Web Services Eventing (WS-Eventing)

August 2004

Authors

Don Box, Microsoft
Luis Felipe Cabrera, Microsoft
Craig Critchley, Microsoft
Francisco Curbera, IBM
Donald Ferguson, IBM
Alan Geller (Editor), Microsoft
Steve Graham, IBM
David Hull, TIBCO Software
Gopal Kakivaya, Microsoft
Amelia Lewis, TIBCO Software
Brad Lovering, Microsoft
Matt Mihic, BEA Systems
Peter Niblett, IBM
David Orchard, BEA Systems
Junaid Saiyed, Sun Microsystems
Shivajee Samdarshi, TIBCO Software
Jeffrey Schlimmer, Microsoft
Igor Sedukhin, Computer Associates
John Shewchuk, Microsoft
Bill Smith, Sun Microsystems
Sanjiva Weerawarana, IBM
David Wortendyke, Microsoft

Copyright Notice

Abstract

This specification describes a protocol that allows Web services to subscribe to or accept subscriptions for event notification messages.

Composable Architecture

By using the XML, SOAP [SOAP 1.1, SOAP 1.2], and WSDL [WSDL 1.1] extensibility models, the Web service specifications (WS-*) are designed to be composed with each other to provide a rich set of tools to provide security in the Web services environment. This specification specifically relies on other Web service specifications to provide secure, reliable, and/or transacted message delivery and to express Web service and client policy.

Status

This specification is a public draft release and is provided for review and evaluation only. The authors hope to solicit your contributions and suggestions in the near future. The authors make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

Web services often want to receive messages when events occur in other services and applications. A mechanism for registering interest is needed because the set of Web services interested in receiving such messages is often unknown in advance or will change over time. This specification defines a protocol for one Web service (called a "subscriber") to register interest (called a "subscription") with another Web service (called an "event source") in receiving messages about events (called "notifications" or "event messages"). The subscriber may manage the subscription by interacting with a Web service (called the "subscription manager") designated by the event source.

To improve robustness, a subscription may be leased by an event source to a subscriber, and the subscription expires over time. The subscription manager provides the ability for the subscriber to renew or cancel the subscription before it expires.

There are many mechanisms by which event sources may deliver events to event sinks. This specification provides an extensible way for subscribers to identify the delivery mechanism they prefer. While asynchronous, pushed delivery is defined here, the intent is that there should be no limitation or restriction on the delivery mechanisms capable of being supported by this specification.

1.1 Requirements

This specification intends to meet the following requirements:

- Define means to create and delete event subscriptions.

- Define expiration for subscriptions and allow them to be renewed.
- Define how one Web service can subscribe on behalf of another.
- Define how an event source delegates subscription management to another Web service.
- Allow subscribers to specify how event messages should be delivered.
- Leverage other Web service specifications for secure, reliable, transacted message delivery.
- Support complex eventing topologies that allow the originating event source and the final event sink to be decoupled.
- Provide extensibility for more sophisticated and/or currently unanticipated subscription scenarios.
- Support a variety of encoding formats, including (but not limited to) both SOAP 1.1 [SOAP 1.1] and SOAP 1.2 [SOAP 1.2] Envelopes.

1.2 Delivery Modes

While the general pattern of asynchronous, event-based messages is extremely common, different applications often require different event message delivery mechanisms. For instance, in some cases a simple asynchronous message is optimal, while other situations may work better if the event consumer can poll for event messages in order to control the flow and timing of message arrival. Some consumers will require event messages to be wrapped in a standard "event" SOAP envelope, while others will prefer messages to be delivered unwrapped. Some consumers may require event messages to be delivered reliably, while others may be willing to accept best-effort event delivery.

In order to support this broad variety of event delivery requirements, this specification introduces an abstraction called a Delivery Mode. This concept is used as an extension point, so that event sources and event consumers may freely create new delivery mechanisms that are tailored to their specific requirements. This specification provides a minimal amount of support for delivery mode negotiation by allowing an event source to provide a list of supported delivery modes in response to a subscription request specifying a delivery mode it does not support.

This specification defines a single delivery mode, Push Mode, which is simple asynchronous messaging.

As an example of a possible extension, a feature may allow a subscriber to request that event messages be "wrapped" in a standard message. This feature is requested by specifying a new delivery mode, e.g. http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Wrap, in the Subscribe request. Use of this delivery mode would indicate that notification messages should be "wrapped". Similarly, this approach could be generalized to allow the subscriber to provide other information regarding their preferences for notification message delivery.

1.3 Subscription Managers

In some scenarios the event source itself manages the subscriptions it has created. In other scenarios, for example a geographically distributed publish-and-subscribe system, it may be useful to delegate the management of a subscription to another Web service. To support this flexibility, the response to a subscription request to an event source will

include the EPR of a service that the subscriber may interact with to manage this subscription. This EPR should be the target for future requests to renew or cancel the subscription. It may address the same Web service (Address and ReferenceProperties) as the event source itself, or it may address some other Web service to which the event source has delegated management of this subscription; however, the full subscription manager EPR (Address and Reference Properties and ReferenceParameters) must be unique for each subscription.

We use the term "subscription manager" in this specification to refer to the Web service that manages the subscription, whether it is the event source itself or some separate Web service.

1.4 Example

Table 1 lists a hypothetical request to create a subscription for storm warnings.

Table 1: Hypothetical request to create a subscription

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ew='http://www.example.com/warnings' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
(09)     </wsa:Action>
(10)     <wsa:MessageID>
(11)         uuid:d7c5726b-de29-4313-b4d4-b3425b200839
(12)     </wsa:MessageID>
(13)     <wsa:ReplyTo>
(14)       <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
(15)     </wsa:ReplyTo>
(16)     <wsa:To>http://www.example.org/oceanwatch/EventSource</wsa:To>
(17)   </s12:Header>
(18)   <s12:Body>
(19)     <wse:Subscribe>
(20)       <wse:Delivery>
(21)         <wse:NotifyTo>
(22)           <wsa:Address>
(23)               http://www.example.com/MyEventSink/OnStormWarning
(24)           </wsa:Address>
(25)           <wsa:ReferenceProperties>
(26)             <ew:MySubscription>2597</ew:MySubscription>
(27)           </wsa:ReferenceProperties>
(28)         </wse:NotifyTo>
(29)       </wse:Delivery>
(30)     </wse:Subscribe>
```

