsd:long"/>
   </xsd:simpleType>
   <xsd:simpleType name="GeneralizedTimeType">
      <xsd:restriction base="xsd:time"/>
   </xsd:simpleType>
   <xsd:complexType name="HeartbeatType">
      <xsd:sequence>
         <xsd:element name="systemLable" type="impl:SystemLabelType"/>
         <xsd:element name="period" type="impl:HeartbeatPeriodType"/>
         <xsd:element name="timeStamp" type="impl:GeneralizedTimeType"/>
```

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

(2) ITU Heartbeat Service WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wst="http://www.itu.int/xml-namespace/itu-t/q.818" xmlns:wsdl-
soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:impl="http://www.itu.int/xml-
namespace/itu-t/q.818" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/q.818">
   <wsdl:types>
      <xsd:schema>
         <xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/q.818"
schemaLocation="HeartbeatService.xsd"/>
      </xsd:schema>
   </wsdl:types>
   <wsdl:message name="HeartbeatMsg">
      <wsdl:part name="HeartbeatMsg" type="wst:HeartbeatType"/>
   </wsdl:message>
   <wsdl:message name="PeriodGetValue">
        <wsdl:part name="PeriodGetValue" type="wst:HeartbeatPeriodType"/>
   </wsdl:message>
   <wsdl:message name="PeriodSetValue">
        <wsdl:part name="PeriodSetValue" type="wst:HeartbeatPeriodType"/>
   </wsdl:message>
   <wsdl:message name="SystemLabelGetValue">
        <wsdl:part name="SystemLabelGetValue" type="wst:SystemLabelType"/>
   </wsdl:message>
   <wsdl:message name="SystemLabelSetValue">
        <wsdl:part name="SystemLabelSetValue" type="wst:SystemLabelType"/>
   </wsdl:message>
   <wsdl:portType name="HeartbeatServicePort">
      <wsdl:operation name="HeartbeatOp">
         <wsdl:input name="Heartbeat" message="impl:HeartbeatMsg"/>
      </wsdl:operation>
      <wsdl:operation name="PeriodGet">
         <wsdl:output name="PeriodGetResult" message="impl:PeriodGetValue"/>
      </wsdl:operation>
      <wsdl:operation name="PeriodSet">
         <wsdl:input name="NewPeriod" message="impl:PeriodSetValue"/>
      </wsdl:operation>
      <wsdl:operation name="SystemLabelGet">
```

```
         <wsdl:output name="SystemLabelGetResult" message="impl:SystemLabelGetValue"/>
      </wsdl:operation>
      <wsdl:operation name="SystemLabelSet">
         <wsdl:input name="NewSystemLabel" message="impl:SystemLabelSetValue"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="HeartbeatServiceBinding" type="impl:HeartbeatServicePort">
      <wsdl-soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="HeartbeatOp">
         <wsdl-soap:operation soapAction="HeartbeatOp"/>
         <wsdl:input>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
      </wsdl:operation>
      <wsdl:operation name="PeriodGet">
           <wsdl-soap:operation soapAction="PeriodGet"/>
         <wsdl:output>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="PeriodSet">
           <wsdl-soap:operation soapAction="PeriodSet"/>
         <wsdl:input>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
      </wsdl:operation>
      <wsdl:operation name="SystemLabelGet">
           <wsdl-soap:operation soapAction="SystemLabelGet"/>
         <wsdl:output>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="SystemLabelSet">
           <wsdl-soap:operation soapAction="SystemLabelSet"/>
         <wsdl:input>
            <wsdl-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="impl:HeartbeatServiceBinding">
               <wsdl-soap:address location=""/>
      </wsdl:port>
   </wsdl:service>
</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"?>
<xsd:schema xmlns="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:impl="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.itu.int/xml-
namespace/itu-t/q.818" elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:simpleType name="NameType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="ScopeType">
      <xsd:union>
         <xsd:simpleType>
            <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:union>
   </xsd:simpleType>
   <xsd:complexType name="StringSetType">
      <xsd:sequence>
         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="AttributeValueType">
      <xsd:sequence>
         <xsd:element name="attributeName" type="xsd:string" minOccurs="1"
maxOccurs="unbounded"/>
         <xsd:element name="value" type="xsd:anyType" minOccurs="1"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="AttributeSetType">
      <xsd:sequence>
```

```
         <xsd:element name="element" type="impl:AttributeValueType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="GetResultsType">
      <xsd:sequence>
         <xsd:element name="name" type="impl:NameType" minOccurs="1"
maxOccurs="unbounded"/>
         <xsd:element name="attributes" type="impl:AttributeSetType" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="failedAttributes" type="impl:StringSetType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="GetResultsSetType">
      <xsd:sequence>
         <xsd:element name="element" type="impl:GetResultsType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="ModificationOpType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="set"/>
         <xsd:enumeration value="add"/>
         <xsd:enumeration value="remove"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="ModificationType">
     <xsd:sequence>
         <xsd:element name="attribute" type="xsd:string" minOccurs="1"
maxOccurs="unbounded"/>
         <xsd:element name="op" type="impl:ModificationOpType" minOccurs="1"
maxOccurs="unbounded"/>
         <xsd:element name="value" type="xsd:anyType" minOccurs="1"
maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ModificationSeqType">
      <xsd:sequence>
         <xsd:element name="element" type="impl:ModificationType" minOccurs="1"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="UpdateResultsType">
      <xsd:sequence>
         <xsd:element name="name" type="impl:NameType" minOccurs="1"
maxOccurs="unbounded"/>
```

```
         <xsd:element name="failedAttributes" type="impl:StringSetType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="UpdateResultsSetType">
      <xsd:sequence>
         <xsd:element name="element" type="impl:UpdateResultsType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="DeleteResultsType">
