Web Services Notification (WS-Notification)

Version 1.0

1/20/2004

Authors

Dave Chappell, Sonic Software
Steve Graham, IBM (editor)
Amy Lewis, TIBCO Software
Nataraj Nagaratnam, IBM
Peter Niblett, IBM (editor)
Jay Parikh, Akamai Technologies
Sanjay Patil, SAP
Shivajee Samdarshi, TIBCO Software
Steve Tuecke, Globus / Argonne National Laboratory
William Vambenepe, Hewlett-Packard Development Company
Bill Weihl, Akamai Technologies

Abstract

The Notification model, where a service provider, or other entity, initiates messages based on a subscription or registration of interest from a service requestor, is a common pattern that should be standardized in Web services.

This document specifies a standard Web services approach to notification using a topic-based pub/sub pattern. It includes: standard message exchanges to be implemented by service providers that wish to participate in Notifications, standard message exchanges for a notification broker service provider (allowing publication of messages from entities that are not themselves service providers), operational requirements expected of service providers and requestors that participate in notifications, and an XML model that describes topics.

Status

This WS-Notification specification is an initial draft release and is provided for review and evaluation only. The Companies hope to solicit your contributions and suggestions in the near future. The Companies make no warranties or representations regarding the specification in any manner whatsoever.

Table of Contents

1 Introduction

The Event-driven, or Notification-based, interaction pattern is a commonly used pattern for inter-object communications. Examples exist in many domains, for example in publish/subscribe systems provided by Message Oriented Middleware vendors, or in system and device management domains.

This notification pattern is increasingly being used in a Web services context; some of its characteristics include:

- Entities interested in receiving information (which we call **NotificationConsumers**) can be registered dynamically with Web services able to produce the information (**NotificationProducers**). The dynamic nature of this registration means:
  1. A NotificationProducer need have no knowledge of a NotificationConsumer prior to its registration.
  2. When registering, a NotificationConsumer can inform the NotificationProducer of the nature of the information it is interested in, and may additionally specify trigger conditions which determine when the information is sent.
- There may be more than one NotificationConsumer registered with a NotificationProducer for any class of information. The NotificationProducer delivers the information to all NotificationConsumers that have registered an interest.
- Whenever the NotificationProducer has some information to communicate, it delivers it to the appropriate registered NotificationConsumers as a NotificationMessage. A given NotificationConsumer may receive zero or more NotificationMessages from a given NotificationProducer throughout the time during which it is registered.

WS-Notification standardizes the concepts, message exchanges, WSDL 1.1 and XML Schema renderings required to express the pattern. The benefits of standardization include:

- **Interoperation between NotificationProducers and NotificationConsumers.** WS-Notification specifies a standard set of message exchanges that define the roles of NotificationProducer and NotificationConsumer. Any services that implement these message exchanges will be able to exchange NotificationMessages, subject to there being a common transport binding.
- **Interoperation between middleware providers.** The WS-Notification interfaces have been defined in a way which allows the implementation to be delegated to a middleware provider. This specification allows different middleware providers to interoperate.
- **Standardized mechanism to develop Topic taxonomies.** Interoperation is facilitated by having a standard way to name and describe Topics that is not tied to a particular implementation.
- **Standardized concepts and terminology.** A common set of concepts and terms simplifies the job of the application developer as well as aiding interoperation.

1.1 Goals and Requirements

The goals of WS-Notification are to standardize the terminology, concepts, message exchanges and the WSDL needed to express the notification pattern, and to provide a language to describe Topics.

1.1.1 Requirements

In meeting these goals, the specification must address the following specific requirements:

- **WS-Notification must be independent of binding-level details:** Transport protocol details must be orthogonal to the subscription and the delivery of the notifications, so that the specification can be used over a variety of different transports.
- **Must allow for Message Oriented Middleware implementations.** The design of WS-Notification must allow a service that is acting as a NotificationProducer to delegate its implementation of WS-Notification semantics to a Message Oriented Middleware provider.
- **Must support both direct and brokered Notification:** A NotificationConsumer may have a direct relationship with a NotificationProducer, or may connect to it via an intermediary service termed a NotificationBroker. The specification must allow a NotificationProducer or NotificationConsumer to participate in both brokered and non-brokered configurations.
- **Must allow for federation of brokers.** It must be possible to build configurations with multiple intermediary broker services in a dynamic fashion. This is to allow for greater scalability, and to permit sharing of administrative workload.
- **Must allow for simple transformation and aggregation of Topics:** It must be possible to construct configurations (using intermediary brokers) where the Topic subscribed to by the NotificationConsumer differs from the Topic published to by the NotificationProducer, yet NotificationMessages from the NotificationProducer are routed to the NotificationConsumer by a broker that is acting according to administratively-defined rules.
- **Must provide runtime metadata about Notifications:** It must be clear to a potential Subscriber what elements available for subscription are provided by a NotificationProducer, and in what formats the subscription for notification can be made.
- **Relationship to other WS-* specifications:** WS-Notification must be composable with other Web services specifications, in particular WS-Security, WS-Policy, WS-Federation, WS-Addressing, WS-Coordination, WS-ResourceProperties, WS-ResourceLifetime, Web Services for Remote Portals, and WS-ReliableMessaging and the WS-Resource framework [WS-Resource].

1.1.2 Non-Goals

The following topics are outside the scope of this specification:

- **Defining the format of Messages used to pass notification data:** The data carried in NotificationMessages is application-domain specific, and WS-Notification does not prescribe any particular format for this data**.**
- **Defining any particular "standard Events" or "standard Messages"**
- **Defining the mapping between Situations and NotificationMessages.**

- **Defining the means by which NotificationProducers and NotificationBrokers are discovered by subscribers**
- **Defining the specific policy language to be used to govern specifics of the notification message exchange between the NotificationProducer and the NotificationConsumer**

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

When describing abstract data models, this specification uses the notational convention used by the [XML Infoset]. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

When describing concrete XML schemas, this specification uses the notational convention of [WS-Security]. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xsd:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xsd:anyAttribute/>).

1.3 Namespaces

The following namespaces are used in this document:

Prefix	Namespace
s12	http://www.w3.org/2003/05/soap-envelope
xsd	http://www.w3.org/2001/XMLSchema
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
wsa	http://schemas.xmlsoap.org/ws/2003/02/addressing
wsnt	http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification
wsrl	http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceLifetime
wsrp	http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties

2 Relationship to Other Specifications

WS-Notification leverages the work defined in the WS-Resource [WS-Resource] family of specifications. WS-Resource defines the relationship between a Web service and stateful entities (known as WS-Resources, or just resources) by defining a so-called *implied resource pattern*. WS-Resource standardizes the use of WS-Addressing to specify how a Web service message MUST contain the identity of a WS-Resource

to be used as context for processing the message. Several aspects of WS-Notification, particularly Subscriptions, utilize this concept.

WS-Notification also uses WS-ResourceProperties [WS-ResourceProperties]. WS-ResourceProperties defines a mechanism by which a WSDL portType can be associated with an XML element that describes the data associated with a resource. This specification includes standard message exchanges associated with reading and writing a resource's data.

WS-Notification uses WS-ResourceLifetime [WS-ResourceLifetime] to provide direct, immediate destruction of resources as well as scheduled destruction of resources, based on a leasing or time-based model.

3 Terminology and Concepts

The following definitions outline the terminology and usage in this specification. This section gives only brief description of these terms.

Situation:

- A Situation is some occurrence within a Web service or its environment of interest to third parties.
- A Situation could be a change of the internal state of a Web service or an associated resource or could be environmental, such as a timer event. It could also be an external event, such as a piece of news that has been supplied by a news-feed service.
- WS-Notification does not specify what a Situation is or is not, nor does it define the relationship between a Situation and the NotificationMessage(s) that are used to describe it.

NotificationMessage:

- A NotificationMessage is an artifact of a Situation containing information about that Situation that some entity wishes to communicate to other entities.
- A NotificationMessage is represented as an XML element with a Namespace qualified QName and a type defined using XML Schema.
- A typical usage pattern is to define a single NotificationMessage type (to be precise, its defining XML element) for each kind of Situation, containing information pertinent to that kind of Situation; in this case one can think of a NotificationMessage instance as in some sense *being* (or at least representing) the Situation.
- A designer could choose to associate several different NotificationMessage types with a Situation, for example, describing different aspects of the Situation, destined for different target recipients, etc. Conversely it is possible that several essentially different Situations give rise to NotificationMessages of the same type.

Notification:

- A Notification is the act of transmitting a NotificationMessage to an interested party.

Publisher:

- A Publisher is an entity that creates NotificationMessages, based upon Situation(s) that it is capable of detecting and translating into NotificationMessage artifacts. It does not need to be a Web service.
- A Publisher MAY be a Web service that implements the message exchanges associated with the NotificationProducer interface, in which case it also distributes the NotificationMessages to the relevant NotificationConsumers.
- If a Publisher does not implement the message exchanges associated with NotificationProducer, then it is not required to support the Subscribe request message and does not have to maintain knowledge of the NotificationConsumers that are subscribed to it; a NotificationBroker takes care of this on its behalf.

PublisherRegistration:

- A PublisherRegistration is a resource, following the implied resource pattern identified in WS-Resource. A PublisherRegistration represents the relationship between a Publisher and a NotificationBroker, in particular which topic(s) the publisher is permitted to publish to.
- A PublisherRegistration resource is created when a Publisher sends the RegisterPublisher request message to a NotificationBroker.
- PublisherRegistration resources are manipulated by messages sent to a PublisherRegistrationManager Web service.

PublisherRegistrationManager:

- A PublisherRegistrationManager is a Web service that implements message exchanges associated with the PublisherRegistrationManager interface.
- A PublisherRegistrationManager provides services that allow a service requestor to query and manipulate PublisherRegistration resources that it manages. A PublisherRegistrationManager is a Web service that participates in the implied resource pattern.

NotificationProducer:

- A NotificationProducer is a Web service that implements the message exchanges associated with the NotificationProducer interface and supports one or more Topics.
- A NotificationProducer is capable of distributing NotificationMessages. It maintains a list of Subscription resources and when it has a NotificationMessage to distribute; it matches the NotificationMessage (and its associated Topic) against the interest registered in each Subscription in its list. If it identifies a match it issues a Notification to the NotificationConsumer associated with that Subscription.
- A Web Service that implements the message exchanges associated with NotificationProducer can be a Publisher (i.e. it creates the NotificationMessages itself) or it can be a NotificationBroker, distributing NotificationMessages that were produced by a separate Publisher entity.
- It is the factory for Subscription resources.

Topic:

- A Topic is the concept used to categorize Notifications and their related NotificationMessage schemas.

- Topics are used as part of the matching process that determines which (if any) subscribing NotificationConsumers should receive a NotificationMessage.
- Every NotificationMessage instance generated by a Publisher is associated with a Topic. The relation between Situation and Topic is not specified by WS-Notification but MAY be specified by the designer of the TopicSpace.
- A synonym in some other publish/subscribe models is *subject*.

NotificationConsumer:

- A NotificationConsumer is a Web Service that receives NotificationMessages from a NotificationProducer.
- A NotificationConsumer may implement the generic Notify message exchange, or it may be able to process one or more domain-specific NotificationMessage types.

Subscription:

- A Subscription is a WS-Resource, following the implied resource pattern defined in [WS-Resource]. A Subscription represents the relationship between a NotificationConsumer, NotificationProducer, Topic and various other optional filter expressions, policies and context information.
- A Subscription resource is created when a Subscriber sends the Subscribe request message to a NotificationProducer.
- Subscription resources are manipulated by messages sent to the SubscriptionManager Web service associated with the Subscription resource, using the implied resource pattern.

SubscriptionManager

- A SubscriptionManager is a Web service that implements message exchanges associated with the SubscriptionManager interface.
- A SubscriptionManager provides services that allow a service requestor to query and manipulate Subscription resources that it manages. A SubscriptionManager is a Web service that participates in the implied resource pattern.

Subscriber:

- A Subscriber is an entity (often a Web service) that acts as a service requestor, sending the subscribe request message to a NotificationProducer.
- Note that a Subscriber may be a different entity than the NotificationConsumer that actually receives the NotificationMessages.

NotificationBroker:

- A NotificationBroker is an intermediary Web Service that decouples NotificationConsumers from Publishers.
- It implements both the NotificationProducer and NotificationConsumer interfaces.
- Because a NotificationBroker is an intermediary, it provides additional capabilities to the basic NotificationProducer interface:
    - It can relieve a Publisher from having to implement message exchanges associated with NotificationProducer; the NotificationBroker takes on the duties of a SubscriptionManager (managing subscriptions)

and NotificationProducer (distributing NotificationMessages) on behalf of the Publisher.

- It can reduce the number of inter-service connections and references, if there are many Publishers and many NotificationConsumers
- It can act as a *finder service*. Potential Publishers and Subscribers can in effect find each other by utilizing a common NotificationBroker.
- It can provide anonymous Notification, so that the Publishers and NotificationConsumers need not be aware of each others identity.

- An implementation of a NotificationBroker may provide additional added-value function that is beyond the scope of this specification, for example logging NotificationMessages, or transforming Topics and/or NotificationMessage content. Additional function provided by a NotificationBroker can apply to all Publishers that utilize it.

Demand-Based Publisher:

- Some Publishers may be interested in knowing whether they have any Subscribers or not, since producing a NotificationMessage may be a costly process. Such Publishers can register with the NotificationBroker as a *Demand-Based Publisher*.
- Demand-Based Publishers implement message exchanges associated with the NotificationProducer interface.
- The NotificationBroker subscribes to the Demand-Based Publisher. When the NotificationBroker knows that there are no Subscribers for the NotificationMessages from a Demand-Based Publisher it pauses its Subscription with that Publisher; when it knows that there are some Subscribers, it resumes the Subscription.
- This way the Demand-Based Publisher doesn't need to produce messages when there are no Subscribers, however a Demand-Based Publisher is only required to support a single Subscriber on any given Topic, and so can delegate the management of multiple Subscribers, delivery to multiple NotificationConsumers and other related issues (for example security) to the NotificationBroker.

Topic Space:

- A forest of Topic Trees grouped together into the same namespace for administrative purposes.

Topic Tree:

- A hierarchical grouping of Topics.

WS-Resource:

- When a stateful resource participates in the implied resource pattern, we refer to it as a *WS-Resource* [WS-Resource].

Implied Resource Pattern

- The way WS-Addressing is used to associate a stateful resource with the execution of message exchanges implemented by a Web service.
- An EndpointReference that follows the implied resource pattern MUST include a ReferenceProperties child element that identifies the WS-Resource to be

associated with the execution of all message exchanges performed using this EndpointReference.

