

INTERNATIONAL TELECOMMUNICATION UNION

**TELECOMMUNICATION
STANDARDIZATION SECTOR**

STUDY PERIOD 2009-2012

STUDY GROUP 2

TD 92 (WP 2/2)-E

English only

Original: English

Question(s): 9/2

Geneva, 9-18 November 2010

TEMPORARY DOCUMENT

Source: Editors, Q.ws-xml

Title: Draft Q.ws-xml: Web-Service based Management Services (V0.2)

Draft Q.ws-xml

Web-Service based Management Services

Summary

This draft Recommendation defines a set of services required to support service-oriented Web-service based interfaces, and along with ITU-T X.ws-xml composes a framework for Web-service 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. A WSDL module defining the interfaces to the network management specific support services is also provided.

Keywords

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

Attention: This is not a publication made available to the public, but an internal ITU-T Document intended only for use by the Member States of ITU, by ITU-T Sector Members and Associates, and their respective staff and collaborators in their ITU related work. It shall not be made available to, and used by, any other persons or entities without the prior written consent of ITU-T.

Contents

1	SCOPE	3
2	REFERENCES.....	3
3	DEFINITIONS	3
3.1	TERMS DEFINED ELSEWHERE:.....	3
3.2	TERMS DEFINED IN THIS RECOMMENDATION	4
4	ABBREVIATIONS AND ACRONYMS	4
5	CONVENTIONS	4
6	WEB-SERVICE BASED NETWORK MANAGEMENT SERVICES GOAL AND REQUIREMENTS	4
6.1	GOALS.....	4
6.1.1	<i>Application interoperability</i>	5
6.1.2	<i>Common usage of well-known Web-Services</i>	5
6.2	INFORMATION MODELING DEPENDENCIES	5
6.2.1	<i>Access granularity</i>	5
6.2.2	<i>Representation of containment and naming</i>	5
6.2.3	<i>Object creation and deletion</i>	6
6.3	SCOPING AND FILTERING	6
6.3.1	<i>Scoping</i>	6
6.3.2	<i>Filtering</i>	7
6.4	NOTIFICATIONS	7
7	FRAMEWORK OVERVIEW AND PROTOCOL REQUIREMENTS.....	7
7.1	FRAMEWORK OVER VIEW	7
7.2	PROTOCOL REQUIREMENTS	8
8	USAGE OF WELL-KNOWN WEB-SERVICES.....	9
8.1	WEB-SERVICE NOTIFICATION	9
8.1.1	<i>Mapping of operations from M.3702</i>	9
8.1.2	<i>Notification format definitions</i>	12
8.2	UDDI SERVICE REGISTRATION.....	12
8.2.1	<i>UDDI Information Model used in network management</i>	13
8.2.2	<i>Register a new service and access it using UDDI</i>	14
9	FRAMEWORK SUPPORT SERVICES.....	15
9.1	MULTIPLE OBJECT OPERATION SERVICE	15
9.2	HEARTBEAT SERVICE	17
10	COMPLIANCE AND CONFORMANCE	18
ANNEX A WSDL DEFINITION OF FRAMEWORK SUPPORT SERVICES (THIS ANNEX FORMS AN INTEGRAL PART OF THIS RECOMMENDATION)		19
A.1	ITU NOTIFICATION SERVICE WSDL AND XML SCHEMA DEFINITION.....	19
A.2	ITU UDDI SERIVCE REGISTRATION WSDL AND XML SCHEMA DEFINITION	27
A.3	ITU HEARTBEAT SERIVCE REGISTRATION WSDL AND XML SCHEMA DEFINITION.....	27
A.4	ITU MULTIPLE OBJECT OPERATION SERICE WSDL AND XML SCHEMA DEFINITION	27

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-Service/XML is now an additional paradigm for network management.

This Recommendation, together with [ITU-T X.ws-xml], sets out to define a framework for defining how interfaces supported by management systems and network elements should be modelled using Web-Service/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-service notification is used in network management interfaces;
- How new service is registered and accessed using UDDI;
- How to monitor the availability of Web-Service notification forwarding channels;
- How to access multiple management object in one operation;
- Compliance and conformance requirements.