```
(31)    </s12:Body>
(32) </s12:Envelope>
```

Lines (07-09) in Table 1 indicate the message is a request to create a subscription, and Line (16) indicates that it is sent to a hypothetical event source of ocean events.

While Lines (13-15) indicate where a reply should be sent, Lines (20-29) indicate where and how notifications should be delivered; there is no requirement that these match. The absence of a Mode attribute on Line (20) indicates that notifications should be delivered using Push mode; that is, they should be asynchronously sent as SOAP messages to the endpoint described in lines (21-28). Note that Lines (25-27) illustrate a typical pattern where the event sink lists a reference property (Line 26) that identifies the subscription and will be included in each notification.

Table 2 lists a hypothetical response to the request in Table 1.

Table 2: Hypothetical response to a subscribe request

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)      http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse
(09)     </wsa:Action>
(10)     <wsa:RelatesTo>
(11)         uuid:d7c5726b-de29-4313-b4d4-b3425b200839
(12)     </wsa:RelatesTo>
(13)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(14)   </s12:Header>
(15)   <s12:Body>
(16)     <wse:SubscribeResponse>
(17)       <wse:SubscriptionManager>
(18)         <wsa:Address>
(19)           http://www.example.org/oceanwatch/SubscriptionManager
(20)         </wsa:Address>
(21)         <wsa:ReferenceParameters>
(22)           <ow:MyId>
(23)             28
(24)           </ow:MyId>
(25)         </wsa:ReferenceParameters>
(26)       </wse:SubscriptionManager>
(27)       <wse:Expires>P0Y0M0DT30H0M0S</wse:Expires>
(28)     </wse:SubscribeResponse>
(29)   </s12:Body>
(30) </s12:Envelope>
```

Lines (07-09) in Table 2 indicate this message is a response to a request to create a subscription, and Lines (10-12) indicate that it is a response to the request in Table 1. Lines (17-26) provide the subscription manager EPR for this subscription, and Line (27) indicates the subscription will expire in 30 hours unless renewed.

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 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 [RFC 2119].

This specification uses the following syntax to define normative outlines for messages:

- The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- Characters are appended to elements and attributes to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.
- An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. If a receiver does not recognize an extension, the receiver SHOULD NOT process the message and MAY fault.
- XML namespace prefixes (see Table 3) are used to indicate the namespace of the element being defined.

2.2 XML Namespaces

The XML namespace URI that MUST be used by implementations of this specification is:

```
http://schemas.xmlsoap.org/ws/2004/08/eventing
```

Table 3 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 3: Prefixes and XML namespaces used in this specification.

Prefix	XML Namespace	Specification(s)
s	(Either SOAP 1.1 or 1.2)	(Either SOAP 1.1 or 1.2)
s11	http://schemas.xmlsoap.org/soap/envelope/	SOAP 1.1 [SOAP 1.1]
s12	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2 [SOAP 1.2]
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL [WSDL 1.1]
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	WS-Addressing [WS-

		Addressing]
wse	http://schemas.xmlsoap.org/ws/2004/08/eventing	This specification
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]

2.3 Terminology

Delivery Mode
: The mechanism by which event messages are delivered from the source to the sink.

Event Source
: A Web service that sends notifications and accepts requests to create subscriptions.

Event Sink
: A Web service that receives notifications.

Notification
: A one-way message sent to indicate that an event has occurred.

Push Mode
: A delivery mechanism where the source sends event messages to the sink as individual, unsolicited, asynchronous SOAP messages.

Subscriber
: A Web service that sends requests to create, renew, and/or delete subscriptions.

Subscription Manager
: A Web service that accepts requests to manage get the status of, renew, and/or delete subscriptions on behalf of an event source.

2.4 Compliance

An implementation is not compliant with this specification if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace identifier for this specification (listed in Section 2.2) within SOAP Envelopes unless it is compliant with this specification.

SOAP is not a requirement for using the constructs defined in this specification.

Normative text within this specification takes precedence over normative outlines, which in turn takes precedence over the XML Schema and WSDL descriptions.

3. Subscription Messages

To create, renew, and delete subscriptions, subscribers send request messages to event sources and subscription managers.

When an event source accepts a request to create a subscription, it typically does so for a given amount of time, although an event source may accept an indefinite subscription with no time-based expiration. If the subscription manager accepts a renewal request, it updates that amount of time. During that time, notifications are delivered by the event source to the requested event sink. An event source may support filtering to limit notifications that are delivered to the event sink; if it does, and a subscribe request contains a filter, the event source sends only notifications that match the requested filter. The event source sends notifications until one of the following happens: the subscription manager accepts an unsubscribe request for the subscription; the subscription expires without being renewed; or the event source cancels the subscription prematurely. In this last case, the event source makes a best effort to indicate why the subscription ended.

In the absence of reliable messaging at the application layer (e.g. [WS-ReliableMessaging]), messages defined herein are delivered using the quality of service of the underlying transport(s) and on a best-effort basis at the application layer.

3.1 Subscribe

To create a subscription, a subscriber sends a request message of the following form to an event source:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
    </wsa:Action>
    ...
  </s:Header>
  <s:Body ...>
    <wse:Subscribe ...>
      <wse:EndTo>endpoint-reference</wse:EndTo> ?
      <wse:Delivery Mode="xs:anyURI"? >xs:any</wse:Delivery>
      <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires> ?
      <wse:Filter Dialect="xs:anyURI"? > xs:any </wse:Filter> ?
      ...
    </wse:Subscribe>
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
: If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Body/*/wse:EndTo
: Where to send a SubscriptionEnd message if the subscription is terminated unexpectedly. (See 3.4 Subscription End.) If present, this element MUST be of type wsa:EndpointReferenceType. Default is not to send this message.

/s:Envelope/s:Body/*/wse:Delivery
: A delivery destination for notification messages, using some delivery mode. See section 1.2 Delivery Modes for details.

/s:Envelope/s:Body/*/wse:Delivery/@Mode
: The delivery mode to be used for notification messages sent in relation to this subscription. Implied value is 'http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push', which indicates that Push Mode delivery should be used. See section 1.2 Delivery Modes for details.

  If the event source does not support the requested delivery mode, the request MUST fail, and the event source MAY generate a wse:DeliveryModeRequestedUnavailable fault indicating that the requested delivery mode is not supported.

/s:Envelope/s:Body/*/wse:Delivery/@Mode='http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push'

Value of /s:Envelope/s:Body/*/wse:Delivery is a single element, wse:NotifyTo, that contains the endpoint reference to which notification messages should be sent.

/s:Envelope/s:Body/*/wse:Expires

Requested expiration time for the subscription. (No implied value.) The event source defines the actual expiration and is not constrained to use a time less or greater than the requested expiration. The expiration time may be a specific time or a duration from the subscription's creation time. Both specific times and durations are interpreted based on the event source's clock.

If this element does not appear, then the request is for a subscription that will not expire. That is, the subscriber is requesting the event source to create a subscription with an indefinite lifetime. If the event source grants such a subscription, it may be terminated by the subscriber using an Unsubscribe request, or it may be terminated by the event source at any time for reasons such as connection termination, resource constraints, or system shut-down.

If the expiration time is either a zero duration or a specific time that occurs in the past according to the event source, then the request MUST fail, and the event source MAY generate a wse:InvalidExpirationTime fault indicating that an invalid expiration time was requested.