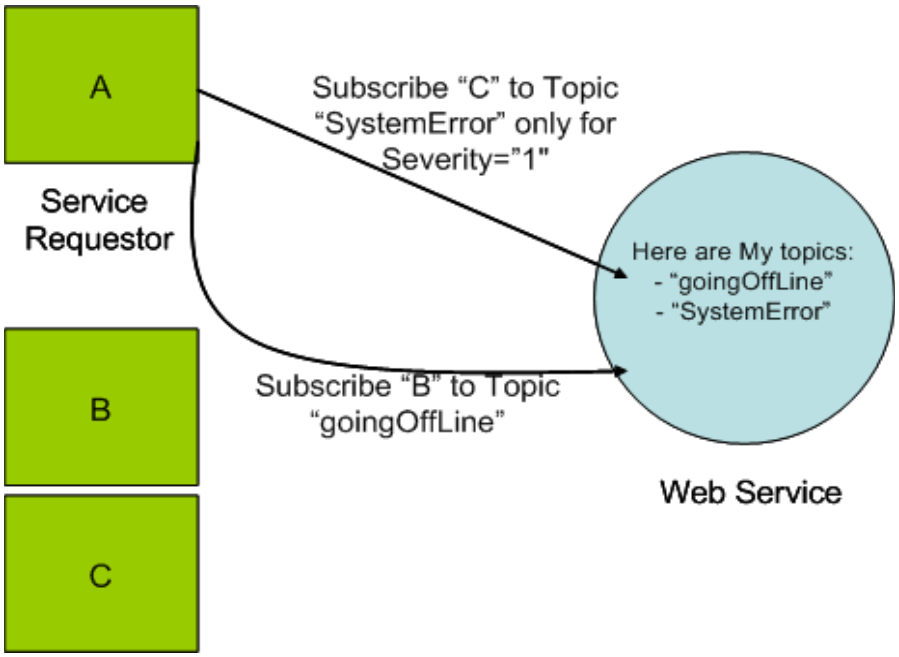
- A message that follows the implied resource pattern MUST be sent to a Web service referred to by an EndpointReference that follows the implied resource pattern, and MUST include the ReferenceProperties information from that EndpointReference, as specified by WS-Addressing.
- A Web service that follows the implied resource pattern MUST use the ReferenceProperties information from a message that follows the implied resource pattern in order to identify the WS-Resource to associate with the execution requested by that message.

WS-Resource Qualified Endpoint Reference

- An Endpoint Reference that associates a WS-Resource with a Web service.
- The identity of the WS-Resource MUST be contained within the ReferenceProperties element of the Endpoint Reference.
- The address of the Web service associated with the WS-Resource MUST be contained in the Address element of the Endpoint Reference.

4 Example

Consider the following depiction:

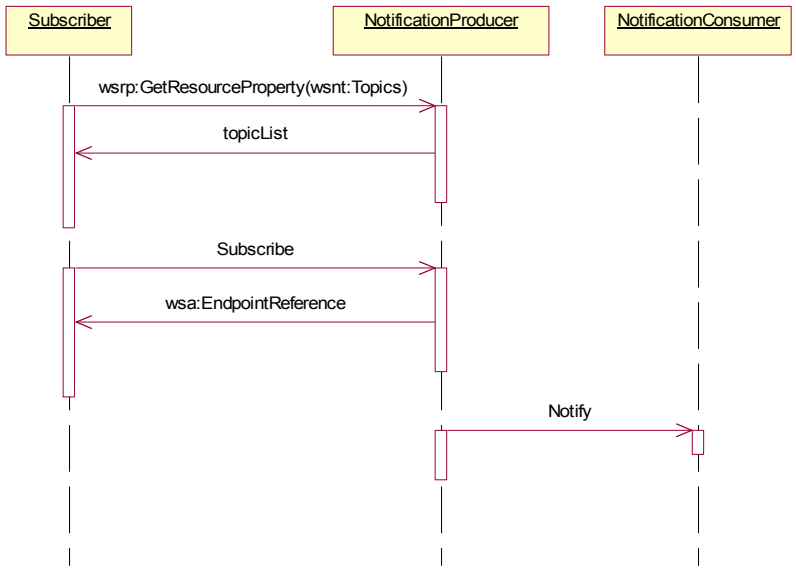


The figure above shows a Web service (a simple Web application server manager, for example) that supports two topics: "goingOffLine" and "systemErrorDetected". The Web service provides an operation to allow requestors to retrieve the list of topics that it supports.

The service requestor (A), playing the role of subscriber, registers interest in the topics associated with that Web service, by sending the Subscribe request message to the NotificationProducer Web service, as depicted by arrows on the above figure.

When certain Situations happen within the operating environment of the Web service, one or more NotificationMessages are generated on one or more of its Topics. NotificationMessages are sent to the interested parties (NotificationConsumers) to notify them of the Situation.

The following diagram illustrates a possible sequence of operations:

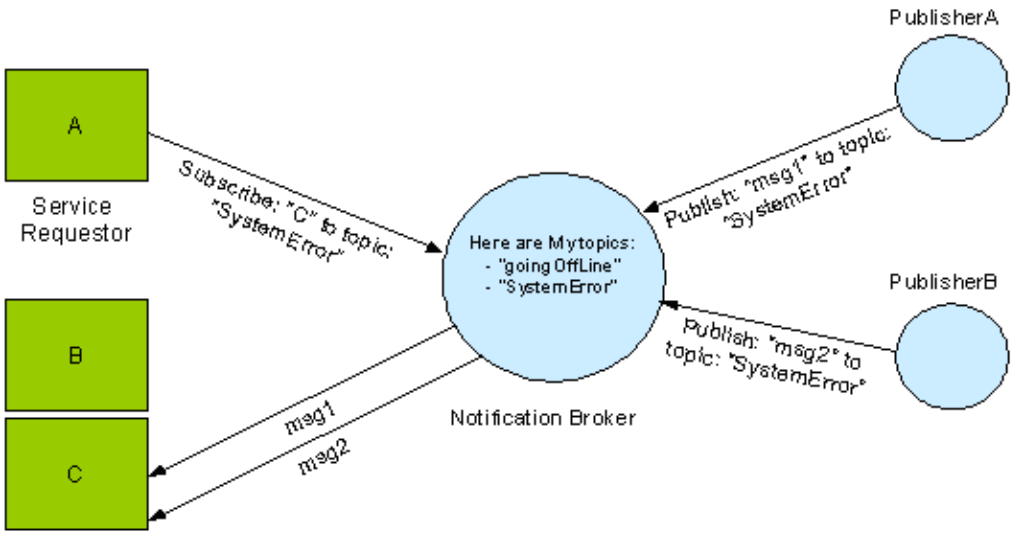


The Subscriber verifies that a topic of interest is supported by a chosen NotificationProducer. This verification is done by sending a GetResourceProperty request message [WS-ResourceProperties]retrieving the current values of the "Topics" resource property of the NotificationProducer. Note that this step is optional; a Subscriber may attempt to create a subscription without first verifying that the NotificationProducer supports the Topic of interest.

The Subscriber then sends a Subscribe request message to the NotificationProducer, indicating the address of the NotificationConsumer, the Topic(s) for the Subscription, and other related Subscription information. In response to this message, the NotificationProducer creates a Subscription resource and returns an EndpointReference [WS-Addressing] to this Subscription resource.

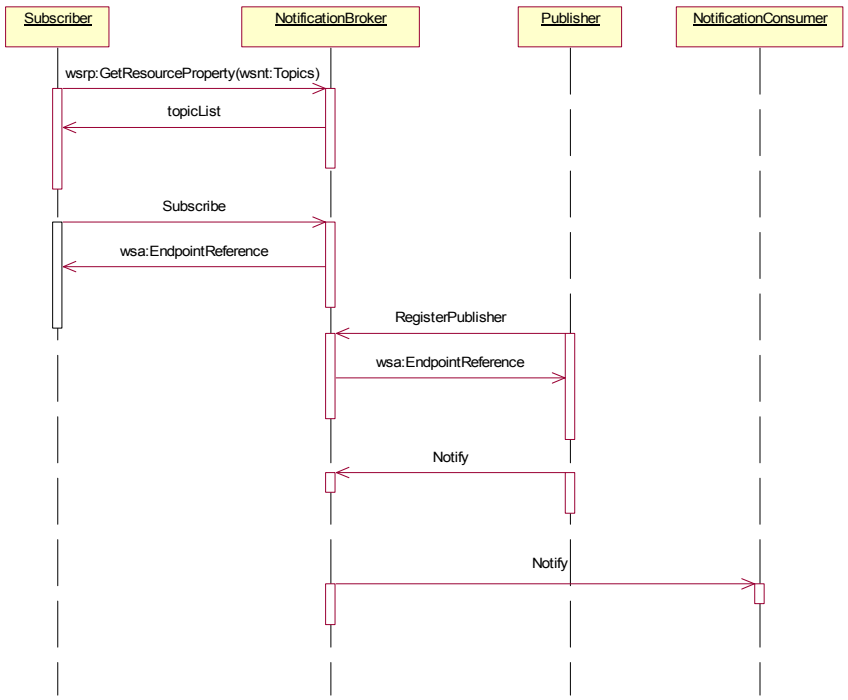
At some later time, the NotificationProducer issues a Notification that matches the Subscription. The NotificationProducer uses the Notify message to deliver this to the NotificationConsumer, or, if indicated on the Subscription, sends the NotificationMessage directly, without using the generic Notify message exchange.

Some notification patterns involve an intermediary, or broker. This arrangement is depicted in the following figure:



In the above figure, service requestor "A" subscribes the NotificationConsumer named "C" to the "SystemError" Topic provided by the NotificationBroker. Other entities (for example, publisherA) are allowed to publish msg1 onto the SystemError Topic. Some time later, another entity (publisherB) publishes msg2 onto that Topic. The NotificationBroker delivers both msg1 and msg2 to "C" as they both match the subscription that "A" established for "C".

The interactions between the Subscriber, NotificationBroker and Publisher are shown in the following interaction diagram:



In the brokered case, the sequence of message exchanges between Subscriber and the NotificationBroker is the same as the sequence of message exchanges between Subscriber and NotificationProducer in the non-brokered case.

Instead of interacting directly with the ultimate NotificationConsumers, Publishers interact with the NotificationBroker using a sequence of message exchanges supported by the broker. The case depicted above shows the publisher pre-registering with the broker by sending a RegisterPublisher message, establishing with the broker the intention of the Publisher to publish on one or more topics. The response of this message is an EndpointReference to a PublicationRegistration resource, that embodies the relationship between this Publisher and the NotificationBroker. Subsequent to this registration, the Publisher publishes a NotificationMessage to the broker, using the Notify message exchange. At some point subsequent to this publication, the broker delivers the NotificationMessages to any NotificationConsumer identified by Subscriptions which match the publication.

Note: the Notify message exchange can be used for two purposes: to publish a NotificationMessage to the broker and to deliver a NotificationMessage to a NotificationConsumer. This allows chaining of brokers and anonymous intermediation of brokers between NotificationProducers (publishers) and NotificationConsumers.

5 Publishing

There are three distinct stages in the Notification process

1. Observation of the Situation and its noteworthy characteristics;
2. Creation of the NotificationMessage artifact that captures the noteworthy characteristics of the Situation; and
3. Distribution of copies of the NotificationMessage to zero or more interested parties.

Stages 1 and 2 happen largely outside of the scope of the WS-Notification architecture; this specification does not restrict the means by which these stages must occur. We refer to an entity that performs stages 1 and 2 as a Publisher,

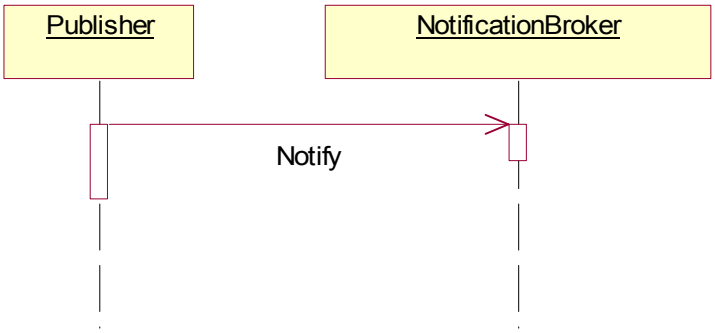
However, the WS-Notification specification does specify how dissemination of messages SHOULD occur. There are two dominant patterns by which NotificationMessages are disseminated in WS-Notification: direct and brokered.

In the direct case, the publishing Web service implements message exchanges associated with the NotificationProducer interface; it is responsible for accepting Subscribe messages and sending NotificationMessages to interested parties. The implementer of this Web service can choose to program this behavior or delegate to specialized implementations of the Subscribe and NotificationMessage delivery behavior.

In the brokered case, an intermediary - a NotificationBroker - is responsible for disseminating messages produced by one or more Publishers to zero or more NotificationConsumers.

There are three patterns associated with the relationship between the Publisher and the NotificationBroker: simple publishing, composable publishing and demand-based publishing.

The following figure illustrates simple publishing:

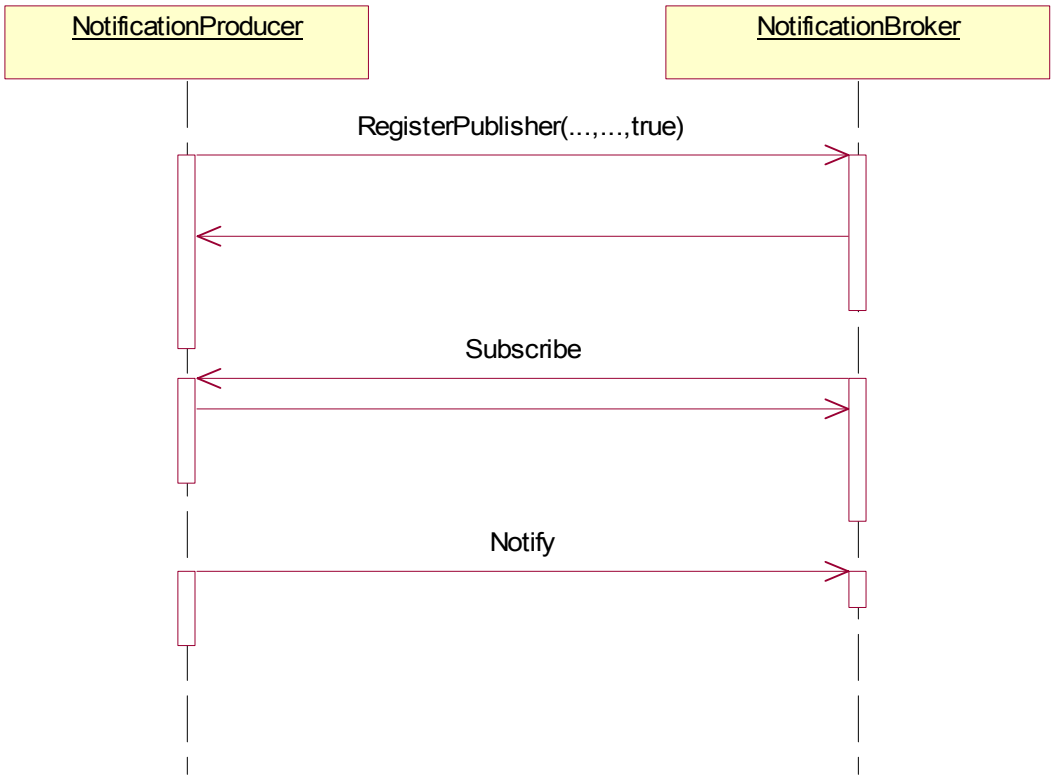


In the simple publishing scenario, the Publisher entity is responsible only for the core Publisher functions - observing the Situation and formatting the NotificationMessage artifact that describes the Situation. The dissemination step occurs when the Publisher sends the Notify message to the NotificationBroker.

In the composable publishing pattern, the role of the Publisher is played by a Web service that implements NotificationProducer. The act of observing the Situation and formatting the NotificationMessage happens within the implementation logic of the NotificationProducer itself. The NotificationMessage is disseminated by the NotificationProducer sending the Notify message to a NotificationBroker. The NotificationMessage may also be disseminated by sending the Notify message to any NotificationConsumer that had been previously subscribed to the NotificationProducer.

Note: in either of the above two cases, the NotificationBroker MAY require the Publisher to register with it prior to sending the Notify message. For example, if the broker wishes to control who can publish to a given Topic, it can perform an access control check during this registration. However a NotificationBroker MAY choose to allow Publishers to publish without pre-registration if it so chooses.

