Web Services Brokered Notification 1.3 (WS-BrokeredNotification)

Public Review Draft 01, 7 July 2005

Document identifier:

wsn-ws-brokered-notification-1.3-spec-pr-01

Location:

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

Editors:

Dave Chappell, Sonic Software <chappell@sonicsoftware.com>

Lily Liu, webMethods <lily.liu@webmethods.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].

This document defines the Web services interface for the NotificationBroker. A NotificationBroker is an intermediary, which, among other things, allows publication of messages from entities that are not themselves service providers. It includes standard message exchanges to be implemented by NotificationBroker service providers along with operational requirements expected of service providers and requestors that participate in brokered notifications. This work relies upon WS-BaseNotification.

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-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 the TC) are publicly archived and can be viewed at http://lists.oasis-open.org/archives/wsn-comment/.

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

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 or in system and device management domains. Message brokers are involved in many of these systems, such as the ones provided by Message Oriented Middleware vendors.

This specification defines the Web services interface for the NotificationBroker. A NotificationBroker is an intermediary between message Publishers and message Subscribers. Common functions of Publishers and Subscribers, such as messaging dissemination and security measurements, can be implemented at the NotificationBroker to produce lightweight Producers and Consumers. A NotificationBroker decouples NotificationProducers and Notification Consumers and can provide advanced messaging features such as demand-based publishing and load-balancing. A NotificationBroker also allows publication of messages from entities that are not themselves service providers. This is very similar to a traditional Message Oriented Middleware model.

The NotificationBroker interface includes standard message exchanges to be implemented by NotificationBroker service providers along with operational requirements expected of service providers and requestors that participate in brokered notifications.

1.1 Goals and Requirements

The goal of WS-BrokeredNotification is to standardize message exchanges involved in Web services publish and subscribe of a message broker. The overall objectives of WS-Notification are presented in [WS-BaseNotification]. The following section lists the specific subset of those objectives realized by WS-BrokeredNotification.

1.1.1 Requirements

In meeting this goal, the WS-BrokeredNotification specification must explicitly address the following requirements:

- **Must allow for a notification broker as an intermediary.** A NotificationBroker is an intermediary Web service that decouples NotificationConsumers from Publishers. A notification broker can relieve a Publisher from having to implement message exchanges associated with NotificationProducer; the NotificationBroker takes on the duties of subscription management and distributing Notifications on behalf of the Publisher. It implements NotificationProducer interface. It may implement SubscriptionManager or may delegate the subscription management work to another component.
- **Must allow for federation of brokers.** It must be possible to build configurations with multiple intermediary broker services in a dynamic fashion. This specification must allow for a variety of broker topology usage patterns. Among other things, these allow for greater scalability and permit sharing of administrative workload.

- **Must provide runtime metadata:** There must be a mechanism that lets a potential Subscriber discover what elements available for a subscription are provided by a NotificationBroker, and in what formats the subscription for a notification can be made.
- **Must conform to WS-BaseNotification:** A NotificationBroker must support required message exchanges defined by the [WS-BaseNotification] specification. It must conform to the NotificationProducer and the NotificationConsumer interfaces defined in WS-BaseNotification.
- **WS-BrokeredNotification 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 not exclude non-service producers and subscribers:** WS-BrokeredNotification design must not exclude a non-service entity to deliver a notification message to a NotificationBroker. It must not exclude a NotificationBroker to send a notification message to a non-service consumer.
- **Must provide publisher registration:** WS-BrokeredNotification must define standard message exchanges for registering a NotificationPublisher with a NotificationBroker.

1.1.2 Non-Goals

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

- **Defining the format of notification payloads:** The data carried in Notification payloads is application-domain specific, and WS-BrokeredNotification does not prescribe any particular format for this data**.**
- **Defining any Events or Notifications:** The WS-BrokeredNotification specification does not define any “standard” or “built-in” notification situations, events, or messages.
- **Defining the means by which NotificationBrokers are discovered by subscribers:** It is beyond the scope of this specification to define the mechanisms for runtime discovery of NotificationBrokers.