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T X.ws-xml] ITU-T Recommendation X.ws-xml (20XX), *Title*

[W3C] W3C Recommendation XXXX(20XX), *Title*

2007-08-28 *Semantic Annotations for WSDL and XML Schema*

2007-06-26 *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer Recommendation*

2007-06-26 *Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts Recommendation*

2007-06-26 *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*

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 terms.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

UDDI	Universal Description Discovery and Integration
W3C	World Wide Web Consortium
WS	Web-Service
XML	eXtensible Markup Language
XSD	XML Schema Definition

To be extended.

5 Conventions

This Recommendation has no conventions.

6 Web-service based network management services goal and requirements

This clause describes the key goals of the services framework and the requirements that help the Web-service based network management services support these goals. Sub-clause 4.1 introduces the goals of the Web-service 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-service 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-service 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-service based information models are specified in ITU-T X.ws-xml, 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.ws-xml 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-service notification and Web-service UDDI service registration centre, which will impact system interoperability (i.e., issues involving the use of Web-service 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-Service based managed objects detailed in ITU-T X.ws-xml describe the rules for modeling manageable resources. They also embody several decisions that must be supported by the network management Web-service based services framework. This clause summarizes those points.

6.2.1 Access granularity

Web-service 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-service. ITU-T X.ws-xml uses a *service-grain* modeling approach, which means each modeled resource is only accessible through a specific Web-service. The objects that represent manageable resources are called *managed objects*.

6.2.2 Representation of containment and naming

Containment is a logical representation of how 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.ws-xml 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-service 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

To be provided.

6.3 Scoping and filtering

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 and filtering. 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 and filtered request) is performed only on those Entities which meet criteria defined by a filter.

The use of scoping and filtering in this framework supports:

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

6.3.1 Scoping

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).

6.3.2 Filtering

Filters allow for the specification of criteria that Entities must meet in order to have a management operation performed. Together with scoping, filtering allows a single operation to be performed across multiple managed objects with a single operation request.

A filter parameter is used to determine whether or not an operation should be performed on a managed object. A filter parameter applies a test that is either satisfied or not by a particular managed object. The filter is expressed in terms of assertions about the presence or value of certain attributes of the managed object, and it is satisfied if and only if it evaluates to TRUE.