The last pattern, the *demand-based* pattern, requires the Publisher to be a NotificationProducer, and thereby accept the Subscribe message. Demand-based publication is intended for use in cases where the act of observing the Situation or the act of formatting the NotificationMessage artifact might be expensive to perform, and therefore should be avoided if there are no interested parties for that NotificationMessage. To use this pattern, the Publisher must register with the NotificationBroker, using the registration to express the intent to provide demand-based publishing only. Based upon this style of registration, the NotificationBroker sends the Subscribe message to the Publisher (recall: in this situation the Publisher must implement the message exchanges associated with the NotificationProducer interface).



Furthermore, the NotificationBroker is expected to pause its Subscription whenever it has no active Subscribers for the information provided by the Publisher. When the NotificationBroker does have active Subscribers, it is obliged to resume its Subscription to the Publisher.

6 Topics

This chapter expands on the concept of a Topic, discusses how WS-Notification uses XML to describe a Topic's metadata, and defines an XML path expression language to refer to Topics.

6.1 Topics and Topic Spaces

As part of the publication of a NotificationMessage, the Publisher associates it with one or more Topics. When a Subscriber creates a Subscription, it associates the Subscription with one or more Topics. The NotificationProducer uses these Topic lists as part of the matching process: a NotificationMessage is delivered to a NotificationConsumer if the list of Topics associated with the Subscription has a non-empty intersection with the list of Topics associated with the NotificationMessage.

In order to avoid naming collisions, and to facilitate interoperation between independently developed NotificationProducers and Subscribers, every WS-Notification Topic is assigned to an XML Namespace. The set of Topics associated with a given XML Namespace is termed a *Topic Space*. Any XML namespace has the potential to scope a single collection of Topics. Of course, not every XML namespace will define a Topic Space.

Each Topic can have zero or more *child topics* and a child topic can itself contain further child topics. A Topic without a *parent* is termed a *root topic*. A particular root topic and all its descendents form a hierarchy (termed a *Topic Tree*).

The rationale for hierarchical topic structures is:

- They allow Subscribers to subscribe against multiple Topics. For example a Subscriber can subscribe against an entire Topic Tree, or a subset of the Topics in a Topic Tree. This reduces the number of subscription requests that a Subscriber needs to issue if it is interested in a large sub-tree. It also means that a Subscriber can receive NotificationMessages related to descendent topics without having to be specifically aware of their existence.
- They provide a convenient way to manage large Topic Spaces (for example when administering security policies).

Note: Although WS-Notification permits hierarchical topic structures, there is no requirement or expectation that all Topic Spaces will contain them. It is perfectly possible for a Topic Space to contain only root topics (possibly only a single root topic). A NotificationProducer is not required to support structured topics. It may support only topics from Topic Spaces that contain only root Topics; even if it does support topics from a Topic Space that contains topic hierarchies, it may choose only to support root topics from that topic space.

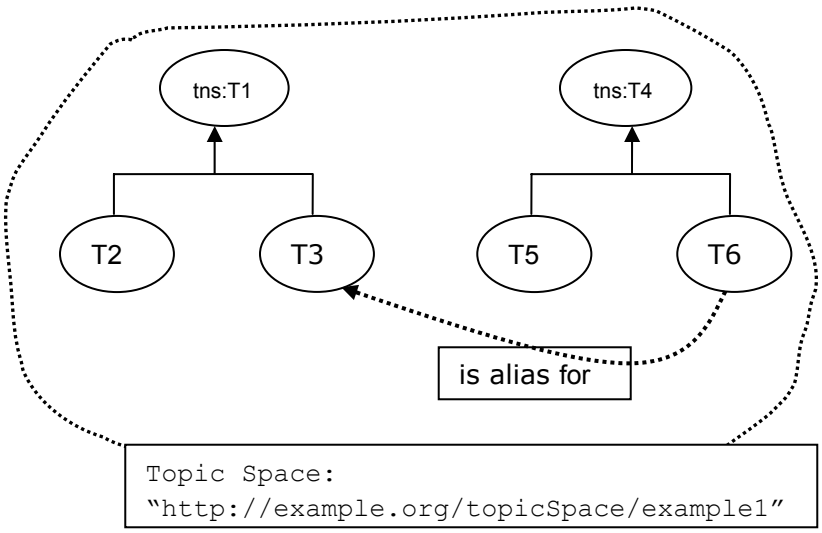
A Topic Space is thus a collection (forest) of Topic Trees. The Topic Space contains additional metadata relating to its member Topics. The metadata describing a particular Topic Space can be modeled as an XML document (see section 6.3).

Each Topic has a local name, an NCName. All root topics must have unique names within their Topic Space. In this way, a root Topic can be uniquely referenced by a QName formed by combining the XML Namespace associated with the Topic Space and the local name of the root topic. Child topics can only be referred to relative to their ancestor root topic's QName using a TopicPathExpression (see section 6.5).

No Topic can contain two immediate child topics with the same name, however Topics with the same name can appear elsewhere in a Topic Tree, and no relationship is implied. Similarly two separate Topic Trees in the same Topic Space may contain descendent Topics with the same name; these are not necessarily related to each other in any way either.

6.2 Example Topic Space

Consider a Topic Space that can be depicted as follows. The Topic Space is contained in the "http://example.org/topicSpace/example1" namespace. This Topic Space has two root Topics, named t1 and t4. Topic t1 has two child topics, t2 and t3. Topic t4 has two child topics, t5 and t6. Topic t6 is an alias for t1's child topic t3.



This topic space and its metadata can be described using the following XML instance document:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsnt:topicSpace name="TopicSpaceExample1"
   targetNamespace="http://example.org/topicSpace/example1"
   xmlns:tns="http://example.org/topicSpace/example1"
   xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification" >
   <wsnt:topic name="t1">
      <wsnt:topic name="t2" messageTypes="tns:m1 tns:m2"/>
      <wsnt:topic name="t3" messageTypes="tns:m3"/>
   </wsnt:topic>
   <wsnt:topic name="t4">
      <wsnt:topic name="t5" messageTypes="tns:m3"/>
      <wsnt:topic name="t6" aliasRef="tns:t1/t3"/>
   </wsnt:topic>
</wsnt:topicSpace>
```

We describe the details behind modeling topic spaces and topics in the following sections.

6.3 Modeling Topic Spaces in XML

The WS-Notification XML Schema contains element and type definitions used to create topic space instance documents. An instance document is associated with a single Topic Space and contains the names of Topics in that Topic Space along with their metadata. It may include all the topics in that topic space, or just a subset of them. The following is a non-normative description of a TopicSpace element:

```
...
<TopicSpace name=NCName? targetNamespace=anyURI …>
  <Topic … />*
…
</TopicSpace>
```

A TopicSpace element is further constrained in the following way:

/wsnt:TopicSpace

The top-level element in a topic space instance document. It contains Topic declaration elements and associates them with the XML Namespace for the topic space

/wsnt:TopicSpace/@name

An optional name that can be assigned to the TopicSpace element for light-weight documentation purposes.

/wsnt:TopicSpace/@targetNameSpace

The XML Namespace for this topic space. It is expressed as a URI. This forms the namespace component of the QName of each root Topic in the Topic Space.

/wsnt:topicSpace/Topic

The TopicSpace has a collection of zero or more child Topic elements that define the roots of the Topic Trees within the Topic Space. The TopicSpace element may contain any number of Topic elements. The value of /Topic/@name MUST be unique amongst all root Topics defined in the TopicSpace.

/wsnt:TopicSpace/{any}

This is an extensibility mechanism to allow additional elements to be specified.

/wsnt:TopicSpace/@{any}

This is an extensibility mechanism to allow additional attributes to be specified.

6.4 Modeling Topics in XML

WS-Notification defines an XML representation of a Topic that can be represented in the following non-normative fashion::

```
<TopicSpace name=… targetNamespace=…>
  <Topic name=NCName messageTypes=list of QName? final=boolean?
     aliasRef=TopicPathExpression? >

      <MessagePattern>wsrp:QueryExpression</MessagePattern>?
      <Topic … />*
…
  </Topic>
</TopicSpace>
```

A Topic element is further constrained in the following way:

/wsnt:Topic/@namespace

The namespace of a Topic is defined as the targetNamespace of the TopicSpace element ancestor of the Topic. As we saw in section 6.3,

individual root topics are modeled by defining Topic child elements of the TopicSpace element.

/wsnt:Topic/@name

The NCName of this topic. This attribute is required. These NCNames must all be unique with respect to the parent element (TopicSpace or Topic) that contains this Topic. In the case of a root Topic, the @namespace and @name attributes combine to form the QName of the root Topic.

/wsnt:Topic/@messageTypes

An optional list of the QNames of XML elements that define the types of NotificationMessage that may be used with the Topic. A Publisher using a given Topic MUST NOT generate a NotificationMessage whose type is not included in this list, although the special value xsd:any indicates that any NotificatonMessage type MAY be used. A given QName MAY appear multiple times in the list; second or subsequent appearance of a given QName MUST BE ignored. If this list is empty, or the attribute not defined, the default value of "xsd:any" is assumed.

/wsnt:Topic/@aliasRef

An optional attribute that indicates that this Topic definition is an alias for another Topic (or combination of Topics). This mechanism can be used to permit alternative spellings of a given Topic name, or to allow a Topic (sub)tree from one Topic Space to be imported into a Topic definition in another Topic Space. The value of @aliasRef is itself a TopicPathExpression that may resolve to multiple Topics, including further aliases (even possibly itself). Publishing or subscribing using a Topic which is an alias is equivalent to publishing or subscribing to all the non-alias Topics which result from the process of alias resolution. This process is described in greater detail in the next section. A Topic containing an aliasRef attribute MUST NOT contain any child Topic elements. A Topic containing an aliasRef attribute MAY contain @messageTypes, MessagePattern, or @final– however if it does their values MUST BE ignored.

/wsnt:Topic/@final

An optional attribute whose value is of type xsd:boolean. The default value is "false". If the value is "true" it indicates that the NotificationProducer cannot dynamically add child Topics to this Topic. This means that it is an error if a Publisher or Subscriber attempts to use a TopicPathExpression that references child Topics of a Topic that is marked as @final="true" – other than child Topics that are explicitly included in the definition of the Topic.

/wsnt:Topic/MessagePattern

An optional QueryExpression as defined by WS-ResourceProperties. This QueryExpression is used to describe the pattern of the message that will appear on the Topic. Conceptually, the MessagePattern component can be thought of as the object of an boolean() expression, evaluated against a NotificationMessage. This boolean() expression, with the value of MessagePattern as parameter, is guaranteed to evaluate to "true" when evaluated in the context of any NotificationMessage that is associated with the Topic. The MessagePattern component constrains the NotificationMessages that can be used with the Topic. It is additional to the

constraint contained in @messageTypes, and provides a further refinement to that constraint.

/wsnt:Topic/MessagePattern/@dialect

A URI that identifies the language of the QueryExpression. WS-ResourceProperties defines standard URIs for XPath 1.0 and XPath 2.0 languages. Designers MAY define and use other domain-specific URIs to identify the dialect of the QueryExpression.

/wsnt:Topic/Topic

Declares a child Topic. A Topic may contain any number of child Topic elements; however the value of the @name attribute of a child Topic must be unique amongst all the child Topics of its immediate parent.

/wsnt:Topic/{any}

This is an extensibility mechanism to allow additional elements to be specified.

/wsnt:Topic/@{any}

This is an extensibility mechanism to allow additional attributes to be specified.

6.5 Topic Path Expressions

Topics are referred to by TopicPathExpressions. There are several places in WS-Notification where these expressions are used:

1. As a component of the Subscribe message request to a NotificationProducer;
2. As a component of the Notify message to a NotificationConsumer or NotificationBroker;
3. In the Topics Resource Property element associated with the NotificationProducer role
4. In the aliasRef attribute of a Topic element.

In some of the places a full TopicPathExpression is permitted, in others it is only a subset called a ConcreteTopicPath (see the end of this section for details).

TopicPathExpressions are like[1] XPath [XPath] expressions that resolve to node-set objects.

The simplest useful TopicPathExpression is simply the QName of a root topic. For example:

```
"tns:t1"
```

This TopicPathExpression identifies the root Topic t1 within the Topic Space corresponding to the namespace prefix tns.

The TopicPathExpression to identify the Topic named "t3", child of Topic tns:t1 is as follows:

```
"tns:t1/t3"
```

[1] As an aside, the authors considered trying to rework the topic space document to allow topic path expressions to be XPath expressions, but this yielded very unnatural XML Documents.

Just as with XPath, the TopicPathExpression syntax uses the slash ("/") to describe *child of*. The equivalent XPath expression for this would be:

```
"/TopicSpace/Topic[@name="t1"]/Topic[@name="t3"]"
```

It is hopefully clear why straight XPath was not appropriate for TopicPathExpressions.

The wildcard character * is used to identify a node-set consisting of a collection of child Topics. For example

```
"tns:t1/*"
```

This TopicPathExpression identifies all the child Topics of the root Topic t1. Note that this TopicPathExpression does not include the root Topic t1 itself, and it does not include any grandchildren or further descendents of t1.

The special syntax "//*" is used to identify a set of Topics consisting of all descendents (not just immediate children). If included at the end of a TopicPathExpression it indicates that the TopicPathExpression matches a Topic subtree. For example:

```
"tns:t1/t3//*"
```

This identifies tns:t1/t3 itself and all its descendent Topics.

To include the root Topic with the entire Topic Tree, the following TopicPathExpression would be used:

```
"tns:t1//*"
```

Wildcard characters may be interspersed with fixed child Topic names, to build up longer paths, for example:

```
"tns:t1/*/t3/*"
```

This TopicPathExpression identifies all grandchildren of tns:t1 that have the name t3.

The //* wildcard can also be used in the middle of a TopicPathExpression, for example

```
"tns:t1//*/t3"
```

This TopicPathExpression identifies all descendents of tns:t1 that have the name t3.

A TopicPathExpression MAY contain two or more wildcards (both * and //*).

TopicPathExpression may be combined together with the conjunction operator as follows:

```
 "tns:t1/t2 | tns:t4/t5"
```

A TopicPathExpression using | can include root Topics from different TopicSpaces. Note: a TopicPathExpression containing a conjunction operator is equivalent to the set union of the Topics described by combining the TopicPathExpression on either side of the conjunction operator.

A **concreteTopicPath** is a TopicPathExpression that contains no wildcards and no '|' operators; it is a TopicPathExpression that identifies one Topic within a Topic Space.

6.5.1 Formal Definition of the TopicPathExpression

A TopicPathExpression is a token (as defined by XML Schema) with an additional constraint on its format.

The constraint is the string must contain a TopicPathExpression. The representation for the TopicPathExpression grammar uses the simple Extended Backus Naur Form (EBNF) also used in [XML].

