Web Services Topics 1.3 (WS-Topics)

Committee Specification, 31 July 2006

Document identifier:

wsn-ws_topics-1.3-spec-cs-01

Location:

http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-cs-01.pdf

Editors:

William Vambenepe, HP <vbp@hp.com>
Steve Graham, IBM <sggraham@us.ibm.com>
Peter Niblett, IBM <peter_niblett@uk.ibm.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 a mechanism to organize and categorize items of interest for subscription known as “topics”. These are used in conjunction with the notification mechanisms defined in WS-BaseNotification. WS-Topics defines three topic expression dialects that can be used as subscription expressions in subscribe request messages and other parts of the WS-Notification system. It further specifies an XML model for

describing metadata associated with topics. This specification should be read in conjunction with the WS-Base Notification specification.

Status:

This document is published by the OASIS WS-Notification Technical Committee as a “Committee Specification”.

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 provided by Message Oriented Middleware vendors, or in system and device management domains.

This document defines a mechanism to organize and categorize items of interest for subscription known as “topics”. These are used in conjunction with the notification mechanisms defined in WS-Base Notification.

WS-Topics defines four topic expression dialects that can be used as subscription expressions in subscribe request messages and other parts of the WS-Notification system. It further specifies an XML model for describing metadata associated with topics. This specification should be read in conjunction with the WS-BaseNotification specification.

1.1 Goals and Requirements

The goal of the WS-Topics specification is to define a mechanism to organize and categorize items of interest for subscription known as “topics”. It defines a set of topic expression dialects that can be used as subscription expressions in subscribe request messages and other parts of the WS-Notification system.

1.1.1 Requirements

In meeting this goal, the specification must address the following specific 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 permit transformation and aggregation of Topics: It must be possible to construct configurations (using intermediary brokers) where the Topic subscribed to by the NotificationConsumer differs from the Topic published to by the NotificationProducer, yet Notifications from the NotificationProducer are routed to the NotificationConsumer by a broker that is acting according to administratively-defined rules.
- Must permit non-centralized development of a topic tree: It must be possible for actors to define additional topics based on existing topics without requiring coordination with the actor responsible for creating the topics that are being built on.

1.1.2 Non-Goals

The following aspects are outside the scope of these specifications:

- Defining the format of notification payloads: The data carried in notification messages is application-domain specific, and this specification does not prescribe any particular format for this data.

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 [RFC2119].

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”. 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.
- Elements with simple content are conventionally assigned a value which corresponds to the type of their content, as defined in the normative schema.
- The use of {any} indicates the presence of an element wildcard (<xs:any/>).
- The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).
- In the interest of brevity, some extensibility points have been omitted from the Pseudo-schemas.

```
<!-- 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 file describing the elements 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 included in the appendices.

1.3 Namespaces

The following namespaces are used in this document:

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema
wsnt	http://docs.oasis-open.org/wsn/b-2
wstop	http://docs.oasis-open.org/wsn/t-1

2 Terminology and Concepts

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

Topic:

- A Topic is the concept used to categorize Notifications and their related Notification schemas.
- Topics are used as part of the matching process that determines which (if any) subscribing NotificationConsumers should receive a Notification.
- When it generates a Notification, a Publisher can associate it with one or more Topics. The relation between Situation (as defined in [WS-BaseNotification]) and Topic is not specified by WS-Notification but MAY be specified by the designer of the Topic Namespace.
- A synonym in some other publish/subscribe models is *subject*.

Topic Namespace:

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

Topic Tree:

- A hierarchical grouping of Topics.

Topic Set:

- The collection of Topics supported by a NotificationProducer

3 Topics and Topic Namespaces

The WS-Notification specifications allow the use of Topics as a way to organize and categorize a set of Notifications. The Topics mechanism provides a convenient means by which subscribers can reason about Notifications of interest. Topics appear in several places within the WS-Notification system. As part of the publication of a Notification, a Publisher may associate it with one or more Topics. When a Subscriber creates a Subscription, it may supply a Topic filter expression, associating the Subscription with one or more Topics. The NotificationProducer uses these sets of Topics as part of the matching process: a Notification is delivered to a NotificationConsumer if the set of Topics associated with the Subscription has a non-empty intersection with the set of Topics associated with the Notification.

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

It is important to understand the distinction between a Topic Namespace and the set of Topics (the “Topic Set”) supported by a NotificationProducer. A Topic Namespace is just an abstract set of Topic definitions. While it is certainly possible for a given Topic Namespace to be used by exactly one Notification Producer, there is no expectation that this will be the case. Topics from a single Topic Namespace can be referenced in the Topic Sets of many different NotificationProducers. Moreover the Topic Set of a NotificationProducer MAY contain Topics from several different Topic Namespaces. This concept is expanded upon in section 11.

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

The rationale for hierarchical topic structures is:

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

Note: Although WS-Notification permits hierarchical topic structures, there is no requirement or expectation that all Topic Namespaces will contain them. It is perfectly possible for a Topic Namespace to contain only root Topics (possibly only a single root Topic). A NotificationProducer may restrict its Topic Set to include only Topics from Topic Namespaces that just contain root Topics; even if it does include Topics from a Topic Namespace that contains topic hierarchies, it may choose only to support root Topics from that Topic Namespace.

A Topic Namespace is thus a collection (forest) of Topic Trees. The Topic Namespace may contain additional metadata relating to its member Topics. The metadata describing a particular Topic Namespace can be modeled as an XML document (see section 5).

Each Topic has a local name, an NCName as defined by [XML-Namespaces]. All root Topics must have unique names within their Topic Namespace. In this way, a root Topic can be uniquely referenced by a QName formed by combining the XML Namespace associated with the Topic Namespace and the local name of the root Topic. Child Topics can be referred to relative to their ancestor root Topic's QName using a path-based TopicExpression dialect (see section 8).

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

WS-Topics allows a Topic Namespace to contain one or more extensions to a Topic Tree that is defined in another Topic Namespace. These extensions can be used as though they were child Topics of Topics in that Topic Namespace. This mechanism allows one organization to define a set of core hierarchical topic structures (in one Topic Namespace), and another organization to add its own Topics (from its own separate Namespace) into this hierarchy.

4 Example

Consider a Topic Namespace that can be depicted as illustrated by Figure 1. The Topic Namespace is contained in the "http://example.org/topicSpace/example1" namespace. This Topic Namespace has two root Topics, named t1 and t4. Topic t1 has two child Topics, t2 and t3. Topic t4 has two child Topics, t5 and t6.

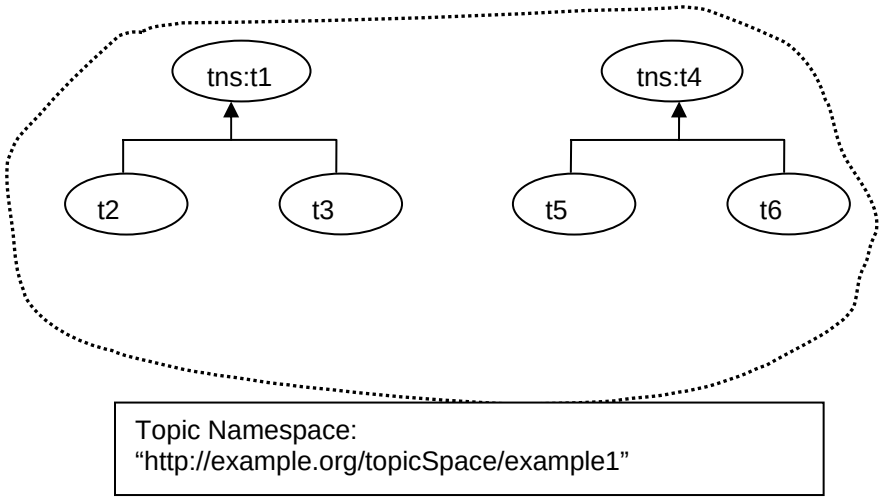

Figure 4: Example Topic Namespace

This Topic Namespace and its metadata can be described using the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<wstop:TopicNamespace name="TopicSpaceExample1"
   targetNamespace="http://example.org/topicSpace/example1"
   xmlns:tns="http://example.org/topicSpace/example1"
   xmlns:xyz="http://example.org/anotherNamespace"
   xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://docs.oasis-open.org/wsn/t-1
                       http://docs.oasis-open.org/wsn/t-1.xsd" >
   <wstop:Topic name="t1">
      <wstop:Topic name="t2" messageTypes="xyz:m1 tns:m2"/>
      <wstop:Topic name="t3" messageTypes="xyz:m3"/>
   </wstop:Topic>
   <wstop:Topic name="t4">
      <wstop:Topic name="t5" messageTypes="tns:m3"/>
      <wstop:Topic name="t6"/>
```



```
   </wstop:Topic>
</wstop:TopicNamespace>
```