Some event sources may not have a "wall time" clock available, and so are only able to accept durations as expirations. If such a source receives a Subscribe request containing a specific time expiration, then the request MAY fail; if so, the event source MAY generate a wse:UnsupportedExpirationType fault indicating that an unsupported expiration type was requested.

/s:Envelope/s:Body/*/wse:Filter

A Boolean expression in some dialect, either as a string or as an XML fragment (see /s:Envelope/s:Body/*/wse:Filter/@Dialect). If the expression evaluates to false for a notification, the notification MUST NOT be sent to the event sink. Implied value is an expression that always returns true. If the event source does not support filtering, then a request that specifies a filter MUST fail, and the event source MAY generate a wse:FilteringNotSupported fault indicating that filtering is not supported.

If the event source supports filtering but cannot honor the requested filtering, the request MUST fail, and the event source MAY generate a wse:FilteringRequestedUnavailable fault indicating that the requested filter dialect is not supported.

/s:Envelope/s:Body/*/wse:Filter/@Dialect

Implied value is 'http://www.w3.org/TR/1999/REC-xpath-19991116'.

While an XPath predicate expression provides great flexibility and power, alternate filter dialects may be defined. For instance, a simpler, less powerful dialect might be defined for resource-constrained implementations, or a new dialect might be defined to support filtering based on data not included in the notification message itself. If desired, a filter dialect could allow the definition of a composite filter that contained multiple filters from other dialects.

/s:Envelope/s:Body/*/wse:Filter/@Dialect=' http://www.w3.org/TR/1999/REC-xpath-19991116'

Value of /s:Envelope/s:Body/*/wse:Filter is an XPath [XPath 1.0] predicate expression (PredicateExpr); the context of the expression is:

- Context Node: the SOAP Envelope containing the notification.
- Context Position: 1.
- Context Size: 1.
- Variable Bindings: None.
- Function Libraries: Core Function Library [XPath 1.0].
- Namespace Declarations: The [in-scope namespaces] property [XML Infoset] of /s:Envelope/s:Body/*/wse:Filter.

Other message information headers defined by WS-Addressing [WS-Addressing] MAY be included in the request and response messages, according to the usage and semantics defined in WS-Addressing.

Other components of the outline above are not further constrained by this specification.

If the event source accepts a request to create a subscription, it MUST reply with a response of the following form:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body ...>
    <wse:SubscribeResponse ...>
      <wse:SubscriptionManager>
        wsa:EndpointReferenceType
      </wse:SubscriptionManager>
      <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires>
      ...
    </wse:SubscribeResponse>
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

/s:Envelope/S:Header/wsa:RelatesTo

MUST be the value of the wsa:MessageID of the corresponding request.

/s:Envelope/s:Body/*/wse:SubscriptionManager

The EPR of the subscription manager for this subscription.

In some cases, it is convenient for all EPRs issued by a single event source to address a single Web service and use a reference parameter to distinguish among the active subscriptions. For convenience in this common situation, this specification defines a global element, wsa:Identifier of type xs:anyURI, that MAY be used as a distinguishing reference parameter if desired by the event source.

/s:Envelope/s:Body/*/wse:Expires

The expiration time assigned by the event source. The expiration time MAY be either an absolute time or a duration but SHOULD be of the same type as the requested expiration (if any).

If this element does not appear, then the subscription will not expire. That is, the subscription has an indefinite lifetime. It may be terminated by the subscriber using an Unsubscribe request, or it may be terminated by the event source at any time for reasons such as connection termination, resource constraints, or system shut-down.

Other components of the outline above are not further constrained by this specification.

If the event source chooses not to accept a subscription, the request MUST fail, and the event source MAY generate a wse:EventSourceUnableToProcess fault indicating that the request was not accepted.

Table 4 lists another hypothetical request to create a subscription.

Table 4: Second hypothetical request to create a subscription

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ew='http://www.example.com/warnings' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
(09)     </wsa:Action>
(10)     <wsa:MessageID>
(11)         uuid:e1886c5c-5e86-48d1-8c77-fc1c28d47180
(12)     </wsa:MessageID>
(13)     <wsa:ReplyTo>
(14)       <wsa:Address>http://www.example.com/MyEvEntsink</wsa:Address>
(15)       <wsa:ReferenceProperties>
(16)         <ew:MySubscription>2597</ew:MySubscription>
(17)       </wsa:ReferenceProperties>
(18)     </wsa:ReplyTo>
(19)     <wsa:To>http://www.example.org/oceanwatch/EventSource</wsa:To>
(20)   </s12:Header>
(21)   <s12:Body>
(22)     <wse:Subscribe>
(23)       <wse:EndTo>
(24)         <wsa:Address>
(25)           http://www.example.com/MyEventSink
(26)         </wsa:Address>
(27)         <wsa:ReferenceProperties>
(28)           <ew:MySubscription>2597</ew:MySubscription>
(29)         </wsa:ReferenceProperties>
(30)       </wse:EndTo>
```

```
(31)       <wse:Delivery>
(32)         <wse:NotifyTo>
(33)           <wsa:Address>
(34)               http://www.other.example.com/OnStormWarning
(35)           </wsa:Address>
(36)           <wsa:ReferenceProperties>
(37)             <ew:MySubscription>2597</ew:MySubscription>
(38)           </wsa:ReferenceProperties>
(39)         </wse:NotifyTo>
(40)       </wse:Delivery>
(41)       <wse:Expires>2004-06-26T21:07:00.000-08:00</wse:Expires>
(42)       <wse:Filter xmlns:ow='http://www.example.org/oceanwatch'
(43)           Dialect='http://www.example.org/topicFilter' >
(44)         weather.storms
(45)       </wse:Filter>
(46)     </wse:Subscribe>
(47)   </s12:Body>
(48) </s12:Envelope>
```

Like the request in Table 1, Lines (07-09) of Table 4 indicate the message is a request to create a subscription. Line (19) indicates that it is sent to a hypothetical event source of ocean events.

Lines (13-18) indicate where to send the response to this request, Lines (23-30) indicate where to send a SubscriptionEnd message if necessary, and Lines (31-34) indicate how and where to send notifications.

Line (41) indicates the event sink would prefer to have the subscription expire on 26 June 2004 at 9:07 PM Pacific time.

Lines (42-45) indicate the event sink only wants weather reports where topic is storms, using a custom filter dialect.

Table 5 lists a hypothetical response to the request in Table 4.

Table 5: Hypothetical response to second subscribe request

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ew='http://www.example.com/warnings'
(06)     xmlns:ow='http://www.example.org/oceanwatch' >
(07)   <s12:Header>
(08)     <wsa:Action>
(09)      http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse
(10)     </wsa:Action>
(11)     <wsa:RelatesTo>
(12)         uuid:e1886c5c-5e86-48d1-8c77-fc1c28d47180
(13)     </wsa:RelatesTo>
```

```
(14)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(15)     <ew:MySubscription>2597</ew:MySubscription>
(16)   </s12:Header>
(17)   <s12:Body>
(18)     <wse:SubscribeResponse>
(19)       <wse:SubscriptionManager>
(20)         <wsa:Address>
(21)           http://www.example.org/oceanwatch/SubscriptionManager
(22)         </wsa:Address>
(23)         <wsa:ReferenceParameters>
(24)           <wse:Identifier>
(25)             uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(26)           </wse:Identifier>
(27)         </wsa:ReferenceParameters>
(28)       </wse:SubscriptionManager>
(29)       <wse:Expires>2004-07-01T00:00:00.000-00:00</wse:Expires>
(30)     </wse:SubscribeResponse>
(31)   </s12:Body>
(32) </s12:Envelope>
```

