Web Services Base Notification 1.3 (WS-BaseNotification)

Public Review Draft 01, 07 July 2005

Document identifier

wsn-ws_base_notification-1.3-spec-pr-01

Location:

http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-pr-01.pdf

Editors:

Steve Graham, IBM <sggraham@us.ibm.com>
David Hull, Tibco <dmh@tibco.com>
Bryan Murray, Hewlett-Packard Company <bryan.murray@hp.com>

Abstract:

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.

WS-Notification is a family of related specifications that define a standard Web services approach to notification using a topic-based publish/subscribe 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. The WS-Notification family of documents includes three normative specifications: [WS-BaseNotification], [WS-BrokeredNotification], and WS-Topics.

Status:

On July 7th, 2005, the OASIS WS-Notification Technical Committee approved this document for publication as a Public Review Draft. Committee members should send comments on this specification to the wsn@lists.oasis-open.org list. Others may submit comments to the TC via the web form found on the TC's web page at http://www.oasis-wsn-ws_base_notification-1.3-spec-pr-01

Copyright © OASIS Open 2004-2005. All Rights Reserved.
open.org/committees/wsn. Click the button for "Send A Comment" at the top of the page.
Submitted comments (for this work as well as other works of that TC) are publicly
archived and can be viewed at:
For information on whether any patents have been disclosed that may be essential to
implementing this specification, and any offers of patent licensing terms, please refer to
the Intellectual Property Rights section of the WSN TC web page (http://www.oasis-
open.org/committees/wsn/).
# Table of Contents

1 Introduction
   1.1 Goals and Requirements
      1.1.1 Requirements
      1.1.2 Non-Goals
   1.2 Terminology
   1.3 Namespaces
   1.4 Fault Definitions

2 Terminology and Concepts
   2.1 Definitions
   2.2 Production vs. Delivery

3 NotificationConsumer Interface
   3.1 Notification Metadata
   3.2 Notify
      3.2.1 Example SOAP Encoding of the Notify Message

4 NotificationProducer Interface
   4.1 NotificationProducer Resource Properties
   4.2 Subscribe
      4.2.1 Example SOAP Encoding of the Subscribe Message Exchange
   4.3 GetCurrentMessage
      4.3.1 Example SOAP Encoding of the GetCurrentMessage Message Exchange

5 Pull-Style Notification
   5.1 PullPoint Interface
      5.1.1 Accumulating Notification Messages
      5.1.2 GetMessages
      5.1.3 Example SOAP Encoding of the GetMessages Message Exchange
      5.1.4 Destroy
      5.1.5 Example SOAP Encoding of the Destroy Message Exchange
   5.2 CreatePullPoint Interface
      5.2.1 Example SOAP Encoding of the CreatePullPoint Message Exchange

6 SubscriptionManager Interface
   6.1 Base Subscription Manager
      6.1.1 Renew
      6.1.2 Unsubscribe
   6.2 Pausable Subscription Manager
      6.2.1 PauseSubscription
      6.2.2 Example SOAP Encoding of the PauseSubscription Message Exchange
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>ResumeSubscription</td>
<td>35</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Example SOAP Encoding of the ResumeSubscription Message Exchange</td>
<td>36</td>
</tr>
<tr>
<td>6.4</td>
<td>Subscriptions as WS-Resources</td>
<td>37</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Subscription Resource Properties</td>
<td>37</td>
</tr>
<tr>
<td>7</td>
<td>Security Considerations</td>
<td>39</td>
</tr>
<tr>
<td>7.1</td>
<td>Securing the Message Exchanges</td>
<td>39</td>
</tr>
<tr>
<td>7.2</td>
<td>Securing Subscriptions and Notifications</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>References</td>
<td>41</td>
</tr>
<tr>
<td>8.1</td>
<td>Normative</td>
<td>41</td>
</tr>
<tr>
<td>8.2</td>
<td>Non-Normative</td>
<td>41</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Acknowledgments</td>
<td>41</td>
</tr>
<tr>
<td>Appendix B</td>
<td>XML Schema</td>
<td>42</td>
</tr>
<tr>
<td>Appendix C</td>
<td>WSDL 1.1</td>
<td>54</td>
</tr>
<tr>
<td>Appendix D</td>
<td>Revision History</td>
<td>63</td>
</tr>
<tr>
<td>Appendix E</td>
<td>Notices</td>
<td>65</td>
</tr>
</tbody>
</table>
1 Introduction

The Event-driven, or Notification-based, interaction pattern is commonly used in 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.

The WS-Notification family of specifications defines a standard Web services approach to notification. This document is the base specification on which all the other specifications in the family depend. It defines the normative Web services interfaces for two of the important roles in the notification pattern, namely the NotificationProducer and NotificationConsumer roles. This specification includes standard message exchanges to be implemented by service providers that wish to act in these roles, along with operational requirements expected of them.

In the remainder of this section and section 2 we will give a brief introduction to the Notification pattern, and the terms we will use in this specification.

In the Notification pattern a Web service, or other entity, disseminates information to a set of other Web services, without having to have prior knowledge of these other Web services.

This specification defines a role called the NotificationProducer. A NotificationProducer is capable of producing a set of Notification messages. A NotificationProducer accepts incoming Subscribe requests. Each Subscribe request contains a reference to a NotificationConsumer and identifies the subset of the Notifications the NotificationProducer should produce. This subset can be described by identifying one or more boolean filters, including filtering by Topic, as discussed in [WS-Topics]. The NotificationProducer agrees to produce Notification Messages as requested in the Subscribe request, or returns a fault if the subscription cannot be handled.

The production of Notifications may be realized in a number of ways. One particular configuration, in which the NotificationProducer reproduces Notifications produced by other entities, is described in the [WS-Brokered Notification] specification. Alternatively, a NotificationProducer may produce Notifications itself. An implementer interested only in such direct, point-to-point, notification need only refer to this WS-BaseNotification specification.

1.1 Goals and Requirements

The goal of WS-BaseNotification is to standardize the terminology, concepts, operations, WSDL and XML needed to express the basic roles involved in Web services publish and subscribe for notification message exchange.

1.1.1 Requirements

In meeting these goals, the WS-BaseNotification specification must explicitly address the following requirements:

- **Must support resource-constrained devices.** The specifications must be factored in a way that allows resource-constrained devices to participate in the Notification pattern. Such devices will be able to send information to, and receive information from Web services, without having to implement all the features of the specifications.

- **Must provide runtime metadata:** There must be a mechanism that lets a potential Subscriber discover what elements are provided for subscription by a NotificationProducer, and in what formats the subscription for notification can be made.

In addition, the WS-BaseNotification must allow for the following requirements to be met:
• **WS-BaseNotification 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 the WS-BaseNotification must allow a service that is acting as a NotificationProducer to delegate its implementation of WS-BaseNotification semantics to a Message Oriented Middleware provider.

• **Relationship to other WS-* specifications:** WS-BaseNotification must be composable with other Web services specifications.

### 1.1.2 Non-Goals

The following topics are outside the scope of the WS-BaseNotification specification:

• **Defining the format of notification payloads:** The data carried in Notification payloads is application-domain specific, and WS-BaseNotification does not prescribe any particular format for this data.

• **Defining any Notifications.** The WS-BaseNotification specification does not define any "standard" or "built-in" notification situations or messages.

• **Defining the mapping between Situations and Notifications.** The WS-BaseNotification specification does not define the circumstances under which a potential producer of information should decide if and when it should actually produce notifications. However they do define the form and semantics of the notification once it has decided to do so.

• **Defining the means by which NotificationProducers are discovered by subscribers.** It is beyond the scope of this specification to define the mechanisms for runtime discovery of NotificationProducers.

• **Defining message ordering or interleaving policies for delivery is beyond the scope of WS-BaseNotification.**

### 1.2 Terminology

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]).

This specification uses a notational convention, referred to as "Pseudo-schemas" in a fashion similar to the WSDL 2.0 Part 1 specification [WSDL 2.0]. A Pseudo-schema uses a BNF-style convention to describe attributes and elements:

- `?` denotes optionality (i.e. zero or one occurrences),
- `*` denotes zero or more occurrences,
- `+` one or more occurrences,
- `|` represents choice.
- `[]` and `[]` are used to form groups,
• Attributes are conventionally assigned a value which corresponds to their type, as defined in the normative schema.

```xml
<!-- sample pseudo-schema -->
<element
    required_attribute_of_type_QName="xs:QName"
    optional_attribute_of_type_string="xs:string" >
    <required_element />
    <optional_element /> ?
    <one_or_more_of_these_elements /> +
    [ <choice_1 /> | <choice_2 /> ] *
</element>
```

Where there is disagreement between the separate XML schema and WSDL files describing the messages defined by this specification and the normative descriptive text (excluding any pseudo-schema) in this document, the normative descriptive text will take precedence over the separate files. The separate files take precedence over any pseudo-schema and over any schema and WSDL included in the appendices.

### 1.3 Namespaces
The following namespaces are used in this document:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a> OR <a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a></td>
</tr>
<tr>
<td>xsd</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
</tr>
<tr>
<td>wsa</td>
<td><a href="http://www.w3.org/2005/03/addressing">http://www.w3.org/2005/03/addressing</a></td>
</tr>
<tr>
<td>wsrfrp</td>
<td><a href="http://docs.oasis-open.org/wsrfrp-1">http://docs.oasis-open.org/wsrfrp-1</a></td>
</tr>
<tr>
<td>wsrfbf</td>
<td><a href="http://docs.oasis-open.org/wsrfbf-1">http://docs.oasis-open.org/wsrfbf-1</a></td>
</tr>
<tr>
<td>wsn</td>
<td><a href="http://docs.oasis-open.org/wsn/b-1">http://docs.oasis-open.org/wsn/b-1</a></td>
</tr>
<tr>
<td>wsntw</td>
<td><a href="http://docs.oasis-open.org/wsn/bw-1">http://docs.oasis-open.org/wsn/bw-1</a></td>
</tr>
</tbody>
</table>

### 1.4 Fault Definitions
All faults generated by a NotificationProducer or SubscriptionManager SHOULD be compliant with the WS-BaseFaults [WS-BaseFaults] specification.

All faults defined by this specification MUST use the following URI for the WS-Addressing [action] Message Addressing Property:

http://docs.oasis-open.org/wsn/fault.
2 Terminology and Concepts

2.1 Definitions

The following definitions outline the terminology and usage in this specification.

**Situation:**
- A Situation is some occurrence known to a NotificationProducer and of potential interest to third parties.
- A Situation could be a change of the internal state of a 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 Notification(s) that are used to describe it.

**Notification:**
- A Notification is an artifact of a Situation containing information about that Situation that some entity wishes to communicate to other entities.
- A Notification 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 Notification 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 Notification instance as in some sense being (or at least representing) the Situation.
- A designer could choose to associate several different Notification 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 Notifications of the same type.

**NotificationProducer:**
- A NotificationProducer is a Web service that implements the message exchanges associated with the NotificationProducer interface.
- A NotificationProducer is capable of producing Notifications for those NotificationConsumers for which Subscriptions have been registered, based on Situations that occur and on the parameters supplied with the requests from which the Subscriptions were created.
- A Web Service that implements the message exchanges associated with NotificationProducer may directly produce Notifications itself, or it may be a NotificationBroker, reproducing Notifications that were produced by separate Publisher and/or NotificationProducer entities.
- It is the factory for Subscription resources.