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

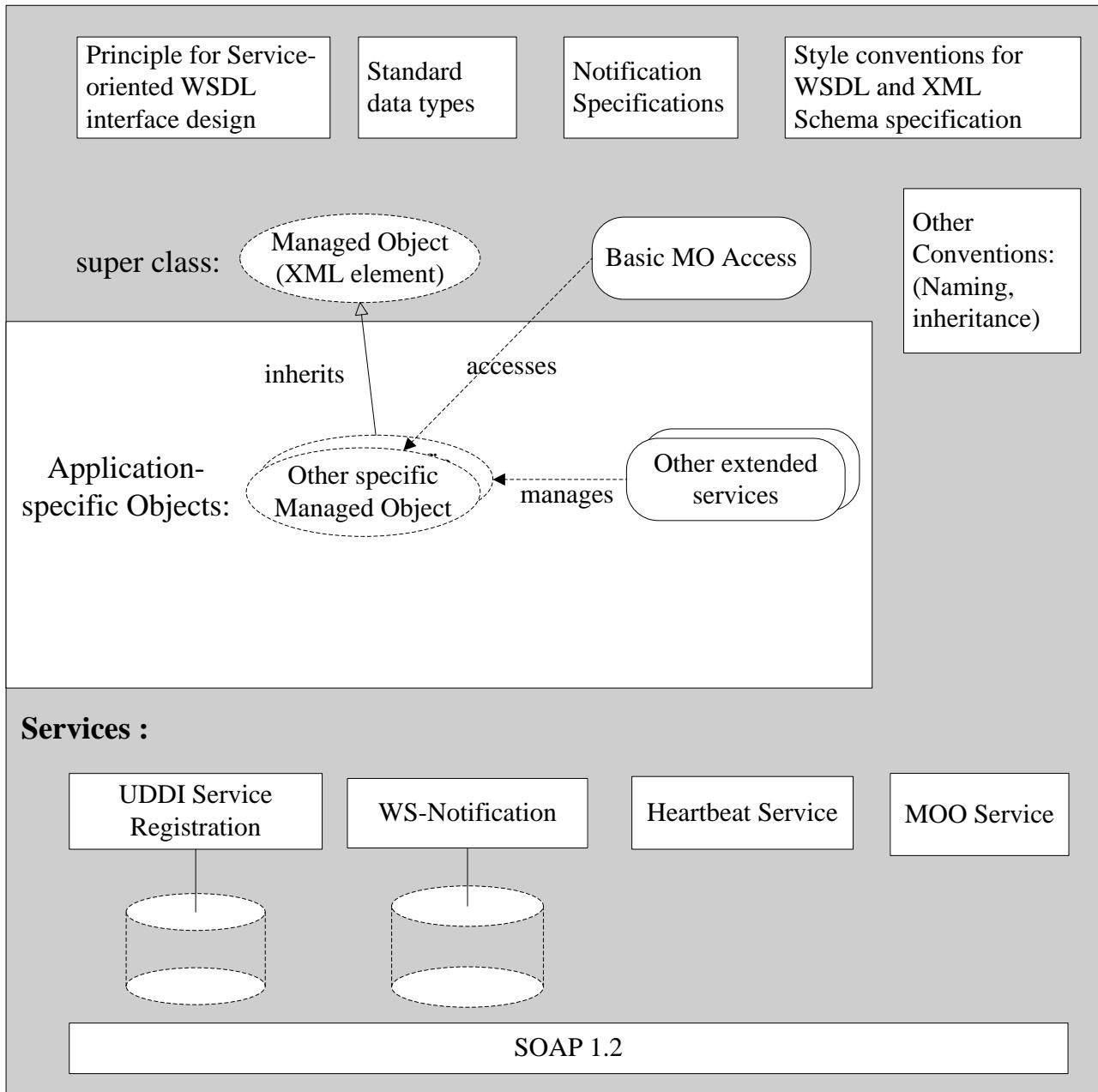


Figure 1 Overview of Framework

To be completed.

7.2 Protocol requirements

This clause defines the versions of the services that are required to support this framework. Web-services and protocol specifications are defined by the W3C. Table 1 shows which version of the applicable W3C specification must be supported to comply with this framework and indicates the clause in this Recommendation where detailed requirements are defined for the service. A later version of a service that includes all the required capabilities of the stated version complies with this framework.

Table 1/Q.ws-xml – Web services versions

Service	Version	Clause
WSDL Specification (W3C)	2.0	
XML Schema (W3C)	1.1 (2004 October)	
Web-Service notification (OASIS)	1.3	
UDDI service registration (OASIS)	3.0	
SOAP (W3C)	1.2	

To be extended and confirmed.

8 Usage of well-known Web-services

8.1 Web-Service Notification

This framework uses the OASIS Web-Service Notification specification for transferring the notification.

8.1.1 Mapping of operations from M.3702

The following table indicate the mapping from [ITU-T M.3702] to the web-service notifications.

[ITU-T M.3702]	Q.ws-xml	WS-Notification
1. <code>subscribeNotification()</code> : <code>input :managerId</code> <code>notificationTypes</code> <code>filteringCriteria</code> <code>destination</code> <code>output :subscriptionId</code> <code>status</code>	<code>subscribeNotification():</code> Request: <code>name="managerId" type="string"</code> <code>name="notificationTypes" type="string"</code> <code>name="filteringCriteria"</code> <code>type="impl:FilterType"</code> <code>name="destination"</code> <code>type="wsa:EndpointReferenceType"</code> Response : <code>name="subscriptionId" type="string"</code>	<code>Subscribe</code> <code>(consumerReference,</code> <code>filter,</code> <code>subscriptionPolicy,</code> <code>initialTerminationTime)</code> <code>returns:</code> <code>subscriptionReference</code>
2. <code>unsubscribeNotification()</code> : <code>input :managerId</code> <code>subscriptionId</code> <code>output:status</code>	<code>unsubscribeNotification()</code> Request: <code>name="managerId" type="string"</code> <code>name="subscriptionId" type="string"</code> Response :	<code>Unsubscribe</code> <code>(subscriptionReference)</code>