Like the response in Table 2, Lines (08-10) of Table 5 indicate this message is a response to a request to create a subscription, and Lines (11-13) indicate that it is a response to the request in Table 4. Lines (14-15) indicate the response is sent to the event sink indicated in Lines (13-18) of Table 4. Lines (19-28) provide the address of the subscription manager for this subscription; note that this particular response uses the global wse:Identifier element defined by this specification. Finally, Line (29) indicates the subscription will expire on 1 July 2004 unless renewed; there is no requirement that this time be necessarily longer or shorter than the requested expiration (Line (41) of Table 4).

3.2 Renew

To update the expiration for a subscription, subscription managers MUST support requests to renew subscriptions.

To renew a subscription, the subscriber sends a request of the following form to the subscription manager:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/08/eventing/Renew
    </wsa:Action>
    ...
  </s:Header>
  <s:Body ...>
    <wse:Renew ...>
      <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires> ?
```

```
      ...
    </wse:Renew>
  </s:Body>
</s:Envelope>
```

Components of the outline listed above are additionally constrained as for a request to create a subscription (see Section 3.1 Subscribe). Other components of the outline above are not further constrained by this specification.

If the subscription manager accepts a request to renew a subscription, it MUST reply with a response of the following form:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/08/eventing/RenewResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body ...>
    <wse:RenewResponse ...>
      <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires> ?
      ...
    </wse:RenewResponse>
  </s:Body>
</s:Envelope>
```

Components of the outline listed above are constrained as for a response to a subscribe request (see Section 3.1 Subscribe) with the following addition(s):

/s:Envelope/s:Body/*/wse:Expires
: If the requested expiration is a duration, then the implied start of that duration is the time when the subscription manager starts processing the Renew request.

If the subscription manager chooses not to renew this subscription, the request MUST fail, and the subscription manager MAY generate a wse:UnableToRenew fault indicating that the renewal was not accepted.

Other components of the outline above are not further constrained by this specification.

Table 6 lists a hypothetical request to renew the subscription created in Table 5.

Table 6: Hypothetical request to renew second subscription

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/08/eventing/Renew
(09)     </wsa:Action>
```

```
(10)     <wsa:MessageID>
(11)         uuid:bd88b3df-5db4-4392-9621-aee9160721f6
(12)     </wsa:MessageID>
(13)     <wsa:ReplyTo>
(14)       <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
(15)     </wsa:ReplyTo>
(16)     <wsa:To>
(17)       http://www.example.org/oceanwatch/SubscriptionManager
(18)     </wsa:To>
(19)     <wse:Identifier>
(20)       uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(21)     </wse:Identifier>
(22)   </s12:Header>
(23)   <s12:Body>
(24)     <wse:Renew>
(25)       <wse:Expires>2004-06-26T21:07:00.000-08:00</wse:Expires>
(26)     </wse:Renew>
(27)   </s12:Body>
(28) </s12:Envelope>
```

Lines (07-09) indicate this is a request to renew a subscription. Lines (19-21) contain the reference parameter that indicates the subscription to be renewed is the one created in Table 5. Line (25) in Table 6 indicates the request is to extend the subscription until 26 June 2004 at 9:07 PM Pacific.

Table 7 lists a hypothetical response to the request in Table 6.

Table 7: Hypothetical response to renew request

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)      http://schemas.xmlsoap.org/ws/2004/08/eventing/RenewResponse
(09)     </wsa:Action>
(10)     <wsa:RelatesTo>
(11)         uuid:bd88b3df-5db4-4392-9621-aee9160721f6
(12)     </wsa:RelatesTo>
(13)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(14)   </s12:Header>
(15)   <s12:Body>
(16)     <wse:RenewResponse>
(17)       <wse:Expires>2004-06-26T12:00:00.000-00:00</wse:Expires>
(18)     </wse:RenewResponse>
```

```
(19)   </s12:Body>
(20) </s12:Envelope>
```

Line (17) in Table 7 indicates the subscription has been extended only until 26 June 2004 at noon.

3.3 GetStatus

To get the status of a subscription, the subscriber sends a request of the following form to the subscription manager:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatus
    </wsa:Action>
    ...
  </s:Header>
  <s:Body ...>
    <wse:GetStatus ...>
      ...
    </wse:GetStatus>
  </s:Body>
</s:Envelope>
```

Components of the outline listed above are additionally constrained as for a request to renew a subscription (see Section 3.2 Renew). Other components of the outline above are not further constrained by this specification.

If the subscription is valid and has not expired, the subscription manager MUST reply with a response of the following form:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatusResponse
    </wsa:Action>
    ...
  </s:Header>
  <s:Body ...>
    <wse:GetStatusResponse ...>
      <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires> ?
      ...
    </wse:GetStatusResponse>
  </s:Body>
</s:Envelope>
```

Components of the outline listed above are constrained as for a response to a renew request (see Section 3.2 Renew). Other components of the outline above are not further constrained by this specification.

Table 8 lists a hypothetical request to get the status of the subscription created in Table 5.

Table 8: Hypothetical request to get the status of the second subscription

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatus
(09)     </wsa:Action>
(10)     <wsa:MessageID>
(11)         uuid:bd88b3df-5db4-4392-9621-aee9160721f6
(12)     </wsa:MessageID>
(13)     <wsa:ReplyTo>
(14)       <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
(15)     </wsa:ReplyTo>
(16)     <wsa:To>
(17)       http://www.example.org/oceanwatch/SubscriptionManager
(18)     </wsa:To>
(19)     <wse:Identifier>
(20)       uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(21)     </wse:Identifier>
(22)   </s12:Header>
(23)   <s12:Body>
(24)     <wse:GetStatus />
(25)   </s12:Body>
(26) </s12:Envelope>
```

Lines (07-09) indicate this is a request to get the status of a subscription. Lines (16-21) indicate that the request is sent to the subscription manager for the subscription created in Table 5.

Table 9 lists a hypothetical response to the request in Table 8.

Table 9: Hypothetical response to get status request

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)      http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatusResponse
(09)     </wsa:Action>
```

```
(10)     <wsa:RelatesTo>
(11)         uuid:bd88b3df-5db4-4392-9621-aee9160721f6
(12)     </wsa:RelatesTo>
(13)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(14)   </s12:Header>
(15)   <s12:Body>
(16)     <wse:GetStatusResponse>
(17)       <wse:Expires>2004-06-26T12:00:00.000-00:00</wse:Expires>
(18)     </wse:GetStatusResponse>
(19)   </s12:Body>
(20) </s12:Envelope>
```

Line (17) in Table 9 indicates the subscription will expire on 26 June 2004 at noon.

