Web Services Topics (WS-Topics)

Version 1.0

3/5/2004

Authors

Steve Graham, IBM (editor)
Peter Niblett, IBM (editor)
Dave Chappell, Sonic Software
Amy Lewis, TIBCO Software
Nataraj Nagaratnam, IBM
Jay Parikh, Akamai Technologies
Sanjay Patil, SAP AG
Shivajee Samdarshi, TIBCO Software
Igor Sedukhin, Computer Associates International
David Snelling, Fujitsu Laboratories of Europe
Steve Tuecke, Globus / Argonne National Laboratory
William Vambenepe, Hewlett-Packard
Bill Weihl, Akamai Technologies

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 white papers and 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: a white paper: *Publish-Subscribe Notification for Web services* as well as 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-Base Notification*. 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 and the *Publish-Subscribe Notification for Web Services* document.

Status

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

Table of Contents

1 Introduction

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

This 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 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 and the *Publish-Subscribe Notification for Web Services* white paper.

1.1 Notational Conventions

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

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

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

1.2 Namespaces

The following namespaces are used in this document:

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

2 Terminology and Concepts

Please refer to [WS-Notification Whitepaper] for a list of terms and their definitions.

3 Topics and Topic Spaces

A collection of related Topics is used to organize and categorize a set of notification messages. It 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 NotificationMessage, the Publisher associates it with one or more Topics. When a Subscriber creates a Subscription, it associates the Subscription with one or more Topics. The NotificationProducer uses these Topic lists as part of the matching process: a NotificationMessage is delivered to a NotificationConsumer if the list of Topics associated with the Subscription has a non-empty intersection with the list of Topics associated with the NotificationMessage.

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

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

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

The rationale for hierarchical topic structures is:

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

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

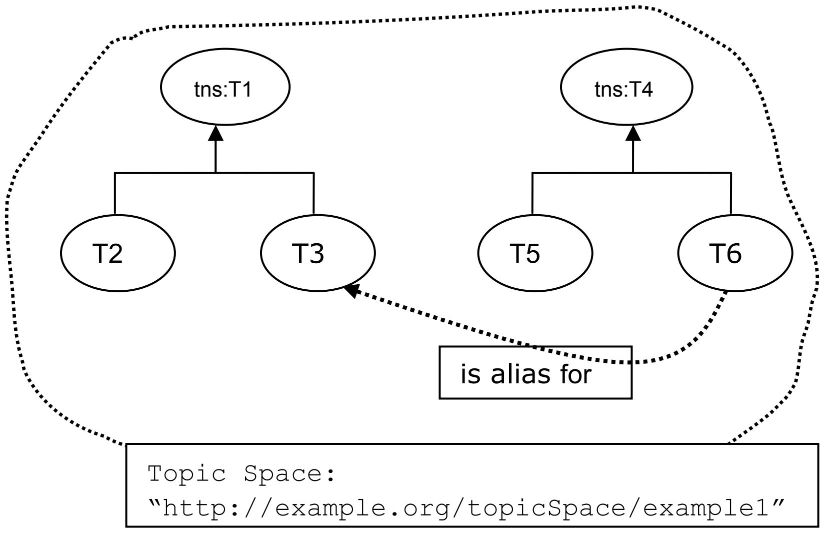
A Topic Space is thus a collection (forest) of Topic Trees. The Topic Space contains additional metadata relating to its member Topics. The metadata describing a particular Topic Space can be modeled as an XML document (see section 5).

Each Topic has a local name, an NCName. All root topics must have unique names within their Topic Space. In this way, a root Topic can be uniquely referenced by a QName formed by combining the XML Namespace associated with the Topic Space and the local name of the root topic. Child topics can only be referred to relative to their ancestor root topic's QName using a path-based TopicExpression dialect (see section 7).