```
[1] TopicPathExpression ::=  TopicPath | TopicExpression
[2] TopicExpression     ::=  TopicPathExpression Conjunction
                             TopicPathExpression
[3] Conjunction         ::=  '|'
[4] TopicPath           ::=  RootTopic ChildTopicExpression*
[5] RootTopic           ::=  QName | '*'
    [ vc: If a namespace is included in the RootTopic, it must correspond to a
    valid Topic Space Document and the local name must correspond to the name
    of a root topic defined in that namespace.]
[6] ChildTopicExpression ::= '/' (ChildTopicName | '*' | '/*')
[7] ChildTopicName      ::=  NCName
    [ vc: The NCName must correspond to the name of a topic within the
    descendant path from the RootTopic, where each forward slash denotes
    another level of child topic elements in the path.]
```

Note: White space is not permitted within a TopicExpression.

6.5.2 AliasRefs and their resolution

The aliasRef attribute is optional attribute on a Topic element that indicates that the Topic is an alias for another Topic (or combination of topics). This mechanism can be used to permit alternative spellings of a given Topic name, or to allow a Topic (sub)tree from one TopicSpace to be *imported* into a Topic definition in another Topic Space. In this example Topic t6 is defined as an alias for tns:t1/t3

```
<wsnt:Topic name="t6" aliasRef="tns:t1/t3"/>
```

Syntactically an aliasRef is permitted to be any TopicPathExpression, including a wild cards or '|' operators. This means that an aliasRef might reference another alias Topic definition, or might be a wild card expression that includes a mixture of alias and non-alias definitions.

An alias is resolved into a set of zero or more non-aliased TopicPathExpressions using the following rules:

1. If the aliasRef is a concrete TopicPathExpression that is not an alias, then the alias resolves to that concreteTopicPath.
2. If the aliasRef is a concreteTopicPath that is itself an alias, then resolution proceeds recursively from this alias.
3. If the aliasRef is a TopicPathExpression with *, //* but no '|' operators, then the alias resolves to this concreteTopicPath (no deeper examination of aliases is performed in this case).
4. If a TopicPathExpression contains '|' operators, then each component of the expression is treated separately and creates a new resolution branch. Resolution proceeds on each branch individually, using rules 1,2,3, the resolution of each branch is aggregated to the resolved TopicPathExpression set.
5. If a circular reference is encountered (aliasRef pointing directly or indirectly back to itself) then the branch in question contributes nothing to the resolved TopicPathExpression set.

If a TopicPathExpression (full or concrete) is supplied as a parameter on a message exchange defined by WS-Notification, it is subjected to the alias resolution process described above. The resulting resolved TopicPathExpression set is then used in place of the original parameter. If the resolved TopicPathExpression set is empty, the operation fails. If the operation required a concrete topicPath and the resolved set contains multiple topics, or contains wild card topic expressions, then the operation fails.

6.6 Growing a Topic Tree

If a Topic in the TopicSpace is marked with the final attribute, with value="true", then no further child Topics can be added dynamically to that Topic.

If a Topic is not marked with the final attribute with value="true", then a NotificationProducer could potentially add further child Topics to that Topic, and permit Subscriptions to such child Topics. This specification does not define the circumstances under which this occurs, and it is up to the NotificationProducer to determine if and when it permits additional children (it is not obligated to allow children to be added just because a Topic may be marked with final="false").

When a NotificationProducer accepts Topics that are not previously defined in the TopicSpace, it is not obliged to update any actual instance document that contains the TopicSpace definition. Rather, the extension exists only for that NotificationProducer and any Subscriber that interacts with it. Circumstances under which a NotificationProducer MAY add new child Topics to a Topic include:

- a Subscriber attempting to subscribe to a TopicPathExpression that suggests a new child Topic;
- a Publisher attempting to publish to a TopicPathExpression that suggests a new child Topic;
- the NotificationProducer implementation encountering a new circumstance that doesn't fit well with any of the existing child Topics (for example a new company starts trading on a stock market, and a stock ticker service wishes to include it);
- an administrator explicitly adding support for a new child Topic using some administrative portType (not defined by WS-Notification) implemented by the NotificationProducer.

6.7 The "ad-hoc" topic space

Associating a TopicSpace with an XML namespace provides an unambiguous naming scheme for Topics. This is important when two entities which have no prior knowledge of each other attempt (for example a Subscriber which has just discovered a NotificationBroker) to interact.

However there are circumstances where someone wishes to implement a Publisher for which there is no suitable pre-existing TopicSpace – and where the implementer does not wish to incur the overhead of creating a new TopicSpace (assigning a unique namespace, and creating the TopicSpace element within some XML instance document).

To help such users, WS-Notification defines a special built-in TopicSpace called the *ad-hoc* TopicSpace.

The ad-hoc TopicSpace has no pre-defined root Topics, but allows new root Topics to be added dynamically (in the same way that a non-final Topic allows new child Topics to be added to it). Any Topic that is added dynamically to the ad-hoc TopicSpace itself permits the addition of further child Topics, and allows any type of NotificationMessage element to be associated with it. There is no concept of Topic aliasing in the ad-hoc TopicSpace.

The ad-hoc TopicSpace is defined by the following namespace URI (http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification/adHoc) and is accessed using TopicPathExpressions that reference this namespace.

A NotificationProducer or Subscriber can use this TopicSpace to define *ad-hoc Topics* dynamically, without having to associate them with their own TopicSpace. Caution should be used when employing ad-hoc Topics, as there is no way for a NotificationConsumer to distinguish between it and other similarly-named ad-hoc Topics supported by any number of NotificationProducers.

6.8 NotificationProducer and Topics

A NotificationProducer uses Topics to group NotificationMessages related to some Situation. A NotificationProducer can support one or more Topics, from multiple TopicSpaces. A NotificationProducer can support an entire Topic Tree, or just a subset of the Topics in that Topic Tree. The set of Topics currently supported by a NotificationProducer can be determined by accessing the Topics Resource Property element (see section 8.1). This Resource Property contains the set of Topics that the NotificationProducer expects to handle.

The list of Topics supported by the NotificationProducer MAY change over time. Reasons for the set of Topics changing include:

- The NotificationProducer supporting additional Topics from a TopicSpace that is already partially supported;
- The NotificationProducer supporting additional Topics from a TopicSpace not previously supported;
- The NotificationProducer supporting extension Topics to a (new or already supported) TopicSpace, as discussed in the previous section;
- The NotificationProducer ceasing to support Topics previously listed.

This specification does not require a NotificationProducer to support any or all of the types of changes just listed, and does not dictate the set of conditions under which the list of supported Topics will change.

7 NotificationConsumer Interface

WS-Notification allows a NotificationProducer to send a NotificationMessage to a NotificationConsumer in one of two ways:

1. The NotificationProducer MAY simply send the "raw" NotificationMessage (i.e. the application-specific content) to the NotificationConsumer.
2. The NotificationProducer MAY send the NotificationMessage data using the Notify message described below

The second option (the Notify message) allows the NotificationProducer to supply additional WS-Notification-defined information (such as the Topic) in addition to the application-specific NotificationMessage content. It also means that the

NotificationConsumer can receive a wide range of NotificationMessages without having to declare support for each one in its WSDL portType. Furthermore, this form of Notification allows a batch of multiple NotificationMessages to be delivered using a single message.

When a Subscriber sends a Subscribe request message, it indicates which kind of Notification is required (the raw NotificationMessage, or the Notify Message). The NotificationProducer MUST observe this Subscription parameter, and use the form that has been requested.

This means that a NotificationConsumer MAY choose to implement the Notify Message, or to implement each NotificationMessage explicitly in its WSDL (or both). When requesting creation of a Subscription on behalf of a NotificationConsumer, a Subscriber SHOULD ensure that the NotificationConsumer is able to handle the kind of Notification it has requested for the given Subscription.

7.1 Notify

A NotificationProducer MAY send the Notify message to the NotificationConsumer in order to deliver one or more NotificationMessage(s). The format of the Notify message is:

```
…
<wsnt:Notify>
   <wsnt:NotificationMessage>
      <wsnt:Topic>wsnt:ConcreteTopicPathExpression</wsnt:Topic>
      <wsnt:ProducerReference>?
         wsa:EndpointReference
      </wsnt:ProducerReference>
      <wsnt:Message>xsd:any</wsnt:Message>
   <wsnt:NotificationMessage>+
</wsnt:Notify>
…
```

The components of the Notify message are further described as follows:

/wsnt:Notify

Contains a collection of one or more NotificationMessages to be delivered. This message MAY follow the implied resource pattern as outlined in [WS-Resource].

/wsnt:Notify/wsnt:NotificationMessage

Contains a NotificationMessage to be delivered.

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

The concrete TopicPathExpression describing the Topic associated with this NotificationMessage. This element describes the topic(s) that matched to a subscription, causing the NotificationProducer to send the Notify message to the NotificationConsumer.

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

An OPTIONAL EndpointReference to the NotificationProducer that originally generated the NotificationMessage artifact.

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

A copy of the actual NotificationMessage payload.

No response is expected from the NotificationConsumer upon receipt of this message.

7.1.1 Example SOAP Encoding of the Notify Message

The following is a non-normative example of a Notify message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:ncex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
   http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification/Notify
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.consumer.org/ConsumerEndpoint
    </wsa:To>
    <ncex:NCResourceId>
        uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
    </ncex:NCResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:Notify>
      <wsnt:NotificationMessage>
        <wsnt:Topic>wsrp:ResourcePropertiesValueChanges</wsnt:Topic>
          <wsnt:ProducerReference
                xmlns:npex="http://www.producer.org/RefProp">
            <wsa:Address>
                http://www.producer.org/ProducerEndpoint
            </wsa:Address>
            <wsa:ReferenceProperties>
              <npex:NPResourceId>
                  uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
              </npex:NPResourceId>
            </wsa:ReferenceProperties>
          </wsnt:ProducerReference>
        <wsnt:Message>
          <npex:NotifyContent>exampleNotifyContent</npex:NotifyContent>
        </wsnt:Message>
      <wsnt:NotificationMessage>
    </wsnt:Notify>
  </s12:Body>
</s12:Envelope>
```

Note the use of the wsa:ReferenceProperties elements in the message. These are examples of WS-Resource qualified EndpointReferences, following the implied resource pattern.

8 NotificationProducer Interface

This section describes the message exchanges and resource properties that a NotificationProducer MUST support. These message exchanges allow the NotificationProducer to advertise its support for one or more Topics, and allow a Subscriber to create Subscriptions or to control the delivery of NotificationMessages by the NotificationProducer.

8.1 NotificationProducer Resource Properties

In addition to the message exchanges described in this specification, a NotificationProducer MUST also support the required message exchanges defined in the WS-ResourceProperties specification and MAY support the optional message exchanges defined in the WS-ResourceProperties specification.

The Resource Properties document defined by the NotificationProducer MUST include a reference to the following resource property elements:

```
…
   targetNamespace=
      ”http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification”
…

  <xsd:element name=”Topics”>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name=”Topic” type=”wsnt:TopicPathExpression”
                     minOccurs=”0” maxOccurs=”unbounded” />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name=”FixedTopicSet” type=”xsd:boolean”/>

…
```

These Resource Property elements are further constrained as follows:

/wsnt:Topics

The collection of Topics supported by the NotificationProducer.

/wsnt:Topics/Topic

If a Topic appears as a child of the Topics element, a Subscriber can reasonably expect the NotificationProducer to deliver NotificationMessages on this Topic. It is not a guarantee that it will receive any NotificationMessages; the NotificationProducer may not actually produce any NotificationMessages on the particular Topic during the time that the Subscriber is registered.

Inclusion of a given Topic under the Topics element does not imply that any child topic of that given Topic is also supported. In order to indicate a Topic and all its descendent topics the TopicPathExpression MUST use the “//*” construct.

/wsnt:FixedTopicSet

Indicates if the collection of Topics contained within the Topics Resource Property may change. This value is “true” if the collection of Topics supported by the NotificationProducer does not change and “false” if the NotificationProducer allows the collection to change (for example if it allows additional Topics to be supported should publishers or subscribers request them).

8.2 Subscribe

A Subscriber sends a Subscribe message to a NotificationProducer in order to register the interest of a NotificationConsumer for NotificationMessages related to one or more Topics.

As part of the processing of a Subscribe request message, the NotificationProducer MUST create a Subscription resource representing the Subscription. It returns an EndpointReference in the response to the Subscribe request. This EndpointReference is a WS-Resource-Qualified EndpointReference as defined in [WS-Resource] and includes the address of a SubscriptionManager service and a reference property containing the identity of the Subscription resource.

Sending two identical Subscribe request messages to a NotificationProducer will result in the creation of two Subscription resources. The NotificationConsumer will be associated with both Subscription resources and so will receive two copies of any matching NotificationMessage.

The format of the Subscribe message is:

```
…
<wsnt:SubscribeRequest>
   <wsnt:ConsumerReference>
      wsa:endpointReference
   </wsnt: ConsumerReference>
   <wsnt:TopicPathExpression />
   <wsnt:UseNotify> xsd:boolean </wsnt:UseNotify>?
   <wsnt:Precondition> wsrp:QueryExpression </Precondition>?
   <wsnt:Selector> wsrp:QueryExpression </wsnt:Selector>?
   <wsnt:SubscriptionPolicy> wsp:Policy </wsnt:SubscriptionPolicy>?
   <wsrl:InitialTerminationTime>
      xsd:dateTime
   </wsrl:InitialTerminationTime>?
</wsnt: SubscribeRequest>
…
```

This request message MUST follow the implied resource pattern as outlined in [WS-Resource]. The components of the Subscribe request message are further described as follows:

/wsnt:ConsumerReference

An endpoint reference element from WS-Addressing [WS-Addressing], used to identify the NotificationConsumer.

The Web service identified by the endpoint reference SHOULD implement the message exchanges defined by NotificationConsumer (e.g. the Notify message).

/wsnt:ConsumerReference/wsa:ReferenceProperties

The ReferenceProperties MAY contain an identifier of the resource to which the NotificationMessage should be delivered. This is used in the situation where the NotificationConsumer itself is implementing the implied resource pattern [WS-Resource].

The ReferenceProperties MAY contain other elements useful in the NotificationMessage delivery. For example the ReferenceProperties could contain a "correlationContext" - an element that is returned as part of each NotificationMessage delivered to the NotificationConsumer.

/wsnt:TopicPathExpression

A TopicPathExpression that identifies one or more Topics supported by the NotificationProducer.

/wsnt:UseNotify

A boolean value that indicates how the NotificationProducer is to deliver NotificationMessages to the NotificationSource. This component is optional, if missing the default value is "true".

If this value is "true" then the NoticationProducer MUST use the wsnt:Notify message to deliver NotificationMessages to the NotificationConsumer. In this case the NotificationConsumer referred to in the wsnt:ConsumerReference element MUST implement the Notify message and include a Notify operation in its portType definition.

If this value is "false" then the NoticationProducer MUST deliver the NotificationMessage using the raw NotificationMessage itself. In this case the NotificationConsumer referred to in the wsnt:ConsumerReference element MUST implement all the NotificationMessages associated with the Topic or Topics referred to by the TopicPathExpression, and include corresponding operations in its portType definition.

/wsnt:Precondition

This component contains a QueryExpression as defined by WS-ResourceProperties.

This component expresses a NotificationProducer-specific precondition associated with generation of NotificationMessages by the NotificationProducer. A NotificationProducer object is free to define the syntax and meaning of this expression

Non-normative examples of a possible precondition include:

- An XPath expression on the Resource Properties of the NotificationProducer that evaluates to a Boolean. The NotificationProducer must only deliver NotificationMessages to the NotificationConsumer if this expression evaluates to "true".
- A database query operation to be performed in order to generate a NotificationMessage, or periodic sequence of NotificationMessages.

/wsnt:Precondition/@dialect

A URI that identifies the language of the QueryExpression. WS-ResourceProperties defines standard URIs for XPath 1.0 and XPath 2.0 languages. Designers MAY define and use other domain-specific URIs to identify the dialect of the QueryExpression.

/wsnt:Selector

A QueryExpression as defined by WS-ResourceProperties. A wsnt:Selector expression MUST evaluates to a Boolean. The evaluation context is NotificationMessage(s) that are Published on the Topic(s) identified by the TopicPathExpression defined by the wsnt:TopicPathExpression component of this message.