3. suspendSubscription(): input : managerId subscriptionId output: status	suspendSubscription(): Request: name="managerId" type="string" name="subscriptionId" type="string" Response : 	PauseSubscription (subscriptionReference)
4. resumeSubscriptions(): input : managerId subscriptionId output : status	resumeSubscriptions(): Request: name="managerId" type="string" name="subscriptionId" type="string" Response : 	ResumeSubscription (subscriptionReference)
5. getNotificationTypes(): input : notificationIRPId output: notificationTypeList status	getNotificationTypes(): Request: name="notificationIRPId" type="string" Response : name="notificationTypeList" type="impl:typeList"	--
6. querySubscription(): input : subscriptionId output: subscribedNotificationType subscriptionStatus destination filteringCriteria status	querySubscription(): Request: name="subscriptionId" type="string" Response: name="notificationTypes" type="string" name="subscriptionStatus" type="impl:status" name="destination" type="wsa:EndpointReferenceType" name="filteringCriteria" type="impl:FilterType"	--
7. modifySubscription(): input : subscriptionId filteringCriteria destination notificationTypes output: status	modifySubscription(): Request: name="subscriptionId" type="string" name="filteringCriteria" type="impl:FilterType" name="destination"	--

	<pre> e " type="wsa:EndpointReferenceTyp e" name="<u>notificationTypes</u>" type="string" Response : </pre>	
8. <u>listAllSubscriptionIds()</u> : <u>input</u> :managerId <u>output</u> :subscriptionIdSet status	<u>listAllSubscriptionIds()</u> : Request: name="managerId " type="string" Response : name="subscriptionIdSet" type="impl:IdSet"	--
9. <u>report notification:</u> <u>input</u> : notification	--	<u>Notify()</u> :

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

Operation name		Q. ws-xml	WS-Notificaiton
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中, which can be “raw” or any value in “Notify message”. “Notify message” can provide more information in a message according to WSN definition WSN. In the mapping of such an operation, the value of “subscriptionPolicy” should use the “Notify message”
		--	initialTerminationTime initialTerminationTime indicate the termination time of a subscription. When the value of attribute “ <u>xsi:nil</u> ” 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 such corresponding operations defined in WSN, and there should be implemented by Agent itself.		
querySubscription			
modifySubscription			
listAllSubscriptionIds			

The detailed interface definition can be found in Annex A.1.

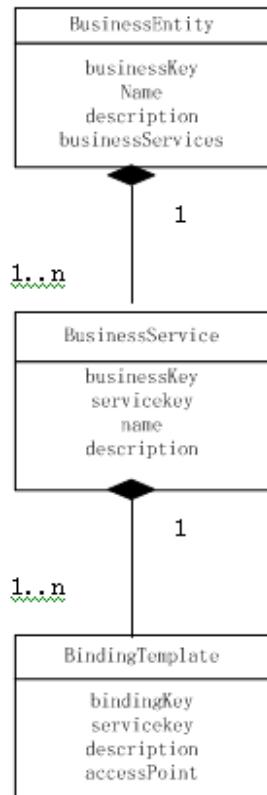
8.1.2 Notification format definitions

To be extended.