1.2 Notational Conventions

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

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

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,
- `[' and `]' are used to form groups,
- `|' represents choice.
- Attributes are conventionally assigned a value which corresponds to their type, as defined in the normative schema.

```
<!-- 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:

Prefix	Namespace
s	http://schemas.xmlsoap.org/soap/envelope/ OR http://www.w3.org/2003/05/soap-envelope
xsd	http://www.w3.org/2001/XMLSchema
wsa	http://www.w3.org/2005/03/addressing
wsn-b	http://docs.oasis-open.org/wsn/b-1
wsn-br	http://docs.oasis-open.org/wsn/br-1

wsn-bw	http://docs.oasis-open.org/wsn/bw-1
wsn-brw	http://docs.oasis-open.org/wsn/brw-1
wsrf-bf	http://docs.oasis-open.org/wsrf/bf-1
wsrf-bfw	http://docs.oasis-open.org/wsrf/bfw-1

1.4 Fault Definitions

All faults generated by a NotificationBroker, RegisterPublisher, or PublisherRegistrationManager 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 Relationship to Other Specifications

This specification builds on the basic notification mechanism defined in [WS-BaseNotification], by adding the concept of an intermediary NotificationBroker, and describing additional variants on the publisher role. A NotificationBroker takes on the role of both NotificationProducer and NotificationConsumer (as defined in [WS-BaseNotification]), and its interactions with other NotificationProducers and NotificationConsumers are largely defined by the WS-BaseNotification specification.

This means that a NotificationBroker, implemented to conform to this specification, must also conform to [WS-BaseNotification]. Such a NotificationBroker can deliver notifications to NotificationConsumers that are implemented to conform to [WS-BaseNotification], and can subscribe to Notifications distributed by NotificationProducers as defined in [WS-BaseNotification].

A NotificationBroker may support hierarchical topics as defined in [WS-Topics]. By supporting topics, NotificationBroker can manage enterprise messaging systems more efficiently.

WS-BrokeredNotification must be composable with other Web services specifications.

3 Terminology and Concepts

In addition to the terminology and usage described in the WS-BaseNotification specification, the following are the terms defined in this specification:

Publisher:

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

NotificationBroker:

- A NotificationBroker is an intermediary Web service that decouples NotificationConsumers from Publishers. A NotificationBroker is capable of subscribing to notifications, either on behalf of NotificationConsumers, or for the purpose of messaging management. It is capable disseminating notifications on behalf of Publishers to NotificationConsumers.
- A NotificationBroker aggregates NotificationProducer, NotificationConsumer, RegisterPublisher, and CreatePullPoint interfaces.
- Acting as an intermediary, a NotificationBroker provides additional capabilities to the base NotificationProducer interface:
    - It can relieve a Publisher from having to implement message exchanges associated with NotificationProducer; the NotificationBroker takes on the duties of a SubscriptionManager (managing subscriptions) and NotificationProducer (distributing Notifications) on behalf of the Publisher.
    - It can reduce the number of inter-service connections and references, if there are many Publishers and many NotificationConsumers.
    - It can act as a finder service. Potential Publishers and Subscribers can in effect find each other by utilizing a common NotificationBroker.
    - It can provide anonymous Notification, so that the Publishers and the NotificationConsumers need not be aware of each other's identity.
- An implementation of a NotificationBroker may provide additional added-value function that is beyond the scope of this specification, for example, logging Notifications, or

transforming Topics and/or Notification content. Additional function provided by a NotificationBroker can apply to all Publishers that utilize it.

- It may be the factory for Subscription resources or it may delegate the subscription factory to another component.
- A NotificationBroker provides publisher registration functions.
- A NotificationBroker may subscribe and disseminate messages that are not WS-Notification conforming.

PublisherRegistration:

- PublisherRegistration is a resource. A PublisherRegistration represents the relationship between a Publisher and a NotificationBroker, in particular, which topic(s) the publisher is permitted to publish to.
- A PublisherRegistration resource is created when a Publisher sends the RegisterPublisher request message to a NotificationBroker and the NotificationBroker succeeds in processing the registration.
- PublisherRegistration resources can be manipulated by messages sent to a PublisherRegistrationManager Web service.

RegisterPublisher:

- A RegisterPublisher is a Web service that implements the message exchanges associated with the RegisterPublisher interface. A PublisherRegistration resource is created as a result of a RegisterPublisher request to a NotificationBroker.

PublisherRegistrationManager:

- A PublisherRegistrationManager is a Web service that implements the message exchanges associated with the PublisherRegistrationManager interface.
- A publisher registration resource can be manipulated through PublisherRegistrationManager message exchanges.
- A PublisherRegistrationManager provides services that allow a service requestor to query and manipulate PublisherRegistration resources that it manages.