3.4 Unsubscribe

Though subscriptions expire eventually, to minimize resources, the subscribing event sink SHOULD explicitly delete a subscription when it no longer wants notifications associated with the subscription.

To explicitly delete a subscription, a subscribing event sink sends a request of the following form to the subscription manager:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/08/eventing/Unsubscribe
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wse:Unsubscribe ...>
      ...
    </wse:Unsubscribe>
  </s:Body>
</s:Envelope>
```

Components of the outline above are additionally constrained only as for a request to renew a subscription (see Section 3.2 Renew). For example, the faults listed there are also defined for a request to delete a subscription.

If the subscription manager accepts a request to delete a subscription, it MUST reply with a response of the following form:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/08/eventing/UnsubscribeResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
```

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

Components of the outline listed above are not further constrained by this specification.

Table 10 lists a hypothetical request to delete the subscription created in Table 5.

Table 10: Hypothetical unsubscribe request to delete second subscription

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/08/eventing/Unsubscribe
(09)     </wsa:Action>
(10)     <wsa:MessageID>
(11)         uuid:2653f89f-25bc-4c2a-a7c4-620504f6b216
(12)     </wsa:MessageID>
(13)     <wsa:ReplyTo>
(14)       <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
(15)     </wsa:ReplyTo>
(16)     <wsa:To>
(17)       http://www.example.org/oceanwatch/SubscriptionManager
(18)     </wsa:To>
(19)     <wse:Identifier>
(20)       uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(21)     </wse:Identifier>
(22)   </s12:Header>
(23)   <s12:Body>
(24)     <wse:Unsubscribe />
(25)   </s12:Body>
(26) </s12:Envelope>
```

Lines (07-09) in Table 10 indicate the message is a request to delete a subscription. Lines (16-21) indicate that the request is addressed to the manager for the subscription created in Table 5.

Table 11 lists a hypothetical response to the request in Table 10.

Table 11: Hypothetical response to unsubscribe request

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing' >
(04)   <s12:Header>
```

```
(05)     <wsa:Action>
(06)   http://schemas.xmlsoap.org/ws/2004/08/eventing/UnsubscribeResponse
(07)     </wsa:Action>
(08)     <wsa:RelatesTo>
(09)         uuid:2653f89f-25bc-4c2a-a7c4-620504f6b216
(10)     </wsa:RelatesTo>
(11)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(12)   </s12:Header>
(13)   <s12:Body />
(14) </s12:Envelope>
```

3.5 Subscription End

If the event source terminates a subscription unexpectedly, the event source SHOULD send a Subscription End SOAP message to the endpoint reference indicated when the subscription was created (see 3.1 Subscribe.) The message MUST be of the following form:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscriptionEnd
    </wsa:Action> ?
    ...
  </s:Header>
  <s:Body ...>
    <wse:SubscriptionEnd ...>
      <wse:SubscriptionManager>
        endpoint-reference
      </wse:SubscriptionManager>
      <wse:Status>
        [
    http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryFailure |
    http://schemas.xmlsoap.org/ws/2004/08/eventing/SourceShuttingDown |
    http://schemas.xmlsoap.org/ws/2004/08/eventing/SourceCanceling
        ]
      </wse:Status>
      <wse:Reason xml:lang="language identifier" >xs:string</wse:Reason> ?
      ...
    </wsa:SubscriptionEnd>
    ...
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

/s:Envelope/s:Body/*/wse:SubscriptionManager

Endpoint reference of the subscription manager. This element may be used to identify the subscription that has been terminated.

/s:Envelope/s:Body/wse:SubscriptionEnd/wse:Status = "http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryFailure"

This value MUST be used if the event source terminated the subscription because of problems delivering notifications.

/s:Envelope/s:Body/wse:SubscriptionEnd/wse:Status = "http://schemas.xmlsoap.org/ws/2004/08/eventing/SourceShuttingDown"

This value MUST be used if the event source terminated the subscription because the source is being shut down in a controlled manner; that is, if the event source is being shut down but has the opportunity to send a SubscriptionEnd message before it exits.

/s:Envelope/s:Body/wse:SubscriptionEnd/wse:Status = "http://schemas.xmlsoap.org/ws/2004/08/eventing/SourceCanceling"

This value MUST be used if the event source terminated the subscription for some other reason before it expired.

/s:Envelope/s:Body/wse:SubscriptionEnd/wse:Reason

This optional element contains text, in the language specified by the @xml:lang attribute, describing the reason for the unexpected subscription termination.

Other message information headers defined by WS-Addressing [WS-Addressing] MAY be included in the message, according to the usage and semantics defined in WS-Addressing.

Other components of the outline above are not further constrained by this specification.

Table 12 lists a hypothetical SubscriptionEnd message corresponding to an early termination of the subscription created in Table 4.

Table 12: Hypothetical subscription end message

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
(05)     xmlns:ew='http://www.example.com/warnings' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscriptionEnd
(09)     </wsa:Action>
(10)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(11)     <ew:MySubscription>2597</ew:MySubscription>
(12)   </s12:Header>
(13)   <s12:Body>
(14)     <wse:SubscriptionEnd>
(15)       <wse:SubscriptionManager>
(16)         <wsa:Address>
(17)           http://www.example.org/oceanwatch/SubscriptionManager
(18)         </wsa:Address>
```