**NotificationConsumer:**
- A NotificationConsumer is an endpoint, represented by a WS-Addressing endpoint reference, designated to receive Notifications produced by a NotificationProducer as a result of a subscription.
- A NotificationConsumer may accept the generic Notify message, or it may be able to process one or more domain-specific Notification types.
Subscription:

- A Subscription represents the relationship between a NotificationConsumer and a NotificationProducer, including any filtering parameters such as Topic and various other optional filter expressions, along with any relevant policies and context information.

- A Subscription resource is created when a Subscriber sends the SubscribeRequest message to a NotificationProducer.

- Subscription resources are manipulated by messages sent to the SubscriptionManager Web service associated with the Subscription resource.

SubscriptionManager

- A SubscriptionManager is an endpoint, represented by an endpoint reference [WS-Addressing] that implements message exchanges associated with the SubscriptionManager interface.

- A SubscriptionManager provides operations that allow a service requestor to query and manipulate Subscription resources that it manages.

- A SubscriptionManager is subordinate to the NotificationProducer, and MAY be implemented by the NotificationProducer service provider. However WS-Notification permits it to be implemented by a separate service provider, should an implementer so desire.

Subscriber:

- A Subscriber is any entity that sends the SubscribeRequest message to a NotificationProducer.

- Note that a Subscriber may be a different entity from the NotificationConsumer for which Notifications are actually produced.

2.2 Production vs. Delivery

Various statements in this document refer to a NotificationProducer producing a Notification. Producing a Notification means: supplying a Notification to a delivery mechanism for transmission to a NotificationConsumer. Depending on the actual delivery mechanism, this transmission might be reliable or might be done on a best-effort basis. A Notification which is never produced will definitely never be delivered, but the converse is not necessarily true: a Notification which is produced may or may not actually be delivered, depending on the delivery mechanism, the validity of the NotificationConsumer address, the state of the network, and so forth.
3 NotificationConsumer Interface

WS-BaseNotification allows a NotificationConsumer to receive a Notification in one of two forms:

1. The NotificationConsumer MAY simply receive the “raw” Notification (i.e. the application-specific content).

2. The NotificationConsumer MAY receive the Notification data as a Notify message as described below.

The second option (the Notify message) provides a well specified mechanism by which the NotificationProducer can supply additional information defined by WS-BaseNotification (such as the Topic) in addition to the application-specific Notification content. It also allows the NotificationConsumer to receive a wide range of Notifications without having to explicitly provide support for each one in its WSDL. This form of Notification also allows a batch of several Notifications to be combined into a single physical message.

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

This means that a NotificationConsumer MAY choose to implement the Notify Message, or to implement raw Notification(s) explicitly (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 form of Notification it has requested for the given Subscription.

3.1 Notification Metadata

In order to inform the NotificationConsumer about the Situation that produced a Notification and to allow the NotificationConsumer to manipulate the underlying subscription, the NotificationProducer MAY include any combination of the following metadata elements in the Notifications it produces:

- wsnt:SubscriptionReference
  - An EndpointReference to the Subscription WS-Resource [WS-Resource] that is associated with the Notify message.

- wsnt:Topic
  - A TopicExpression describing exactly one Topic, which MUST be the Topic that is associated with the Notification. This element describes the Topic that matched to a subscription, causing the NotificationProducer to send the Notify message to the NotificationConsumer.

- wsnt:Topic/@Dialect
  - The dialect used in the TopicExpression. This MUST be the same dialect used by the Subscriber when it created the Subscription that yielded this Notify message.

- wsnt:ProducerReference
  - An EndpointReference to the NotificationProducer that produced the Notification artifact.

When using the Notify message format, these elements MUST appear within the wsnt:NotificationMessage element as described below. When using the raw message format, these elements MAY appear as components of the message (such as SOAP header elements) in accordance with any contract with the NotificationConsumer.
3.2 Notify

A NotificationProducer MAY produce a Notify message containing one or more Notification(s).

The format of the Notify message is:

```xml
...<wsnt:Notify>

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

The WS-Addressing [action] Message Addressing Property MUST contain the URI

http://docs.oasis-open.org/wsn/bw-1/NotificationConsumer/Notify.

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

- /wsnt:Notify
  Contains a collection of one or more Notifications.

- /wsnt:Notify/wsnt:NotificationMessage
  Contains a Notification payload along with any metadata components as defined in section 3.1.

- /wsnt:Notify/wsnt:NotificationMessage/wsnt:Message
  A copy of the actual Notification payload.

- /wsnt:Notify/{any}
  The Notify message also allows for open content, in order to accommodate elements that may be needed by extensions built on BaseNotification, including those providing additional filtering mechanisms.

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

3.2.1 Example SOAP Encoding of the Notify Message

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

```xml
<s:Envelope ...>
  <s:Header>
    ...
  </s:Header>
</s:Envelope>
```
<wsa:Action>
  http://docs.oasis-open.org/wsn/bw-1/NotificationConsumer/Notify
</wsa:Action>
...
</s:Header>
<s:Body>
  <wsnt:Notify>
    <wsnt:NotificationMessage>
      <wsnt:SubscriptionReference>
        <wsa:Address>
          http://www.producer.org/SubsMgrEndpoint
        </wsa:Address>
      </wsnt:SubscriptionReference>
      <wsnt:Topic Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple">
        npex:SomeTopic
      </wsnt:Topic>
      <wsnt:ProducerReference>
        <wsa:Address>
          http://www.producer.org/ProducerEndpoint
        </wsa:Address>
      </wsnt:ProducerReference>
      <wsnt:Message>
        <npex:NotifyContent>exampleNotifyContent</npex:NotifyContent>
      </wsnt:Message>
    </wsnt:NotificationMessage>
  </wsnt:Notify>
</s:Body>
</s:Envelope>
4 NotificationProducer Interface

This section describes the message exchanges 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.

4.1 NotificationProducer Resource Properties

In addition to the message exchanges described in this specification, a NotificationProducer MAY 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. In such a case, this specification defines several resource properties, which MUST conform to the following schema fragment for content:

```xml
... targetNamespace="http://docs.oasis-open.org/wsn/b-1"
...
<xsd:element name="TopicExpression" type="wsnt:TopicExpressionType"/>
<xsd:element name="FixedTopicSet" type="xsd:boolean" default="true"/>
<xsd:element name="TopicExpressionDialect" type="xsd:anyURI"/>
...
```

These properties must also conform to the following schema fragment for cardinality:

```xml
<xsd:element ref="wsnt:TopicExpression"
    minOccurs="0" maxOccurs="unbounded" />
<xsd:element ref="wsnt:FixedTopicSet"
    minOccurs="0" maxOccurs="1" />
<xsd:element ref="wsnt:TopicExpressionDialect"
    minOccurs="0" maxOccurs="unbounded" />
```

These Resource Property elements are further constrained as follows:

/wsnt:TopicExpression

This resource property contains a collection of topics supported by the NotificationProducer. The set of topics is expressed using one or more wsnt:TopicExpression resource property element(s). Each wsnt:TopicExpression resource property element is a TopicExpression. The dialect of TopicExpression used can be any dialect. It is RECOMMENDED to use one of the TopicExpression dialects described in [WS-Topics]. A NotificationProducer MAY use an expression that refers to multiple topics, if the dialect used has this capability. The same topic may be referred to...
by multiple wsnt:TopicExpression resource property element(s), for example, in different resource property elements each using a different dialect.

If a topic is identified by one of the wsnt:TopicExpression resource property elements, a Subscriber can reasonably expect that the NotificationProducer will not return an InvalidTopicExpressionFault for subscription requests for the topic. It is not a guarantee that it will receive any Notifications; the NotificationProducer may not actually produce any Notifications on the particular topic during the time that the Subscriber is registered.

/wsnt:FixedTopicSet
Indicates if the collection of topics contained within the wsnt:TopicExpression 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). This property is optional and defaults to “true” if missing.

/wsnt:TopicExpressionDialect
Indicates one or more TopicExpression dialect(s) that are supported by the NotificationProducer. If a URI corresponding to a dialect appears in this resource property, a subscriber is assured that a subscription request containing a valid topic expression using that dialect will be accepted by the NotificationProducer. The TopicExpressionDialect property is a “fixed property”, i.e. its value, for any given NotificationProducer WS-Resource, does not change over time.

4.2 Subscribe
A NotificationProducer is capable of producing a sequence of zero or more Notifications. A Subscriber can register the interest of a NotificationConsumer to receive a subset of this sequence. A Subscriber sends a Subscribe message to a NotificationProducer in order to register this interest.

If the processing of a Subscribe message is successful, the NotificationProducer MUST produce a response message, as described below, containing an endpoint reference representing a Subscription created as a result of processing the Subscribe request.

Sending two identical Subscribe messages to a NotificationProducer MUST result in the creation of two Subscriptions. The NotificationConsumer will be associated with both Subscriptions with the result that two copies of any matching Notification will be produced for that consumer.

A given NotificationConsumer may be the object of more than one Subscription, and separate NotificationConsumers may subscribe to the same subset of Notifications. In such situations, WS-BaseNotification places no restrictions on the order in which Notifications are produced.

Notifications for different NotificationConsumers may be produced in different orders, even when the associated subscription requests are otherwise identical, and Notifications from separate Subscriptions with the same NotificationConsumer may be interleaved in any manner.

NotificationProducers MAY advertise more constrained behavior through policy assertions or other means.

The format of the Subscribe message is:

```xml
...
<wsnt:Subscribe>
```
<wsnt:ConsumerReference>
  wsa:EndpointReference
</wsnt:ConsumerReference>
<wsnt:Filter>
  [ <wsnt:TopicExpression Dialect="xsd:anyURI">
    {any} ?
  </wsnt:TopicExpression> | 
  <wsnt:ProducerProperties Dialect="xsd:anyURI">
    {any} ?
  </wsnt:ProducerProperties> | 
  <wsnt:MessageContent Dialect="xsd:anyURI">
    {any}
  </wsnt:MessageContent> | 
  {any} *
  ] *
</wsnt:Filter> ?
<wsnt:InitialTerminationTime>
  [xsd:dateTime | xsd:duration]
</wsnt:InitialTerminationTime> ?
<wsnt:SubscriptionPolicy>
  [ <wsnt:UseRaw/> | 
    {any}
  ] *
</wsnt:SubscriptionPolicy> ?
{any}*
</wsnt:Subscribe>


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

/wsnt:ConsumerReference

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

This component SHOULD provide all the necessary information to specify how the NotificationProducer should send notifications to the NotificationConsumer. However, there may be cases when the ConsumerReference EPR may not include all the details that the NotificationProducer expects.

The NotificationProducer should specify via WSDL, policy assertions, meta-data or by some other means, the information it expects to be present in a ConsumerReference. If a ConsumerReference does not contain sufficient information, the NotificationProducer MAY choose to fault or it MAY choose to use out of band mechanisms to obtain the required information.

In cases where the optional wsnt:UseRaw policy component is not specified, the Web service identified by the endpoint reference MUST implement the message exchanges defined by NotificationConsumer (i.e., the Notify message).
The Filter component is the means by which a Subscriber expresses the subset of Notifications that the NotificationConsumer should receive. This subset is expressed by the child elements of the Filter. The child elements are a sequence of zero or more expressions evaluating to a Boolean that constrain the set of possible Notifications. Each expression is evaluated in a manner specific to that kind of expression (see below); the order and timing of the evaluation is determined by the NotificationProducer. All filter expressions MUST evaluate to true in order for the notification message to be sent.

If no Filter component appears in a Subscribe message, then the Subscriber’s intent is for the NotificationConsumer to receive every message produced by the NotificationProducer. The NotificationProducer MAY reject Subscribe requests that do not contain a Filter component.

The NotificationProducer MUST respond with an InvalidFilterFault message if any child expression element is not supported by the NotificationProducer. For example, if the NotificationProducer does not support the concept of Topics, it MUST respond with an InvalidFilterFault message if a Subscribe message contains a Filter component that includes a TopicExpression child. The fault MUST include the filter QNames for the filters it did not understand.

This specification defines the filters TopicExpression, ProducerProperties, and MessageContent. A NotificationProducer MAY accept these filters and/or any other filters that may be defined.

A filter limiting notification messages to those that are associated with at least one topic matching the TopicExpression. The TopicExpression identifies one or more topics supported by the NotificationProducer.

A REQUIRED attribute of type URI that identifies the language of the TopicExpression. WS-Topics defines an initial set of standard URIs for TopicExpressions. Designers MAY define and use other domain-specific URIs to identify the dialect of the TopicExpression. The NotificationProducer MAY refuse to process the Subscribe request if the dialect used by the Subscriber in the TopicExpression is not one of the dialects supported by the NotificationProducer.

The NotificationProducer MUST respond with a TopicExpressionDialectUnknownFault if it understands the TopicExpression element, but does not understand the specified TopicExpression dialect. Note that a NotificationProducer may understand the meaning of a given dialect URI without actually supporting that dialect.

If the TopicExpression is incompatible with or does not comply with the rules of the dialect, the NotificationProducer MUST respond with a InvalidTopicExpressionFault.

If the TopicExpression dialect is understood and the expression references a topic which is not supported by the NotificationProducer, the NotificationProducer MAY return a TopicNotSupportedFault.
This component contains a filter expression evaluated on the ResourceProperties [WS-
ResourceProperties] of the NotificationProducer. The expression MUST be a Boolean
equation.

This attribute contains a URI specifying the type of ProducerProperties filter expression
contained by the element. Some standard URIs are defined by the WS-
ResourceProperties specification. Designers MAY define and use other domain-specific
URIs to identify the dialect of the ProducerProperties filter expression.

The NotificationProducer MAY refuse to process the Subscribe request if the Dialect
used by the Subscriber is not one of the dialects supported by the NotificationProducer.

A filter limiting notification messages to the set for which the specified expression
evaluated over the Notification Message to be produced evaluates to true. A
wsnt:MessageContent expression MUST evaluate to a Boolean. The evaluation context
is Notification payload.

This attribute contains a URI specifying the type of MessageContent filter expression
contained by the element. There are two well known dialects identified by this
specification, corresponding to two versions of the XPath language.

http://www.w3.org/TR/1999/REC-xpath-19991116
This URI identifies the XPath 1.0 language. The contents of the
MessageContent expression MUST be a string containing a valid XPath
1.0 expression.

http://www.w3.org/TR/2005/WD-xpath2-20050404
This URI identifies the Xpath 2.0 (working draft) language. The contents
of the MessageContent expression MUST be a string containing a valid
XPath 2.0 expression. Note: an additional URI will be added to represent
the W3C Recommendation form of the XPath 2.0 language.

Designers MAY define and use other domain-specific URIs to identify the dialect of the
MessageContent filter expression.

The NotificationProducer MAY refuse to process the Subscribe request if the dialect used
by the Subscriber is not one of the dialects supported by the NotificationProducer.

This component contains the service requestor’s suggestion for the initial termination
time of the Subscription being created. There are two forms of this component, absolute
time and duration. If the type of this component is xsd:dateTime, the value of the
component is to be interpreted as an absolute time. If the type of this component is
xsd:duration, the value of the component is to be interpreted as a duration to be added to
the current time. All time measurements are determined by the NotificationProducer.

The resulting absolute time, whether computed from a duration or given explicitly in the
request message, is used to initialize the value of the TerminationTime resource property
of the Subscription resource.
If the NotificationProducer is unable or unwilling to set the TerminationTime resource property of the Subscription resource to the requested time or a value greater, or if this requested time is not in the future, then the NotificationProducer MUST return an UnacceptableInitialTerminationTimeFault message.

The use of the xsi:nil attribute with value “true” indicates there is no scheduled termination time requested for the Subscription, implying that the requested Subscription has infinite duration.

If the element does not include the time zone designation, the value of the element MUST be interpreted as universal time (UTC) time.

If this component is not included, the initial value of the TerminationTime resource property is dependent on the implementation of the NotificationProducer.

/wsnt:SubscriptionPolicy
This optional component is an open component intended to be used in an application specific way to specify policy related requirements/assertions associated with the subscribe requests. This mechanism could be used to govern the message rate (e.g. maximum 3 messages per second), reliability of the Notification delivery, etc. The semantics of how the NotificationProducer MUST or MAY react to the policy requirements and assertions appearing in this component are specific to the actual policy grammar used.

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.

/wsnt:SubscriptionPolicy/wsnt:UseRaw
An element whose presence indicates that the NotificationProducer is to produce Notifications without using the Notify wrapper. This component is optional, if missing the default behavior is to use the Notify wrapper for all notification messages. This element MUST NOT occur more than once in a Subscribe request message.

If this element is not present then the NotificationProducer MUST use the wsnt:Notify message in Notifications produced for this subscription. In this case the NotificationConsumer referred to in the wsnt:ConsumerReference element SHOULD implement the Notify message and include a Notify operation in its portType definition.

If this element is present then the NotificationProducer SHOULD produce the raw Notification itself.

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

```xml
<wsnt:SubscribeResponse>
  <wsnt:SubscriptionReference>
    wsa:EndpointReference
  </wsnt:SubscriptionReference>
  <wsnt:CurrentTime>xsd:dateTime</wsnt:CurrentTime> ?
  <wsnt:TerminationTime>xsd:dateTime</wsnt:TerminationTime> ?
  {any}*
</wsnt:SubscribeResponse>
```
The WS-Addressing [action] Message Addressing Property MUST contain the URI


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

/wsnt:SubscriptionReference

A WS-Resource reference to the Subscription WS-Resource created as a result of the
Subscribe message.

/wsnt:CurrentTime

This OPTIONAL component SHOULD be returned if the SubscriptionManager uses
scheduled termination from WS-ResourceLifetime. The value of this component is the
value of the CurrentTime resource property of the Subscription at the time the response
message is created.

/wsnt:TerminationTime

This OPTIONAL component SHOULD be returned if the SubscriptionManager uses
scheduled termination from WS-ResourceLifetime. The value of this component is the
value of the TerminationTime resource property of the Subscription at the time the
response message is created.

If the NotificationProducer does not respond to the Subscribe request message with the
SubscribeResponse message, then it MUST send a fault. This specification defines the following
faults associated with failure to process the Subscribe request message:

ResourceUnknownFault

- The NotificationProducer is a WS-Resource, and the resource identified in the message
  is not known to the Web service. This fault is specified by the WS-Resource [WS-

InvalidFilterFault

- The Subscribe message contained a filter that was not understood or supported by the
  NotificationProducer.

TopicExpressionDialectUnknownFault

- The Subscribe message contained a TopicExpression filter having a dialect that was not
  understood or supported by the NotificationProducer.

InvalidTopicExpressionFault

- The Subscribe message contained a TopicExpression filter where the contents of the
  filter did not match the dialect specified.

TopicNotSupportedFault

- The Subscribe message contained a TopicExpression filter that referenced a topic that
  was not supported by the NotificationProducer.

InvalidProducerPropertiesExpressionFault

- The Subscribe message contained a ProducerProperties filter that did not represent a
  valid boolean expression.

InvalidMessageContentExpressionFault
• The Subscribe message contained a MessageContent filter that did not represent a valid boolean expression.

InvalidUseRawValueFault

• The RawValue element appeared more than once in the SubscriptionPolicy.

UnacceptableInitialTerminationTimeFault

• The value of InitialTerminationTime specified in the Subscribe message was not acceptable to the NotificationProducer. The NotificationProducer MAY include a hint in the fault message indicating acceptable values for InitialTerminationTime.

SubscribeCreationFailedFault

• The NotificationProducer failed to process the Subscribe message. The NotificationProducer SHOULD use a more specific fault message if possible. The NotificationProducer MAY include a hint in the fault message indicating why it failed to process the Subscribe message.

4.2.1 Example SOAP Encoding of the Subscribe Message Exchange

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

```xml
<s:Envelope ...
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/NotificationProducer/SubscribeRequest
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsnt:Subscribe>
      <wsnt:ConsumerReference>
        <wsa:Address>
          http://www.consumer.org/ConsumerEndpoint
        </wsa:Address>
      </wsnt:ConsumerReference>
      <wsnt:Filter>
        <wsnt:TopicExpression Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple">
          npex:SomeTopic
        </wsnt:TopicExpression>
        <wsnt:MessageContent Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
          boolean(ncex:Producer="15")
        </wsnt:MessageContent>
        <wsnt:Filter>
          <wsnt:InitialTerminationTime>2005-12-25T00:00:00.000002</wsnt:InitialTerminationTime>
        </wsnt:Filter>
      </wsnt:Filter>
    </wsnt:Subscribe>
  </s:Body>
</s:Envelope>
```
The following is a non-normative example of a Subscribe response message using SOAP:

```xml
<s:Envelope ...
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/NotificationProducer/SubscribeResponse
    </wsa:Action>
  ...
  </s:Header>
  <s:Body>
    <wsnt:SubscribeResponse>
      <wsnt:SubscriptionReference>
        <wsa:Address>
          http://www.producer.org/SubsMgrEndpoint
        </wsa:Address>
      </wsnt:SubscriptionReference>
    </wsnt:SubscribeResponse>
  </s:Body>
</s:Envelope>
```

### 4.3 GetCurrentMessage

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

In certain circumstances, a NotificationProducer MAY choose to not cache the last Notification to one or more Topics it supports. In such cases, the NotificationProducer MUST respond with a fault message indicating that no current message is available on that Topic. The NotificationProducer MAY choose to communicate its caching policy by some means not specified in this document, such as using a policy assertion.

The format of the GetCurrentMessage request message is:

```xml
... <wsnt:GetCurrentMessage>
  <wsnt:Topic Dialect="xsd:anyURI">
    {any}
  </wsnt:Topic>
  {any} *
</wsnt:GetCurrentMessage> ...
```


The components of the GetCurrentMessage request message are further described as follows:
A TopicExpression that identifies exactly one Topic.

If the NotificationProducer successfully processes the GetCurrentMessage request, it MUST respond with a GetCurrentMessageResponse message. This response has the following form:

```xml
<wsn:GetCurrentMessageResponse>
    {any} *
</wsnt:GetCurrentMessageResponse>
```


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

/wsnt:GetCurrentMessageResponse/{any}

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

If the NotificationProducer does not respond to the GetCurrentMessage request message with the GetCurrentMessageResponse message, then it MUST send a fault. This specification defines the following faults associated with failure to process the GetCurrentMessage request message:

- **ResourceUnknownFault**
  - The NotificationProducer is acting as a WS-Resource, and the resource identified in the message (which follows the WS-Resource Access Pattern) is not known to the Web service. This fault is specified by the WS-Resource [WS-Resource] specification.

- **TopicExpressionDialectUnknownFault**
  - The Topic element of the GetCurrentMessage message had a dialect that was not understood or supported by the NotificationProducer.

- **InvalidTopicExpressionFault**
  - The Topic element of the GetCurrentMessage message had contents that did not match the dialect specified.

- **TopicNotSupportedFault**
  - The Topic element of the GetCurrentMessage message referenced a topic that was not supported by the NotificationProducer.

- **MultipleTopicsSpecifiedFault**
  - The GetCurrentMessage message referenced more than one topic.

- **NoCurrentMessageOnTopicFault**
  - The topic referenced in the GetCurrentMessage message does not have any pending messages.

### 4.3.1 Example SOAP Encoding of the GetCurrentMessage Message Exchange

The following is a non-normative example of a GetCurrentMessage request message using SOAP:
The following is a non-normative example of a GetCurrentMessage response message using SOAP:

```xml
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/NotificationProducer/GetCurrentMessageResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsnt:GetCurrentMessageResponse>
      <npex:NotifyContent>exampleNotifyContent</npex:NotifyContent>
    </wsnt:GetCurrentMessageResponse>
  </s:Body>
</s:Envelope>
```
5 Pull-Style Notification

There are certain circumstances in which the basic “push-style” of NotificationMessage delivery is not appropriate. For example, certain NotificationConsumers are behind a firewall such that the NotificationProducer cannot initiate a message exchange to send the Notification. A similar circumstance exists for NotificationConsumers that are unable or unwilling to provide an endpoint to which the NotificationProducer can send Notification Messages. In other situations, the NotificationConsumer prefers to control the timing of receipt of Notification Messages, instead of receiving Notification Messages at unpredictable intervals, it may prefer to “pull” “retrieve” the Notification Messages at a time of its own choosing.

For these and other reasons, WS-BaseNotification defines a pair of portTypes: a PullPoint interface, defining an endpoint that accumulates Notification Messages and allows a requestor to retrieve accumulated Notification Messages and a CreatePullPoint interface that acts as a factory for PullPoint resources. The intended pattern of use is that a Subscriber or other party creates a PullPoint through the factory interface, and then uses it as the ConsumerReference in one or more Subscribe requests. The actual consumer then pulls Notifications from the PullPoint.

5.1 PullPoint Interface

In support of the “pull-style” of Notification Message delivery, this specification describes a PullPoint resource by defining a PullPoint interface. The PullPoint interface provides the means by which NotificationProducers can insert Notification Messages into the PullPoint, by which requestors can retrieve messages from the PullPoint and the means by which requestors can destroy the PullPoint. Note: the PullPoint MAY be a WS-Resource, and if it is, request messages defined in this specification MUST follow the WS-Resource Access Pattern defined by [WS-Resource] and the PullPoint WS-Resource MUST support the immediate termination interface defined by WS-RF Resource Lifetime and it MAY support the scheduled termination interface defined by WS-RF Resource Lifetime.

5.1.1 Accumulating Notification Messages

The PullPoint interface supports the NotificationConsumer interface (as defined in section 3). This interface allows NotificationProducers to send Notification Messages to the PullPoint using either a “raw” or a “wrappered” approach. Notification Messages received by the PullPoint through its NotificationConsumer interface are accumulated by the PullPoint on a best effort basis. If the PullPoint is no longer capable of accumulating additional Notification Messages, it MAY ignore Notification Messages sent to its NotificationConsumer interface, or it MAY dispose of any previously accumulated Notification Messages and continue to accumulate incoming Notification Messages. The PullPoint interface does not define additional constraints on its use of the NotificationConsumer interface.

5.1.2 GetMessages

The PullPoint interface provides a message exchange to allow requestors to retrieve (or pull) Notification Messages from the PullPoint.

If a requestor wishes to retrieve Notification Messages accumulated by the PullPoint, it sends a GetMessages request to the PullPoint endpoint. The format of the GetMessages request message is:
The WS-Addressing [action] Message Addressing Property MUST contain the URI
http://docs.oasis-open.org/wsn/bw-1/PullPoint/GetMessagesRequest.

The components of the GetMessages request message are further described as follows:

/wsnt:GetMessages
This component requests that messages held by the PullPoint be returned. A response is
generated without waiting for messages to be accumulated by the PullPoint.

/wsnt:GetMessages/wsnt:MaximumNumber
This optional non-negative integer indicates the maximum number of accumulated
Notification Messages to be returned in the response message. If the total number of
Notification Messages accumulated by the PullPoint is less than or equal to this number,
all of the Notification Messages are returned in the response message. If the total
number of accumulated Notification Messages exceeds this value, the PullPoint MUST
return exactly this number of accumulated Notification Messages; the implementation of
the PullPoint is free to determine which subset of messages are returned in the response
message in this case. If the MaximumNumber component is not present in the request
message, the requestor is asking to receive all messages held by the PullPoint.

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

/wsnt:GetMessagesResponse
This component contains zero or more accumulated Notification Messages. The number
of messages appearing is limited by the wsnt:MaximumNumber component of the
GetMessages request message. Each accumulated Notification Message appearing in a
GetMessagesResponse is “removed” from the PullPoint and MUST not appear in the
response message of subsequent GetMessages requests.

/wsnt:GetMessagesResponse/wsnt:NotificationMessage
The content of this component is an accumulated NotificationMessage. The
NotificationMessage component is described as part of the Notify message (section 3.2)
the GetMessagesResponse message does not define additional constraints on the
NotificationMessage component. If the accumulated NotificationMessage was added to
the PullPoint through a Notify message, the contents of the NotificationMessage is
exactly the contents of the NotificationMessage component of the Notify message. If the
accumulated NotificationMessage was added to the PullPoint through a “raw” message,
then the contents of the NotificationMessage is formed by inserting the body of the “raw”
message into the wsnt:Message component of the

If the PullPoint does not respond to the GetMessages request message with the
GetMessagesResponse message, then it MUST send a fault. This specification defines the
following faults associated with failure to process the GetMessages request message:

ResourceUnknownFault

- The resource identified in the message (which follows the WS-Resource Access Pattern)
is not known to the Web service. This fault is specified by the WS-Resource [WS-

5.1.3 Example SOAP Encoding of the GetMessages Message
Exchange

The following is a non-normative example of a GetMessages request message using SOAP:

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/PullPoint/GetMessagesRequest
    </wsa:Action>
  </s:Header>
  <s:Body>
    <wsnt:GetMessages>
      <wsnt:MaximumNumber>10</wsnt:MaximumNumber>
    </wsnt:GetMessages>
  </s:Body>
</s:Envelope>
```

The following is a non-normative example of a GetMessages response message using SOAP:

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/PullPoint/GetMessagesResponse
    </wsa:Action>
  </s:Header>
  <s:Body>
    <wsnt:GetMessagesResponse>
      <wsnt:NotificationMessage>
        ...
      <wsnt:NotificationMessage>
      <wsnt:NotificationMessage>
        ...
    </wsnt:GetMessagesResponse>
  </s:Body>
</s:Envelope>
```
5.1.4 Destroy

The PullPoint interface provides a destroy operation, providing a means by which a requestor can terminate the PullPoint resource. To terminate a PullPoint resource, a requestor MUST send a Destroy request message to the PullPoint. The Destroy request message has the following form:

```xml
<wsnt:Destroy>
    {any} *
</wsnt:Destroy>
```


Upon receipt of the Destroy request, the PullPoint MUST attempt to destroy itself. If the Destroy request message is successfully processed, the PullPoint MUST respond with the following message:

```xml
<wsnt:DestroyResponse>
    {any} *
</wsnt:DestroyResponse>
```


If the PullPoint does not respond to the Destroy request message with the DestroyResponse message, then it MUST send a fault. This specification defines the following faults associated with failure to process the Destroy request message:

- UnableToDestroyPullPointFault.

5.1.5 Example SOAP Encoding of the Destroy Message Exchange

The following is a non-normative example of a Destroy request message using SOAP:

```xml
<s:Envelope ... >
    <s:Header>
        <wsa:Action>
            http://docs.oasis-open.org/wsn/bw-1/PullPoint/DestroyRequest
        </wsa:Action>
    </s:Header>
    <s:Body>
        ...
    </s:Body>
</s:Envelope>
```
The following is a non-normative example of a Destroy response message using SOAP:

```
<wsnt:Destroy/>
</s:Body>
</s:Envelope>

The following is a non-normative example of a Destroy response message using SOAP:

```
<s:Envelope ... >
 <s:Header>
  <wsa:Action>
   http://docs.oasis-open.org/wsn/bw-1/PullPoint/DestroyResponse
  </wsa:Action>
  ...
 </s:Header>
 <s:Body>
  <wsnt:DestroyResponse/>
 </s:Body>
</s:Envelope>
```

### 5.2 Create PullPoint Interface

This specification describes a CreatePullPoint interface standardizing the means by which a PullPoint resource is created.

If a requestor wishes to create a new PullPoint resource, it MUST send a CreatePullPoint request to an endpoint supporting the PullPoint interface. The CreatePullPoint request message has the following form:

```
<wsnt:CreatePullPoint>
 {any} *
</wsnt:CreatePullPoint>
```

The WS-Addressing [action] Message Addressing Property MUST contain the URI http://docs.oasis-open.org/wsn/bw-1/CreatePullPoint/CreatePullPointRequest

Upon receipt of the CreatePullPoint request, the receiver MUST attempt to create a new PullPoint resource. If the CreatePullPoint request message is successfully processed, the receiver MUST respond with the following message:

```
<wsnt:CreatePullPointResponse>
 <wsnt:PullPoint>wsa:EndpointReferenceType</wsnt:PullPoint>
 {any} *
</wsnt:CreatePullPointResponse>
```


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

```
/wsnt:CreatePullPointResponse/wsnt:PullPoint
```
This component is an EndpointReference, as defined by WS-Addressing, that is a reference to the PullPoint resource created during the processing of the CreatePullPoint request message.

If the receiver does not respond to the CreatePullPoint request message with the CreatePullPointResponse message, then it MUST send a fault. This specification defines the following faults associated with failure to process the Destroy request message:

- UnableToCreatePullPointFault.

### 5.2.1 Example SOAP Encoding of the CreatePullPoint Message Exchange

The following is a non-normative example of a CreatePullPoint request message using SOAP:

```xml
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/PullPoint/CreatePullPointRequest
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsnt:CreatePullPoint/>
  </s:Body>
</s:Envelope>
```

The following is a non-normative example of a CreatePullPoint response message using SOAP:

```xml
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/PullPoint/CreatePullPointResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsnt:CreatePullPointResponse>
      <wsnt:PullPoint>
        <wsa:Address>...</wsa:Address>
        ...
      </wsnt:PullPoint>
    </wsnt:CreatePullPointResponse>
  </s:Body>
</s:Envelope>
```
6 SubscriptionManager Interface

The SubscriptionManager interface defines message exchanges to manipulate Subscription resources. There are two styles of SubscriptionManager interface: base and pausable. All SubscriptionManagers MUST implement the message exchanges described in the Base Subscription Manager section.