- A PublisherRegistrationManager is subordinate to the NotificationBroker, and MAY be implemented by the NotificationBroker service provider. However WS-BrokeredNotification permits it to be implemented by a separate service provider, should an implementer so desire.

Demand-Based Publishing:

- Some Publishers may be interested in knowing whether they have any Subscribers or not, since producing a Notification may be a costly process. Such Publishers can register with the NotificationBroker as a Demand-Based Publisher.
- Demand-Based Publishers implement message exchanges associated with the NotificationProducer interface.

- The NotificationBroker subscribes to the Demand-Based Publisher. When the NotificationBroker knows that there are no Subscribers for the Notifications from a Demand-Based Publisher it pauses its Subscription with that Publisher; when it knows that there are some Subscribers, it resumes the Subscription.
- This way the Demand-Based Publisher does not need to produce messages when there are no Subscribers, however a Demand-Based Publisher is only required to support a single Subscriber on any given Topic, and so can delegate the management of multiple Subscribers, the delivery to multiple NotificationConsumers, and other related issues (for example security) to the NotificationBroker.

4 Publishing

There are three distinct stages in the Notification process

- Observation of the Situation and its noteworthy characteristics;
- Creation of the Notification artifact that captures the noteworthy characteristics of the Situation; and
- Distribution of copies of the Notification to zero or more interested parties.

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

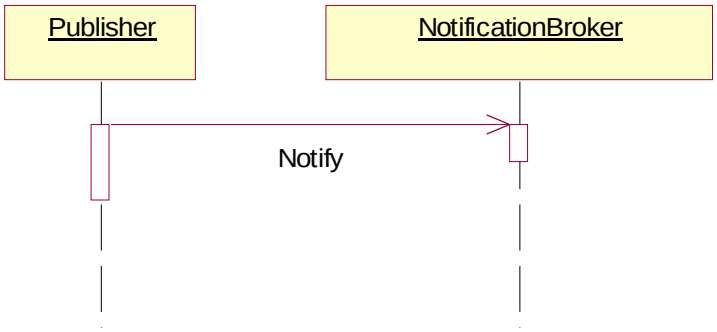
However, the WS-Notification family of specifications does specify how dissemination of messages SHOULD occur. There are two dominant patterns by which Notifications are disseminated in WS-Notification: direct and brokered.

In the direct case, the publishing Web service implements message exchanges associated with the NotificationProducer interface; it is responsible for accepting Subscribe messages and sending Notifications to interested parties. The implementer of this Web service can choose to program this behavior or delegate to specialized implementations of the Subscribe and Notification delivery behavior. This case is addressed by the WS-BaseNotification specification [WS-BaseNotification].

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

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

The following figure illustrates simple publishing:



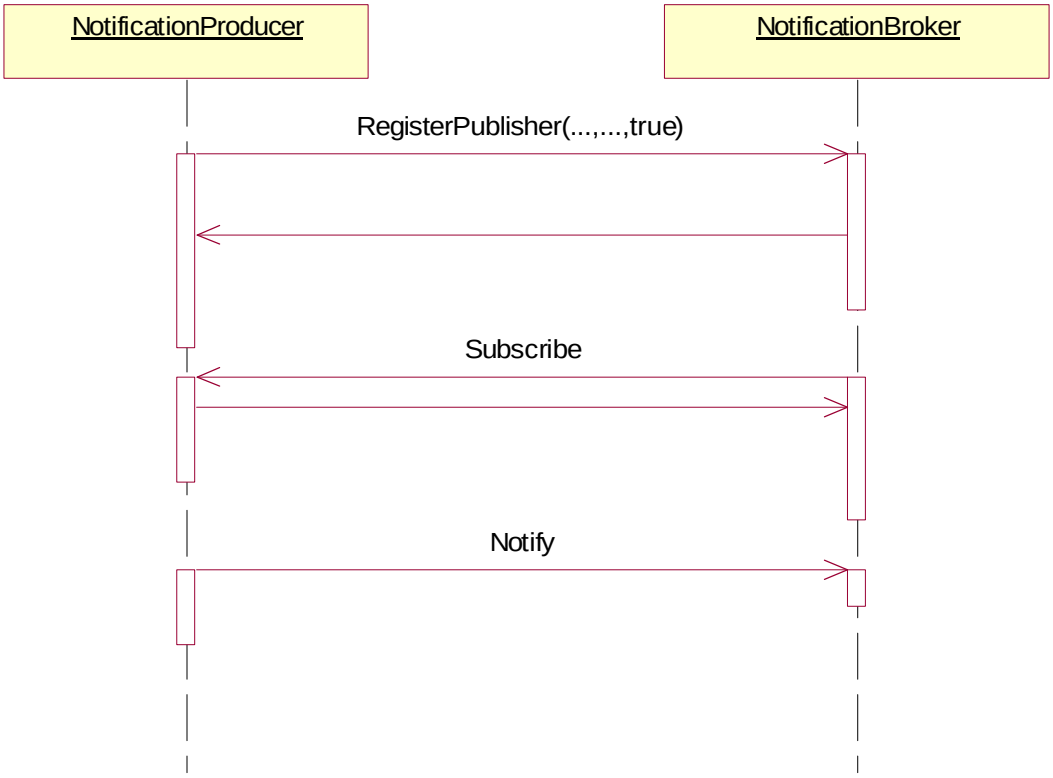
In the simple publishing scenario, the Publisher entity is responsible only for the core Publisher functions - observing the Situation and formatting the Notification artifact that describes the Situation. The dissemination step occurs when the Publisher sends the Notify message to the NotificationBroker.

In the broker initiated publishing pattern, the role of the Publisher is played by a Web service that implements NotificationProducer. The act of observing the Situation and formatting the Notification happens within the implementation logic of the NotificationProducer itself. The Notification is disseminated by the NotificationProducer sending the Notify message to a NotificationBroker. The Notification may also be disseminated by sending the Notify message to any NotificationConsumer that are subscribing to the NotificationProducer.

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

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

Furthermore, the NotificationBroker is expected to pause its Subscription whenever it has no



active Subscribers for the information provided by the Publisher. When the NotificationBroker does have active Subscribers, it is obliged to resume its Subscription to the Publisher.

5 NotificationBroker Interface

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

NotificationBroker MAY be a WS-Resource, and if it is, it MUST support the required message exchanges defined by the WS-ResourceProperties specification [WS-ResourceProperties] and MAY support the optional message exchanges defined by WS-ResourceProperties.

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

- NotificationProducer

- NotificationConsumer
- RegisterPublisher
- CreatePullPoint

The NotificationBroker portType aggregates the four portTypes and is not the only way to implement a broker. A distributed broker implementation can be achieved by hosting NotificationProducer, NotificationConsumer, CreatePullPoint, or RegisterPublisher portTypes at one or more physical endpoints.

The NotificationBroker does not specify any subscription durability or continuity. NotificationBrokers SHOULD advertise their durability or reliability features, either through policies or other means.

NotificationBrokers MAY offer flow control and MAY implement Pull-Style notifications. If so, NotificationBrokers SHOULD advertise these features, either through policies or other means.

5.1 NotificationBroker Resource Properties

In addition to the message exchanges described in this specification, a NotificationBroker 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. If it does so, the Resource Properties document defined by the NotificationBroker MUST include references to resource properties defined in NotificationProducer and NotificationConsumer, and also MUST include a reference to the following resource property element:

```
…
  targetNamespace="http://docs.oasis-open.org/wsn/br-1>
…
  <xsd:element name=”RequiresRegistration” type=”xsd:boolean”/>
…
```

Furthermore, this reference MUST reflect the minOccurs and maxOccurs properties as follows:

```
  <xsd:element ref="wsn-br:RequiresRegistration"
                        minOccurs="1" maxOccurs="1" />
```

This resource property element is further constrained as follows:

/wsn-br:RequiresRegistration

The value is “true” if the NotificationBroker requires a publisher to register (see 6.1) before sending it a Notify (i.e. publish) message on a Topic. The default is “false”.

5.2 Notify

The NotificationBroker MUST support the Notify message exchange from the NotificationConsumer interface [WS-BaseNotification], with the following clarifications/restrictions:

A Publisher sends a Notify message to a NotificationBroker in order to publish a Notification on a given Topic. As a result of the Publisher sending this message, Notifications are delivered to all NotificationConsumers subscribed on the given Topic. For some Topics (those that require a Publisher to pre-register), the sender must be a registered Publisher in order to successfully publish a Notification on the given Topic (see 6.1).

5.3 Subscribe

The NotificationBroker MUST support the Subscribe message exchange from the NotificationProducer interface [WS-BaseNotification]. A NotificationBroker MAY support any TopicExpression dialect.

A NotificationBroker is capable of routing or 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 NotificationBroker in order to register this interest.

If the processing of a Subscribe message is successful, the NotificationBroker MUST produce a response message, as described in WS-BaseNotification, containing an endpoint reference to a Subscription resource representing a Subscription created as a result of processing the Subscribe request. Otherwise, the NotificationBroker must fault. WS-BaseNotification defines a set of these faults.

5.4 GetCurrentMessage

The NotificationBroker MUST support the GetCurrentMessage message exchange from the NotificationProducer interface [WS-BaseNotification].

As defined in WS-BaseNotification, in response to a GetCurrentMessage message, the NotificationBroker MAY return the last Notification published on a given Topic. This is a non-destructive read, allowing a newly-subscribed NotificationConsumer to get the last Notification that other NotificationConsumers have received.

5.5 RegisterPublisher

The NotificationBroker MUST support the RegisterPublisher message exchange from the RegisterPublisher interface.

A Publisher can register its interest to publish messages through the NotificationBroker by sending a RegisterPublisherRequest. The NotificationBroker is responsible for managing the registration, and sending a RegisterPublisherResponse to the Publisher if the registration process succeeds. Otherwise, the NotificationBroker MUST fault. These message exchanges are further specified in the following Section 6.

5.6 CreatePullPoint

The NotificationBroker MUST support the CreatePullPoint interface. The 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 the NotificationBroker.

The NotificationBroker MAY support pull-style notification and attempt to create a PullPoint resource upon receiving a CreatePullPoint request. The NotificationBroker does not define additional constraints to its usage of the CreatePullPoint interface.

If the NotificationBroker does not support pull-style notification, it MUST response with the following fault upon receiving a CreatePullPoint request:

PullNotificationNotSupportedFault

- The NotificationBroker does not support pull-style notification.

6 RegisterPublisher Interface

The RegisterPublisher interface contains message exchanges for publisher registration. NotificationBroker implements the RegisterPublisher interface and is responsible for publisher registration. A NotificationBroker may reject processing certain publisher registrations for reasons such as lacking of authorization.

6.1 RegisterPublisher

The RegisterPublisher message is used by the Publisher to confirm its ability to publish on a given Topic or set of Topics. If an entity wishes to register a publisher, it MUST send a RegisterPublisher request message to the NotificationBroker. The format of the RegisterPublisher request message is:

```
…
<wsn-br:RegisterPublisherRequest>
   <wsn-br:PublisherReference>
      wsa:EndpointReference
   </wsn-br:PublisherReference>?
   <wsn-br:Topic dialect = “xsd:anyURI”>
      {any}
   </wsn-br:Topic>*
   <wsn-br:Demand>
       xsd:Boolean
   </wsn-br:Demand>?
   <wsn-br:InitialTerminationTime>
      xsd:dateTime
   </wsn-br:InitialTerminationTime>?
   {any} *
</wsn-br:RegisterPublisherRequest>
…
```

The WS-Addressing [action] Message Addressing Property MUST contain the URI http://docs.oasis-open.org/wsn/brw-1/RegisterPublisher/RegisterPublisherRequest.

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

/wsn-br:PublisherReference

An OPTIONAL endpoint reference element from WS-Addressing [WS-Addressing], used to identify an entity that wishes to become a Publisher. This component MUST appear if the /wsn-br:Demand component has value “true”. If this component is missing, the Publisher is either not a Web service, or does not wish to receive messages from the NotificationBroker.

/wsn-br:Topic

A set of TopicExpressions that identifies one or more Topics. If included, the given Publisher is registered to publish only on the set of Topics identified by this component. If this is missing the Publisher is registered to publish on any Topic supported by the NotificationBroker.

/wsn-br:Demand

A Boolean element with the default value “false”. If its value is “true”, then the intent of the Publisher is to use a demand-based model from the NotificationBroker (see Section 4). In this case, the NotificationBroker must observe the rules associated with demand-based publishing, including establishing a Subscription with the Publisher on those Topics and pausing/resuming those Subscriptions as the NotificationBroker receives Subscriptions for those Topics.

/wsn-br:InitialTerminationTime

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

The publisher should take care when choosing a value for InitialTerminationTime, and any subsequent values that modify the TerminationTime property of the publisher registration. It is RECOMMENDED that the publisher choose termination time values that are significantly (several magnitude) greater than the network latency expected in the interaction between the publisher and the broker. In so doing, the designer avoids undesirable results, such as the termination time having expired prior to the receipt of the published message. The WS-Resource Lifetime specification [WS-ResourceLifetime] (Section 6.1 Regarding time) contains further suggestions on how designers should reason about time values in a WS-Resource Lifetime application.

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

/ wsn-br:RegisterPublisherRequest/{any}

The RegisterPublisherRequest message allows for open content, in order to accommodate elements that may be needed by extensions built on WS-BrokeredNotification.

If a /wsn-br:Topic component is included in the message, the NotificationBroker MUST register the Web service specified by the /wsn-br:PublisherReference component as a Publisher on the set of Topics identified by the /wsn-br:Topic component. If for any reason the registration fails, the NotificationBroker MUST fault.

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

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