The NotificationProducer must only deliver NotificationMessages to the NotificationConsumer if this expression evaluates to "true".

/wsnt:Selector/@dialect

A URI that identifies the language of the QueryExpression. WS-ResourceProperties defines standard URIs for XPath 1.0 and XPath 2.0 languages. Designers MAY define and use other domain-specific URIs to identify the dialect of the QueryExpression.

/wsnt:SubscriptionPolicy

A WS-Policy element indicating the requested policies to apply to the Subscription. This mechanism could be used to govern the message rate (e.g. maximum 3 messages per second), reliability of the NotificationMessage delivery, etc.

If this component is not specified in the Subscribe request message, then the NotificationProducer SHOULD use other means (such as directly contacting the NotificationConsumer) to resolve any policy-related inquiries.

/wsrl:InitialTerminationTime

The generic (not application specific) semantics of this component are described in [WS-ResourceLifetime]. This component contains the service requestor's suggestion for the initial termination time of the Subscription being created. This time is relative to the time source used by the NotificationProducer. If the NotificationProducer is unable or unwilling to set the TerminationTime to the given value or greater, then the Subscribe request MUST fault. If the value is not "in the future" relative to the current time as known by the NotificationProducer, the Subscribe request MUST fault. The use of the xsi:nil attribute with value "true" indicates there is no scheduled termination time requested for the Subscription. If the element does not include the time zone designation, the value of the element MUST be interpreted as universal time (UTC) time.

The response to the Subscribe request message is a message of the following form:

```
…
<wsnt:SubscribeResponse>
   <wsnt:SubscriptionReference>
      <wsa:Address> Address of Subscription Manager </wsa:Address>
      <wsa:ReferenceProperties>
```

```
          Subscription Identifier
      </wsa:ReferenceProperties>
      …
   </wsnt:SubscriptionReference>
…
</wsnt:SubscribeResponse>
…
```

The contents of the SubscribeResponse message are further described as follows:

/wsnt:SubscriptionReference/wsa:Address

The address of a Web service that implements the SubscriptionManager set of messages. This SubscriptionManager is associated with the Subscription resource created by the Subscribe message following the implied resource pattern.

/wsnt:SubscriptionReference/wsa:ReferenceProperties

An identifier of the Subscription resource following to the implied resource pattern.

Instead of the SubscribeResponse message, the NotificationProducer may also send the following faults in response to a Subscribe message:

- Unable to create subscription
- Others, TBD

8.2.1 Example SOAP Encoding of the Subscribe Message Exchange

The following is a non-normative example of a Subscribe request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/SubscribeRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/ProducerEndpoint
    </wsa:To>
    <npex:NPResourceId>
         uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
    </npex:NPResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:SubscribeRequest>
      <wsnt:ConsumerReference
            xmlns:ncex="http://www.consumer.org/RefProp">
```

```
        <wsa:Address>
           http://www.producer.org/ProducerEndpoint
        </wsa:Address>
        <wsa:ReferenceProperties>
           <ncex:NCResourceId>
              uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
           </ncex:NCResourceId>
        </wsa:ReferenceProperties>
      </wsnt:ConsumerReference>
      <wsnt:TopicPathExpression>
        wsrp:ResourcePropertiesValueChanges
      </wsnt:TopicPathExpression>
      <wsnt:UseNotify>true</wsnt:UseNotify>
      <wsnt:Precondition
           dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
        boolean(ncex:NumberOfBlocks > 20 and ncex:BlockSize=1024)
      </Precondition>
      <wsnt:Selector
            dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
          boolean(ncex:Producer="15")
      </wsnt:Selector>
      <wsrl:InitialTerminationTime>
         2003-12-25T00:00:00.00000Z
      </wsrl:InitialTerminationTime>
    </wsnt: SubscribeRequest>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a Subscribe response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:ncex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/SubscribeResponse
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.consumer.org/ConsumerEndpoint
    </wsa:To>
    <ncex:NCResourceId>
        uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
    </ncex:NCResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:SubscribeResponse>
      <wsnt:SubscriptionReference
            xmlns:npex="http://www.producer.org/RefProp">
        <wsa:Address>
```

```
          http://www.producer.org/SubsMgrEndpoint
        </wsa:Address>
        <wsa:ReferenceProperties>
          <npex:NSResourceId>
              uuid:8fefcd11-7d3d-66b344a2-ca44-9876bacd44e9
          </npex:NSResourceId>
        </wsa:ReferenceProperties>
      </wsnt:SubscriptionReference>
    </wsnt:SubscribeResponse>
  </s12:Body>
</s12:Envelope>
```

8.3 GetCurrentMessage

In response to a GetCurrentMessage message, the NotificationProducer MUST return the last NotificationMessage published to a given Topic. This is a non-destructive read, and allows a newly-subscribed NotificationConsumer to get the last NotificationMessage that other NotificationConsumers have received.

If the Topic is paused when this request is received, the response message contains the last NotificationMessage (if any) that was published prior to the pause taking effect.

The format of the GetCurrentMessage request message is:

```
…
<wsnt:GetCurrentMessageRequest>
   <wsnt:Topic>wsnt:ConcreteTopicPathExpression</wsnt:Topic>
</wsnt:GetCurrentMessageRequest>
…
```

This request message MUST follow the implied resource pattern as outlined in [WS-Resource]. The components of the GetCurrentMessage request message are further described as follows:

/wsnt:Topic

The concrete TopicPathExpression identifying a Topic.

The response of the GetCurrentMessage request message is a message of the following form:

```
…
<wsnt:GetCurrentMessageResponse>
   xsd:any
</wsnt:GetCurrentMessageResponse>
…
```

The contents of the GetCurrentMessage response message are further described as follows:

/wsnt:GetCurrentMessageResponse/{any}

Contains the last NotificationMessage associated with the Topic identified by the request message.

Instead of the GetCurrentMessageResponse message, the NotificationProducer may also send the following faults in response to a GetCurrentMessage message:

- Invalid TopicPathExpression
- Given TopicPathExpression did not match any Topic supported by the NotificationProducer
- The Topic is supported but there is no NotificationMessage currently available on that Topic
- Others, TBD

8.3.1 Example SOAP Encoding of the GetCurrentMessage Message Exchange

The following is a non-normative example of a GetCurrentMessage request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/GetCurrentMessageRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/ProducerEndpoint
    </wsa:To>
    <npex:NPResourceId>
        uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
    </npex:NPResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:GetCurrentMessageRequest>
      <wsnt:Topic>wsrp:ResourcePropertiesValueChanges</wsnt:Topic>
    </wsnt:GetCurrentMessageRequest>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a GetCurrentMessage response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
       "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/GetCurrentMessageResponse
```

```
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/ProducerEndpoint
    </wsa:To>
    <npex:NPResourceId>
        uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
    </npex:NPResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:GetCurrentMessageResponse>
       <npex:NotifyContent>exampleNotifyContent</npex:NotifyContent>
    </wsnt:GetCurrentMessageResponse>
  </s12:Body>
</s12:Envelope>
```

8.4 Pause

If a requestor wishes to temporarily suspend the delivery of NotificationMessages by the NotificationProducer, it MUST send a Pause request message. The Pause request message has the following form:

```
…
<wsnt:PauseRequest>
   <wsnt:TopicPathExpression>?
</wsnt:PauseRequest>
…
```

This request message MUST follow the implied resource pattern as outlined in [WS-Resource]. The contents of the Pause request message are further described as follows:

/wsnt:TopicPathExpression

If a TopicPathExpression is given, NotificationMessage delivery is paused on all Subscriptions associated with any Topic identified by the TopicPathExpression. In this situation, we say the Topic is currently “paused”.

If no TopicPathExpression is given, NotificationMessage delivery is paused on ALL Subscriptions associated with the NotificationProducer. In this situation, we say the NotificationProducer is in the “paused” state.

NotificationMessage delivery can be resumed by sending a Resume request message (see 8.5).

Note: Pausing or Resuming a Topic or an entire NotificationProducer has no effect on the TerminationTime of the affected Subscription resources [WS-ResourceLifetime]. A Subscription’s TerminationTime may elapse while the Topic or NotificationProducer is in a paused state, initiating the scheduled destruction of the Subscription resource.

In an asynchronous implementation, there could be some NotificationMessages *in-flight* at the time the NotificationProducer receives a Pause request. A NotificationProducer MAY attempt to deliver these in-flight NotificationMessages.

If the NotificationProducer accepts the PauserRequest message, it MUST respond with a message of the following form:

```
…
```

```
<wsnt:PauseResponse />
…
```

Instead of the PauseResponse message, the NotificationProducer may also send the following faults in response to a Pause request message:

- Invalid TopicPathExpression
- Given TopicPathExpression did not match any Topic supported by the NotificationProducer
- Pause operation could not be performed
- Others, TBD

8.4.1 Example SOAP Encoding of the Pause Message Exchange

The following is a non-normative example of a Pause request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
    "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/PauseRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/ProducerEndpoint
    </wsa:To>
    <npex:NPResourceId>
        uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
    </npex:NPResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:PauseRequest>
      <wsnt:Topic>wsrp:ResourcePropertiesValueChanges</wsnt:Topic>
    </wsnt:PauseRequest>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a Pause response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:ncex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
```

```
Notification/PauseResponse
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.consumer.org/ConsumerEndpoint
    </wsa:To>
    <ncex:NCResourceId>
        uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
    </ncex:NCResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:PauseResponse/>
  </s12:Body>
</s12:Envelope>
```

8.5 Resume

This request message MUST follow the implied resource pattern as outlined in [WS-Resource]. If a requestor wishes to resume the delivery of NotificationMessages by the NotificationProducer, it MUST send a resume request message. The resume request message has the following form:

```
…
<wsnt:ResumeRequest>
   <wsnt:topicPathExpression>?
</wsnt:ResumeRequest>
…
```

The contents of the ResumeRequest message are further described as follows:

wsnt:ResumeRequest/wsnt:topicPathExpression

> If a topicPathExpression is given, delivery is resumed on all subscriptions associated with any topic identified by the topicPathExpression. In this case, the topic is no longer "paused". However, delivery of notifications will not be resumed if the entire NotificationProducer is in the "paused" state. If a topic is not in the "paused" state, and it is the subject of a resume operation, no change occurs.
>
> If no topicPathExpression is given, the NotificationProducer is transitioned out of the "paused" state, and delivery is resumed on those subscriptions whose topic is currently not paused. If the NotificationProducer is not in the "paused" state, and the resume operation is invoked with no topicPathExpression, no change occurs.

When a topic is resumed the NotificationProducer will generate Notifications for situations that it detects after the resumption of the topic. It does not generate Notifications relating to situations it detected while the topic was paused. Similarly, if the entire NotificationProducer is resumed, it does not generate Notifications relating to situations that it detected while it was paused.

If the NotificationProducer accepts the ResumeRequest message, it must respond with a message of the following form:

```
…
<wsnt:ResumeResponse />
```

```
…
```

Instead of the ResumeResponse message, the NotificationProducer may also send the following faults in response to a ResumeRequest message:

- Invalid topic path expression
- Given topic path did not match any topic supported by the NotificationProducer
- Resume operation could not be performed
- Others, TBD

8.5.1 Example SOAP Encoding of the Resume Message Exchange

The following is a non-normative example of a Resume request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/ResumeRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/ProducerEndpoint
    </wsa:To>
    <npex:NPResourceId>
        uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
    </npex:NPResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:ResumeRequest>
      <wsnt:Topic>wsrp:ResourcePropertiesValueChanges</wsnt:Topic>
    </wsnt:ResumeRequest>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a Resume request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:ncex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/ResumeResponse
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
```

```
        http://www.consumer.org/ConsumerEndpoint
    </wsa:To>
    <ncex:NCResourceId>
        uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
    </ncex:NCResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:ResumeResponse/>
  </s12:Body>
</s12:Envelope>
```

9 SubscriptionManager Interface

The SubscriptionManager interface defines message exchanges to manipulate Subscription resources. The SubscriptionManager follows the implied resource pattern as described in [WS-Resource].

A SubscriptionManager MUST support the required message exchanges associated with the WS-ResourceProperties specification [WS-ResourceProperties] and MAY support the optional message exchanges defined by WS-ResourceProperties.

The SubscriptionManager MUST support the message exchanges defined for both forms of resource lifetime (immediate and scheduled destruction) by WS-ResourceLifetime [WS-ResourceLifetime]. These message exchanges define the means by which Subscription resources can be explicitly destroyed, or destroyed using a scheduled (time-based) mechanism.

9.1 Subscription Resource Properties

In addition to the Resource Properties required by WS-ResourceLifetime, the Subscription resource MUST include a reference to each of the following Resource Property elements in its Resource Properties document:

```
…
  <xsd:element name=”ConsumerReference”
               type=”wsa:EndpointReferenceType”
               minOccurs=”1” maxOccurs=”1” />
  <xsd:element name=”TopicPathExpression”
               type=”wsnt:TopicPathExpression”
               minOccurs=”1” maxOccurs=”1” />
  <xsd:element name=”UseNotify”
               type=”xsd:boolean”
               minOccurs=”1” maxOccurs=”1” />
  <xsd:element name=”Precondition”
               type=”wsrp:QueryExpressionType”
               minOccurs=”0” maxOccurs=”1” />
  <xsd:element name=”Selector”
               type=”wsrp:QueryExpressionType”
               minOccurs=”0” maxOccurs=”1” />
  <xsd:element name=”SubscriptionPolicy”
               type=”wsp:Policy”
               minOccurs=”0” maxOccurs=”1” />
  <xsd:element name=”CreationTime”
               type=”xsd:dateTime”
               minOccurs=”1” maxOccurs=”1” />
```

These Resource Property elements are further described as follows:

/wsnt:ConsumerReference and /wsnt:TopicPathExpression and /wsnt:UseNotify and /wsnt:Precondition and /wsnt:Selector and /wsnt:SubscriptionPolicy

These elements are as defined in the description of the Subscribe request message (see 8.2).

/wsnt:CreationTime

Indicates the date and time at which the Subscription was created.

The following properties MAY be modified by the requestor, by sending a SetResourceProperties request message as defined in the WS-ResourceProperties specification:

/wsnt:ConsumerReference and /wsnt:Precondition and /wsnt:SubscriptionPolicy

9.2 PauseSubscription

To temporarily suspend the delivery of NotificationMessages on the given Subscription, a requestor MUST send a PauseSubscription request message to the SubscriptionManager. The PauseSubscription request message has the following form:

```
…
<wsnt:PauseSubscriptionRequest />
…
```

This request message MUST follow the implied resource pattern as outlined in [WS-Resource].

Upon successful processing of this message the Subscription is in the *paused* state. Delivery of NotificationMessages can be resumed by sending the ResumeSubscription request message (see 9.3).

In an asynchronous implementation, there could be some NotificationMessages *in-flight* at the time the Subscription was paused. A NotificationProducer MAY attempt to deliver these in-flight NotificationMessages.

Note: transitioning between the paused state and the resumed state has no effect on the TerminationTime of the Subscription resource [WS-ResourceLifetime]. A Subscription's TerminationTime may elapse while it is in a paused state, initiating the scheduled destruction of the Subscription resource.