8.2 UDDI Service Registration

To be extended

8.2.1 UDDI Information Model used in network management



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

- **businessEntity**: Describes a business or other organization that typically provides Web services. It contains descriptive information about the business or provider and about the services it offers. This would include information such as name and description in multiple languages. A given instance of the businessEntity structure is uniquely identified by its businessKey. businessServices is a list of business services provided by the businessEntity.
- **businessService**: A businessService is the logical child of a single businessEntity. businessService describes a collection of related Web services offered by a businessEntity. It contains information such as ServiceKey , a given businessService entity is uniquely identified by its serviceKey. And also includes name and description. The businessKey attribute uniquely identifies the businessEntity which is the provider of the businessService. Every businessService is "contained" in exactly one businessEntity.
- **bindingTemplate**: Describes the technical information necessary to use a particular Web service. Each bindingTemplate is contained in a businessService. It includes information such as descriptive information and bindingKey, a given bindingTemplate entity is uniquely identified by its bindingKey. And the serviceKey attribute uniquely identifies the businessService that contains the bindingTemplate. The accessPoint is a string used to convey the network address suitable for invoking the Web service being described.

8.2.2 Register a new service and access it using UDDI

8.2.2.1 Using Publication API to publish a new Web service in UDDI

(1) get_authToken:

arguments: • **userId**: This required attribute argument is the user identifier that an individual authorized user was assigned by a UDDI node. Nodes should provide a means for individuals to obtain a userID and password credentials that will be valid at the given node.

• **cred**: This required attribute argument is the password or credential that is associated with the user.

return: • **authToken**: Upon successful completion this API call returns an authToken structure that contains a valid authInfo element that can be used in subsequent calls to API calls that require an authInfo value.

(2) save_business:

arguments: • **authInfo**: This argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call. Registries that serve multiple publishers and registries that restrict who can publish in them typically require authInfo for this call.

• **businessEntity**: Required repeating element containing one or more businessEntity structures.

return: • **businessKey**: This is assigned as a result of processing the save_business are included in the returned data.

(3) save_service:

arguments: • **authInfo**: This argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call. Registries that serve multiple publishers and registries that restrict who can publish in them typically require authInfo for this call.

• **businessService**: Required repeating element containing one or more complete businessService elements.

return : • **serviceKey** and **bindingKey** values that were assigned as a result of processing the save_service API are included in the businessService data.

8.2.2.2 Using Inquiry API to find the Web service in UDDI

(1) find_business:

arguments: • **name** :This optional collection of string values represents one or more names potentially qualified with xml:lang attributes. Since "exactMatch"

is the default 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: Returns a **businessList** that matches the criteria specified in the arguments.BusinessList ,this structure is a sequence of businesses.

(2) `find_service:`

- arguments:
- **name:** This optional collection of string values represents one or more names potentially qualified with xml:lang attributes. Since "exactMatch" is the default 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: Returns a **serviceList** that matches the criteria specified in the arguments.ServiceList ,this structure is a sequence of businesses.

(3) `find_binding:`

- arguments:
- **serviceKey:** This optional uddi_key is used to specify a particular instance of a businessService element in the registered data. Only bindings in the specific businessService data identified by the serviceKey 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: Returns a **bindingDetail** that contains zero or more bindingTemplate structures matching the criteria specified in the argument list.Each bindingTemplate structure contains the description and accessPoint.

9 Framework support services

9.1 Multiple object operation service

To be extended

(1) get attribute values of multiple object

MOO Service	Q.ws-xml
<pre>GetResultsSetType scopedGet (in NameType baseName, in ScopeType scope, in FilterType filter, in LanguageType language, in StringSetType attributes, in unsigned short howMany, out GetResultsIterator resultsIterator) raises (InvalidParameter, InvalidFilter, FilterComplexityLimit, Itut_x780::ApplicationError);</pre>	<p>Request:</p> <p>name="baseName" type="impl:NameType" name="scope" type="impl:ScopeType" name="filter" type="impl:FilterType" name="language" type="impl:Language" name="attributes" type="impl:StringSetType" name="howmany" type="xsd:short"</p> <p>Response:</p> <p>name="results" type="impl:GetResultsSetType"</p>