```
…
<wsn-br:RegisterPublisherResponse>
   <wsn-br:PublisherRegistrationReference>
      <wsa:Address>
         Address of PublisherRegistration Manager
      </wsa:Address>
      <wsa:ReferenceParameters>
          PublisherRegistration Identifier
      </wsa:ReferenceParameters>
      …
   </wsn-br:PublisherRegistrationReference>
</wsn-br:RegisterPublisherResponse>
…
```

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

http://docs.oasis-open.org/wsn/brw-1/RegisterPublisher/RegisterPublisherResponse

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

/wsn-br:PublisherRegistrationReference

A WS-Addressing endpoint reference to the PublisherRegistration resource created by the RegisterPublisher request message.

If the NotificationBroker does not succeed in responding to the RegisterPublisher request message with the RegisterPublisherResponse message, then it MUST send a fault. The NotificationBroker MUST fault if it rejects the publisher registration. This specification defines the following faults associated with failure to process the RegisterPublisher request message:

ResourceUnknownFault

- The NotificationBroker 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.

InvalidTopicExpressionFault

- The TopicExpression presented in the request message is invalid.

TopicNotSupportedFault

- The TopicExpression does not match any Topic supported by the NotificationBroker.

PublisherRegistrationRejectedFault

- The publisher registration is rejected by the NotificationBroker.

PublisherRegistrationFailedFault

- The publisher registration process has failed.

6.1.1 Example SOAP Encoding of the RegisterPublisher Message Exchange

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

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
        http://docs.oasis-open.org/wsn/brw-1/RegisterPublisher/
RegisterPublisherRequest
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsn-br:RegisterPublisher>
      <wsn-br:PublisherReference>
         <wsa:Address>
            http://www.producer.org/PublisherEndpoint
         </wsa:Address>
         <wsa:ReferenceParameters>
           <npex: NPResourceDisambiguator>
               uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
           </npex: NPResourceDisambiguator>
         </wsa:ReferenceParameters>
      </wsn-br:PublisherReference>
      <wsn-br:Topic  Dialect="http://docs.oasis-open.org/wsn/t-
1/SimpleTopicExpression">
          npex:SomeTopic
      </wsn-br:Topic>
      <wsn-br:Demand>
         true
      </wsn-br:Demand>
```

```
      <wsn-br:InitialTerminationTime>
         2003-12-25T00:00:00.00000Z
      </wsn-br:InitialTerminationTime>
    </wsn-br:RegisterPublisher>
  </s:Body>
</s:Envelope>
```

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

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
        http://docs.oasis-open.org/wsn/brw-
1/RegisterPublisher/RegisterPublisherResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsn-br:RegisterPublisherResponse>
      <wsn-br:PublisherRegistrationReference>
        <wsa:Address>
           http://www.producer.org/PublisherRegisterEndpoint
        </wsa:Address>
        <wsa:ReferenceParameters>
          <npex:NPubResourceId>
             uuid:95fefeb3-f37d-5dfe-44fe-221d9fceec99
          </npex:NPubResourceId>
        </wsa:ReferenceParameters>
      </wsn-br:PublisherRegistrationReference>
    </wsn-br:RegisterPublisherResponse>
  </s:Body>
</s:Envelope>
```

7 PublisherRegistrationManager Interface

The PublisherRegistrationManager interface defines message exchanges to manipulate PublisherRegistration resources. The PublisherRegistrationManager 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 PublisherRegistrationManager 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.

If the PublisherRegistrationManager does not respond to a request message with a respond message defined in this specification, then it MUST send a fault. The WS-ResourceProperties and WS-ResourceLifetime define some of these fault messages.

7.1 PublisherRegistration Resource Properties

In addition to the message exchanges described in this specification, a PublisherRegistrationManager 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. If it does so, the Resource Properties document defined by the PublisherRegistrationManager MUST also include references to the following resource property elements:

```
……
  targetNamespace="http://docs.oasis-open.org/wsn/br-1"
…
  <xsd:element name=”PublisherReference”
          type=”wsa:EndpointReference”/>
  <xsd:element name=”Topic” type=”wsn-b:TopicExpressionType”/>
  <xsd:element name=”Demand” type=”xsd:boolean” />
  <xsd:element name=”CreationTime” type=”xsd:dateTime” />
…
```

Furthermore, these references MUST reflect the minOccurs and maxOccurs properties as follows:

```
  <xsd:element ref=”wsn-br:PublisherReference”
               minOccurs=”0” maxOccurs=”1” />
  <xsd:element ref=”wsn-br:Topic”
               minOccurs=”0” maxOccurs=”unbounded” />
  <xsd:element ref=”wsn-br:Demand”
               minOccurs=”1” maxOccurs=”1” />
  <xsd:element ref=”wsn-br:CreationTime”
               minOccurs=”0” maxOccurs=”1” />
```

These resource property elements are further constrained as follows:

/wsn-br:PublisherReference, /wsn-br:Topic, and /wsn-br:Demand

These elements are defined in the description of the RegisterPublisher request message (see 6.1).

/wsn-br:CreationTime

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

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

- /wsn-br:TopicExpression and /wsn-br:Demand
    - Note: /wsn-br:Demand may not take the value “true” if there is no /wsn-br:PublisherReference resource property element in the resource property document.

7.2 Destroy

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

```
<wsn-br:DestroyRequest>
  {any} *
</wsn-br:DestroyRequest>
```

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

http://docs.oasis-open.org/wsn/brw-1/ PublisherRegistrationManager/DestroyRequest.

The Destroy request message allows for open content and contains an extension component / wsn-br:DestroyRequest/{any}.

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

```
<wsn-br:DestroyResponse/>
```

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

http://docs.oasis-open.org/wsn/brw-1/PublisherRegistrationManager/DestroyResponse.

If the PublisherRegistrationManager 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:

ResourceUnknownFault

- The PublisherRegistrationManager 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-Resource] specification.

ResourceNotDestroyedFault

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

7.2.1 Example SOAP Encoding of the Destroy Message Exchange

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