For those environments that choose to implement a Subscription as a WS-Resource, the SubscriptionManager MUST implement the resource properties and message exchanges defined in Subscriptions as WS-Resources section. Both the Base Subscription Manager and the Pausable Subscription Manager MAY expose Subscriptions as WS-Resources.

6.1 Base Subscription Manager

The basic behavior of a SubscriptionManager is to renew the duration of a Subscription resource and terminate a Subscription.

6.1.1 Renew

To modify the current lifetime of a Subscription, a requestor sends a Renew request message to the SubscriptionManager. The Renew request message MUST have the following form:

```
...  
<wsnt:Renew>
  <wsnt:TerminationTime>
    [xsd:dateTime | xsd:duration]
  </wsnt:TerminationTime>
</wsnt:Renew>
...  
```


The components of the Renew request message are further described as follows:

**/wsnt:TerminationTime**

This component contains the service requestor’s suggestion for change in expiration, or termination time of the Subscription resource. There are two forms of this component, absolute time and duration. If the type of this component is xsd:dateTime, the value of the component is to be interpreted as an “absolute time”. If the type of this component is xsd:duration, the value of the component is to be interpreted as a “duration”.

The duration form is used to “compute” the “absolute time” form in the following fashion.

The value of this component in “absolute time” form is computed by adding the xsd:duration value to the current time value of the SubscriptionManager.

The “absolute time” form (whether computed from a duration, or contained within the request message) is used to reset the value of the expiration or termination time component of the Subscription resource.

If the SubscriptionManager is unable or unwilling to reset the termination time component of the Subscription resource to the given value of the “absolute time” form or a value
greater, then the Renew request MUST fault. If the value is not “in the future” relative to
the current time as known by the SubscriptionManager, the Renew request MUST fault.

The use of the xsi:nil attribute with value “true” indicates there is no scheduled
termination time requested for the Subscription, implying that the Subscription should be
renewed to have an infinite duration. If the element does not include the time zone
designation, the value of the element MUST be interpreted as universal time (UTC) time.

If the SubscriptionManager successfully processes the Renew request message, it MUST
respond with a RenewResponse message. This response has the following form:

```xml
<wsnt:RenewResponse>
  <wsnt:TerminationTime>xsd:dateTime</wsnt:TerminationTime>
  <wsnt:CurrentTime>xsd:dateTime</wsnt:CurrentTime>
</wsnt:RenewResponse>
```

The WS-Addressing [action] Message Addressing Property MUST contain the URI
http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/RenewResponse.

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

- `/wsnt:TerminationTime`
  - The value of this component is the value of the termination time component of the
    Subscription at the time the response message is created.

- `/wsnt:CurrentTime`
  - This OPTIONAL component contains the value of the CurrentTime according to the
    SubscriptionManager at the time the response message is created.

If the SubscriptionManager does not respond to the Renew request message with the
RenewResponse message, then it MUST send a fault. This specification defines the following
faults associated with failure to process the Renew request message:

- `ResourceUnknownFault`
  - The SubscriptionManager is acting as a WS-Resource, and the resource identified in the
    message (which follows the WS-Resource Access Pattern) is not known to the Web
    service. This fault is specified by the WS-Resource [WS-Resource] specification.

- `UnacceptableTerminationTimeFault`
  - The value of InitialTerminationTime specified in the Subscribe message was not
    acceptable to the NotificationProducer. The NotificationProducer MAY include a hint in
    the fault message indicating acceptable values for InitialTerminationTime.

### 6.1.1.1 Example SOAP Encoding of the Renew Message

**Exchange**

The following is a non-normative example of a Renew request message using SOAP:

```xml
<s:Envelope ...
  <s:Header>
    <wsa:Action>

```
The following is a non-normative example of a Renew response message using SOAP:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/RenewResponse
    </wsa:Action>
  </s:Header>
  <s:Body>
    <wsnt:RenewResponse>
      <wsnt:TerminationTime>2005-12-26T00:00:00.000000Z</wsnt:TerminationTime>
    </wsnt:RenewResponse>
  </s:Body>
</s:Envelope>
```

### 6.1.2 Unsubscribe

To terminate a Subscription, a requestor sends an Unsubscribe request message to the SubscriptionManager. The Unsubscribe request message MUST have the following form:

```
... ...
<wsnt:Unsubscribe>
  {any}
</wsnt:Unsubscribe>
... ...
```

The WS-Addressing [action] Message Addressing Property MUST contain the URI

```
http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/UnsubscribeRequest.
```

Upon receipt of the Unsubscribe request, the SubscriptionManager MUST attempt to destroy the Subscription resource. If the SubscriptionManager successfully processes the Unsubscribe request message, it MUST respond with the following message:

```
... ...
<wsnt:UnsubscribeResponse>
  {any}
</wsnt:UnsubscribeResponse>
... ...
```
The WS-Addressing [action] Message Addressing Property MUST contain the URI
http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/UnsubscribeResponse.

If the SubscriptionManager does not respond to the Unsubscribe request message with the
UnsubscribeResponse message, then it MUST send a fault. This specification defines the
following faults associated with failure to process the Unsubscribe request message:

ResourceUnknownFault

- The SubscriptionManager is acting as a WS-Resource, and the resource identified in the
  message (which follows the WS-Resource Access Pattern) is not known to the Web
  service. This fault is specified by the WS-Resource [WS-Resource] specification.

UnableToDestroySubscriptionFault

- The SubscriptionManager was unable to destroy the Subscription resource for some
  reason.

6.1.2.1 Example SOAP Encoding of the Unsubscribe Message

Exchange

The following is a non-normative example of an Unsubscribe request message using SOAP:

```xml
<envelope ... >
  <Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/UnsubscribeRequest
    </wsa:Action>
    ...
  </Header>
  <Body>
    <wsnt:Unsubscribe />
  </Body>
</envelope>
```

The following is a non-normative example of an Unsubscribe response message using SOAP:

```xml
<envelope ... >
  <Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/UnsubscribeResponse
    </wsa:Action>
    ...
  </Header>
  <Body>
    <wsnt:UnsubscribeResponse />
  </Body>
</envelope>
```
6.2 Pausable Subscription Manager

A Pausable Subscription Manager implements all the message exchanges defined in the Base Subscription Manager and augments this with a set of advanced capabilities to allow third parties to pause and resume subscriptions.

6.2.1 PauseSubscription

To temporarily suspend the production of Notifications on the given Subscription, a requestor MAY send a PauseSubscription request message to the SubscriptionManager. The PauseSubscription request message has the following form:

```
...<wsnt:PauseSubscription>{any}</wsnt:PauseSubscription>
...
```

The WS-Addressing [action] Message Addressing Property MUST contain the URI http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/PauseSubscriptionRequest.

Upon successful processing of this message the Subscription is in the paused state. Production of Notifications can be resumed by sending the ResumeSubscription request message (see 6.3). In an asynchronous implementation, a NotificationProducer may continue producing notifications while the PauseSubscription request is in transit. In general, there is no way of knowing exactly when the pause will take effect.

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

When a Subscription is resumed after having been paused, the NotificationProducer MAY choose from one of the courses of action listed below, or any other action. NotificationProducers MAY advertise their behavior in this situation via policy assertions. In the absence of a specific policy assertion, Subscribers SHOULD NOT assume any particular behavior on the part of the NotificationProducer. Possible actions include:

1. Produce Notifications 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. Produce a Notification 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. Produce no Notifications until a Situation occurs after the Subscription has been resumed.

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

```
...<wsnt:PauseSubscriptionResponse>{any}</wsnt:PauseSubscriptionResponse>
...
```

The WS-Addressing [action] Message Addressing Property MUST contain the URI http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/PauseSubscriptionResponse.
If the SubscriptionManager does not respond to the PauseSubscription request message with the
PauseSubscription response message, then it MUST send a fault. This specification defines the
following faults associated with failure to process the PauseSubscription request message:

- ResourceUnknownFault
  - The resource identified in the message is not known to the Web service. This fault is
- PauseFailedFault
  - The Pause operation could not be performed on the Subscription.

### 6.2.2 Example SOAP Encoding of the PauseSubscription Message Exchange

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

```xml
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/PauseSubscriptionRequest
    </wsa:Action>
  </s:Header>
  <wsnt:PauseSubscription/>
</s:Envelope>
```

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

```xml
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/PauseSubscriptionResponse
    </wsa:Action>
  </s:Header>
  <wsnt:PauseSubscriptionResponse/>
</s:Envelope>
```

### 6.3 ResumeSubscription

If a requestor wishes to resume the production of Notifications on the given Subscription, it must
send a ResumeSubscription request message. The ResumeSubscription request message has
the following form:
The WS-Addressing [action] Message Addressing Property MUST contain the URI
   http://docs.oasis-open.org/wsn/bw-1/SubscriptionManager/ResumeSubscriptionRequest.
Upon successful processing of this message the Subscription is no longer 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.
If the SubscriptionManager accepts the ResumeSubscription request message, it MUST respond
with a message of the following form:

       <wsnt:ResumeSubscriptionResponse>
           {any}
       </wsnt:ResumeSubscriptionResponse>...

The WS-Addressing [action] Message Addressing Property MUST contain the URI
   http://docs.oasis-open.org/wsn/bw-
       1/SubscriptionManager/ResumeSubscriptionResponse.
If the SubscriptionManager does not respond to the ResumeSubscriptionMessage request
message with the ResumeSubscriptionResponse message, then it MUST send a fault. This
specification defines the following faults associated with failure to process the
ResumeSubscriptionMessage request message:
ResourceUnknownFault
   • The resource identified in the message is not known to the Web service. This fault is
ResumeFailedFault
   • The Resume operation could not be performed on the Subscription.

6.3.1 Example SOAP Encoding of the ResumeSubscription Message Exchange
The following is a non-normative example of a ResumeSubscription request message using
SOAP:

       <s:Envelope ... >
           <s:Header>
               <wsa:Action>
                   http://docs.oasis-open.org/wsn/bw-
                       1/SubscriptionManager/ResumeSubscriptionRequest
               </wsa:Action>
           </s:Header>
       </s:Body>
The following is a non-normative example of a ResumeSubscription response message using SOAP:

```xml
<wsnt:ResumeSubscription/>
</s:Body>
</s:Envelope>
```

6.4 Subscriptions as WS-Resources

Either Base Subscription Managers or Pausable Subscription Managers MAY expose subscriptions as WS-Resources. A Subscription Manager that exposes Subscriptions as WS-Resources 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.

It is RECOMMENDED that Subscription Managers that expose Subscriptions as WS-Resources support the message exchanges and resource properties defined for both forms of resource lifetime (immediate and scheduled destruction) by WS-ResourceLifetime [WS-ResourceLifetime]. These message exchanges MAY be used to define the means by which Subscription resources can be explicitly destroyed, or destroyed using a scheduled (time-based) mechanism. Such Subscription Managers MAY support other means of destroying subscriptions.

Subscription Managers MUST (for example in its WSDL definition) declare to requestors which means of Subscription destruction is supported.

6.4.1 Subscription Resource Properties

In cases where subscriptions are exposed as WS-Resources, this specification defines several resource properties, which MUST conform to the following schema fragment for content:

```xml
... targetNamespace="http://docs.oasis-open.org/wns/b-1"
...
<xsd:element name="ConsumerReference"
    type="wsa:EndpointReferenceType" />
<xsd:element name="Filter"
    type="wsnt:FilterType" />
```
These Resource Property elements are further described as follows:

/\wsnt:ConsumerReference and /\wsnt:Filter and /\wsnt:SubscriptionPolicy

These elements are as defined in the description of the Subscribe request message (see 4.2). SubscriptionManagers MAY prohibit any combination of these properties from being set. If setting a given property is disallowed, the SubscriptionManager MUST fault on receiving a request to change it. SubscriptionManagers SHOULD describe using metadata techniques defined by WS-ResourceFramework which properties may be set.

If any of these properties is settable, it is implementation dependent whether production of notifications will be attempted without interruption when a change is made. The SubscriptionManager may choose to treat the change in properties as equivalent to a cancellation followed by the creation of a new subscription. In this case, messages may potentially be lost in the interim.

A SubscriptionManager MAY choose to provide a stronger guarantee of continuity of notification production, and if so it SHOULD advertise this capability, for example in a policy assertion. In the absence of any such policy assertion, a subscriber SHOULD NOT make any assumption of continuity.

/\wsnt:CreationTime

Indicates the date and time at which the Subscription was created. This is an optional component, supporting resource constrained devices which cannot associate a creation time with subscription resources they create. This property is read-only.

SubscriptionManagers MUST fault on receiving a request to change it.

If the SubscriptionManager also accepts the SetResourceProperties request message as defined in WS-ResourceProperties, the following properties MAY be modified by the requestor:

/\wsnt:ConsumerReference and /\wsnt:Filter and /\wsnt:SubscriptionPolicy
7 Security Considerations

This section deals with the security aspects of WS-BaseNotification. It deals with (a) securing the message exchanges defined in this specification, and (b) authorization and denial of service considerations specific to the Notification pattern.

7.1 Securing the Message Exchanges

In the Notification pattern, Notifications are sent to a NotificationConsumer, Subscribers exchange SubscriptionRequest and SubscriptionResponse messages with NotificationProducers, and any party with access to a SubscriptionManager endpoint may perform operations on the underlying Subscription resources. In cases where this communication must be secured it is RECOMMENDED that this be done using the mechanisms described in WS-Security.

Communication between a NotificationProducer and NotificationConsumer will typically comprise a number of Notification messages. In cases where this communication must be secure, it is RECOMMENDED that a security context be established, allowing for potentially more efficient means of authentication. Note that the keys used to secure this channel may differ from any keys used in the Subscribe request/response operation that created the associated Subscription.

7.2 Securing Subscriptions and Notifications

As WS-BaseNotification provides mechanisms for subscribing to topics, security policies should be established such that

1. only authorized principals can subscribe to receive Notifications
2. only authorized principals can modify or delete Subscriptions

It is recommended that the authorization policies be specified at the granularity of the Topic, if Topics are supported. It should be noted that even though Subscriptions may be done by authorized principals, the Notifications may be produced for 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 Notifications 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 Notifications 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-BaseNotification may use WS-ResourceProperties and WS-ResourceLifetime, the security considerations outlined in WS-those specifications need to be taken into account where appropriate. 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.
In addition to the usual concerns of authorization and message integrity which apply to all web services, notification presents issues all its own due to the third-party nature of subscription. Since the NotificationProducer is agreeing to produce Notifications for a consumer based on the requests of a Subscriber, it must assure itself that there is no harm in producing these Notifications. A malicious Subscriber may request Notifications be sent to a party that is not authorized to receive them. It may also mount DOS attacks by requesting large volumes of Notifications be sent to parties that cannot handle them.

The NotificationProducer may address these risks in many different ways, including but not limited to:

- Simply trusting all Subscribers, perhaps because all parties are known to be on a closed, trusted, network, or because the consequences of unauthorized Subscriptions are otherwise known to be negligible.
- Requiring all Subscribers to provide secure credentials proving that they are trusted to make Subscriptions.
- Refusing to produce notifications for NotificationConsumers that are not known to be authorized.
- Explicitly confirming with NotificationConsumers whether they wish to receive the Notifications that the Subscriber has requested.
- Some combination of the above, depending on the identity of the Subscriber and NotificationProducer

NotificationProducers SHOULD advertise, whether through policy assertions or other means, what security measures they take.
8 References

8.1 Normative


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

8.2 Non-Normative

[SOAP 1.1] http://www.w3.org/TR/SOAP

[SOAP 1.2] http://www.w3.org/TR/soap12-part1/


Appendix A. Acknowledgments

Special thanks to the Global Grid Forum’s Open Grid Services Infrastructure working group, which defined the OGSI v1.0 [OGSI] specification which was a large inspiration for the ideas expressed in this specification.

The following individuals were members of the committee during the development of this specification:

Sid Askary, Fred Carter (AmberPoint), Martin Chapman (Oracle), Dave Chappell (Sonic Software), Rick Cobb (KnowNow), Ugo Corda (SeeBeyond Technology Corporation), John Fuller, Stephen Graham (IBM), David Hull (Tibco), Hideharu Kato (Hitachi), Lily Liu (webMethods, Inc.), Tom Maguire (IBM), Susan Malaika (IBM), Samuel Meder (Argonne National Laboratory), Bryan Murray (Hewlett-Packard), Peter Niblett (IBM), Sanjay Patil (SAP), Mark Peel (Novell), Matt Roberts (IBM), Igor Sedukhin (Computer Associates), David Snelling (Fujitsu), Latha Srinivasan
In addition, the following people made contributions to this specification:
Tim Banks (IBM), Nick Butler (IBM), Doug Davis (IBM), John Dinger (IBM), Don Ferguson (IBM),
Jeff Frey (IBM), Andreas Koeppel (SAP), Heather Kreger (IBM), Amy Lewis (TIBCO Software),
Kevin Liu (SAP), Nataraj Nagaratnam (IBM), Martin Nally (IBM), Jeff Nick (IBM), Jay Parikh
(Akamai Technologies), Claus von Riegen (SAP), Rick Rineholt (IBM), John Rofrano (IBM),
Shivajee Samdarshi (TIBCO Software), Eugène Sindambiwe (SAP), Jay Unger (IBM), Bill Weihl
(Akamai Technologies), Mark Weitzel (IBM), and Dan Wolfson (IBM).

Appendix B. XML Schema

The XML types and elements used in this specification are defined in the following XML Schema:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
OASIS takes no position regarding the validity or scope of any
intellectual property or other rights that might be claimed to pertain
to the implementation or use of the technology described in this
document or the extent to which any license under such rights might or
might not be available; neither does it represent that it has made any
effort to identify any such rights. Information on OASIS's procedures
with respect to rights in OASIS specifications can be found at the
OASIS website. Copies of claims of rights made available for
publication and any assurances of licenses to be made available, or the
result of an attempt made to obtain a general license or permission for
the use of such proprietary rights by implementors or users of this
specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any
copyrights, patents or patent applications, or other proprietary rights
which may cover technology that may be required to implement this
specification. Please address the information to the OASIS Executive
Director.

Copyright (C) OASIS Open (2004-2005). All Rights Reserved.
```
This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

-->

<xsd:schema
    targetNamespace="http://docs.oasis-open.org/wsn/b-1"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-1"
    xmlns:wsa="http://www.w3.org/2005/03/addressing"
    xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

    <!-- Import -->
    <xsd:import namespace="http://www.w3.org/2005/03/addressing"
                 schemaLocation="http://www.w3.org/2005/03/addressing"/>

    <xsd:import namespace="http://docs.oasis-open.org/wsrf/bf-1"
                 schemaLocation="http://docs.oasis-open.org/wsrf/bf-1"/>

    <!-- Misc. Helper Types -->
    <xsd:complexType name="QueryExpressionType" mixed="true">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="1" processContents="lax"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
<xsd:attribute name="Dialect" type="xsd:anyURI" use="required"/>
</xsd:complexType>

<xsd:complexType name="TopicExpressionType" mixed="true">
  <xsd:sequence>
    <xsd:any minOccurs="0" maxOccurs="1" processContents="lax" />
  </xsd:sequence>
  <xsd:attribute name="Dialect" type="xsd:anyURI" use="required" />
  <xsd:anyAttribute/>
</xsd:complexType>

<xsd:complexType name="FilterType">
  <xsd:sequence>
    <xsd:any minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- =============== Resource Property Related ===================

<!-- ======== Resource Properties for NotificationProducer ========
<xsd:element name="TopicExpression" type="wsnt:TopicExpressionType"/>
<xsd:element name="FixedTopicSet" type="xsd:boolean" default="true"/>
<xsd:element name="TopicExpressionDialect" type="xsd:anyURI"/>

<xsd:element name="NotificationProducerRP">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsnt:TopicExpression" minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="wsnt:FixedTopicSet" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="wsnt:TopicExpressionDialect" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

</xsd:complexType>

<!-- ======== Resource Properties for SubscriptionManager ======= -->
<xsd:element name="ConsumerReference" type="wsa:EndpointReferenceType" />
<xsd:element name="Filter" type="wsnt:FilterType" />
<xsd:element name="SubscriptionPolicy">
  <xsd:complexType>
    <xsd:sequence>
      <s:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="CreationTime" type="xsd:dateTime" />

<xsd:element name="SubscriptionManagerRP" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="wsnt:ConsumerReference"
                minOccurs="1" maxOccurs="1" />
            <xsd:element ref="wsnt:Filter"
                minOccurs="0" maxOccurs="1" />
            <xsd:element ref="wsnt:SubscriptionPolicy"
                minOccurs="0" maxOccurs="1" />
            <xsd:element ref="wsnt:CreationTime"
                minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- ================= Notification Metadata ===================== -->
<xsd:element name="SubscriptionReference"
    type="wsa:EndpointReferenceType" />
<xsd:element name="Topic"
    type="wsnt:TopicExpressionType" />
<xsd:element name="ProducerReference"
    type="wsa:EndpointReferenceType" />

<!-- ================== Message Helper Types ===================== -->
<xsd:complexType name="NotificationMessageHolderType" >
    <xsd:sequence>
        <xsd:element ref="wsnt:SubscriptionReference"
            minOccurs="0" maxOccurs="1" />
        <xsd:element ref="wsnt:Topic"
            minOccurs="0" maxOccurs="1" />
        <xsd:element ref="wsnt:ProducerReference"
            minOccurs="0" maxOccurs="1" />
        <xsd:element name="Message">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="##any" processContents="lax"
                        minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="NotificationMessage"
    type="wsnt:NotificationMessageHolderType"/>
<!--- Message Types for NotificationConsumer --->
<xsd:element name="Notify">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsnt:NotificationMessage" minOccurs="1" maxOccurs="unbounded" />
      <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Message Types for NotificationProducer --->
<xsd:simpleType name="AbsoluteOrRelativeTimeType">
  <xsd:union memberTypes="xsd:dateTime xsd:duration" />
</xsd:simpleType>