      <xsd:sequence>
         <xsd:element name="name" type="impl:NameType" minOccurs="1"
maxOccurs="unbounded"/>
         <xsd:element name="noDeletable" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="DeleteResultsSetType">
      <xsd:sequence>
         <xsd:element name="element" type="impl:DeleteResultsType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedGetRequestType">
      <xsd:sequence>
          <xsd:element name="baseName" type="x782:NameType"/>
          <xsd:element name="scope" type="q818:ScopeType"/>
          <xsd:element name="moClassList" type="x782:MOClassListType" minOccur=”0”
minOccur=”1”/>
          <xsd:element name="attributes" type="q818:StringSetType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedGetResponseType">
      <xsd:sequence>
          <xsd:element name="results" type="impl:GetResultsSetType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedUpdateRequestType">
      <xsd:sequence>
          <xsd:element name="baseName" type="impl:NameType"/>
          <xsd:element name="scope" type="impl:ScopeType"/>
          <xsd:element name="moClassList" type="x782:MOClassListType" minOccur=”0”
minOccur=”1”/>
          <xsd:element name="modifications" type="impl:ModificationSeqType"/>
          <xsd:element name="failuresOnly" type="xsd:boolean"/>
```

```
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedUpdateResponseType">
      <xsd:sequence>
          <xsd:element name="results" type="impl:UpdateResultsSetType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedDeleteRequestType">
      <xsd:sequence>
          <xsd:element name="baseName" type="impl:NameType"/>
          <xsd:element name="scope" type="impl:ScopeType"/>
          <xsd:element name="moClassList" type="x782:MOClassListType" minOccur=”0”
minOccur=”1”/>
          <xsd:element name="failuresOnly" type="xsd:boolean"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ScopedDeleteResponseType">
      <xsd:sequence>
          <xsd:element name="results" type="impl:DeleteResultsSetType"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

(2) ITU MOO Service WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wst="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:wsdl-soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:impl="http://www.itu.int/xml-namespace/itu-t/q.ws-xml"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/q.818">
   <wsdl:types>
      <xsd:schema>
         <xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/q.818"
schemaLocation="MOOService.xsd"/>
      </xsd:schema>
   </wsdl:types>
   <wsdl:message name="ScopedGetRequest">
      <wsdl:part name="ScopedGetRequest" type="wst:ScopedGetRequestType"/>
   </wsdl:message>
   <wsdl:message name="ScopedGetResponse">
      <wsdl:part name="ScopedGetResponse" type="wst:ScopedGetResponseType"/>
   </wsdl:message>
   <wsdl:message name="ScopedUpdateRequest">
```

```
      <wsdl:part name="ScopedUpdateRequest" type="wst:ScopedUpdateRequestType"/>
   </wsdl:message>
   <wsdl:message name="ScopedUpdateResponse">
      <wsdl:part name="ScopedUpdateResponse" type="wst:ScopedUpdateResponseType"/>
   </wsdl:message>
   <wsdl:message name="ScopedDeleteRequest">
      <wsdl:part name="ScopedDeleteRequest" type="wst:ScopedDeleteRequestType"/>
   </wsdl:message>
   <wsdl:message name="ScopedDeleteResponse">
      <wsdl:part name="ScopedDeleteResponse" type="wst:ScopedDeleteResponseType"/>
   </wsdl:message>
   <wsdl:portType name="MOOServicePort">
      <wsdl:operation name="ScopedGet">
         <wsdl:input name="ScopedGetRequest" message="impl:ScopedGetRequest"/>
         <wsdl:output name="ScopedGetResponse" message="impl:ScopedGetResponse"/>
      </wsdl:operation>
      <wsdl:operation name="ScopedUpdate">
         <wsdl:input name="ScopedUpdateRequest" message="impl:ScopedUpdateRequest"/>
         <wsdl:output name="ScopedUpdateResponse" message="impl:ScopedUpdateResponse"/>
      </wsdl:operation>
      <wsdl:operation name="ScopedDelete">
         <wsdl:input name="ScopedDeleteRequest" message="impl:ScopedDeleteRequest"/>
         <wsdl:output name="ScopedDeleteResponse" message="impl:ScopedDeleteResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MOOServiceBinding" type="impl:MOOServicePort">
      <wsdl-soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="ScopedGet">
         <wsdl-soap:operation soapAction="ScopedGet"/>
         <wsdl:input>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ScopedUpdate">
         <wsdl-soap:operation soapAction="ScopedUpdate"/>
         <wsdl:input>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
```

```
         <wsdl:output>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ScopedDelete">
         <wsdl-soap:operation soapAction="ScopedDelete"/>
         <wsdl:input>
            <wsdl-soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </wsdl:input>
         <wsdl:output>
            <wsdl-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="impl:MOOServiceBinding">
         <wsdl-soap:address location=" "/>
      </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"?>
<wsdl:definitions targetNamespace="urn:ContainmentService-pcs"
xmlns:impl="urn:ContainmentService-pcs" xmlns:intf="urn:ContainmentService-pcs"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
  <schema targetNamespace="urn:ContainmentService-pcs"
xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
       <simpleType name="ScopeChoiceType">
           <restriction base="string">
               <enumeration value="baseObjectOnly"/>
               <enumeration value="wholeSubtree"/>
               <enumeration value="individualLevel"/>
               <enumeration value="baseToLevel"/>
           </restriction>
       </simpleType>
   <complexType name="ArrayOf_soapenc_string">
    <complexContent>
```

```
    <restriction base="soapenc:Array">
     <attribute ref="soapenc:arrayType" wsdl:arrayType="soapenc:string[]"/>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="ContaintInput">
    <sequence>
       <element name="name" type="xsd:string"/>
       <element name="scope" type="impl:ScopeChoiceType"/>
       <element name="deepnum" type="xsd:int"/>
    </sequence>
  </complexType>
 </schema>
```