```
<s:Envelope ... >
  <s:Header>
```

```
    <wsa:Action>
      http://docs.oasis-open.org/wsn/brw-
1/PublisherRegistrationManager/DestroyRequest
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsn-br:DestroyRequest/>
  </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/brw-
1/PublisherRegistrationManager/DestroyResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsn-br:DestroyResponse/>
  </s:Body>
</s:Envelope>
```

8 Security Considerations

Baseline security considerations for WS-Notification are discussed in WS-BaseNotification specification. This section only covers additional broker specific security measurements.

8.1 Securing PublisherRegistration

In addition to the security policies for Notification process and Subscription process, WS-BrokeredNotification should provide policies such that:

1. only authorized Publishers can register with a NotificationBroker
2. only messages of the authorized Publishers and of registered topics, can be accepted by a NotificationBroker
3. only authorized principals can modify or delete PublisherRegistration resource

Given that WS-BrokeredNotification may implement WS-ResourceProperties and WS-ResourceLifetime, the security considerations outlined in these 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.

A NotificationBroker can support the security measurements of NotificationProducers and NotificationConsumers mentioned in WS-BaseNotification. Acting as an intermediary, NotificationBroker MAY also provide convenience to security management, including but not limited to:

- Controlling who can publish on a topic at publisher registration time
- Refusing to relay messages from unauthorized publishers
- Imposing security measurements on all messaging routing through the broker
- Providing convenience in messaging security management based on topics.

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

9 References

9.1 Normative

[RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

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

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

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

[WS-Addressing] http://www.w3.org/TR/ws-addr-core

[WS-BaseNotification] http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-cd-01.pdf

[WS-Topics] http://docs.oasis-open.org/wsn/wsn-WS-Topics-1.3-cd-01.pdf

[WS-Resource] http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-cd-01.pdf

[WS-ResourceLifetime] http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-cd-01.pdf

[WS-ResourceProperties] http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-cd-01.pdf

[WS-BaseFaults] http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-cd-01.pdf

9.2 Non-Normative

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

[WS-Security] http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

Appendix A. Acknowledgments

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

Sid Askary, Individual, Fred Carter, AmberPoint, Martin Chapman, Oracle, Dave Chappell, Sonic Software, Rick Cobb, KnowNow, Ugo Corda, SeeBeyond Technology Corporation, John Fuller, Individual, Stephen Graham, IBM, David Hull, Tibco, Hideharu Kato, Hitachi, Lily Liu, webMethods, 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, Hewlett-Packard, William Vambenepe, Hewlett-Packard, Kirk Wilson, Computer Associates.

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

In addition, the following people who are not members of the committee 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), Igor Sedukhin (Computer Associates), Eugène Sindambiwe (SAP), Jay Unger (IBM), Bill Weihl (Akamai Technologies), Mark Weitzel (IBM), Dan Wolfson (IBM).

Appendix B. XML Schema

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

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

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

  <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"/>

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

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

<!-- ====== Resource Properties for PublisherRegistration ========= -->
   <xsd:element name="PublisherReference"
                      type="wsa:EndpointReferenceType"/>
   <xsd:element name="Topic"
                      type="wsn-b:TopicExpressionType"/>
   <xsd:element name="Demand"
                      type="xsd:boolean"/>
   <xsd:element name="CreationTime"
                      type="xsd:dateTime"/>
   <xsd:element name="NotificationBrokerRP">
            <xsd:complexType>
               <xsd:sequence>
         <!-- From NotificationProducer -->
                  <xsd:element ref="wsn-b:TopicExpression"
                     minOccurs="0" maxOccurs="unbounded" />
                  <xsd:element ref="wsn-b:FixedTopicSet"
                     minOccurs="0" maxOccurs="1" />
                  <xsd:element ref="wsn-b:TopicExpressionDialect"
                     minOccurs="0" maxOccurs="unbounded" />

         <!-- NotificationBroker specific  -->
                  <xsd:element ref="wsn-br:RequiresRegistration"
                                minOccurs="1" maxOccurs="1" />
               </xsd:sequence>
```

```
            </xsd:complexType>
   </xsd:element>

<!-- ====== Resource Properties for PublisherRegistration ========= -->
   <xsd:element name="PublisherRegistrationRP">
            <xsd:complexType>
               <xsd:sequence>
         <!-- From WS-ResourceLifetime ScheduledResourceTermination -->
               <xsd:element ref="wsn-b:CurrentTime"
                  minOccurs="0" maxOccurs="1" />
               <xsd:element ref="wsn-b:TerminationTime"
                           minOccurs="1" maxOccurs="1" />

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

<!-- ========== Message Types for NotificationBroker  ============ -->
   <xsd:element name="RegisterPublisher">
            <xsd:complexType>
               <xsd:sequence>
               <xsd:element name="PublisherReference"
                           type="wsa:EndpointReferenceType"
                           minOccurs="0" maxOccurs="1" />
               <xsd:element name="Topic"
                           type="wsn-b:TopicExpressionType"
                           minOccurs="0" maxOccurs="unbounded" />
               <xsd:element name="Demand"
                           type="xsd:boolean" default="false"
                           minOccurs="0" maxOccurs="1" />
               <xsd:element name="InitialTerminationTime"
                           type="xsd:dateTime"
                           minOccurs="0" maxOccurs="1" />
               <xsd:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            </xsd:complexType>
   </xsd:element>

   <xsd:element name="RegisterPublisherResponse">
            <xsd:complexType>
               <xsd:sequence>
```