This Topic Namespace defines six Topics – the two root Topics and their four children. Continuing with our example, we introduce a NotficationProducer that wishes to use three of these Topics,

- The root Topic tns:t1
- The t2 child of tns:t1
- The t5 child of tns:t4

The NotificationProducer supports these Topics by adding them to its Topic Set. The Topic Set can itself be represented as an XML document as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wstop:TopicSet xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
xmlns:tns="http://example.org/topics/example1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://docs.oasis-open.org/wsn/t-1
                    http://docs.oasis-open.org/wsn/t-1.xsd">

   <tns:t1 wstop:topic="true">
         <t2 wstop:topic="true"/>
   </tns:t1>
   <tns:t4>
         <t5 wstop:topic="true"/>
   </tns:t4>

</wstop:TopicSet>
```

The Topic Set document has a root element called TopicSet, and each Topic supported by the NotificationProducer is represented by an element in the document. The Topic name is used as this element's QName, and its position in the document hierarchy matches the position of the Topic in the Topic hierarchy. So root Topics (for example `tns:t1`) appear as children of the TopicSet element, and other Topics are represented by elements that are children of the element that corresponds to their parent Topic.

Elements that represent Topics are marked with a wstop:topic attribute taking the value "true". This allows the NotificationProducer to insert additional elements that represent other items of metadata; these other items can be distinguished from the elements that represent Topics since they don't have @wstop:topic="true". It also means that the document can represent a Topic Set which includes child Topics without including their parents. In this example the TopicSet document contains a `tns:t4` element, which allows it to include `tns:t4/t5`. However since the `tns:t4` element does not have a @wstop:topic="true" the `tns:t4` it does not represent a Topic, so the root Topic does not form part of this Topic Set

We describe the details behind modeling Topic Namespaces and Topics in the following sections.

5 Modeling Topic Namespaces in XML

The WS-Topics XML Schema contains element and type definitions used to create Topic Namespace documents. A Topic Namespace document is associated with a single Topic Namespace and contains the names of Topics in that Topic Namespace along with their metadata. It might include all the Topics in that Topic Namespace, or just a subset of them.

The following pseudo-schema gives a non-normative description of a TopicNamespace element:

```
<TopicNamespace name=xsd:NCName? targetNamespace=xsd:anyURI
final=xsd:boolean? >
     <Topic … />*
</TopicNamespace>
```

A TopicNamespace element is constrained in the following way:

/wstop:TopicNamespace

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

/wstop:TopicNamespace/@name

A name that can be assigned to the TopicNamespace element for light-weight documentation purposes.

/wstop:TopicNamespace/@targetNameSpace

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

/wstop:TopicNamespace/@final

An attribute whose value is of type xsd:boolean. The default value (to be assumed if the attribute is omitted) is “false”. If the value is “true” it indicates that any Topic which appears in a NotificationProducer’s Topic Set and uses this target namespace MUST have its root explicitly defined in the TopicNamespace.

/wstop:TopicNamespace/Topic

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

/wstop:TopicNamespace/{any}

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

/wstop:TopicNamespace/@{any}

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

6 Modeling Topics in XML

WS-Notification defines an XML representation of a Topic that can be represented as follows:

```
<TopicNamespace name=… targetNamespace=…>
  <Topic name=xsd:NCName messageTypes=list of xsd:QName?
         final=xsd:boolean? parent=ConcreteTopicExpression? >
      <MessagePattern>QueryExpressionType</MessagePattern>?
      <Topic … />*
  </Topic>
  …
</TopicNamespace>
```

A Topic element is further constrained in the following way:

/wstop:Topic

This describes the definition of a Topic. It contains a MessagePattern child element (which can be omitted) followed by zero or more child Topic elements.

The namespace of a Topic is defined as the targetNamespace of the TopicNamespace element ancestor of the Topic. As we saw in section 5, individual root Topics are modeled by defining Topic child elements of the TopicNamespace element.

/wstop:Topic/@name

The NCName of this Topic. This attribute is required. These NCNames must all be unique with respect to the parent element (TopicNamespace or Topic) that contains this Topic. In the case of a root Topic, Topic/@name gives the local name of the Topic, while its namespace is given by the @targetNamespace attribute of the containing TopicNamespace element. A root Topic can be identified using a QName whose prefix is bound to this namespace and whose local part is the local name.

/wstop:Topic/@messageTypes

A list of the QNames of XML global element declarations (GEDs) that define the kinds of Notification that can be used with the Topic. If the list is present then a Publisher using a given Topic MUST NOT generate a Notification with root element whose QName is not included in this list. If the list is empty, or the attribute is not defined, then a Notification can have any XML element as root. A given QName can appear multiple times in the list; second or subsequent appearance of a given QName are not meaningful and SHOULD be ignored.

/wstop:Topic/@final

An attribute whose value is of type xsd:boolean. The default value (to be assumed if the attribute is omitted) is “false”. If the value is “true” it indicates that the NotificationProducer MUST NOT use child Topics of this Topic other than those explicitly shown in this TopicSpace document. This means that it is an error if a Publisher or Subscriber attempts to use a TopicExpression that references child Topics of a Topic that is marked as @final=”true” – other than child Topics that are explicitly included in the definition of the Topic.

/wstop:Topic/@parent

An attribute whose value is a ConcreteTopicExpression. If present it designates a parent

Topic and indicates that this root Topic, and any child Topics descended from it, are extensions of that parent. See section 6.1 for a description of extension Topics. This attribute MUST NOT be used on Topics other than root Topics.

/wstop:Topic/MessagePattern

A QueryExpression. If it is present, this QueryExpression is used to describe the pattern of the message that will appear on the Topic. Conceptually, the MessagePattern component can be thought of as the object of a boolean() expression, evaluated against a Notification. This boolean() expression, with the value of MessagePattern as parameter, is guaranteed to evaluate to "true" when evaluated in the context of any Notification that is associated with the Topic. The MessagePattern component constrains the Notification Messages that can be used with the Topic. It is additional to the constraint contained in @messageTypes, and provides a further refinement to that constraint.

/wstop:Topic/MessagePattern/@Dialect

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

/wstop:Topic/Topic

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

/wstop:Topic/{any}

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

/wstop:Topic/@{any}

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

6.1 Extension Topics

A NotificationProducer MAY support Topics that are marked as Extensions of other Topics by the wstop:/Topic/@parent attribute. Support for such Topics is OPTIONAL, a NotificationProducer MAY choose not to support Topic Namespaces that contain Extension Topics.

If the @parent attribute is used, the following constraints MUST be obeyed by the designer of the Topic Namespace:

1. The Topic containing the @parent attribute (the "Extension Topic") MUST be a root Topic in its Topic Namespace
2. The Topic referenced by the @parent attribute (the "Parent Topic") MUST be from a different Topic Namespace. It need not be a root Topic in that Namespace.
3. The Topic referenced by the @parent attribute can be an Extension Topic or the child of an Extension Topic, however it MUST be possible to follow a chain of Extension/parent/root Topics back to a root Topic that is not an Extension Topic. Moreover a given Topic Namespace MUST NOT appear more than once in this chain. This means that circular references, e.g. A extends B / B extends A are NOT permitted.
4. The Parent Topic MUST NOT be marked as final.

Although it appears as a root topic in its namespace, an Extension Topic, or its descendents, can only be referenced using a path-based TopicExpression dialect in which the path passes through the Parent Topic. In the case where the Parent itself is Extension Topic (or is descended from one) this requirement applies recursively to the Parent Topic as well. Note that if the dialect permits them, wild cards can be used in the TopicExpression to avoid having to include the Parent Topic(s) explicitly in the path expression.

7 Modeling Topic Sets in XML

The WS-Topics XML Schema contains element and type definitions used to create Topic Set documents. A Topic Set document gives an XML representation of the set of Topics supported by a NotificationProducer. It has the wstop:TopicSet element as its document root, and contains zero or more XML elements that represent the Topics in the Topic Set.

- If a Topic is defined as a root Topic of its Topic Namespace, and is not marked as an Extension Topic, then it MUST appear as an immediate child of wstop:TopicSet. In addition, if this Topic comes from any Namespace other than the ad-hoc Topic Namespace described in section 10, then it MUST be represented by a namespace-qualified element, with a Namespace name that is the targetNamespace of the Topic Namespace.
- If a Topic is an Extension Topic, then it MUST NOT appear as an immediate child of wstop:TopicSet, however it MUST be represented by a namespace-qualified element, with a Namespace name that is the targetNamespace of the Topic Namespace.
- If a Topic is not a root Topic it MUST be represented by a non-qualified (NCName) element, and MUST NOT appear as an immediate child of wstop:TopicSet.

Section 4 includes an example TopicSet showing both root and child Topics.

The following pseudo-schema gives a non-normative description of a TopicSet element:

```
<TopicSet>
  {any}*
</TopicSet>
```

A TopicSet document is constrained in the following way:

/wstop:TopicSet

The top-level element in a Topic Set document. It contains a Topic element corresponding to each supported Topic, along with OPTIONAL provider-specific additional elements. There MUST NOT be a default XML namespace in scope for any of the descendents of TopicSet (this ensures that all root Topics in the Topic Set can be identified by virtue of having QName prefixes)

/wstop:TopicSet/{any}

The TopicSet contains an element corresponding to each Topic that is included in the Topic Set. The Topic name is used as the local part of the element name, and the element is qualified with a Namespace if and only if it represents a root Topic from a Topic Namespace other than the ad-hoc Topic Namespace. The position of the element in the document hierarchy matches the position of the Topic in the Topic hierarchy. The TopicSet element can contain additional elements that do not represent Topics in the Set – it MUST contain additional, appropriately named elements where these are needed to ensure the correct position in the hierarchy of the elements that do represent Topics in the Set. It MAY contain additional elements that carry Producer-specific metadata.

/wstop:TopicSet//*/@topic

This is an attribute of type xsd:boolean, used to distinguish elements that represent Topics in the set from those that do not. An element in the content of wstop:TopicSet MUST have a wstop:@topic attribute with a value of “true” if and only if it represents a Topic in the Topic Set.

/wstop:TopicSet/@{any}

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

If a Topic is defined as an Extension of another Topic then its Parent Topic MUST be represented by an element in the TopicSet (although it need not have wstop:@topic=”true”), and the element representing the Extension Topic MUST be a child of the element representing the Parent Topic. This means that all Extension Topics can be referenced using paths that include the root Topic from the Parent Topic’s Topic Namespace.

8 Topic Expression Dialects

Topics are referred to by TopicExpressions. There are several places in WS-Notification where these expressions can appear:

- As a component of the Subscribe message request to a NotificationProducer;
- As a component of a Notification message sent to a NotificationConsumer or NotificationBroker;
- In the TopicExpression Resource Property element(s) associated with the NotificationProducer role

A non-normative syntax for a TopicExpression is shown below:

```
<wsnt:TopicExpression Dialect= xsd:anyURI?>
   {any}?
</wsnt:TopicExpression>
```

A TopicExpression has two components:

/wsnt:TopicExpression/@Dialect

The Dialect component contains a URI which identifies the type of grammar used in the TopicExpression. This URI may be one from the set defined in this document, or may be a URI defined elsewhere.

/wsnt:TopicExpression/{any}

The content of the TopicExpression is an expression in the grammar defined by the expression language identified by the @Dialect component.

The purpose of a TopicExpression is to identify a set of one or more Topics. These Topics can come from one or more Topic Namespaces.

This specification defines a number of Dialects that can be used to construct TopicExpressions.

These Dialects make use of Namespace prefixes as defined in [XML-Namespaces]. The namespace declarations that specify the mapping of a prefix to an actual namespace URI can be found amongst any namespace declaration in scope for the TopicExpression. Note: Some XML processors might modify the namespace declarations. Designers should be aware that such transforms exist and might render the expression incoherent; as it is likely the change in namespace declaration will not update a QName embedded within a string.

8.1 Simple TopicExpression Dialect

This specification defines a simple TopicExpression dialect with the following URI:

```
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple
```

This dialect is defined to standardize a very simple Topic Expression language for use by resource constrained entities in the WS-Notification system that deal only with simple Topic Namespaces. In this dialect the TopicExpression is simply the QName of a root Topic, consisting

of a namespace prefix that identifies the Topic Space, and a local name that identifies the root Topic within that Topic Space.

A TopicExpression in this dialect is a token (as defined by XML Schema) with an additional constraint on its format. The constraint is the token must contain a TopicExpression. The grammar is defined using the simple Extended Backus Naur Form (EBNF) also used in [XML]:

```
[1] TopicExpression     ::=    RootTopic
[2] RootTopic           ::=    QName
    [ vc: If a namespace prefix is included in the RootTopic, it must correspond to a valid
    Topic Namespace definition and the local name must correspond to the name of a root
    Topic defined in that namespace.]
```

Because the only valid TopicExpression in this dialect is a QName, only root Topics can be addressed by this grammar. For those entities that support only this dialect of TopicExpression, only simple Topic Namespaces (TopicNamespaces that only define root Topics) SHOULD be used.

Although an Extension Topic is a root Topic in its own namespace, Extension Topics can not be referenced using this dialect. An Extension Topic MUST only be referenced using a path than includes its Parent Topic.

An example TopicExpression within this dialect is shown below:

```
…
   xmlns:tns="http://example.org/topics/example1"
…

<wsnt:TopicExpression
   Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple">
          tns:t1
</wsnt:TopicExpression>
```

This TopicExpression identifies the root Topic t1 within the Topic Namespace corresponding to the namespace prefix tns:.

8.2 Concrete TopicExpression Dialect

This specification defines a path-based TopicExpression dialect with the following URI:

```
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete
```

The Concrete TopicExpression is used to identify a single Topic within a Topic Namespace, using a path notation. As it uses a path notation, it can identify any Topic within a Topic Namespace – it is not limited to root Topics.

A TopicExpression in this dialect is a token (as defined by XML Schema) with an additional constraint on its format. The constraint is the token must contain a TopicExpression. The grammar is defined using the simple Extended Backus Naur Form (EBNF) also used in [XML]:

```
[3] TopicExpression     ::=    TopicPath
[4] TopicPath           ::=    RootTopic ChildTopicExpression*
```


[5] RootTopic ::= QName
[vc: If a namespace prefix is included in the RootTopic, it must correspond to a valid Topic Namespace definition and the local name must correspond to the name of a root Topic defined in that namespace.]
[6] ChildTopicExpression ::= '/' ChildTopicName
[7] ChildTopicName ::= QName | NCName
[vc: The NCName or local part of the QName, must correspond to the name of a Topic within the descendant path from the RootTopic, where each forward slash denotes another level of child Topic elements in the path.]

Note: White space is not permitted within a Concrete TopicExpression.

An example TopicExpression within this dialect is shown below:

```
…
   xmlns:tns="http://example.org/topics/example1"
…

<wsnt:TopicExpression
 Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete">
          tns:t1/t3
</wsnt:TopicExpression>
```

This TopicExpression identifies the Topic named "t3", child of Topic tns:t1.

As with XPath, this TopicExpression syntax uses the slash ("/") to describe *child of*.

This dialect allows namespace prefixes to be included in the path. Prefixes are used to switch between namespaces when passing from a parent Topic to an Extension Topic as shown in the following example:

```
…
   xmlns:tns1=http://example.org/topics/example1"
   xmlns:tns2=http://example.org/topics/example2"
…
<wsnt:TopicExpression
 Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete">
          tns1:t1/tns2:t3
</wsnt:TopicExpression>
```

This TopicExpression identifies the Topic named "t3" from Topic Namespace http://example.org/topics/example2", which was defined in that namespace as an extension of Topic t1 from Topic Namespace http://example.org/topics/example1".

An Extension Topic can only be referenced using a path than includes its Parent Topic in the manner just shown. In this example it would not be valid to attempt to refer to the topic by using the expression `tns2:t3.`

Namespace prefixes MUST only be used on root Topics (this includes Extension Topics since these are by definition root Topics).

Note: The Simple TopicExpression dialect defined in the previous section is a subset of the Concrete TopicExpression dialect.

8.3 Full TopicExpression Dialect

This specification defines a fully featured path-based TopicExpression dialect with the following URI:

```
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Full
```

This dialect allows TopicExpressions that identify more than one Topic (possibly from multiple Topic Namespaces). It extends the Concrete TopicExpression dialect, in the sense that every expression in the Concrete TopicExpression dialect is also valid in the Full TopicExpression dialect, and has the same meaning.

Full TopicExpressions are XPath 1.0 [XPATH] relative location path expressions with some additional syntactic constraints listed in this section. The XPath expression is evaluated over a NotificationProducer's TopicSet document as defined in section 7. The TopicExpression identifies the set of Topics that correspond to the elements in the node-set that results from evaluating the location path contained in the TopicExpression, using standard XPath 1.0. The initial context node for this evaluation is the wstop:TopicSet root element. Note that some of the elements returned by the evaluation might not correspond to Topics (these are elements which do not have @topic="true").

The Full TopicExpression dialect does not permit the use of the entire XPath language. This specification provides syntactic constraints on the contents of the Full TopicExpression that limit the constructs that can be used.

A TopicExpression in this dialect is a token (as defined by XML Schema) with an additional constraint on its format. The constraint is that the token must conform to production rule [1] in the following grammar. This grammar is defined using the simple Extended Backus Naur Form (EBNF) also used in [XML]:

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

As with the ConcreteTopicExpression, the ChildTopicName [8] MAY contain a namespace prefix to allow an expression to include an extension Topic. Namespace prefixes MUST only be used on root Topics (note that an extension Topic is by definition a root Topic).

655 Note: An Extension Topic is not permitted to appear as the as an immediate child of the
656 wstop:TopicSet element. This means that an Extension Topic can only be referenced using a
657 path than includes its Parent Topic (possibly wildcarded).

658 Note: White space is not permitted within a Full TopicExpression.

659 Note: The Concrete TopicExpression dialect defined in the previous section is a subset of the Full
660 TopicExpression dialect that contains no wildcards, '//' separators, or '|' operators.

661 The dialect is further explained by the following examples (for the sake of brevity, the examples
662 show only the content of the TopicExpression element):

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

665 `“tns:t1/*”`

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

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

671 `“tns:t1/*/t3”`

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

673 The wildcard * can also be used in place of a root Topic name, for example:

674 `“tns:*”`

675 This TopicExpression identifies all root Topics in the tns: Topic Namespace.

676 As in full XPath the // separator is used to identify all descendents (subject of course to the
677 constraints implied by the remainder of the path), not just immediate children.

678 If the TopicExpression ends with the characters “//.” this indicates that the TopicExpression
679 matches a Topic sub-tree. For example:

680 `“tns:t1/t3//.”`

681 This identifies the sub-tree consisting of tns:t1/t3 and all its descendents.

682 If the TopicExpression ends with the characters “//*” this indicates that the TopicExpression
683 matches all the descendents of a Topic. For example:

684 `“tns:t1/t3//*”`

685 This identifies the sub-tree consisting of the descendents of tns:t1/t3 but, unlike the previous
686 example, does not include tns:t1/t3 itself.

687 To include all the Topics in the entire Topic Namespace the following TopicExpression can be
688 used:

689 `“tns://*”`

690 The // separator can also be used in the middle of a TopicExpression, for example


```
“tns:t1//t3”
```

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

A Full TopicExpression can contain two or more wildcards (both * and //).

Full TopicExpressions can be combined together with the conjunction operator as follows:

```
“tns:t1/t2|tns:t4/t5”
```

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

8.4 XPath TopicExpression Dialect

This specification defines a fully conformant XPath 1.0 TopicExpression dialect with the following URI:

```
http://www.w3.org/TR/1999/REC-xpath-19991116
```

This dialect allows TopicExpressions that identify more than one Topic (possibly from multiple Topic Namespaces). It extends the Full TopicExpression dialect, in the sense that every expression in the Full TopicExpression dialect is also valid in the XPath TopicExpression dialect, and has the same meaning.

The XPath TopicExpression is evaluated over the NotificationProducer’s TopicSet document in the same way as the Full TopicExpression that is described section 8.3. The only difference between the two dialects is that the XPath TopicExpression permits a richer set of selection possibilities, since the full range of XPath 1.0 is available.

Any valid XPath expression is permitted, however if an expression does not return a node-set containing elements that correspond to Topics then it does not identify any Topics. For example, the following XPath expressions are valid XPath TopicExpressions, but none of them identify any Topics, so including any of these as a Filter in a Subscribe request will result in no Notifications being delivered to the NotificationConsumer:

- 123
- //@topic=true
- //@topic
- //*[@topic=false]

The first of these evaluates to a number and the second is a boolean. Neither of these are node-sets, so neither identifies any Topics. The third of these evaluates to a node-set, but it is a node-set that only contains attributes. The last one evaluates to a node-set that contains elements, but it only selects the elements that do not correspond to Topics.

8.5 Validating TopicExpressions

If the NotificationProducer permits it, a TopicExpression MAY be used as a Filter in the Subscribe message [WS-BaseNotification]. Such TopicExpressions might refer to one or more Topics which

might or might not exist in the Topic Namespace, or in the Topic Set supported by the NotificationProducer.

The NotificationProducer MUST validate the TopicExpression as follows:

If the TopicExpression explicitly refers to a Topic that is not permitted by the Topic Namespace, then the NotificationProducer MUST respond with a Fault. A Topic is not permitted if it is a root Topic that is not defined in the Topic Namespace, and that Topic Namespace has @final="true", or if it descends from a root Topic that is not defined in the Topic Namespace, and that Topic Namespace has @final="true". A Topic is also not permitted if it, or any of its ancestors, are not defined in the Topic Namespace and are the child of a Topic that is defined with @final='true'.

If the NotificationProducer has a fixed Topic Set, and the intersection of the Topics selected by the TopicExpression with this Topic Set is empty, then the NotificationProducer MUST respond with a Fault.

If the TopicExpression has a path that references a Topic Namespace that is not supported by the NotificationProducer then the NotificationProducer MAY respond with a Fault, regardless of whether the Topic Set is fixed or not

Here are some examples to illustrate these rules:

Suppose that Topic Namespace tns1 (with @final="true") contains root Topics tns1:A (@final= "true") and tns1:B (@final = "false"), and that NotificationProducer (X) has a fixed Topic Set consisting just of tns1:B.

- Any subscribe with a TopicExpression containing `tns1:D` is rejected
- Any subscribe with a TopicExpression containing `tns1:A/X` is rejected
- A subscribe to `tns1:B/X` is rejected, but would be permitted if X did not have a fixed Topic Set.
- A subscribe to `tns1:A` is rejected, but would be permitted if X did not have a fixed Topic Set.
- A subscribe to `tns1:*` is permitted (and is equivalent in this case to a subscribe to `tns1:B`)
- A subscribe to `tns1://*` is permitted (and is equivalent in this case to a subscribe to `tns1:B`)
- A subscribe to `tns1:A|tns1:B` is permitted (and is equivalent in this case to a subscribe to `tns1:B`)

9 Growing a Topic Tree

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

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

Similarly, if the TopicNamespace is not marked with the ‘final’ attribute with value=”true”, then a NotificationProducer MAY add root Topics to its Topic Set that use that Topic Namespace’s URI but which were not defined in the TopicNamespace document.

When a NotificationProducer accepts Topics that are not previously defined in the Topic Namespace, it adds them to its TopicSet document, but it is not obliged to update any actual document that contains the Topic Namespace definition. Rather, the extension exists only for that NotificationProducer and any Publisher or Subscriber that interacts with it. Circumstances under which a NotificationProducer is permitted to add new child Topics to a Topic include:

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

If a Notification Producer accepts a new Topic into its Topic Set, then messages produced on that new Topic are eligible for selection by any wild-carded subscriptions that were in effect before the Topic was added. The NotificationProducer MUST behave as if each subscription’s TopicExpression is re-evaluated against the Topic Set as each message is processed, although implementers are free to choose any approach that produces this effect.

10 The “ad-hoc” Topic Namespace

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

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

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

The ad-hoc Topic Namespace has no pre-defined root Topics, but it is not final and so it allows new root Topics to be added dynamically (in the same way that a non-final Topic allows new child Topics to be added to it). Any Topic that is added dynamically to the ad-hoc Topic Namespace itself permits the addition of further child Topics, and allows any type of Notification element to be associated with it.

The ad-hoc Topic Namespace is indicated by omitting the namespace URI, i.e. a namespace of "", and is accessed by using TopicExpressions which are unqualified.

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

11 NotificationProducers and Topics

A NotificationProducer MAY use Topics to group Notifications related to some Situation (see [WS-BaseNotification] for a definition of NotificationProducer, Notification and Situation). A NotificationProducer can support zero or more Topics, and these can come from multiple Topic Namespaces. A NotificationProducer can support an entire Topic Tree, or just a subset of the Topics in a Topic Tree.