(2) ITU Containment Service WSDL definition

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wst="http://www.itu.int/xml-namespace/itu-t/q.818"
xmlns:wsdl-soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:impl="http://www.itu.int/xml-namespace/itu-t/q.ws-xml"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/q.818">
 </wsdl:types>
   <wsdl:message name="getContainedResponse">
      <wsdl:part name="getContainedReturn" type="impl:ArrayOf_soapenc_string"/>
   </wsdl:message>
   <wsdl:message name="getContainedRequest">
      <wsdl:part name="name" type="impl:ContaintInput"/>
   </wsdl:message>
   <wsdl:message name="existRequest">
      <wsdl:part name="name" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="existResponse">
      <wsdl:part name="existReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="scopedRequest">
      <wsdl:part name="name" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="scopedResponse">
      <wsdl:part name="scopedReturn" type="impl:ScopedGetResponse"/>
   </wsdl:message>
   <wsdl:portType name="Containment">
      <wsdl:operation name="exist" parameterOrder="name">
```

```
         <wsdl:input message="impl:existRequest" name="existRequest"/>
         <wsdl:output message="impl:existResponse" name="existResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getContained" parameterOrder="name">
         <wsdl:input message="impl:getContainedRequest" name="getContainedRequest"/>
         <wsdl:output message="impl:getContainedResponse" name="getContainedResponse"/>
      </wsdl:operation>
      <wsdl:operation name="scoped" parameterOrder="name">
         <wsdl:input message="impl:scopedRequest" name="scopedRequest"/>
         <wsdl:output message="impl:scopedResponse" name="scopedResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="ContainmentServiceSoapBinding" type="impl:Containment">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="exist">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="existRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="existResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getContained">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getContainedRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getContainedResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="scoped">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="scopedRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="scopedResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ContainmentService-pcs" use="encoded"/>
```

```
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="ContainmentService">
      <wsdl:port binding="impl:ContainmentServiceSoapBinding" name="ContainmentService">
         <wsdlsoap:address location=" "/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

A.5 Notificaiton Type XML Schema definition

This Annex provides the XML Schema definitions for the common notifications contents defined in clause 8.3.1.2 and 8.3.1.3. The common data types referenced in this Annex is from [ITU-T X.782].

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:q818="http://www.itu.int/xml-namespace/itu-t/q818"
xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x782"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/q818"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/x782"
schemaLocation="x782.xsd"/>

<!-- 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="x782: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="q.818: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="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="q.818: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="q.818: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="q.818: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="q.818: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="q.818: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>
```

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.

_____