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


 <xsd:complexType name="InvalidTopicExpressionFaultType">
          <xsd:complexContent>
             <xsd:extension base="wsrf-bf:BaseFaultType"/>
          </xsd:complexContent>
 </xsd:complexType>
 <xsd:element name="InvalidTopicExpressionFault"
                   type="wsn-br:InvalidTopicExpressionFaultType"/>

 <xsd:complexType name="TopicNotSupportedFaultType">
          <xsd:complexContent>
             <xsd:extension base="wsrf-bf:BaseFaultType"/>
          </xsd:complexContent>
 </xsd:complexType>
 <xsd:element name="TopicNotSupportedFault"
              type="wsn-br:TopicNotSupportedFaultType"/>

 <xsd:complexType name="PublisherRegistrationRejectedFaultType">
          <xsd:complexContent>
             <xsd:extension base="wsrf-bf:BaseFaultType"/>
          </xsd:complexContent>
       </xsd:complexType>
 <xsd:element name="PublisherRegistrationRejectedFault"
              type="wsn-br:PublisherRegistrationRejectedFaultType"/>

 <xsd:complexType name="PublisherRegistrationFailedFaultType">
          <xsd:complexContent>
             <xsd:extension base="wsrf-bf:BaseFaultType"/>
          </xsd:complexContent>
       </xsd:complexType>
 <xsd:element name="PublisherRegistrationFailedFault"
              type="wsn-br:PublisherRegistrationFailedFaultType"/>

<xsd:complexType name="PullNotificationNotSupportedType">
  <xsd:complexContent>
    <xsd:extension base="wsrf-bf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="PullNotificationNotSupportedFault"
             type="wsn-br:PullNotificationNotSupportedType"/>

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

  <xsd:complexType name="ResourceNotDestroyedFaultType">
    <xsd:complexContent>
      <xsd:extension base="wsrf-bf:BaseFaultType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="ResourceNotDestroyedFault"
               type="wsn-br:ResourceNotDestroyedFaultType"/>