<xsd:element name="CurrentTime" type="xsd:dateTime" />

<xsd:element name="TerminationTime" nillable="true" type="xsd:dateTime" />

<xsd:element name="ProducerProperties" type="wsnt:QueryExpressionType" />

<xsd:element name="MessageContent" type="wsnt:QueryExpressionType" />

<xsd:element name="UseRaw"/>

<xsd:element name="Subscribe">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ConsumerReference" type="wsa:EndpointReferenceType" minOccurs="1" maxOccurs="1" />
      <xsd:element name="Filter" type="wsnt:FilterType" minOccurs="0" maxOccurs="1" />
      <xsd:element name="InitialTerminationTime" type="wsnt:AbsoluteOrRelativeTimeType" nillable="true" minOccurs="0" maxOccurs="1" />
      <xsd:element name="SubscriptionPolicy" minOccurs="0" maxOccurs="1">
        <xsd:complexType>
          <xsd:sequence>
            <!-- Additional elements or text here -->
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </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:element ref="wsnt:CurrentTime" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="wsnt:TerminationTime" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:complexType name="SubscribeCreationFailedFaultType">
      <xsd:sequence>
        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:element>

<xsd:element name="GetCurrentMessage">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Topic" type="wsnt:TopicExpressionType" />
    </xsd:sequence>
    <xsd:complexType name="GetCurrentMessageResponse">
      <xsd:sequence>
        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:element>
<xsd:complexType name="InvalidFilterFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType">
      <xsd:sequence>
        <xsd:element name="UnknownFilter" type="xsd:QName"
          minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="InvalidFilterFault" type="wsnt:InvalidFilterFaultType"/>

<xsd:complexType name="TopicExpressionDialectUnknownFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="TopicExpressionDialectUnknownFault" type="wsnt:TopicExpressionDialectUnknownFaultType"/>

<xsd:complexType name="InvalidTopicExpressionFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="InvalidTopicExpressionFault" type="wsnt:InvalidTopicExpressionFaultType"/>

<xsd:complexType name="TopicNotSupportedFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="TopicNotSupportedFault" type="wsnt:TopicNotSupportedFaultType"/>

<xsd:complexType name="MultipleTopicsSpecifiedFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>
</xsd:complexType>
<xsd:element name="MultipleTopicsSpecifiedFault"
    type="wsnt:MultipleTopicsSpecifiedFaultType"/>
<xsd:complexType name="InvalidProducerPropertiesExpressionFaultType">
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
</xsd:complexType>
<xsd:element name="InvalidProducerPropertiesExpressionFault"
    type="wsnt:InvalidProducerPropertiesExpressionFaultType"/>
<xsd:complexType name="InvalidMessageContentExpressionFaultType">
    <xsd:complexContent>
        <xsd:extension base="wsrf-bf:BaseFaultType"/>
    </xsd:complexContent>
</xsd:complexType>
<xsd:element name="InvalidMessageContentExpressionFault"
    type="wsnt:InvalidMessageContentExpressionFaultType"/>
<xsd:complexType name="InvalidUseRawValueFaultType">
    <xsd:complexContent>
        <xsd:extension base="wsrf-bf:BaseFaultType"/>
    </xsd:complexContent>
</xsd:complexType>
<xsd:element name="InvalidUseRawValueFault"
    type="wsnt:InvalidUseRawValueFaultType"/>
<xsd:complexType name="UnacceptableInitialTerminationTimeFaultType">
    <xsd:complexContent>
        <xsd:extension base="wsrf-bf:BaseFaultType">
            <xsd:sequence>
                <xsd:element name="MinimumTime" type="xsd:dateTime"/>
                <xsd:element name="MaximumTime" type="xsd:dateTime"
                    minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:element name="UnacceptableInitialTerminationTimeFault"
    type="wsnt:UnacceptableInitialTerminationTimeFaultType"/>
<xsd:complexType name="NoCurrentMessageOnTopicFaultType">
    <xsd:complexContent>
        <xsd:extension base="wsrf-bf:BaseFaultType"/>
    </xsd:complexContent>
</xsd:complexType>
<xsd:element name="NoCurrentMessageOnTopicFault"
    type="wsnt:NoCurrentMessageOnTopicFaultType"/>
type="wsnt:NoCurrentMessageOnTopicFaultType"/>

<!-- ======== Message Types for PullPoint ======================== -->
<xsd:element name="GetMessages">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="MaximumNumber" type="xsd:nonNegativeInteger"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="GetMessagesResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsnt:NotificationMessage" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Destroy">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="DestroyResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="UnableToDestroyPullPointType"/>
<xsd:complexContent>
  <xsd:extension base="wsrf-bf:BaseFaultType"/>
</xsd:complexContent>

<xsd:complexType>
  <xsd:element name="UnableToDestroyPullPoint" type="wsnt:UnableToDestroyPullPointType"/>
</xsd:complexType>

<!-- ======== Message Types for Create PullPoint =========== -->

<xsd:element name="CreatePullPoint">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="CreatePullPointResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="PullPoint" type="wsa:EndpointReferenceType"/>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="UnableToCreatePullPointType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="UnableToCreatePullPoint" type="wsnt:UnableToCreatePullPointType"/>

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

<xsd:element name="Renew">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="TerminationTime" type="wsnt:AbsoluteOrRelativeTimeType"
        nillable="true"
        minOccurs="1" maxOccurs="1"/>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="RenewResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsnt:TerminationTime" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="wsnt:CurrentTime" minOccurs="0" maxOccurs="1"/>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="UnacceptableTerminationTimeFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType">
      <xsd:sequence>
        <xsd:element name="MinimumTime" type="xsd:dateTime" minOccurs="0"/>
        <xsd:element name="MaximumTime" type="xsd:dateTime" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="UnacceptableTerminationTimeFault" type="wsnt:UnacceptableTerminationTimeFaultType"/>

<xsd:element name="Unsubscribe">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="UnsubscribeResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="UnableToDestroySubscriptionFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="UnableToDestroySubscriptionFault" type="wsnt:UnableToDestroySubscriptionFaultType"/>

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

<xsd:element name="PauseSubscription">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="PauseSubscriptionResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ResumeSubscription">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ResumeSubscriptionResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="PauseFailedFaultType">
</xsd:element>
Appendix C. WSDL 1.1

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

```xml
<?xml version="1.0" encoding="utf-8"?>
<!--
OASIS takes no position regarding the validity or scope of any
intellectual property or other rights that might be claimed to pertain
to the implementation or use of the technology described in this
document or the extent to which any license under such rights might or
might not be available; neither does it represent that it has made any
effort to identify any such rights. Information on OASIS's procedures
with respect to rights in OASIS specifications can be found at the
OASIS website. Copies of claims of rights made available for
publication and any assurances of licenses to be made available, or the
result of an attempt made to obtain a general license or permission for
the use of such proprietary rights by implementors or users of this
specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any
copyrights, patents or patent applications, or other proprietary rights
which may cover technology that may be required to implement this
specification. Please address the information to the OASIS Executive
Director.

Copyright (C) OASIS Open (2004-2005). All Rights Reserved.-->
```
This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

<!-- ===================== Types Definitions ====================== -->
<wsdl:types>
  <xsd:schema>
    <xsd:import
      namespace="http://docs.oasis-open.org/wsn/bw-1"
      schemaLocation="http://docs.oasis-open.org/wsn/bw-1"/>
  </xsd:schema>
</wsdl:types>

<!-- NotificationConsumer::Notify -->

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

<!-- ========== PullPoint::GetMessages ===========
GetMessages(topicExpression)
returns: a NotificationMessage (xsd:any)
-->
GetMessages(MaximumNumber)
returns: NotificationMessage list

<!-- ========== PullPoint::Destroy ===========
  Destroy()
  returns: void
-->

<!-- ========== PullPoint::CreatePullPoint ===========
  CreatePullPoint()
  returns: PullPoint (wsa:EndpointReference)
-->

wsn-ws_base_notification-1.3-spec-pr-01
element="wsnt:UnableToCreatePullPoint"/>
</wsdl:message>

<!-- ================ SubscriptionManager::Renew ==================
Renew( Duration | AbsoluteTime)
returns: (New Termination Time [CurrentTime])
-->  
<wsdl:message name="RenewRequest">
  <wsdl:part name="RenewRequest"
    element="wsnt:Renew"/>
</wsdl:message>

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

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

<!-- ============== SubscriptionManager::Unsubscribe ===============
Unsubscribe()
returns: empty
-->  
<wsdl:message name="UnsubscribeRequest">
  <wsdl:part name="UnsubscribeRequest"
    element="wsnt:Unsubscribe"/>
</wsdl:message>

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

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

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

<wSDL:message name="PauseSubscriptionResponse">
  <wsdl:part name="PauseSubscriptionResponse" element="wsnt:PauseSubscriptionResponse"/>
</wsdl:message>

<wSDL:message name="PauseFailedFault">
  <wsdl:part name="PauseFailedFault" element="wsnt:PauseFailedFault"/>
</wsdl:message>

<!-- ========= SubscriptionManager::ResumeSubscription ============
  ResumeSubscription()
  returns: empty
  -->

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

<!-- ========= NotificationProducer PortType Definition ========= -->
<wSDL:portType name="NotificationProducer">
  <wsdl:operation name="Subscribe">
    <wsdl:input message="wsntw:SubscribeRequest"/>
    <wsdl:output message="wsntw:SubscribeResponse"/>
    <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"/>
    <wsdl:fault name="InvalidFilterFault" message="wsntw:InvalidFilterFault"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:fault name="TopicExpressionDialectUnknownFault"
message="wsntw:TopicExpressionDialectUnknownFault"/>
<wsdl:fault name="InvalidTopicExpressionFault"
message="wsntw:InvalidTopicExpressionFault"/>
<wsdl:fault name="TopicNotSupportedFault"
message="wsntw:TopicNotSupportedFault"/>
<wsdl:fault name="InvalidProducerPropertiesExpressionFault"
message="wsntw:InvalidProducerPropertiesExpressionFault"/>
<wsdl:fault name="InvalidMessageContentExpressionFault"
message="wsntw:InvalidMessageContentExpressionFault"/>
<wsdl:fault name="InvalidUseRawValueFault"
message="wsntw:InvalidUseRawValueFault"/>
<wsdl:fault name="UnacceptableInitialTerminationTimeFault"
message="wsntw:UnacceptableInitialTerminationTimeFault"/>
<wsdl:fault name="SubscribeCreationFailedFault"
message="wsntw:SubscribeCreationFailedFault"/>
</wsdl:operation>

<wsdl:operation name="GetCurrentMessage">
<wsdl:input message="wsntw:GetCurrentMessageRequest"/>
<wsdl:output message="wsntw:GetCurrentMessageResponse"/>
<wsdl:fault name="ResourceUnknownFault"
message="wsrf-rw:ResourceUnknownFault"/>
<wsdl:fault name="TopicExpressionDialectUnknownFault"
message="wsntw:TopicExpressionDialectUnknownFault"/>
<wsdl:fault name="InvalidTopicExpressionFault"
message="wsntw:InvalidTopicExpressionFault"/>
<wsdl:fault name="TopicNotSupportedFault"
message="wsntw:TopicNotSupportedFault"/>
<wsdl:fault name="NoCurrentMessageOnTopicFault"
message="wsntw:NoCurrentMessageOnTopicFault"/>
</wsdl:operation>
</wsdl:portType>