If the SubscriptionManager accepts the PauseSubscription request message, it must respond with a message of the following form:

```
…
<wsnt:PauseSubscriptionResponse />
…
```

Instead of the PauseSubscriptionResponse message, the SubscriptionManager may also send the following faults in response to a PauseSubscription request message:

- Pause operation could not be performed on the Subscription
- Others, TBD

9.2.1 Example SOAP Encoding of the PauseSubscription Message Exchange

The following is a non-normative example of a PauseSubscription request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/PauseSubsriptionRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/SubsMgrEndpoint
    </wsa:To>
    <npex:NSResourceId>
         uuid:8fefcd11-7d3d-66b344a2-ca44-9876bacd44e9
    </npex:NSResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:PauseSubscriptionRequest/>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a PauseSubscription response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
        "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:ncex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/PauseSubscriptionResponse
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.consumer.org/ConsumerEndpoint
    </wsa:To>
    <ncex:NCResourceId>
        uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
    </ncex:NCResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:PauseSubsciptionResponse/>
  </s12:Body>
</s12:Envelope>
```

9.3 ResumeSubscription

If a requestor wishes to resume the delivery of NotificationMessages on the given Subscription, it must send a ResumeSubscription request message. The ResumeSubscription request message has the following form:

```
…
<wsnt:ResumeSubscriptionRequest />
…
```

This request message MUST follow the implied resource pattern as outlined in [WS-Resource]. Upon successful processing of this message the Subscription is no longer in the *paused* state. However, delivery of NotificationMessages will not be resumed if the NotificationProducer or if all of the Topics associated with the Subscription are in the *paused* state.

If a Subscription is not in the *paused* state, and it is the subject of a ResumeSubscription request message, no change occurs.

When a Subscription is resumed the NotificationProducer MAY choose from one of the following courses of action. The choice MAY be determined by the NotificationProducer itself, or MAY be specified by a Policy associated with the Subscription.

1. Generate NotificationMessages for all the Situations related to the Subscription that occurred while the Subscription was paused (as well as for any new Situations that occur after the Subscription has been resumed).
2. Generate a NotificationMessgae for the last Situation that occurred while the Subscription was paused (as well as for any new situations that occur after the topic has been resumed).
3. Generate no NotificationMessages until a Situation occurs after the Subscription has been resumed.

If the SubscriptionManager accepts the ResumeSubscription request message, it MUST respond with a message of the following form:

```
…
<wsnt:ResumeSubscriptionResponse />
…
```

Instead of the ResumeSubscriptionResponse message, the SubscriptionManager may also send the following faults in response to a ResumeSubscription request message:

- Resume operation could not be performed on the Subscription
- Others, TBD

9.3.1 Example SOAP Encoding of the ResumeSubscription Message Exchange

The following is a non-normative example of a ResumeSubscription request message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
```

```
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/ResumeSubsriptionRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/SubsMgrEndpoint
    </wsa:To>
    <npex:NSResourceId>
         uuid:8fefcd11-7d3d-66b344a2-ca44-9876bacd44e9
    </npex:NSResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:ResumeSubscriptionRequest/>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a ResumeSubscription response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:ncex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/ResumeSubscriptionResponse
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.consumer.org/ConsumerEndpoint
    </wsa:To>
    <ncex:NCResourceId>
        uuid:9fef5fec-6dc3-44a2-ba32-8680cace43f9
    </ncex:NCResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:ResumeSubsciptionResponse/>
  </s12:Body>
</s12:Envelope>
```

10 NotificationBroker Interface

The NotificationBroker interface defines a standard set of message exchanges to describe a message broker, providing an intermediary between Publishers and Subscribers on a collection of Topics. This is very similar to a traditional Message Oriented Middleware model.

A SubscriptionManager MUST support the required message exchanges defined by the WS-ResourceProperties specification [WS-ResourceProperties] and MAY support the optional message exchanges defined by WS-ResourceProperties.

A NotificationBroker MUST also support message exchanges and Resource Property elements defined by the following interfaces:

- NotificationProducer
- NotificationConsumer

10.1 NotificationBroker Resource Properties

In addition to the Resource Property elements associated with the interfaces the Notification Broker extends, the Notification Broker must also include the following reference property element:

```
  <xsd:element name="RequiresRegistration" type="xsd:boolean"/>
```

This resource property element is further constrained as follows:

/wsnt:RequiresRegistration

The value is "true" if the NotificationBroker requires a publisher to register (see 10.3) before sending it a Notify (i.e. publish) message on this Topic. The default is "false".

10.2 Notify

The NotificationBroker MUST support the Notify message exchange from NotificationConsumer interface, with the following clarifications/restrictions:

A Publisher sends a Notify message to a NotificationBroker in order to publish a NotificationMessage on a given Topic. As a result of the Publisher sending this message, NotificationMessages are delivered to all NotificationConsumers subscribed on the given Topic. For some Topics (those that require a Publisher to pre-register), the requestor must be a registered Publisher in order to successfully publish a NotificationMessage to the given Topic (see 10.3).

10.3 RegisterPublisher

The RegisterPublisher message is used by the Publisher to confirm its ability to publish on a given Topic or set of Topics.

If an entity wishes to register a publisher, it MUST send a RegisterPublisher request message to the NotificationBroker. The format of the RegisterPublisher request message is:

```
…
<wsnt:RegisterPublisherRequest>
   <wsnt:PublisherReference>
      wsa:EndpointReference
   </wsnt:PublisherReference>?
   <wsnt:Topic>wsnt:TopicPathExpression</wsnt:Topic>*
   <wsnt:Demand>xsd:boolean</wsnt:Demand>?
   <wsrl:InitialTerminationTime>
      xsd:dateTime
   </wsrl:InitialTerminationTime>?
</wsnt:RegisterPublisherRequest>
```

…

This request message MUST follow the implied resource pattern as outlined in [WS-Resource]. The components of the RegisterPublisher request message are further described as follows:

/wsnt:PublisherReference

An OPTIONAL EndpointReference to an entity that wishes to become a Publisher on one or more Topics supported by the Notification Broker. This component MUST appear if the /wsnt:Demand component has value "true". If this component is missing, the Publisher is either not a Web service, or does not wish to receive messages from the NotificationBroker.

/wsnt:Topic

A set of TopicPathExpressions that identifies one or more Topics. If included, the given Publisher is registered to publish only on the set of Topics identified by this component.

/wsnt:Demand

A boolean. If its value is "true", then the intent of the Publisher is to use a demand-based model from the NotificationBroker (see 5). In this case, the NotificationBroker must observe the rules associated with demand-based publishing, including establishing a Subscription with the Publisher on those Topics and pausing/resuming those Subscriptions as the NotificationBroker receives Subscriptions for those Topics.

/wsrl:InitialTerminationTime

The generic (not application specific) semantics of this component are described in [WS-ResourceLifetime]. This component contains the service requestor's suggestion for the initial termination time of the PublisherRegistration resource being created. This time is relative to the time source used by the NotificationBroker. If the NotificationBroker is unable or unwilling to set the TerminationTime to the given value or greater, then the RegisterPublisher request MUST fault. If the value is not "in the future" relative to the current time as known by the NotificationBroker, the RegisterPublisher request MUST fault. The use of the xsi:nil attribute with value "true" indicates there is no scheduled termination time requested for the RegisterPublisher. If the element does not include the time zone designation, the value of the element MUST be interpreted as universal time (UTC) time.

If a /wsnt:Topic component is included in the message, the NotificationBroker MUST register the Web service specified by the /wsnt:PublisherReference component as a Publisher on the set of Topics identified by the /wsnt:Topic component.

As part of the processing of a RegisterPublisher request, the NotificationBroker creates a PublisherRegistration resource representing the registration. A new resource is created regardless of whether the same Publisher has previously registered with the NotificationBroker. The NotificationBroker returns an EndpointReference in the response to the RegisterPublisher request. This EndpointReference is a WS-Resource-Qualified EndpointReference as defined in [WS-Resource] and includes the address of a PublisherRegistrationManager service and a reference property identifying the PublisherRegistration resource.

If the NotificationBroker accepts the RegisterPublisher request message, it must respond with a message of the following form:

```
…
<wsnt:RegisterPublisherResponse>
   <wsnt:PublisherRegistrationReference>
      <wsa:Address>
         Address of PublisherRegistration Manager
      </wsa:Address>
      <wsa:ReferenceProperties>
          PublisherRegistration Identifier
      </wsa:ReferenceProperties>
      …
   </wsnt:PublisherRegistrationReference>
</wsnt:RegisterPublisherResponse>
…
```

The components of the RegisterPublisher response message are further described as follows:

/wsnt:PublisherRegistrationReference/wsa:Address

The address of a Web service that implements the PublisherRegistrationManager set of messages. This PublisherRegistrationManager is capable of operating on the PublisherRegistration resource created as a result of processing the RegisterPublisher request message.

/wsnt:PublisherRegistrationReference/wsa:ReferenceProperties

- An identifier of the PublisherRegistration resource created as a result of processing the RegisterPublisher request message.

Instead of the RegisterPublisherResponse message, the NotificationBroker may also send the following faults in response to a RegisterPublisherRequest message:

- Invalid TopicPathExpression
- Given TopicPathExpression did not match any Topic supported by the NotificationBroker
- Publisher registration failed
- Others, TBD

10.3.1 Example SOAP Encoding of the RegisterPublisher Message Exchange

The following is a non-normative example of a RegisterPublisher response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
```

```
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.producer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/RegisterPublisherRequest
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/NotificationBrokerEndpoint
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <wsnt:RegisterPublisherRequest>
      <wsnt:PublisherReference>
         <wsa:Address>
            http://www.producer.org/ProducerEndpoint
         </wsa:Address>
         <wsa:ReferenceProperties>
           <npex:NPResourceId>
               uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
           </npex:NPResourceId>
         </wsa:ReferenceProperties>
      </wsnt:PublisherReference>
      <wsnt:Topic>wsrp:ResourcePropertiesValueChanges</wsnt:Topic>
      <wsnt:Demand>true</wsnt:Demand>
      <wsrl:InitialTerminationTime>
         2003-12-25T00:00:00.00000Z
      </wsrl:InitialTerminationTime>
    </wsnt:RegisterPublisherRequest>
  </s12:Body>
</s12:Envelope>
```

The following is a non-normative example of a RegisterPublisher response message using SOAP 1.2 [SOAP 1.2]:

```
<s12:Envelope
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
    xmlns:wsnt=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
    xmlns:wsrp=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
    xmlns:npex="http://www.consumer.org/RefProp">
  <s12:Header>
    <wsa:Action>
        http://www.ibm.com/xmlns/stdwip/web-services/WS-
Notification/RegisterPublisherResponse
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
        http://www.producer.org/ProducerEndpoint
    </wsa:To>
    <npex:NPResourceId>
        uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
    </npex:NPResourceId>
  </s12:Header>
  <s12:Body>
    <wsnt:RegisterPublisherResponse>
```

```
      <wsnt:PublisherRegistrationReference>
        <wsa:Address>
           http://www.producer.org/PublisherEndpoint
        </wsa:Address>
        <wsa:ReferenceProperties>
          <npex:NPubResourceId>
             uuid:95fefeb3-f37d-5dfe-44fe-675d9fce5d22
          </npex:NPubResourceId>
        </wsa:ReferenceProperties>
      </wsnt:PublisherRegistrationReference>
    </wsnt:RegisterPublisherResponse>
  </s12:Body>
</s12:Envelope>
```

11 PublisherRegistrationManager Interface

The PublisherRegistrationManager interface defines message exchanges to manipulate PublisherRegistration resources. The PublisherRegistrationManager follows the implied resource pattern as described in [WS-Resource].

A PublisherRegistrationManager MUST support the required message exchanges associated with the WS-ResourceProperties specification [WS-ResourceProperties] and MAY support the optional message exchanges defined by WS-ResourceProperties.

The PublisherRegistrationManager MUST support the message exchanges defined for both forms of resource lifetime (immediate and scheduled destruction) by WS-ResourceLifetime [WS-ResourceLifetime]. These message exchanges define the means by which PublisherRegistration resources can be explicitly destroyed, or destroyed using a scheduled (time-based) mechanism.

11.1 PublisherRegistration Resource Properties

In addition to the resource properties required by WS-ResourceLifetime, the PublisherRegistration resource MUST include a reference to each of the following resource property elements in its resource properties document:

```
…
  <xsd:element name=”PublisherReference”
               type=”wsa:EndpointReference”
               minOccurs=”0” maxOccurs=”1” />
  <xsd:element name=”TopicPathExpression”
               type=”wsnt:TopicPathExpression”
               minOccurs=”1” maxOccurs=”unbounded” />
  <xsd:element name=”Demand”
               type=”xsd:boolean”
               minOccurs=”1” maxOccurs=”1” />
  <xsd:element name=”CreationTime”
               type=”xsd:dateTime”
               minOccurs=”1” maxOccurs=”1” />
```

These resource property elements are further described as follows:

/wsnt:PublisherReference and /wsnt:TopicPathExpression and wsnt:/Demand

These elements are as defined in the description of the RegisterPublisher request message (see 10.3).

/wsnt:CreationTime

Indicates the date and time at which the PublisherRegistration was created.

The following resource properties MAY be modified by the requestor, by sending a SetResourceProperties request message as defined in the WS-ResourceProperties specification:

- /wsnt:TopicPathExpression

12 Security Considerations

This section deals with the security aspects of the Web services approach, specified in this document, to Notification using a Topic-based pub/sub pattern. Security aspects of this document deals with (a) securing the standard message exchanges defined in this specification, and (b) authorization and denial of service considerations.

12.1 Securing the Message Exchanges

As defined in this document, Subscription messages are exchanged between Subscribers and NotificationBrokers; NotificationMessages are exchanged between Publishers and NotificationBrokers, and NotificationBrokers and NotificationConsumers. It is important to ensure NotificationMessages cannot be tampered with during delivery to NotificationConsumers and to ensure confidentiality of sensitive NotificationMessages. Therefore, when these messages are exchanged between these services, it is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [WS-Security]. In order to properly secure messages, the body and all relevant headers need to be included in the digital signature so as to prove the integrity of the message. In addition the reference properties within an EndpointReference, passed in a Subscribe or RegisterPublisher request message, may be encrypted to ensure their privacy. In the event that a requestor communicates frequently with a Web service to access Resource Properties, either directly through a query or accomplished through Notification of resource property value change, it is RECOMMENDED that a security context be established using the mechanisms described in WS-Trust and WS-SecureConversation allowing for potentially more efficient means of authentication.

It is common for communication between requestors and Web service associated with a resource through the implied resource pattern to exchange multiple messages. As a result, the usage profile is such that it is susceptible to key attacks. For this reason it is strongly RECOMMENDED that the keys used to secure the channel be changed frequently. This "re-keying" can be effected a number of ways. The following list outlines four common techniques:

- Attaching a nonce to each message and using it in a derived key function with the shared secret
- Using a derived key sequence and switch "generations"
- Closing and re-establishing a security context
- Exchanging new secrets between the parties

It should be noted that the mechanisms listed above are independent of the security context token (SCT) and secret returned when subscribed the first time. That is, the keys used to secure the channel during notifications may be independent of the key used to prove the right to subscribe with a NotificationProducer.

The security context MAY be re-established using the mechanisms described in WS-Trust and WS-SecureConversation. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret SHOULD NOT be used to encrypt the new shared secret. Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation.

The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

- **Message alteration** – Alteration is prevented by including signatures of the message information using WS-Security.
- **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security.
- **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy and WS-SecurityPolicy).
- **Authentication** – Authentication is established using the mechanisms described in WS-Security and WS-Trust. Each message is authenticated using the mechanisms described in WS-Security.
- **Accountability** – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- **Availability** – Many services are subject to a variety of availability attacks. Replay is a common attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification. That said, care should be taken to ensure that minimal processing be performed prior to any authenticating sequences.
- **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security and the sequences outlined in WS-ReliableMessaging.

12.2 Securing Subscriptions and Notifications

Given WS-Notification provides mechanisms for publishing, and subscribing to topics, security policies should be established such that

1. only authorized principals can subscribe for Notifications
2. only authorized principals can modify or delete Subscriptions
3. only authorized principals can modify or delete PublisherRegistrations
4. only authorized principals can publish NotificationMessages
5. only authorized principals can create new child Topics (where the TopicSpace definition permits this).

It is recommended that the authorization policies be specified at the granularity of the Topic. It should be noted that even though Subscriptions may be done by authorized principals, the NotificationMessages may be delivered to NotificationConsumers whose identity may be different from the Subscriber. Message protection policies as outlined in the previous section can be used to ensure that sensitive NotificationMessages are not delivered to malicious endpoints. For example, a key may need to be specified or generated during the process of Subscription, so that the NotificationMessages can be encrypted using the key to ensure confidentiality of the messages. The mechanism by which the key is specified is governed by the Subscription policy.

Given that WS-Notification uses WS-ResourceProperties, the security considerations outlined in WS-ResourceProperties need to be taken into account. Authorization policies for those Resource Properties should be put in place so that the implications of providing the state information (through GetResourceProperty request messages) or through notification of state change and modification of the resource properties (through SetResourceProperty request messages), are taken into account.

This specification provides a mechanism by which Subscribers can specify a subscription policy. Such a policy may contain security policy about protecting the message exchanges resulting from the Subscription. Security policy for Subscription message exchanges needs to take this into consideration so that the Subscription policies are protected. Also, given this policy may be contained in the Resource Properties of the subscription maintained by the SubscriptionManager, the Resource Properties must be appropriately secured.

WS-Notification uses WS-ResourceLifetime to manage the lifetime of subscriptions and PublisherRegistrations. Authorization policies should be defined so that the implications, of destroying a resource either through explicit requests or by setting TerminationTime are considered. Therefore, the security considerations outlined in WS-ResourceLifetime need to be taken into account in order to secure Subscriptions and PublisherRegistrations appropriately.

13 Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams. The authors wish to acknowledge the contributions from many people, including:

Tim Banks (IBM), Nick Butler (IBM), Doug Davis (IBM), John Dinger (IBM), Don Ferguson (IBM), Jeff Frey, Andreas Koeppel (SAP), Heather Kreger (IBM), Kevin Liu (SAP),Tom Maguire (IBM), Susan Malaika (IBM), Martin Nally (IBM), Jeff Nick (IBM), Claus von Riegen (SAP), Rick Rineholt (IBM), John Rofrano (IBM), Eugène Sindambiwe (SAP), Jay Unger (IBM), Mark Weitzel (IBM), Dan Wolfson (IBM).

14 References

[SOAP 1.2]

http://www.w3.org/TR/soap12-part1/

[URI]

T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[WS-Addressing]
http://www.ibm.com/developerworks/webservices/library/ws-add/

[WS-Resource]

http://www-106.ibm.com/developerworks/webservices/library/ws-resource/ws-modelingresources.pdf

[WS-ResourceLifetime]

http://www-106.ibm.com/developerworks/webservices/library/ws-resource/ws-resourcelifetime.pdf

[WS-ResourceProperties]

http://www-106.ibm.com/developerworks/webservices/library/ws-resource/ws-resourceproperties.pdf

[WS-Security]

http://www.ibm.com/developerworks/webservices/library/ws-secure/

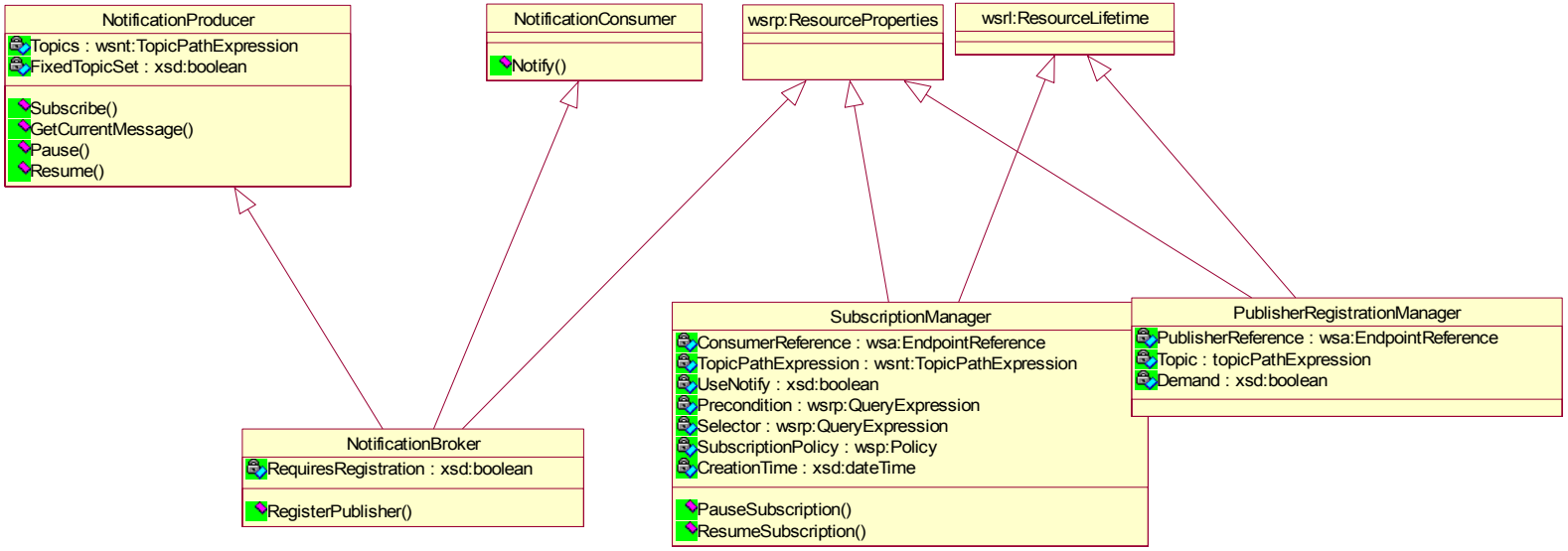
[XML-Infoset]
http://www.w3.org/TR/xml-infoset/

[XML]

http://www.w3.org/TR/REC-xml

[XPATH]
http://www.w3.org/TR/xpath

15 Appendix I – UML



16 Appendix II – XML Schema

The XML types and elements used in WS-Notification are defined in the following XML Schema

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

   Copyright Notice

   Copyright 2003-2004 IBM Corporation
   All rights reserved.
-->

<xsd:schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
  xmlns:wsnt=
    "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
  xmlns:wsrp=
    "http://www.ibm.com/xmlns/stdwip/web-services/WS-
ResourceProperties"
  targetNamespace=
    "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
  elementFormDefault="qualified"  attributeFormDefault="unqualified">

<!-- ======================== Imports  ============================== -->

  <xsd:import namespace=
    "http://schemas.xmlsoap.org/ws/2003/03/addressing"
              schemaLocation=
    "http://schemas.xmlsoap.org/ws/2003/03/addressing"
  />
  <xsd:import namespace=
    "http://schemas.xmlsoap.org/ws/2002/12/policy"
              schemaLocation=
    "http://schemas.xmlsoap.org/ws/2002/12/policy"
  />
  <xsd:import namespace=
    "http://www.ibm.com/xmlns/stdwip/web-services/WS-
ResourceProperties"
              schemaLocation=
  "http://www-106.ibm.com/developerworks/webservices/library/ws-
resource/WS-ResourceProperties.xsd"
  />
  <xsd:import namespace=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceLifetime"
              schemaLocation=
  "http://www-106.ibm.com/developerworks/webservices/library/ws-
resource/WS-ResourceLifetime.xsd"
  />

<!-- ============== utility type definitions  =================== -->
```

```
  <xsd:complexType name="Documentation" mixed="true">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="0"
               maxOccurs="unbounded" namespace="##any"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ExtensibleDocumented" abstract="true"
                   mixed="false">
    <xsd:sequence>
      <xsd:element name="documentation" type="wsnt:Documentation"
                   minOccurs="0" />
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax" />
  </xsd:complexType>

<!-- ============= Topic and Topic-Space Related  ================= -->
  <xsd:complexType name="TopicSpaceType">
    <xsd:complexContent>
       <xsd:extension base="wsnt:ExtensibleDocumented">
         <xsd:sequence>
           <xsd:element name="Topic" type="wsnt:TopicType"
                        minOccurs="0" maxOccurs="unbounded"/>
           <xsd:any namespace="##other"
                    minOccurs="0" maxOccurs="unbounded"
                    processContents="lax"/>
         </xsd:sequence>
         <xsd:attribute name="name" type="xsd:NCName"/>
         <xsd:attribute name="targetNamespace" type="xsd:anyURI"
                        use="required"/>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

  <xsd:element name="TopicSpace" type="wsnt:TopicSpaceType">
    <xsd:unique name="rootTopicUniqueness">
      <xsd:selector xpath="wsnt:Topic"/>
        <xsd:field xpath="@name"/>
    </xsd:unique>
  </xsd:element>

  <xsd:complexType name="TopicType">
    <xsd:complexContent>
      <xsd:extension base="wsnt:ExtensibleDocumented">
        <xsd:sequence>
           <xsd:element ref="wsrp:QueryExpression"
                        minOccurs="0" maxOccurs="1" />
           <xsd:element name="Topic" type="wsnt:TopicType"
                        minOccurs="0" maxOccurs="unbounded">
              <xsd:unique name="childTopicUniqueness">
                 <xsd:selector xpath="wsnt:topic"/>
                 <xsd:field xpath="@name"/>
              </xsd:unique>
           </xsd:element>
           <xsd:any namespace="##other" minOccurs="0"
                                        maxOccurs="unbounded"/>
        </xsd:sequence>
```

```
        <xsd:attribute name="name" use="required" type="xsd:NCName"/>
        <xsd:attribute name="aliasRef"
                       type="wsnt:TopicPathExpression"/>
        <xsd:attribute name="messageTypes" default="xsd:any">
          <xsd:simpleType>
            <xsd:list itemType="xsd:QName"/>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="final" type="xsd:boolean"
                                    default="false"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:simpleType name="TopicPathExpression">
    <xsd:restriction base="xsd:token">
      <xsd:annotation>
        <xsd:documentation>
  The pattern allows strings matching the following EBNF:
    TopicPathExpression  ::=   TopicPath ( '|' TopicPath )*
    TopicPath            ::=   RootTopic ChildTopicExpression*
    RootTopic            ::=   (QName | '*')
    ChildTopicExpression ::=   '/' (NCName | '*' | '/*')

        </xsd:documentation>
      </xsd:annotation>
      <xsd:pattern value=

"(((\w+:)?(\w+))|\*)(/(\w+|\*|/\*))*(\|(((\w+:)?(\w+))|\*)(/(\w+|\*|/\*
))*)*">
      </xsd:pattern>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="ConcreteTopicPathExpression">
    <xsd:restriction base="xsd:token">
      <xsd:annotation>
        <xsd:documentation>
  The pattern allows strings matching the following EBNF:
    ConcreteTopicPath    ::=   RootTopic ChildTopic*
    RootTopic            ::=   QName
    ChildTopic           ::=   '/' NCName

        </xsd:documentation>
      </xsd:annotation>
      <xsd:pattern value="((\w+:)?(\w+))(/(\w+))*" >
      </xsd:pattern>
    </xsd:restriction>
  </xsd:simpleType>

<!-- =============== Resource Property Related  ================== -->
<!-- ======== Resource Properties for NotificationProducer ======== -->
         <xsd:element name="Topics">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="Topic"
                               type="wsnt:TopicPathExpression"
```

```
                                minOccurs="0" maxOccurs="unbounded" />
                  </xsd:sequence>
               </xsd:complexType>
         </xsd:element>

         <xsd:element name="FixedTopicSet" type="xsd:boolean"/>

<!-- ======== Resource Properties for SubscriptionManager ========= -->
  <xsd:element name="ConsumerReference"
               type="wsa:EndpointReferenceType" />
  <xsd:element name="TopicPathExpression"
               type="wsnt:TopicPathExpression" />
  <xsd:element name="UseNotify"
               type="xsd:boolean" />
  <xsd:element name="Precondition"
               type="wsrp:QueryExpressionType" />
  <xsd:element name="Selector"
               type="wsrp:QueryExpressionType" />
  <xsd:element name="SubscriptionPolicy"
               type="wsp:PolicyExpression" />
  <xsd:element name="CreationTime"
               type="xsd:dateTime" />

<!-- ======== Resource Properties for NotificationBroker ========== -->
         <xsd:element name="RequiresRegistration" type="xsd:boolean"/>

<!-- ====== Resource Properties for PublisherRegistration ========= -->
         <xsd:element name="PublisherReference"
                      type="wsa:EndpointReferenceType" />

         <xsd:element name="Demand"
                      type="xsd:boolean" />

         <xsd:element name="PublisherRegistrationResourceProperties">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element ref="wsnt:PublisherReference"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:TopicPathExpression"
                              minOccurs="1" maxOccurs="unbounded" />
                  <xsd:element ref="wsnt:Demand"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:CreationTime"
                               minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
     </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

17 Appendix III – WSDL 1.1

The following illustrates the WSDL 1.1 for the Web service methods described in this specification:

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

```
   Legal Disclaimer

   Copyright Notice

   Copyright 2003-2004 IBM Corporation
   All rights reserved.
-->
<wsdl:definitions name="WS-Notification"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
  xmlns:wsnt=
 "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification"
  xmlns:wsrp=
 "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
  xmlns:wsrl=
 "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceLifetime"
  targetNamespace=
 "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification">

<!-- ========================== Imports ========================== -->
   <wsdl:import
       namespace=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
       location="./WS-ResourceProperties.wsdl" />

   <wsdl:import
       namespace=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceLifetime"
       location="./WS-ResourceLifetime.wsdl" />

<!-- ===================== Types Definitions ===================== -->
   <wsdl:types>
      <xsd:schema
        targetNamespace=
 "http://www.ibm.com/xmlns/stdwip/web-services/WS-Notification" >

         <xsd:include schemaLocation=
 "http://www-106.ibm.com/developerworks/webservices/library/ws-
resource/WS-Notification.xsd" />

         <xsd:import
            namespace=
            "http://schemas.xmlsoap.org/ws/2003/03/addressing"
            schemaLocation=
            "http://schemas.xmlsoap.org/ws/2003/03/addressing"
         />
         <xsd:import namespace=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
              schemaLocation=
  "http://www-106.ibm.com/developerworks/webservices/library/ws-
resource/WS-ResourceProperties.xsd"
         />
         <xsd:import
```

```
            namespace=
  "http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceLifetime"
            schemaLocation=
  "http://www-106.ibm.com/developerworks/webservices/library/ws-
resource/WS-ResourceLifetime.xsd"
         />

<!-- =============== Resource Property Related  ================== -->
<!-- ======== Resource Properties for NotificationProducer ======== -->
         <xsd:element name="NotificationProducerRP" >
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element ref="wsnt:Topics"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:FixedTopicSet"
                               minOccurs="1" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

<!-- ======== Resource Properties for SubscriptionManager ========= -->
         <xsd:element name="SubscriptionManagerRP" >
            <xsd:complexType>
               <xsd:sequence>
         <!-- From WS-ResourceLifetime ScheduledResourceTermination -->
                  <xsd:element ref="wsrl:CurrentTime"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsrl:TerminationTime"
                               minOccurs="1" maxOccurs="1" />