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

4 Example

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



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

```
<?xml version="1.0" encoding="UTF-8"?>
<wstop:topicSpace name="TopicSpaceExample1"
   targetNamespace="http://example.org/topicSpace/example1"
   xmlns:tns="http://example.org/topicSpace/example1"
   xmlns:xyz="http://example.org/anotherNamespace"
   xmlns:wstop=
      "http://www.ibm.com/xmlns/stdwip/web-services/WS-Topics" >
   <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:AliasRef
  dialect="http://www.ibm.com/xmlns/stdwip/web-services/WS-
Topics/TopicExpression/concreteTopicPath" >
            tns:t1/t3
         </wstop:AliasRef>
      </wstop:topic>
   </wstop:topic>
</wstop:topicSpace>
```

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

5 Modeling Topic Spaces in XML

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

```
...

<TopicSpace name=NCName? targetNamespace=anyURI …>

  <Topic … />*

…

</TopicSpace>
```

A TopicSpace element is further constrained in the following way:

/wstop:TopicSpace

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

/wstop:TopicSpace/@name

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

/wstop:TopicSpace/@targetNameSpace

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

/wstop:topicSpace/Topic

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

/wstop:TopicSpace/{any}

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

/wstop:TopicSpace/@{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 in the following non-normative fashion::

```
<TopicSpace name=… targetNamespace=…>
  <Topic name=NCName messageTypes=list of QName? final=boolean?>
  (
      <AliasRef>wsnt:TopicExpression</AliasRef>
  |
      <MessagePattern>wsrp:QueryExpression</MessagePattern>?
      <Topic … />*
…
  )
  </Topic>
</TopicSpace>
```

A Topic element is further constrained in the following way:

/wstop:Topic

This describes the definition of a Topic. Its contents MUST be either a single /AliasRef child element or an optional /MessagePattern child element followed by zero or more child Topic elements.

/wstop:Topic/@namespace

The namespace of a Topic is defined as the targetNamespace of the TopicSpace element ancestor of the Topic. As we saw in section 5, individual root topics are modeled by defining Topic child elements of the TopicSpace 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 (TopicSpace or Topic) that contains this Topic. In the case of a root Topic, the @namespace and @name attributes combine to form the QName of the root Topic.

/wstop:Topic/@messageTypes

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

/wstop:Topic/@final

An optional attribute whose value is of type xsd:boolean. The default value is "false". If the value is "true" it indicates that the NotificationProducer cannot dynamically add child Topics to this Topic. This means that it is an error if a Publisher or Subscriber attempts to use a 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/AliasRef

This element indicates that the Topic definition is an alias for another Topic (or set of Topics). This mechanism can be used to permit alternative spellings of a given Topic name, or to allow a Topic (sub)tree from one Topic Space to be imported into a Topic definition in another Topic Space. The contents of an AliasRef element is a TopicExpression that may resolve to multiple Topics, including further aliases (even possibly itself). Publishing or subscribing using a Topic which is an alias is equivalent to publishing or subscribing to all the non-alias Topics which result from the process of alias resolution. This process is described in greater detail in the next section. A Topic containing an AliasRef child element MAY contain @messageTypes, or @final– however if it does their values SHOULD be ignored. The algorithm for resolving AliasRef elements is described in section 8.

/wstop:Topic/AliasRef/@dialect

A URI that identifies the TopicExpression dialects used in the AliasRef component. This document defines the URIs for three TopicExpression languages. The designer MAY choose from these URIs or use a URI associated with a TopicExpression dialect defined elsewhere.

/wstop:Topic/MessagePattern

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

/wstop:Topic/MessagePattern/@dialect

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

/wstop:Topic/Topic

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

/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.

7 Topic Expressions

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

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

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

```
<wsnt:TopicExpression dialect=anyURI?>
   dialect-specific expression
…
</wsnt:TopicExpression>
```

A topic expression 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 relevant set of Topics from one or more Topic Spaces.