<!-- ========== PullPoint PortType Definition =============== -->
<wsdl:portType name="PullPoint">
<wsdl:operation name="GetMessages">
<wsdl:input name="GetMessagesRequest"
message="wsntw:GetMessagesRequest"/>
<wsdl:output name="GetMessagesResponse"
message="wsntw:GetMessagesResponse"/>
<wsdl:fault name="ResourceUnknownFault"
message="wsrf-rw:ResourceUnknownFault"/>
</wsdl:operation>

<wsdl:operation name="Destroy">
<wsdl:input name="DestroyRequest"
message="wsntw:DestroyRequest"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:output name="DestroyResponse"
message="wsntw:DestroyResponse" />
<wsdl:output name="UnableToDestroyPullPoint"
message="wsntw:UnableToDestroyPullPoint" />
</wsdl:operation>
</wsdl:portType>

<!-- ========== CreatePullPoint PortType Definition =============== -->
<wsdl:portType name="CreatePullPoint">
<wsdl:operation name="CreatePullPoint">
<wsdl:input name="CreatePullPointRequest"
message="wsntw:CreatePullPointRequest" />
<wsdl:output name="CreatePullPointResponse"
message="wsntw:CreatePullPointResponse" />
<wsdl:fault name="UnableToCreatePullPoint"
message="wsntw:UnableToCreatePullPoint" />
</wsdl:operation>
</wsdl:portType>

<!-- ========== SubscriptionManager PortType Definition ===========-->
<wsdl:portType name="SubscriptionManager">
<wsdl:operation name="Renew">
<wsdl:input name="RenewRequest"
message="wsntw:RenewRequest" />
<wsdl:output name="RenewResponse"
message="wsntw:RenewResponse" />
<wsdl:fault name="ResourceUnknownFault"
message="wsrf-rw:ResourceUnknownFault" />
<wsdl:fault name="UnacceptableTerminationTimeFault"
message="wsntw:UnacceptableTerminationTimeFault" />
</wsdl:operation>
<wsdl:operation name="Unsubscribe">
<wsdl:input name="UnsubscribeRequest"
message="wsntw:UnsubscribeRequest" />
<wsdl:output name="UnsubscribeResponse"
message="wsntw:UnsubscribeResponse" />
<wsdl:fault name="ResourceUnknownFault"
message="wsrf-rw:ResourceUnknownFault" />
<wsdl:fault name="UnableToDestroySubscriptionFault"
message="wsntw:UnableToDestroySubscriptionFault" />
</wsdl:operation>
</wsdl:portType>

<!-- ====== PausableSubscriptionManager PortType Definition ======= -->
<wsdl:portType name="PausableSubscriptionManager">
<wsdl:extend target="SubscriptionManager"/>
<!-- ---- PausableConsumer PortType Definition ---- -->
<wsdl:portType name="PausableConsumer">
<wsdl:operation name="Pause">
<wsdl:input name="PauseRequest"
message="wsntw:PauseRequest" />
<wsdl:input name="ExpirationTime"
message="wsntw:ExpirationTime" />
<wsdl:output name="PauseResponse"
message="wsntw:PauseResponse" />
<wsdl:fault name="UnableToPauseConsumerFault"
message="wsntw:UnableToPauseConsumerFault" />
</wsdl:operation>
</wsdl:portType>
</wsdl:extend>
</wsdl:portType>

<!-- ====== PausableSubscriptionManager PortType Definition ======= -->
<wsdl:portType name="PausableSubscriptionManager">
<!-- ---- PausableConsumer PortType Definition ---- -->
<wsdl:portType name="PausableConsumer">
<wsdl:operation name="Pause">
<wsdl:input name="PauseRequest"
message="wsntw:PauseRequest" />
<wsdl:input name="ExpirationTime"
message="wsntw:ExpirationTime" />
<wsdl:output name="PauseResponse"
message="wsntw:PauseResponse" />
<wsdl:fault name="UnableToPauseConsumerFault"
message="wsntw:UnableToPauseConsumerFault" />
</wsdl:operation>
</wsdl:portType>
</wsdl:extends>
</wsdl:portType>
<wsdl:operation name="Renew">
    <wsdl:input name="RenewRequest" message="wsntw:RenewRequest"/>
    <wsdl:output name="RenewResponse" message="wsntw:RenewResponse"/>
    <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"/>
    <wsdl:fault name="UnacceptableTerminationTimeFault" message="wsntw:UnacceptableTerminationTimeFault"/>
</wsdl:operation>

<wsdl:operation name="Unsubscribe">
    <wsdl:input name="UnsubscribeRequest" message="wsntw:UnsubscribeRequest"/>
    <wsdl:output name="UnsubscribeResponse" message="wsntw:UnsubscribeResponse"/>
    <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"/>
    <wsdl:fault name="UnableToDestroySubscriptionFault" message="wsntw:UnableToDestroySubscriptionFault"/>
</wsdl:operation>

<!-- === PausableSubscriptionManager specific operations === -->

<wsdl:operation name="PauseSubscription">
    <wsdl:input message="wsntw:PauseSubscriptionRequest"/>
    <wsdl:output message="wsntw:PauseSubscriptionResponse"/>
    <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"/>
    <wsdl:fault name="PauseFailedFault" message="wsntw:PauseFailedFault"/>
</wsdl:operation>

<wsdl:operation name="ResumeSubscription">
    <wsdl:input message="wsntw:ResumeSubscriptionRequest"/>
    <wsdl:output message="wsntw:ResumeSubscriptionResponse"/>
    <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"/>
    <wsdl:fault name="ResumeFailedFault" message="wsntw:ResumeFailedFault"/>
</wsdl:operation>

</wsdl:portType>
</wsdl:definitions>

Appendix D. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7/8/2005</td>
<td></td>
<td>wsn-ws_base_notification-1.3-spec-pr-01</td>
</tr>
<tr>
<td>Rev</td>
<td>Date</td>
<td>By Whom</td>
<td>What</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>wd-01</td>
<td>2004-05-18</td>
<td>Steve Graham</td>
<td>Initial version created from submission by contributing companies. Minor modifications made to reflect OASIS formatting and namespace URI choices.</td>
</tr>
<tr>
<td>wd-02</td>
<td>2004-05-24</td>
<td>Steve Graham</td>
<td>Added requirements section draft and updated acknowledgments appendix.</td>
</tr>
<tr>
<td>wd-03</td>
<td>2004-06-07</td>
<td>Steve Graham</td>
<td>Minor modifications on wd-02 as reported by Peter N. and William V.</td>
</tr>
<tr>
<td>wd-04</td>
<td>2004-06-21</td>
<td>Steve Graham</td>
<td>Finalize list of acknowledgments and annotate title page with result of TC vote.</td>
</tr>
<tr>
<td>wd-03</td>
<td>2004-07-12</td>
<td>Steve Graham</td>
<td>&quot;rename&quot; 04 back to 03 to minimize reference rework in WSRF.</td>
</tr>
<tr>
<td>wd-04</td>
<td>2004-09-20</td>
<td>Steve Graham</td>
<td>WSN 2.3, 2.11, 2.17, 2.18, 2.19, 2.21, 2.22, 2.29</td>
</tr>
<tr>
<td>1.3, 01a</td>
<td>2004-09-27</td>
<td>Steve Graham</td>
<td>Rename to follow agreed upon format, add resolution to WSN 5.2</td>
</tr>
<tr>
<td>1.3, 01b</td>
<td>2004-11-17</td>
<td>Steve Graham</td>
<td>Post Oct f2f edits</td>
</tr>
<tr>
<td>1.3, 01c</td>
<td>2004-12-7</td>
<td>Steve Graham, David Hull</td>
<td>WSN 1.7, 2.7, 2.13, 2.14, 2.15, 2.24 and WSN2.17 revisited</td>
</tr>
<tr>
<td>1.3, 01d</td>
<td>2004-12-8</td>
<td>Steve Graham</td>
<td>Fix Action URIs and other edits per Tom Maguire's comments</td>
</tr>
<tr>
<td>1.3, 01e</td>
<td>2005-02-19</td>
<td>Bryan Murray</td>
<td>Editorial text and formatting, WSN2.1, WSN2.37, WSN2.38</td>
</tr>
<tr>
<td>1.3, 01e</td>
<td>2005-02-25</td>
<td>Steve Graham</td>
<td>Mod's to Bryan's edits, WSN 2.34, WSN2.35</td>
</tr>
<tr>
<td>1.3, 01e</td>
<td>2005-03-02</td>
<td>Steve Graham</td>
<td>Minor typos reported by Matt Haynos' team.</td>
</tr>
<tr>
<td>1.3, 01f</td>
<td>2005-05-09</td>
<td>Bryan Murray, David Hull</td>
<td>WSN1.5, WSN2.1 (InvalidFilterFault), WSN2.32, WSN2.39, WSN2.43, WSN2.45, WSN2.46</td>
</tr>
<tr>
<td>1.3, 01g</td>
<td>2005-05-18</td>
<td>Bryan Murray</td>
<td>Per face-to-face agreement on hygiene issues. WSN 2.48 (but see comment to list from dmh) (Partially?) WSN 2.49 WSN 2.50 WSN 2.53</td>
</tr>
<tr>
<td>1.3,01g</td>
<td>2005-05-19</td>
<td>David Hull</td>
<td>Integrated definitions from White Paper,</td>
</tr>
<tr>
<td>Rev</td>
<td>Date</td>
<td>By Whom</td>
<td>What</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.3.01 h</td>
<td>2005-05-27</td>
<td>David Hull</td>
<td>including updates for WSN2.39 compliance. NotificationMessage is not Notification throughout. Began integrating security section from whitepaper, along with proposed text for WSN 2.24 Fixed a few more WS 2.39 stragglers</td>
</tr>
<tr>
<td>1.3.01.i</td>
<td>2005-06-02</td>
<td>Steve Graham</td>
<td>Modifications on 01g and 01h text, completion of schema and WSDL related items and 2.47, 2.55, 2.41, 2.23</td>
</tr>
<tr>
<td>1.3.01.j</td>
<td>2005-06-09</td>
<td>Bryan Murray</td>
<td>AI 106, AI 109, WSN1.4, section 7.1 cleanup, remove ref to Primer</td>
</tr>
<tr>
<td>1.3.01.k</td>
<td>2005-06-29</td>
<td>David Hull</td>
<td>Integrated review comments for 2.1, 2.25, 2.33, 2.39, 2.48, 2.50, 2.52. Pseudo-schema descriptions of open content.</td>
</tr>
<tr>
<td>1.3.01.l</td>
<td>2005-07-06</td>
<td>David Hull</td>
<td>Integrated review from chairs: 2.6, 2.34, 2.52, 2.55, typos, abstract, acknowledgements.</td>
</tr>
<tr>
<td>1.3.01.m</td>
<td>2005-07-07</td>
<td>David Hull</td>
<td>Amended requirement for composability, per committee vote. Formatting in section 4.2</td>
</tr>
</tbody>
</table>

**Appendix E. Notices**

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS’s procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.
OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright (C) OASIS Open (2004-2005). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.