```
(19)         <wsa:ReferenceParameters>
(20)           <wse:Identifier>
(21)             uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(22)           </wse:Identifier>
(23)         </wsa:ReferenceParameters>
(24)       </wse:SubscriptionManager>
(25)       <wse:Code>wse:SourceShuttingDown</wse:Code>
(26)       <wse:Reason xml:lang='en-US' >
(27)         Event source going off line.
(28)       </wse:Reason>
(29)     </wse:SubscriptionEnd>
(30)   </s12:Body>
(31) </s12:Envelope>
```

Line (08) is the action URI for Subscription End. Lines (10-11) indicate the message is sent to the EndTo of the subscribe request (Lines (23-30) in Table 4.). Line (25) indicates the event source is shutting down, and Lines (26-28) indicate that the event source was going off line.

4. Notifications

This specification does not constrain notifications because any message MAY be a notification.

However, if a subscribing event sink wishes to have notifications specifically marked, it MAY specify literal SOAP header blocks in the Subscribe request, in the /s:Envelope/s:Body/wse:Subscribe/wse:NotifyTo/wsa:ReferenceProperties or /s:Envelope/s:Body/wse:Subscribe/wse:NotifyTo/wsa:ReferenceProperties elements; per WS-Addressing [WS-Addressing], the event source MUST include each such literal SOAP header block in every notification sent to the endpoint addressed by /s:Envelope/s:Body/wse:Subscribe/wse:NotifyTo.

Table 13 lists a hypothetical notification message sent as part of the subscription created by the subscribe request in Table 4.

Table 13: Hypothetical notification message

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
(04)     xmlns:ew='http://www.example.com/warnings'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://www.example.org/oceanwatch/2003/WindReport
(09)     </wsa:Action>
(10)     <wsa:MessageID>
(11)         uuid:568b4ff2-5bc1-4512-957c-0fa545fd8d7f
(12)     </wsa:MessageID>
(13)     <wsa:To>http://www.other.example.com/OnStormWarning</wsa:To>
```

```
(14)     <ew:MySubscription>2597</ew:MySubscription>
(15)     <ow:EventTopics>weather.report weather.storms</ow:EventTopics>
(16)   </s12:Header>
(17)   <s12:Body>
(18)     <ow:WindReport>
(19)       <ow:Date>030701</ow:Date>
(20)       <ow:Time>0041</ow:Time>
(21)       <ow:Speed>65</ow:Speed>
(22)       <ow:Location>BRADENTON BEACH</ow:Location>
(23)       <ow:County>MANATEE</ow:County>
(24)       <ow:State>FL</ow:State>
(25)       <ow:Lat>2746</ow:Lat>
(26)       <ow:Long>8270</ow:Long>
(27)       <ow:Comments xml:lang='en-US' >
(28)         WINDS 55 WITH GUSTS TO 65. ROOF TORN OFF BOAT HOUSE. REPORTED
(29)         BY STORM SPOTTER. (TBW)
(30)       </ow:Comments>
(31)     </ow:WindReport>
(32)   </s12:Body>
(33) </s12:Envelope>
```

Lines (13-14) indicate the message is sent to the endpoint indicated by the subscribe request (Lines (32-39) in Table 4). Line (15) matches the filter in the subscribe request (Lines (42-45) in Table 4).

5. Faults

All fault messages defined in this specification MUST be sent according to the rules described in WS-Addressing section 4. They are sent to the [fault endpoint], if present and valid. Otherwise they are sent to the [reply endpoint] if present. If neither is present faults may be sent to the [source endpoint].

Endpoints compliant with this specification MUST include required message information headers on all fault messages. Fault messages are correlated as replies using the [relationship] property as defined in WS-Addressing. The [action] property below designates fault messages:

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
```

The definitions of faults use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

The properties above bind to a SOAP 1.2 fault as follows:

```
<S:Envelope>
```

```
 <S:Header>
   <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
   </wsa:Action>
   <!-- Headers elided for clarity.  -->
 </S:Header>
 <S:Body>
  <S:Fault>
   <S:Code>
     <S:Value>[Code]</S:Value>
     <S:Subcode>
      <S:Value>[Subcode]</S:Value>
     </S:Subcode>
   </S:Code>
   <S:Reason>
     <S:Text xml:lang="en">[Reason]</S:Text>
   </S:Reason>
   <S:Detail>
     [Detail]
   </S:Detail>
  </S:Fault>
 </S:Body>
</S:Envelope>
```

The SOAP 1.1 fault is less expressive and map only [Subcode] and [Reason]. These the properties bind to a SOAP 1.1 fault as follows:

```
<S11:Envelope>
 <S11:Body>
  <S11:Fault>
   <faultcode>[Subcode]</faultcode>
   <faultstring xml:lang="en">[Reason]</faultstring>
  </S11:Fault>
 </S11:Body>
</S11:Envelope>
```

5.1 DeliveryModeRequestedUnavailable

This fault is sent when a Subscribe request specifies a delivery mode that is not supported by the event source. Optionally, this fault may contain a list of supported delivery mode URIs in the Detail property.

[Code] s12:Sender

[Subcode] wse:DeliveryModeRequestedUnavailable

[Reason] The requested delivery mode is not supported.

[Detail] <wse:SupportedDeliveryMode> +

Optional; one per delivery mode supported by the receiver

5.2 InvalidExpirationTime

This fault is sent when a Subscribe request specifies an expiration time that is in the past or an expiration duration of zero.

[Code] s12:Sender

[Subcode] wse:InvalidExpirationTime

[Reason] The expiration time requested is invalid.

[Detail] *none*

5.3 UnsupportedExpirationType

This fault is sent when a Subscribe request specifies an expiration time and the event source is only capable of accepting expiration durations; for instance, if the event source does not have access to absolute time.

[Code] s12:Sender

[Subcode] wse:UnsupportedExpirationType

[Reason] Only expiration durations are supported.

[Detail] *none*

5.4 FilteringNotSupported

This fault is sent when a Subscribe request contains a filter and the event source does not support filtering.

[Code] s12:Sender

[Subcode] wse:FilteringNotSupported

[Reason] Filtering is not supported.

[Detail] *none*

5.5 FilteringRequestedUnavailable

This fault is sent when a Subscribe request specifies a filter dialect that the event source does not support. Optionally, this fault may contain a list of supported filter dialect URIs in the Detail property.

[Code] s12:Sender

[Subcode] wse:FilteringRequestedUnavailable

[Reason] The requested filter dialect is not supported.

[Detail] <wse:SupportedDialect> +

Optional; one per filter dialect supported by the receiver

5.6 EventSourceUnableToProcess

This fault is sent when the event source is not capable of fulfilling a Subscribe request for local reasons unrelated to the specific request.

[Code] s12:Receiver

[Subcode] wse:EventSourceUnableToProcess

[Reason] *Text explaining the failure; e.g., "The event source has too many subscribers".*

[Detail] *none*

5.7 UnableToRenew

This fault is sent when the event source is not capable of fulfilling a Renew request for local reasons unrelated to the specific request.

[Code] s12:Receiver

[Subcode] wse:UnableToRenew

[Reason] *Text explaining the failure; e.g., "The event source has too many subscribers".*

[Detail] *none*

5.8 InvalidMessage

If a request message does not comply with the corresponding outline listed above, the request MUST fail and the event source or subscription manager MAY generate the following fault indicating that the request is invalid:

[Code] s12:Sender

[Subcode] wse:InvalidMessage

[Reason] The message is not valid and cannot be processed.

[Detail] *The invalid message*

6. Security Considerations

6.1 Message Security

It is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [WS-Security]. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, any headers identified in the `<wse:NotifyTo>` element and standard messaging headers, such as those from WS-Addressing [WS-Addressing], need to be signed with the body in order to "bind" the two together. For messages with empty bodies, the `<s12:Body>` element should be signed so content cannot be added in transit.

Different security mechanisms may be desired depending on the frequency of messages. For example, for infrequent messages, public key technologies may be adequate for integrity and confidentiality. However, for high-frequency events, it may be more performant to establish a security context for the events using the mechanisms described in WS-Trust [WS-Trust] and WS-SecureConversation [WS-SecureConversation].

It should be noted that if a shared secret is used it is RECOMMENDED that derived keys be used to strengthen the secret as described in WS-SecureConversation.

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

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

6.2 Access Control

It is important for event sources to properly authorize requests. This is especially true for Subscribe requests, as otherwise the ability to subscribe on behalf of a third-party event sink could be used to create a distributed denial-of-service attack.

Some possible schemes for validating Subscribe requests include:

- Send a message to the event sink that describes the requested subscription, and then wait for a confirmation message to be returned by the event sink, before the event source accepts the subscription request. While this provides strong assurance that the event sink actually desires the requested subscription, it does not work for event sinks that are not capable of sending a confirmation, and requires additional logic on the event sink.
- Require user authentication on the Subscribe request, and allow only authorized users to Subscribe.

Other mechanisms are also possible. Note that event sources that are not reachable from the Internet have less need to control Subscribe requests.

7. Implementation Considerations

Implementations SHOULD generate expirations in subscribe and renew request and response messages that are significantly larger than expected network latency.

Event sinks should be prepared to receive notifications after sending a subscribe request but before receiving a subscribe response message. Event sinks should also be prepared to receive notifications after receiving an unsubscribe response message.

8. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including:

Josh Cohen (Microsoft)
Geary Eppley (Microsoft)
Omri Gazitt (Microsoft)
Peter Jarvis (Microsoft)
Chris Kaler (Microsoft)
Ray McCollum (Microsoft)
Toby Nixon (Microsoft)
Denny Page (TIBCO Software)
Krish Srinivasan (Microsoft)
Anders Vinberg (Microsoft)
Alex Weinert (Microsoft).

9. References

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997.

[SOAP 1.1]
D. Box, et al, "Simple Object Access Protocol (SOAP) 1.1," May 2000.

[SOAP 1.2]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

[WS-Addressing]
D. Box et al, "Web Services Addressing (WS-Addressing)," August 2004

[WS-Policy]
D. Box, et al, "Web Services Policy Framework (WS-Policy)," May 2003.

[WS-ReliableMessaging]
R. Bilorusets, et al, "Web Services Reliable Messaging Protocol (WS-ReliableMessaging)," March 2004.

[WS-SecureConversation]
G. Della-Libera et al, "Web Services Secure Conversation Language (WS-SecureConversation)", May, 2004

[WS-Security]
A. Nadalin et al, "Web Services Security: SOAP Message Security 1.0", May, 2004

[WS-SecurityPolicy]
G. Della-Libera, et al, "Web Services Security Policy Language (WS-SecurityPolicy)," December 2002.

[WS-Trust]
S. Anderson, et al, "Web Services Trust Language (WS-Trust)," May 2004.

[WSDL 1.1]
E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March 2001.

[XML Infoset]
J. Cowan, et al, "XML Information Set," October 2001.

[XML Schema, Part 1]
H. Thompson, et al, "XML Schema Part 1: Structures," May 2001.

[XML Schema, Part 2]
P. Biron, et al, "XML Schema Part 2: Datatypes," May 2001.

[XPath 1.0]
J. Clark, et al, "XML Path Language (XPath) Version 1.0," November 1999.

Appendix I – Service Metadata for Eventing

In order to obtain the event-related metadata that describes a service, the mechanisms described in WS-MetadataExchange should be used. The GetMetadata operation defined there allows WSDL and policy information to be retrieved. The WSDL will contain annotations that identify a service as an event source and that identify those messages that describe notification messages. The policy will specify the delivery modes and filter types supported by the event source.

To indicate that notification and solicit-response operations within a WSDL 1.1 portType are events exposed by an event source, this specification defines an @wse:EventSource attribute to annotate the portType for the event source. The normative outline for the @wse:EventSource attribute is:

```
<wsd:definitions ...>

  <wsdl:import
    namespace='http://schemas.xmlsoap.org/ws/2004/08/eventing'
    location='http://schemas.xmlsoap.org/ws/2004/08/eventing.wsdl' />

 [<wsdl:portType [wse:EventSource='xs:boolean']? >
   [<wsdl:operation ...>
     [
      [<wsdl:input .../>] |
      [<wsdl:output .../>] |
      [<wsdl:input .../> <wsdl:output .../>] |
      [<wsdl:output .../> <wsdl:input .../>]
     ]
    </wsdl:operation>]+
  </wsdl:portType>]*
</wsdl:definitions>
```

The following describes additional, normative constraints on the outline listed above:

/wsdl:definitions/wsdl:portType/@wse:EventSource
If omitted, implied value is false.

/wsdl:definitions/wsdl:portType/@wse:EventSource='true'
Indicates the portType supports the Subscribe operation and indicates that notification and solicit-response operations of the portType are events exposed by a service with a port bound to this portType.

Other components of the outline above are not further constrained by this specification.

For example, here is the WSDL 1.1 for a hypothetical storm warning service that exposes a wind report event.

```
<wsdl:definitions
    targetNamespace="http://www.example.org/oceanwatch"
    xmlns:tns="http://www.example.org/oceanwatch"
    xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    xmlns:xs='http://www.w3.org/2001/XMLSchema' >
  <wsdl:import
    namespace='http://schemas.xmlsoap.org/ws/2004/08/eventing'
    location='http://schemas.xmlsoap.org/ws/2004/08/eventing.wsdl' />
  <wsdl:types>
    <xs:schema
      targetNamespace="http://www.example.org/oceanwatch"
      elementFormDefault="qualified"
      blockDefault="#all" >
      <xs:element name="WindReport" >
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Date" type="xs:string" />
            <xs:element name="Time" type="xs:string" />
            <xs:element name="Speed" type="xs:string" />
            <xs:element name="Location" type="xs:string" />
            <xs:element name="County" type="xs:string" />
            <xs:element name="State" type="xs:string" />
            <xs:element name="Lat" type="xs:string" />
            <xs:element name="Long" type="xs:string" />
            <xs:element name="Comments" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name='WindMsg' >
    <wsdl:part name='body' element='tns:WindReport' />
  </wsdl:message>
  <wsdl:portType name='Warnings' wse:EventSource='true' >
    <wsdl:operation name='WindOp' >
      <wsdl:output message='tns:WindMsg' />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

As described here, to subscribe to events exposed by an event source, a subscribing endpoint sends a Subscribe message to the endpoint reference for the event source. If the Subscribe does not include a filter, the event sink should expect to receive events defined by notification operations within the portType and should expect to receive and respond to events defined by solicit-response operations within the portType.

Editor's Note: We anticipate that this WSDL extension may change in subsequent versions of this specification.