7.1 SimpleTopic Expressions

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

```
http://www.ibm.com/xmlns/stdwip/web-services/WS-Topics/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 Spaces.

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 is included in the RootTopic, it must correspond to a
    valid Topic Space 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 spaces, those that define only root topics, SHOULD be used.

An example TopicExpression within this dialect is shown below:

```
…
   xmlns:tns=…

<wsnt:TopicExpression
    dialect="http://www.ibm.com/xmlns/stdwip/web-services/WS-
Topics/TopicExpression/simple">
          tns:t1
</wsnt:TopicExpression>
```

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

7.2 ConcreteTopicPath Expressions

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

```
http://www.ibm.com/xmlns/stdwip/web-services/WS-
Topics/TopicExpression/concreteTopicPath
```

The ConcreteTopicPath expression is used to identify a single Topic within a Topic Space, using a path notation.

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

Note: White space is not permitted within a ConcreteTopicPath expression.

An example TopicExpression within this dialect is shown below:

```
…
   xmlns:tns=…

<wsnt:TopicExpression
    dialect="http://www.ibm.com/xmlns/stdwip/web-services/WS-
Topics/TopicExpression/concreteTopicPath">
          tns:t1/t3
</wsnt:TopicExpression>
```

The TopicPath expression identifies the Topic named "t3", child of Topic tns:t1.

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

Note: The simple Topic Expression dialect defined in the previous section is a subset of the ConcreteTopicPath Expression dialect.

7.3 FullTopicPath Expressions

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

```
http://www.ibm.com/xmlns/stdwip/web-services/WS-
Topics/TopicExpression/FullTopicPath
```

FullTopicPath expressions consist are made up of XPath [XPath] relative location path expressions with optional Namespace prefixes. The XPath expression is evaluated over a document whose nodes are made up of the topics in the topic space, and where topics include their child topics as contained XML elements (note that this document is not the same as the topic space document described earlier, but can be derived from it). The document root element is not itself a topic, so that root topics in the topic space appear as first-level children of the document root. The TopicExpression selects the set of topics that correspond to the node-set that results from evalutaing the location path contained in the TopicExpression, using standard XPath. The initial context node for this evaluation is the document root element.

The FullTopicPath dialect does not permit the use of the entire XPath language. This specification provides syntactic constraints on the contents of the FullTopicPathExpression, that limits 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 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 is included in the RootTopic, it must correspond to a
    valid Topic Space Document and the local name must correspond to the name
    of a root topic defined in that namespace.]
[6] NamespacePrefix         ::=    NCName ‘:’
[7] ChildTopicExpression ::=    ‘/’ ‘/’? (ChildTopicName | ‘*’ | ‘.’ )
[8] ChildTopicName          ::=    NCName
    [ vc: The NCName must correspond to the name of a topic within the
    descendant path from the RootTopic, where each forward slash denotes
    another level of child topic elements in the path.]
```

In this grammar, each TopicPath [4] is to be interpreted as an XPath location path evaluated over the document derived from the Topic Space designated by the NamespacePrefix.

Note: White space is not permitted within a FullTopicPath expression.

Note: The ConcreteTopicPath dialect defined in the previous section is a subset of the FullTopicPath Expression that contains no wildcards and no ‘|’ operators.

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

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

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

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

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

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

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

The wildcard * may also be used in place of a root topic name, for example:

```
"tns:*"
```

This TopicExpression identifies all root topics in the tns: Topic Space.

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

If the TopicExpression ends with the characters "//." this indicates that the TopicExpression matches a Topic subtree. For example:

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

This identifies the subtree consisting of tns:t1/t3 and all its descendents.

If the TopicExpression ends with the characters "//*" this indicates that the TopicExpression matches all the descendents of a topic. For example:

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

This identifies the subtree consisting of the descendents of tns:t1/t3 but, unlike the previous example, does not include tns:t1/t3 itself.

To include all the topics in the entire Topic Space the following TopicExpression can be used:

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

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

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

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

A TopicExpression using | can include root Topics from different Topic Spaces. Note: a 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.