The NotificationProducer MAY support Resource Properties [WS-ResourceProperties] that indicate the set of Topics that it expects to handle. WS-BaseNotification defines two resource properties that can be used for this purpose.

1. The NotificationProducer MAY support the wstop:TopicSet resource property, which returns the entire Topic Set as a single XML element as defined in section 7,
2. The NotificationProducer MAY support the wstop:TopicExpression resource property. This resource property returns a list of TopicExpressions covering the set of supported Topics.

The first approach has the advantage that the ResourceProperty returns the document used to evaluate Topic subscription filters that use the Full or XPATH dialects. It allows the NotificationProducer to insert producer-specific metadata that can be used in filters constructed using the XPATH dialect.

The second approach is simpler in the case where the NotificationProducer only supports Simple or Concrete Topic Expression dialects (it is merely the list of supported expressions). It could be more concise in cases where NotificationProducers support Full or XPath Topic Expression dialects since such a NotificationProducer could use a wildcarded TopicExpression to cover more than one Topic.

A NotificationProducer is free to support either, both, or neither of these ResourceProperties.

This specification defines the following global attribute which MAY be included in the value returned by a ResourceProperty query. It is RECOMMENDED that NotificationProducers include this attribute in TopicExpression ResourceProperty values.

/@wstop:TopicNamespaceLocation

The location from which a TopicNamespace document can be retrieved

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

- The NotificationProducer supporting additional Topics from a Topic Namespace that is already partially supported;
- The NotificationProducer supporting additional Topics from a Topic Namespace not previously supported;
- The NotificationProducer supporting extension Topics to a (new or already supported) Topic Namespace, as discussed in section 9;
- The NotificationProducer ceasing to support Topics previously listed.

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

12 Security Considerations

Security considerations related to the use of Topics are discussed in [WS-BaseNotification] and in [WS-BrokeredNotification]. It is recommended that implementations allow authorization policies be specified at the granularity of the Topic.

13 References

[RFC2119]

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

[WS-BaseNotification]

“Web Services Base Notification 1.3”.
http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-cs-01.pdf

[WS-BrokeredNotification]

“Web Services Brokered Notification 1.3”.
http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-cs-01.pdf

[WS-ResourceProperties]

“Web Services Resource Properties 1.2”, OASIS Standard.
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf

[XML]

“Extensible Markup Language (XML)”, W3C Recommendation.
http://www.w3.org/TR/REC-xml

[XML-Infoset]

“XML Information Set”, W3C Recommendation.
http://www.w3.org/TR/xml-infoset/

[XML-Namespaces]

“Namespaces in XML 1.1”, W3C Recommendation.
http://www.w3.org/TR/xml-names11/

[XPATH]

“XML Path Language (XPath) Version 1.0”, W3C Recommendation.
http://www.w3.org/TR/xpath

Appendix A. Acknowledgments

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 (Hewlett-Packard), William Vambenepe (Hewlett-Packard) and Kirk Wilson (Computer Associates).

Appendix B. XML Schema

The XML types and elements used in this specification 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 (2004-2006). 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:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wstop = "http://docs.oasis-open.org/wsn/t-1"
  targetNamespace = "http://docs.oasis-open.org/wsn/t-1"
  elementFormDefault="qualified"  attributeFormDefault="unqualified">

<!-- =============== utility type definitions  =================== -->
  <xsd:complexType name="Documentation" mixed="true">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="0"
               maxOccurs="unbounded" namespace="##any"/>
    </xsd:sequence>
  </xsd:complexType>

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

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

<!-- ================= Topic-Namespace Related  ================ -->
  <xsd:complexType name="TopicNamespaceType">
    <xsd:complexContent>
       <xsd:extension base="wstop:ExtensibleDocumented">
         <xsd:sequence>
           <xsd:element name="Topic"
                        minOccurs="0" maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:complexContent>
                  <xsd:extension base="wstop:TopicType">
                    <xsd:attribute name="parent"
type="wstop:ConcreteTopicExpression" />
                  </xsd:extension>
                </xsd:complexContent>
              </xsd:complexType>
           </xsd:element>
           <xsd:any namespace="##other"
                    minOccurs="0" maxOccurs="unbounded"
                    processContents="lax"/>
         </xsd:sequence>
         <xsd:attribute name="name" type="xsd:NCName"/>
```



```
        <xsd:attribute name="targetNamespace" type="xsd:anyURI"
                       use="required"/>
        <xsd:attribute name="final" type="xsd:boolean"
                                    default="false"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="TopicNamespace" type="wstop:TopicNamespaceType">
    <xsd:unique name="rootTopicUniqueness">
      <xsd:selector xpath="wstop:Topic"/>
        <xsd:field xpath="@name"/>
    </xsd:unique>
  </xsd:element>

  <xsd:attribute name="topicNamespaceLocation" type="xsd:anyURI"/>



<!-- ====================== Topic Related  ========================== -->

  <xsd:complexType name="TopicType">
    <xsd:complexContent>
      <xsd:extension base="wstop:ExtensibleDocumented">
        <xsd:sequence>
          <xsd:element name="MessagePattern"
                       type="wstop:QueryExpressionType"
                       minOccurs="0" maxOccurs="1" />
          <xsd:element name="Topic" type="wstop:TopicType"