</xsd:schema>
```

Appendix C. WSDL 1.1

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

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

<wsdl:definitions name="WS-BrokeredNotification"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsa="http://www.w3.org/2005/03/addressing"
  xmlns:wsn-br="http://docs.oasis-open.org/wsn/br-1"
  xmlns:wsn-brw="http://docs.oasis-open.org/wsn/brw-1"
  xmlns:wsn-b="http://docs.oasis-open.org/wsn/b-1"
  xmlns:wsn-bw="http://docs.oasis-open.org/wsn/bw-1"
  xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-1"
  xmlns:wsrf-rw="http://docs.oasis-open.org/wsrf/rw-1"
  targetNamespace="http://docs.oasis-open.org/wsn/brw-1">

<!-- ========================== Imports ========================== -->
   <wsdl:import namespace="http://docs.oasis-open.org/wsrf/rw-1"
       location="http://docs.oasis-open.org/wsrf/rw-1"/>

   <wsdl:import namespace="http://docs.oasis-open.org/wsn/bw-1"
       location="http://docs.oasis-open.org/wsn/bw-1"/>

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

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

   <wsdl:message name="RegisterPublisherResponse">
      <wsdl:part name="RegisterPublisherResponse"
                 element="wsn-br:RegisterPublisherResponse"/>
   </wsdl:message>
```

```
   <wsdl:message name="InvalidTopicExpressionFault">
      <wsdl:part name="InvalidTopicExpressionFault"
            element="wsn-br:InvalidTopicExpressionFault" />
   </wsdl:message>

   <wsdl:message name="TopicNotSupportedFault">
      <wsdl:part name="TopicNotSupportedFault"
            element="wsn-br:TopicNotSupportedFault" />
   </wsdl:message>

   <wsdl:message name="PublisherRegistrationRejectedFault">
      <wsdl:part name="PublisherRegistrationRejectedFault"
            element="wsn-br:PublisherRegistrationRejectedFault"/>
   </wsdl:message>

   <wsdl:message name="PublisherRegistrationFailedFault">
      <wsdl:part name="PublisherRegistrationFailedFault"
            element="wsn-br:PublisherRegistrationFailedFault"/>
   </wsdl:message>

   <wsdl:message name="PullNotificationNotSupportedFault">
      <wsdl:part name="PullNotificationNotSupportedFault"
            element="wsn-br:PullNotificationNotSupportedFault"/>
   </wsdl:message>

   <wsdl:message name="DestroyRequest">
      <wsdl:part name="DestroyRequest"
            element="wsn-br:Destroy"/>
   </wsdl:message>

   <wsdl:message name="DestroyResponse">
      <wsdl:part name="DestroyResponse"
            element="wsn-br:DestroyResponse"/>
   </wsdl:message>

   <wsdl:message name="ResourceNotDestroyedFault">
      <wsdl:part name="ResourceNotDestroyedFault"
            element="wsn-br:ResourceNotDestroyedFault"/>
   </wsdl:message>

<!-- ================== PortType Definitions ==================== -->

<!-- ========== RegisterPublisher ============ -->
<wsdl:portType name="RegisterPublisher">
    <wsdl:operation name="RegisterPublisher">
         <wsdl:input  message="wsn-brw:RegisterPublisherRequest"/>
         <wsdl:output message="wsn-brw:RegisterPublisherResponse"/>
         <wsdl:fault  name="ResourceUnknownFault"
                      message="wsrf-rw:ResourceUnknownFault"/>
         <wsdl:fault  name="InvalidTopicExpressionFault"
                      message="wsn-brw:InvalidTopicExpressionFault"/>
```

```
         <wsdl:fault  name="TopicNotSupportedFault"
                      message="wsn-brw:TopicNotSupportedFault"/>
         <wsdl:fault  name="PublisherRegistrationRejectedFault"
                 message="wsn-brw:PublisherRegistrationRejectedFault"/>
         <wsdl:fault  name="PublisherRegistrationFailedFault"
                 message="wsn-brw:PublisherRegistrationFailedFault"/>
    </wsdl:operation>
  </wsdl:portType>


<!-- ========== NotificationBroker PortType Definition ============ -->
   <wsdl:portType name="NotificationBroker">
      <!-- ============= extends NotificationConsumer ============= -->
      <wsdl:operation name="Notify">
         <wsdl:input message="wsn-bw:Notify" />
      </wsdl:operation>

      <!-- ============= extends NotificationProducer ============= -->
      <wsdl:operation name="Subscribe">
         <wsdl:input  message="wsn-bw:SubscribeRequest" />
         <wsdl:output message="wsn-bw:SubscribeResponse" />
         <wsdl:fault  name="ResourceUnknownFault"
                      message="wsrf-rw:ResourceUnknownFault" />
         <wsdl:fault  name="InvalidFilterFault"
                   message="wsn-bw:InvalidFilterFault"/>
         <wsdl:fault  name="TopicExpressionDialectUnknownFault"
                   message="wsn-
bw:TopicExpressionDialectUnknownFault"/>
         <wsdl:fault  name="InvalidTopicExpressionFault"
                      message="wsn-bw:InvalidTopicExpressionFault" />
         <wsdl:fault  name="TopicNotSupportedFault"
                      message="wsn-bw:TopicNotSupportedFault" />
         <wsdl:fault  name="InvalidProducerPropertiesExpressionFault"
             message="wsn-
bw:InvalidProducerPropertiesExpressionFault"/>
         <wsdl:fault  name="InvalidMessageContentExpressionFault"
             message="wsn-bw:InvalidMessageContentExpressionFault"/>
         <wsdl:fault  name="InvalidUseRawValueFault"
             message="wsn-bw:InvalidUseRawValueFault"/>
         <wsdl:fault  name="UnacceptableInitialTerminationTimeFault"
             message="wsn-bw:UnacceptableInitialTerminationTimeFault"/>
         <wsdl:fault  name="SubscribeCreationFailedFault"
                      message="wsn-bw:SubscribeCreationFailedFault"/>
      </wsdl:operation>
      <wsdl:operation name="GetCurrentMessage">
         <wsdl:input  message="wsn-bw:GetCurrentMessageRequest"/>
         <wsdl:output message="wsn-bw:GetCurrentMessageResponse"/>
         <wsdl:fault  name="ResourceUnknownFault"
                      message="wsrf-rw:ResourceUnknownFault"/>
         <wsdl:fault  name="TopicExpressionDialectUnknownFault"
                   message="wsn-bw:TopicExpressionDialectUnknownFault"/>
```

```
         <wsdl:fault  name="InvalidTopicExpressionFault"
                      message="wsn-bw:InvalidTopicExpressionFault"/>
         <wsdl:fault  name="TopicNotSupportedFault"
                      message="wsn-bw:TopicNotSupportedFault"/>
         <wsdl:fault  name="NoCurrentMessageOnTopicFault"
                      message="wsn-bw:NoCurrentMessageOnTopicFault"/>
         <wsdl:fault  name="MultipleTopicsSpecifiedFault"
                      message="wsn-bw:MultipleTopicsSpecifiedFault"/>
      </wsdl:operation>

      <!-- ========= extends RegisterPublisher ======= -->
      <wsdl:operation name="RegisterPublisher">
         <wsdl:input  message="wsn-brw:RegisterPublisherRequest"/>
         <wsdl:output message="wsn-brw:RegisterPublisherResponse"/>
         <wsdl:fault  name="ResourceUnknownFault"
                      message="wsrf-rw:ResourceUnknownFault"/>
         <wsdl:fault  name="InvalidTopicExpressionFault"
                      message="wsn-brw:InvalidTopicExpressionFault"/>
         <wsdl:fault  name="TopicNotSupportedFault"
                      message="wsn-brw:TopicNotSupportedFault"/>
         <wsdl:fault  name="PublisherRegistrationRejectedFault"
                 message="wsn-brw:PublisherRegistrationRejectedFault"/>
         <wsdl:fault  name="PublisherRegistrationFailedFault"
                   message="wsn-brw:PublisherRegistrationFailedFault"/>
      </wsdl:operation>

      <!-- ========= extends CreatePullPoint ======= -->
      <wsdl:operation name="CreatePullPoint">
         <wsdl:input  name="CreatePullPointRequest"
                      message="wsn-bw:CreatePullPointRequest" />
         <wsdl:output name="CreatePullPointResponse"
                      message="wsn-bw:CreatePullPointResponse" />
         <wsdl:fault  name="UnableToCreatePullPoint"
                      message="wsn-bw:UnableToCreatePullPoint" />
         <wsdl:fault  name="PullNotificationNotSupportedFault"
                    message="wsn-brw:PullNotificationNotSupportedFault"/>
      </wsdl:operation>
   </wsdl:portType>

<!-- ===== PublisherRegistrationManager PortType Definition ====== -->
   <wsdl:portType name="PublisherRegistrationManager">

   <!-- === Destroy:ImmediateResourceTermination========== -->
      <wsdl:operation name="Destroy">
         <wsdl:input  name="DestroyRequest"
                      message="wsn-brw:DestroyRequest" />
         <wsdl:output name="DestroyResponse"
                      message="wsn-brw:DestroyResponse" />
        <wsdl:fault name="ResourceUnknownFault"
                   message="wsrf-rw:ResourceUnknownFault" />
         <wsdl:fault  name="ResourceNotDestroyedFault"
```

```
                        message="wsn-brw:ResourceNotDestroyedFault" />
      </wsdl:operation>
   </wsdl:portType>
</wsdl:definitions>
```

Appendix D. 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 implementers 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.