(2) set attribute values of multiple object

MOO Service(Q.816)	Q.ws-xml
<pre>UpdateResultsSetType scopedUpdate (in NameType baseName, in ScopeType scope, in FilterType filter, in LanguageType language, in ModificationSeqType modifications, in boolean failuresOnly, in unsigned short howMany, out UpdateResultsIterator resultsIterator) raises (InvalidParameter, InvalidFilter, filterComplexityLimit, itut_x780::ApplicationError);</pre>	<p>Request:</p> <p>name="baseName" type="impl:NameType" name="scope" type="impl:ScopeType" name="filter" type="impl:FilterType" name="language" type="impl:Language" name="modifications" type="impl:ModificationSeqType" name="failuresOnly" type="xsd:boolean" name="howmany" type="xsd:short"</p> <p>Response:</p> <p>name="results" type="impl:UpdateResultsSetType"</p>

(3) delete attribute values of multiple object

MOO 多对象操作(Q.816)	Q. WS-XML
<pre>DeleteResultsSetType scopedDelete (in NameType baseName, in ScopeType scope,</pre>	<p>Request:</p> <p>name="baseName" type="impl:NameType"</p>

<pre> in FilterType filter, in LanguageType language, in boolean failuresOnly, in unsigned short howMany, out DeleteResultsIterator resultsIterator) throws (InvalidParameter, InvalidFilter, FilterComplexityLimit, Itut_x780::ApplicationError); </pre>	<pre> name="scope" type="impl:ScopeType" name="filter" type="impl:FilterType" name="language" type="impl:Language" name="failuresOnly" type="xsd:boolean" name="howmany" type="xsd:short" </pre> <p>Response:</p> <pre> name="results" type="impl:DeleteResultsSe </pre>

9.2 Heartbeat service

To be extended

The Heartbeat Service is used to verify the operation of the notification channels on a managed system, as well as the communications network between the managed system and managing system.

As can be seen, the Heartbeat service has an attribute named *systemLabel*, and operations to set and get the period between heartbeats. *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. Each channel on the system has a *channelID* to identify the channel. *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* attribute, the ID of the channel through which the notification was sent, the current value for the period, and a timestamp.

9.2.1 Mapping from Q.816

HeartBeat	Q.WS - XML
<pre> void heartbeat (in SystemLabelType systemLabel, in ChannelIDTypechannelID, in HeartbeatPeriodType period, in GeneralizedTimeType timeStamp); </pre>	<p>Request:</p> <pre> name="systemLabel" type="impl:SystemLab name="channelID" type="impl:ChannelIDTy name="period" type="impl:HeartbeatPeriodT name="timeStamp" type="impl:GeneralizedTimeType" </pre> <p>Response:</p>

HeartbeatPeriodType periodGet() //get the period	Request: name="period" type="impl:HeartbeatPeriodTy
void periodSet(in HeartbeatPeriodType period) //set the period	Request: name="period" type="impl:HeartbeatPeriodTy Reponse:

10 Compliance and conformance

To be provided.

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://docs.oasis-open.org/myschema" xmlns:wsnt="http://docs.oasis-open.org/myschema"
  xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/myschema" 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: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://docs.oasis-open.org/myschema"
xmlns:wsdl-soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:impl="http://new.webservice.namespace"
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://new.webservice.namespace">
    <wsdl:types>
        <xsd:schema>
            <xsd:import namespace="http://docs.oasis-open.org/myschema"
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"
type="wsnt:suspendSubscriptionResponseType"/>
        <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>
<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="suspendSubscriptionResponse"
message="impl:suspendSubscriptionResponse"/>
    </wsdl:operation>
    <wsdl:operation name="resumeSubscription">
        <wsdl:input name="resumeSubscriptionRequest"
message="impl:resumeSubscriptionRequest"/>
        <wsdl:output name="resumeSubscriptionResponse"
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: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: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: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: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-soap:address location="http://localhost:8080/wsn-consumer/services/consumer"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

A.2 ITU UDDI Service Registration WSDL and XML Schema definition

A.3 ITU Heartbeat Service Registration WSDL and XML Schema definition

A.4 ITU Multiple Object Operation Service WSDL and XML Schema definition

Bibliography

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