Appendix II – XML Schema

A normative copy of the XML Schema [XML Schema Part 1, Part 2] for this specification may be retrieved by resolving the XML namespace URI for this specification (listed in Section 2.2 XML Namespaces).

A non-normative copy of the XML schema is listed below for convenience.

```
<xs:schema
  targetNamespace="http://schemas.xmlsoap.org/ws/2004/08/eventing"
  xmlns:tns="http://schemas.xmlsoap.org/ws/2004/08/eventing"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  blockDefault="#all">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd" />
  <xs:import namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing" />

  <!-- Types and global elements -->
  <xs:complexType name="DeliveryType" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="Mode" type="xs:anyURI" use="optional" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:simpleType name="NonNegativeDurationType">
    <xs:restriction base="xs:duration">
      <xs:minInclusive value="P0Y0M0DT0H0M0S" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ExpirationType">
      <xs:union memberTypes="xs:dateTime tns:NonNegativeDurationType" />
```

```
</xs:simpleType>

<xs:complexType name="FilterType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Dialect" type="xs:anyURI" use="optional" />
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>

<xs:complexType name="LanguageSpecificStringType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute ref="xml:lang" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="NotifyTo" type="wsa:EndpointReferenceType" />

<!-- Subscribe request -->
<xs:element name="Subscribe">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="EndTo" type="wsa:EndpointReferenceType"
        minOccurs="0" />
      <xs:element name="Delivery" type="tns:DeliveryType" />
      <xs:element name="Expires" type="tns:ExpirationType"
        minOccurs="0" />
      <xs:element name="Filter" type="tns:FilterType" minOccurs="0" />
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>

<xs:element name="Identifier" type="xs:anyURI" />

<!-- Subscribe response -->
<xs:element name="SubscribeResponse">
```

```
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SubscriptionManager"
                  type="wsa:EndpointReferenceType" />
      <xs:element name="Expires" type="tns:ExpirationType" />
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>

<!-- Used in a fault if there's an unsupported dialect -->
<xs:element name="SupportedDialect" type="xs:anyURI" />

<!-- Used in a fault if there's an unsupported delivery mode -->
<xs:element name="SupportedDeliveryMode" type="xs:anyURI" />

<!-- Renew request -->
<xs:element name="Renew">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Expires" type="tns:ExpirationType"
        minOccurs="0" />
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>

<!-- Renew response -->
<xs:element name="RenewResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Expires" type="tns:ExpirationType"
        minOccurs="0" />
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>
```

```
<!-- GetStatus request -->
<xs:element name="GetStatus">
  <xs:complexType>
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>

<!-- GetStatus response -->
<xs:element name="GetStatusResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Expires" type="tns:ExpirationType"
        minOccurs="0" />
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>

<!-- Unsubscribe request -->
<xs:element name="Unsubscribe">
  <xs:complexType>
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>

<!-- count(/s:Envelope/s:Body/*) = 0 for Unsubscribe response -->

<!-- SubscriptionEnd message -->
<xs:element name="SubscriptionEnd">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SubscriptionManager"
```

```
                    type="wsa:EndpointReferenceType" />
        <xs:element name="Status"
          type="tns:OpenSubscriptionEndCodeType" />
        <xs:element name="Reason" type="tns:LanguageSpecificStringType"
          minOccurs="0" maxOccurs="unbounded" />
        <xs:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="SubscriptionEndCodeType">
    <xs:restriction base="xs:anyURI">
      <xs:enumeration value=
  "http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryFailure" />
      <xs:enumeration value=
  "http://schemas.xmlsoap.org/ws/2004/08/eventing/SourceShuttingDown" />
      <xs:enumeration value=
  "http://schemas.xmlsoap.org/ws/2004/08/eventing/SourceCancelling" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="OpenSubscriptionEndCodeType">
    <xs:union memberTypes="tns:SubscriptionEndCodeType xs:anyURI" />
  </xs:simpleType>

  <xs:attribute name="EventSource" type="xs:boolean" />
</xs:schema>
```

Appendix III – WSDL

A normative copy of the WSDL [WSDL 1.1] description can be retrieved from the following address:

```
http://schemas.xmlsoap.org/ws/2004/08/eventing/eventing.wsdl
```

A non-normative copy of the WSDL description is listed below for convenience.

```
<wsdl:definitions
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/08/eventing'
    xmlns:wse='http://schemas.xmlsoap.org/ws/2004/08/eventing'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    xmlns:xs='http://www.w3.org/2001/XMLSchema' >

  <wsdl:types>
    <xs:schema
```

```
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/08/eventing'>
    <xs:include schemaLocation='eventing.xsd' />
  </xs:schema>
</wsdl:types>

<wsdl:message name='SubscribeMsg' >
  <wsdl:part name='body' element='wse:Subscribe' />
</wsdl:message>
<wsdl:message name='SubscribeResponseMsg' >
  <wsdl:part name='body' element='wse:SubscribeResponse' />
</wsdl:message>

<wsdl:message name='RenewMsg' >
  <wsdl:part name='body' element='wse:Renew' />
</wsdl:message>
<wsdl:message name='RenewResponseMsg' >
  <wsdl:part name='body' element='wse:RenewResponse' />
</wsdl:message>

<wsdl:message name='GetStatusMsg' >
  <wsdl:part name='body' element='wse:GetStatus' />
</wsdl:message>
<wsdl:message name='GetStatusResponseMsg' >
  <wsdl:part name='body' element='wse:GetStatusResponse' />
</wsdl:message>

<wsdl:message name='UnsubscribeMsg' >
  <wsdl:part name='body' element='wse:Unsubscribe' />
</wsdl:message>
<wsdl:message name='UnsubscribeResponseMsg' />

<wsdl:message name='SubscriptionEnd' >
  <wsdl:part name='body' element='wse:SubscriptionEnd' />
</wsdl:message>

<wsdl:portType name='EventSource' >
  <wsdl:operation name='SubscribeOp' >
    <wsdl:input message='wse:SubscribeMsg' />
    <wsdl:output message='wse:SubscribeResponseMsg' />
  </wsdl:operation>
  <wsdl:operation name='SubscriptionEnd' >
    <wsdl:output message="wse:SubscriptionEnd' />
  </wsdl:operation>
```

```
  </wsdl:portType>

  <wsdl:portType name='SubscriptionManager' >
    <wsdl:operation name='RenewOp' >
      <wsdl:input message='wse:RenewMsg' />
      <wsdl:output message='wse:RenewResponseMsg' />
    </wsdl:operation>
    <wsdl:operation name='GetStatusOp' >
      <wsdl:input message='wse:GetStatusMsg' />
      <wsdl:output message='wse:GetStatusResponseMsg' />
    </wsdl:operation>
    <wsdl:operation name='UnsubscribeOp' >
      <wsdl:input message='wse:UnsubscribeMsg' />
      <wsdl:output message='wse:UnsubscribeResponseMsg' />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```