         <!-- SubscriptionManager specific  -->
                  <xsd:element ref="wsnt:ConsumerReference"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:TopicPathExpression"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:UseNotify"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:Precondition"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsnt:Selector"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsnt:SubscriptionPolicy"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsnt:CreationTime"
                               minOccurs="1" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

<!-- ======== Resource Properties for NotificationBroker ========== -->
         <xsd:element name="NotificationBrokerRP" >
            <xsd:complexType>
               <xsd:sequence>
         <!-- From NotificationProducer -->
                  <xsd:element ref="wsnt:Topics"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:FixedTopicSet"
```

```
                              minOccurs="1" maxOccurs="1" />
         <!-- SubscriptionManager specific  -->
                  <xsd:element ref="wsnt:RequiresRegistration"
                              minOccurs="1" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

<!-- ====== Resource Properties for PublisherRegistration ========= -->
         <xsd:element name="PublisherRegistrationRP" >
            <xsd:complexType>
               <xsd:sequence>
         <!-- From WS-ResourceLifetime ScheduledResourceTermination -->
                  <xsd:element ref="wsrl:CurrentTime"
                              minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsrl:TerminationTime"
                              minOccurs="1" maxOccurs="1" />

         <!-- PublisherRegistration specific  -->
                  <xsd:element ref="wsnt:PublisherReference"
                              minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsnt:TopicPathExpression"
                              minOccurs="1" maxOccurs="unbounded" />
                  <xsd:element ref="wsnt:Demand"
                              minOccurs="1" maxOccurs="1" />
                  <xsd:element ref="wsnt:CreationTime"
                              minOccurs="1" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

<!-- ================== Message Helper Types  ==================== -->
         <xsd:complexType name="NotificationMessageHolderType" >
            <xsd:sequence>
               <xsd:element name="Topic"
                           type="wsnt:ConcreteTopicPathExpression"
                           minOccurs="1" maxOccurs="1" />
               <xsd:element name="ProducerReference"
                           type="wsa:EndpointReferenceType"
                           minOccurs="1" maxOccurs="1" />
               <xsd:element name="Message" type="xsd:anyType"
                           minOccurs="1" maxOccurs="1" />
            </xsd:sequence>
         </xsd:complexType>

<!-- ================ Message Specific Types  =================== -->
       <xsd:element name="ErrorMessage" type="xsd:string" />

<!-- ========== Message Types for NotificationConsumer  ========== -->
         <xsd:element name="Notify" >
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="NotificationMessage"
                           type="wsnt:NotificationMessageHolderType"
                           minOccurs="1" maxOccurs="unbounded" />
               </xsd:sequence>
            </xsd:complexType>
```

```
         </xsd:element>

<!-- ========== Message Types for NotificationProducer  =========== -->
         <xsd:element name="SubscribeRequest" >
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="ConsumerReference"
                               type="wsa:EndpointReferenceType"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element name="TopicPathExpression"
                               type="wsnt:TopicPathExpression"
                               minOccurs="1" maxOccurs="1" />
                  <xsd:element name="UseNotify"
                               type="xsd:boolean" default="true"
                               minOccurs="0" maxOccurs="1" />
                   <xsd:element name="Precondition"
                               type="wsrp:QueryExpressionType"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element name="Selector"
                               type="wsrp:QueryExpressionType"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element name="SubscriptionPolicy"
                               type="wsp:PolicyExpression"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsrl:InitialTerminationTime"
                               minOccurs="0"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="SubscribeResponse">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="SubscriptionReference"
                               type="wsa:EndpointReferenceType"
                               minOccurs="0" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="GetCurrentMessageRequest">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="Topic"
                           type="wsnt:ConcreteTopicPathExpression" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="GetCurrentMessageResponse">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:any />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
```

```
         <xsd:element name="PauseRequest">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="Topic"
                               type="wsnt:TopicPathExpression"
                               minOccurs="0" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="ResumeRequest">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="Topic"
                               type="wsnt:TopicPathExpression"
                               minOccurs="0" maxOccurs="1" />
              </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

<!-- ========== Message Types for SubscriptionManager  =========== -->

<!-- ========== Message Types for NotificationBroker  ============ -->
         <xsd:element name="RegisterPublisherRequest">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="PublisherReference"
                               type="wsa:EndpointReferenceType"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element name="Topic"
                               type="wsnt:TopicPathExpression"
                               minOccurs="0" maxOccurs="unbounded" />
                  <xsd:element name="Demand"
                               type="xsd:boolean" default="false"
                               minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsrl:InitialTerminationTime"
                               minOccurs="0"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="ResgisterPublisherResponse">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="PublisherRegistrationReference"
                               type="wsa:EndpointReferenceType"
                               minOccurs="0" maxOccurs="1" />
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

      </xsd:schema>
   </wsdl:types>

<!-- ==================== Message Definitions ==================== -->
   <wsdl:message name="ErrorMessage">
      <wsdl:part name="ErrorMessage" element="wsnt:ErrorMessage"/>
```

```
   </wsdl:message>

<!-- ================ NotificationConsumer::Notify ================
  Notify(
    NotificationMessage
      (TopicPathExpression, ProducerReference, Message)*
  returns: n/a (one way)
-->
  <wsdl:message name="Notify">
    <wsdl:part name="Notify" element="wsnt:Notify"/>
  </wsdl:message>

<!-- ============== NotificationProducer::Subscribe ===============
  Subscribe(
   (ConsumerEndpointReference, TopicPathExpression, [UseNotify],
   [Precondition], [Selector], [SubscriptionPolicy],
   [InitialTerminationTime])
  returns: WS-Resource qualified EPR to a Subscription
-->
   <wsdl:message name="SubscribeRequest" >
     <wsdl:part name="SubscribeRequest"
                element="wsnt:SubscribeRequest"/>
   </wsdl:message>
   <wsdl:message name="SubscribeResponse">
      <wsdl:part name="SubscribeResponse"
                 element="wsnt:SubscribeResponse"/>
   </wsdl:message>

<!-- ========== NotificationProducer::GetCurrentMessage ===========
  GetCurrentMessage(topic)
 returns: a NotificationMessage (xsd:any)
     -->
   <wsdl:message name="GetCurrentMessageRequest">
      <wsdl:part name="GetCurrentMessageRequest"
            element="wsnt:GetCurrentMessageRequest"/>
   </wsdl:message>
   <wsdl:message name="GetCurrentMessageResponse">
      <wsdl:part name="GetCurrentMessageResponse"
            element="wsnt:GetCurrentMessageResponse"/>
   </wsdl:message>

<!-- ================ NotificationProducer::Pause =================
   Pause([topicPathExpression])
   returns: empty
-->
   <wsdl:message name="PauseRequest">
      <wsdl:part name="PauseRequest"
                 element="wsnt:PauseRequest"/>
   </wsdl:message>

   <wsdl:message name="PauseResponse">
   </wsdl:message>

<!-- ================ NotificationProducer::Resume =================
   Resume([topicPathExpression])
   returns: empty
-->
```

```
   <wsdl:message name="ResumeRequest">
      <wsdl:part name="ResumeRequest"
                 element="wsnt:ResumeRequest"/>
   </wsdl:message>

   <wsdl:message name="ResumeResponse">
    </wsdl:message>

<!-- ========== SubscriptionManager::PauseSubscription ============
   PauseSubscription()
   returns: empty
-->
   <wsdl:message name="PauseSubscriptionRequest" />
   <wsdl:message name="PauseSubscriptionResponse"/>

<!-- ========= SubscriptionManager::ResumeSubscription ============
   ResumeSubscription()
   returns: empty
-->
   <wsdl:message name="ResumeSubscriptionRequest"/>
   <wsdl:message name="ResumeSubscriptionResponse"/>

<!-- ========== NotificationBroker::RegisterPublisher =============
   RegisterPublisher(PublisherReference, TopicPathExpression* ,
                     [Demand], [InitialTerminationTime])
   returns: WS-Resource qualified EPR to a PublisherRegistration
 **********************************
     -->
   <wsdl:message name="RegisterPublisherRequest">
      <wsdl:part name="RegisterPublisherRequest"
                 element="wsnt:RegisterPublisherRequest"/>
   </wsdl:message>
   <wsdl:message name="RegisterPublisherResponse">
      <wsdl:part name="RegisterPublisherResponse"
                 element="wsnt:ResgisterPublisherResponse"/>
   </wsdl:message>

<!-- =================== PortType Definitions ==================== -->
<!-- ========= NotificationConsumer PortType Definition ========== -->
  <wsdl:portType name="NotificationConsumer">
    <wsdl:operation name="Notify">
      <wsdl:input message="wsnt:Notify" />
    </wsdl:operation>
  </wsdl:portType>

<!-- ========= NotificationProducer PortType Definition ========== -->
  <wsdl:portType name="NotificationProducer"
      wsrp:ResourceProperties ="wsnt:NotificationProducerRP">

      <!-- ========== extends wsrp:ResourceProperties ============ -->
      <wsdl:operation name="GetResourceProperty">
        <wsdl:input message="wsrp:GetResourcePropertyRequest" />
        <wsdl:output message="wsrp:GetResourcePropertyResponse" />
        <wsdl:fault name="UnknownResource"
                    message="wsrp:ErrorMessage" />
        <wsdl:fault name="InvalidResourceProperty"
                    message="wsrp:ErrorMessage" />
```

```
      </wsdl:operation>

      <!-- ========== NotificationProducer Specific ============= -->
      <wsdl:operation name="Subscribe">
         <wsdl:input message="wsnt:SubscribeRequest" />
         <wsdl:output message="wsnt:SubscribeResponse" />
         <wsdl:fault name="SubscribeCreationFailed"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
      <wsdl:operation name="GetCurrentMessage">
         <wsdl:input message="wsnt:GetCurrentMessageRequest"/>
         <wsdl:output message="wsnt:GetCurrentMessageResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="NoCurrentMessageOnTopic"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
      <wsdl:operation name="Pause">
         <wsdl:input message="wsnt:PauseRequest"/>
         <wsdl:output message="wsnt:PauseResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="PauseFailed"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
      <wsdl:operation name="Resume">
         <wsdl:input message="wsnt:ResumeRequest"/>
         <wsdl:output message="wsnt:ResumeResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="ResumeFailed"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
   </wsdl:portType>

<!-- ========== SubscriptionManager PortType Definition ========== -->
   <wsdl:portType name="SubscriptionManager"
      wsrp:ResourceProperties ="wsnt:SubscriptionManagerRP">
      <!-- ========== extends wsrp:ResourceProperties ============ -->
      <wsdl:operation name="GetResourceProperty">
        <wsdl:input message="wsrp:GetResourcePropertyRequest" />
        <wsdl:output message="wsrp:GetResourcePropertyResponse" />
        <wsdl:fault name="UnknownResource"
                    message="wsrp:ErrorMessage" />
        <wsdl:fault name="InvalidResourceProperty"
                    message="wsrp:ErrorMessage" />
      </wsdl:operation>

      <!-- === extends wsrl:ImmediateResourceTermination ========== -->
      <wsdl:operation name="Destroy">
        <wsdl:input message="wsrl:DestroyRequest" />
```

```
        <wsdl:output message="wsrl:DestroyResponse" />
        <wsdl:fault name="UnknownResource"
                    message="wsrl:ErrorMessage" />
        <wsdl:fault name="UnableToDestroyResource"
                    message="wsrl:ErrorMessage" />
      </wsdl:operation>

      <!-- === extends wsrl:ScheduledResourceTermination ========== -->
      <wsdl:operation name="SetTerminationTime">
        <wsdl:input message="wsrl:SetTerminationTimeRequest" />
        <wsdl:output message="wsrl:SetTerminationTimeResponse" />
        <wsdl:fault name="UnknownResource"
                    message="wsrl:ErrorMessage" />
        <wsdl:fault name="TerminationTimeChangeRejected"
                    message="wsrl:ErrorMessage" />
        <wsdl:fault name="UnableToSetTerminationTime"
                    message="wsrl:ErrorMessage" />
      </wsdl:operation>

      <!-- ========= SubscriptionManager specific operations ====== -->
      <wsdl:operation name="PauseSubcription">
         <wsdl:input message="wsnt:PauseSubscriptionRequest"/>
         <wsdl:output message="wsnt:PauseSubscriptionResponse"/>
         <wsdl:fault name="PauseFailed"
                     message="wsnt:ErrorMessage" />

      </wsdl:operation>
      <wsdl:operation name="ResumeSubscription">
         <wsdl:input message="wsnt:ResumeSubscriptionRequest"/>
         <wsdl:output message="wsnt:ResumeSubscriptionResponse"/>
         <wsdl:fault name="ResumeFailed"
                     message="wsnt:ErrorMessage" />
      </wsdl:operation>
   </wsdl:portType>

<!-- ========== NotificationBroker PortType Definition ============ -->
   <wsdl:portType name="NotificationBroker"
      wsrp:ResourceProperties ="wsnt:NotificationBrokerRP">
      <!-- ============= extends NotificationConsumer ============= -->
      <wsdl:operation name="Notify">
         <wsdl:input message="wsnt:Notify" />
      </wsdl:operation>

      <!-- ============= extends NotificationProducer ============= -->
      <!-- ========== extends wsrp:ResourceProperties ============= -->
      <wsdl:operation name="GetResourceProperty">
        <wsdl:input message="wsrp:GetResourcePropertyRequest" />
        <wsdl:output message="wsrp:GetResourcePropertyResponse" />
        <wsdl:fault name="UnknownResource"
                    message="wsrp:ErrorMessage" />
        <wsdl:fault name="InvalidResourceProperty"
                    message="wsrp:ErrorMessage" />
      </wsdl:operation>

      <!-- ========== NotificationProducer Specific ============== -->
      <wsdl:operation name="Subscribe">
         <wsdl:input message="wsnt:SubscribeRequest" />
```

```
         <wsdl:output message="wsnt:SubscribeResponse" />
         <wsdl:fault name="SubscribeCreationFailed"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
      <wsdl:operation name="GetCurrentMessage">
         <wsdl:input message="wsnt:GetCurrentMessageRequest"/>
         <wsdl:output message="wsnt:GetCurrentMessageResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="NoCurrentMessageOnTopic"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
      <wsdl:operation name="Pause">
         <wsdl:input message="wsnt:PauseRequest"/>
         <wsdl:output message="wsnt:PauseResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="PauseFailed"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>
      <wsdl:operation name="Resume">
         <wsdl:input message="wsnt:ResumeRequest"/>
         <wsdl:output message="wsnt:ResumeResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="ResumeFailed"
                     message="wsnt:ErrorMessage"/>
      </wsdl:operation>

      <!-- ========= NotificationBroker specific operations ======= -->
      <wsdl:operation name="RegisterPublisher">
         <wsdl:input message="wsnt:RegisterPublisherRequest"/>
         <wsdl:output message="wsnt:RegisterPublisherResponse"/>
         <wsdl:fault name="InvalidTopicPathExpression"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="TopicNotSupported"
                     message="wsnt:ErrorMessage"/>
         <wsdl:fault name="PublisherRegistrationFailed"
                     message="wsnt:ErrorMessage"/>
       </wsdl:operation>
   </wsdl:portType>

<!-- ===== PublisherRegistrationManager PortType Definition ====== -->
   <wsdl:portType name="PublisherRegistrationManager"
      wsrp:ResourceProperties ="wsnt:PublisherRegistrationManagerRP">

   </wsdl:portType>

</wsdl:definitions>
```