                       minOccurs="0" maxOccurs="unbounded">
            <xsd:unique name="childTopicUniqueness">
              <xsd:selector xpath="wstop:topic"/>
              <xsd:field xpath="@name"/>
            </xsd:unique>
          </xsd:element>
          <xsd:any namespace="##other" minOccurs="0"
                                       maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" use="required" type="xsd:NCName"/>
        <xsd:attribute name="messageTypes">
          <xsd:simpleType>
            <xsd:list itemType="xsd:QName"/>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="final" type="xsd:boolean"
                                    default="false"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

<!-- ================ Topic Set Related  =================== -->

  <xsd:complexType name="TopicSetType">
```



```
    <xsd:complexContent>
       <xsd:extension base="wstop:ExtensibleDocumented">
         <xsd:sequence>
           <xsd:any namespace="##other"
                    minOccurs="0" maxOccurs="unbounded"
                    processContents="lax"/>
         </xsd:sequence>
       </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

  <xsd:element name="TopicSet" type="wstop:TopicSetType"/>
  <xsd:attribute name="topic" type="xsd:boolean" default="false"/>

<!-- ================ Topic Expression Related  ================== -->

  <xsd:simpleType name="FullTopicExpression">
    <xsd:restriction base="xsd:token">
      <xsd:annotation>
        <xsd:documentation>
        TopicPathExpression  ::=   TopicPath ( '|' TopicPath )*
        TopicPath       ::=   RootTopic ChildTopicExpression*
        RootTopic       ::=   NamespacePrefix? ('//')? (NCName | '*')
        NamespacePrefix ::=   NCName ':'
        ChildTopicExpression ::=   '/' '/'? (QName | NCName | '*'| '.')

        </xsd:documentation>
      </xsd:annotation>
      <xsd:pattern value=
         "([\i-[:]][\c-[:]]*:)?(//)?([\i-[:]][\c-[:]]*|\*)((/|//)(([\i-
[:]][\c-[:]]*:)?[\i-[:]][\c-[:]]*|\*|[.]))*(\|([\i-[:]][\c-
[:]]*:)?(//)?([\i-[:]][\c-[:]]*|\*)((/|//)(([\i-[:]][\c-[:]]*:)?[\i-
[:]][\c-[:]]*|\*|[.]))*)*">
      </xsd:pattern>
    </xsd:restriction>
  </xsd:simpleType>

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

        </xsd:documentation>
      </xsd:annotation>
      <xsd:pattern value=
"(([\i-[:]][\c-[:]]*:)?[\i-[:]][\c-[:]]*)(/([\i-[:]][\c-[:]]*:)?[\i-
[:]][\c-[:]]*)*" >
      </xsd:pattern>
    </xsd:restriction>
```



```
  </xsd:simpleType>

  <xsd:simpleType name="SimpleTopicExpression">
    <xsd:restriction base="xsd:QName">
      <xsd:annotation>
        <xsd:documentation>
  The pattern allows strings matching the following EBNF:
    RootTopic            ::=   QName

        </xsd:documentation>
      </xsd:annotation>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```


Appendix C. Revision History

Rev	Date	By Whom	What
wd-01	2004-06-04	William Vambenepe	Initial version created from submission by contributing companies. Minor modifications made to reflect OASIS formatting and namespace URI choices.
b	2005-06-27	Sid Askary	- Added the Section on security - Added the section on faults - Added the concepts from white paper -Corrected typos -Removed references to White Paper - NotificationMessage w/ Notification - Updated status section - Replaced Notional Conventions TODO: - AI 85 - Rewrite of Chapter 5. - Incorporate new Namespace in Schema
c	2005-07-06	Peter Niblett	Updated to use new Namespaces Removed aliases (WSN 4.5) TopicSpace changed to Topic Namespace (WSN 4.2) Added section describing Topic Set document and made corresponding adjustments to the schema and to the definition of FullTopicSet (WSN 4.2) Added an XPath 1.0 Topic Expression Dialect (WSN 4.3) Use wsnt:QueryExpressionType instead of wsrf-rp:QueryExpressionType (WSN

Rev	Date	By Whom	What
			4.26) Updated the references New acknowledgements section Changed SimpleTopicExpression to be xsd:QName instead of xsd:token with a pattern (WSN 4.20) Removed the “special” @messageTypes value of xsd:any, and removed the default value for this attribute from the XML Schema (WSN 4.27) Added “final” attribute to TopicNamespace (WSN 4.22) Renamed the adhoc namespace to “” (WSN 4.9) Added sentence on wildcard resolution with growing topic sets (WSN 4.16) Added global TopicNamespaceLocation attribute (WSN4.21)
d	2005-09-26	Peter Niblett	Corrections to some of the amendments in c, following issue resolution review Term Topic Path changed to become Topic Expression (AI 85)
e	2005-11-24	Peter Niblett	Domain-specific extensions to TopicNamespaces (WSN 4.4) Updated references to and namespace URIs for other WSN specifications (AI 138) Removed reference to WSDL 2.0 (AI 136) Removed section 1.4 (Fault Definitions) as it is not relevant to this specification Replaced section 12 (Security Considerations) with pointers to [WS BaseNotification] and [WS BrokeredNotification], since the material contained was duplicative and not all relevant to this specification Added discussion of TopicSet and

Rev	Date	By Whom	What
			TopicExpression RPs (WSN 4.28) Miscellaneous other corrections (WSN 4.28) Discussion of Namespace prefix binding in TopicExpressions (WSN 4.23 and WSN 4.24) Added description of TopicNamespaceLocation attribute (WSN 4.21) Widened scope of 8.5 to cover all TopicExpressions, not just Full and XPath,
f	2005-12-03	Peter Niblett	Revised the resolution of issue 4.26 to avoid circular dependency of schemas (QueryExpressionType is now defined in this schema).
g	2005-12-06	Peter Niblett	Corrected the namespace and description of TopicSpaceLocation attribute (WSN 4.21) Corrected schemaLocations in the TopicNamespace and TopicSet examples (AI 138) Reworded the definition of wstop:Topic/@parent, and reworded bullet 3 of 6.1 (WSN 4.4) Revised words at the start of section 7, to make them clearer (WSN 4.2)
wd-02a	2006-03-31	Peter Niblett	Miscellaneous errata
wd-02b	2006-05-22	Peter Niblett	WSN 4.29. Specified that an Extension Topic (or child of an Extension Topic) can only be referenced by using path expressions that include the parent of the Extension Topic. If the dialect permits them, wild card characters can be used so that the Parent Topic name does not need to be included explicitly.

1124

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 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.