7.3.1 Validating FullTopicExpressions

If the NotificationProducer permits it, the FullTopicExpression dialect can be used as the TopicExpression in the Subscribe message [WS-BaseNotification]. Such TopicExpressions MAY refer to one or more topics which may or may not exist in the Topic Space, or in the Topic Set supported by the NotificationProducer.

The NotificationProducer MUST validate the TopicExpression as follows.

1. If the TopicExpression explicitly refers to a Topic that is not permitted by the Topic Space, then the NotificationProducer MUST respond with a Fault. A Topic is not permitted if it is a root topic, or a descendent of a root topic, that is not defined in the Topic Space. A Topic is also not permitted if it, or any of its ancestors, are not defined in the Topic Space and are the child of a Topic that is defined with @final='true'.
2. 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.

Here are some examples to illustrate these rules:

Suppose that Topic Space tns1 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)

8 AliasRefs and their resolution

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

```
      <wstop:topic name="t6">
         <wstop:AliasRef
  dialect="http://www.ibm.com/xmlns/stdwip/web-services/WS-
Topics/TopicExpression/concreteTopicPath" >
            tns:t1/t3
         </wstop:AliasRef>
      </wstop:topic>
```

An AliasRef MAY contain any TopicExpression, including those expressing wild cards or '|' operators. This means that an AliasRef might reference another AliasRef Topic definition, or might be a wild card expression that includes a mixture of alias and non-alias definitions.

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

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

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

9 Growing a Topic Tree

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

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

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

- a Subscriber attempting to subscribe to a TopicExpression that suggests a new child Topic;
- a Publisher attempting to publish to 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.

10 The "ad-hoc" Topic Space

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

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

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

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

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

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

11 NotificationProducer and Topics

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

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

- The NotificationProducer supporting additional Topics from a TopicSpace that is already partially supported;

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

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

12 Security Considerations

A non-normative discussion of the security scenarios and considerations associated with the entire family of WS-Notification specifications is contained in [WS-Notification Whitepaper].

13 Acknowledgements

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

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

14 References

[WS-Base Notification]

ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf

[WS-Notification Whitepaper]

http://www-106.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf

[WS-Security]

http://www.oasis-open.org/committees/download.php/5531/oasis-200401-wss-soap-message-security-1.0.pdf

[XML-Infoset]

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

[XML]

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

[XPATH]

http://www.w3.org/TR/xpath

Appendix I – XML Schema

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

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

   Copyright Notice

   (c) Copyright Akamai Technologies,
       Computer Associates International, Inc., Fujitsu Limited,
       Hewlett-Packard Development Company,
       International Business Machines Corporation, SAP AG,
       Sonic Software Corporation, Tibco Software Inc. and
       The University of Chicago  2003, 2004  All rights reserved.

-->

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

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

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


<!-- =============== utility type definitions  ==================== -->
  <xsd:complexType name="Documentation" mixed="true">
    <xsd:sequence>
```

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

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

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

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

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

  <xsd:group name="NonAliasTopicDefinition">
    <xsd:sequence>
      <xsd:element name="MessagePattern"
                   type="wsrp: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:sequence>
  </xsd:group>

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

```
    <xsd:complexContent>
      <xsd:extension base="wstop:ExtensibleDocumented">
        <xsd:sequence>
          <xsd:choice>
            <xsd:element name="AliasRef"
                         type="wsnt:TopicExpressionType"
                         minOccurs="1" maxOccurs="1" />
            <xsd:group ref="wstop:NonAliasTopicDefinition" />
          </xsd:choice>
          <xsd:any namespace="##other" minOccurs="0"
                                       maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" use="required" type="xsd:NCName"/>
        <xsd:attribute name="messageTypes" default="xsd:any">
          <xsd:simpleType>
            <xsd:list itemType="xsd:QName"/>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="final" type="xsd:boolean"
                                     default="false"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

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

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

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

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

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

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

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

</xsd:schema>
```