Web Services Eventing (WS-Eventing)

January 2004

Authors

Luis Felipe Cabrera, Microsoft
Craig Critchley, Microsoft
Gopal Kakivaya, Microsoft
Brad Lovering, Microsoft
Matt Mihic, BEA Systems
David Orchard, BEA Systems
Shivajee Samdarshi, TIBCO Software
Jeffrey Schlimmer (Editor), Microsoft
John Shewchuk, Microsoft
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.

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 an "event sink") to register interest (called a "subscription") with another Web service (called an "event source") in receiving messages about events (called "notifications"). To improve robustness, the subscription is leased by an event source to an event sink, and the subscription expires over time. An event source may allow an event sink to renew the subscription.

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 an event sink can determine which subscriptions it is receiving notifications for.
- Define how one event sink can subscribe on behalf of another.
- Leverage other Web service specifications for secure, reliable, transacted message delivery.
- Provide extensibility for more sophisticated and/or currently unanticipated subscription scenarios.
- Support both SOAP 1.1 [SOAP 1.1] and SOAP 1.2 [SOAP 1.2] Envelopes.

1.2 Example

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

Table 1: Hypothetical request to create a subscription.

```
<s12:Envelope
    xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
    xmlns:wsa='http://schemas.xmlsoap.org/ws/2003/03/addressing'
    xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
    xmlns:ew='http://www.example.com/warnings' >
  <s12:Header>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/01/eventing/Subscribe
    </wsa:Action>
    <wsa:MessageID>
        uuid:d7c5726b-de29-4313-b4d4-b3425b200839
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://www.example.org/oceanwatch/EventSource</wsa:To>
  </s12:Header>
  <s12:Body>
    <wse:Subscribe>
      <wse:NotifyTo>
        <wsa:Address>
            http://www.example.com/MyEventSink/OnStormWarning
```

```
(23)          </wsa:Address>
(24)          <wsa:ReferenceProperties>
(25)            <ew:MySubscription>2597</ew:MySubscription>
(26)          </wsa:ReferenceProperties>
(27)        </wse:NotifyTo>
(28)      </wse:Subscribe>
(29)    </s12:Body>
(30) </s12:Envelope>
(31)
```

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 (20-27) indicate where notifications should be sent; there is no requirement that these match. Note that Lines (24-26) illustrate a typical pattern where the event sink lists a reference property (Line 25) 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/2003/03/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)      http://schemas.xmlsoap.org/ws/2004/01/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:Id>uuid:5005cfe6-c2c6-4296-9c3a-80b9ad111813</wse:Id>
(18)       <wse:Expires>P0Y0M0DT30H0M0S</wse:Expires>
(19)     </wse:SubscribeResponse>
(20)   </s12:Body>
```

```
(21) </s12:Envelope>
(22)
```

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. Line (17) lists the unique identifier for this subscription, and Line (18) 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/2004/01/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]

wsa	http://schemas.xmlsoap.org/ws/2003/03/addressing	WS-Addressing [WS-Addressing]
wse	http://schemas.xmlsoap.org/ws/2004/01/eventing	This specification
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy	WS-PolicyAssertions [WS-PolicyAssertions]
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]

2.3 Terminology

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

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

Event Sink
: A Web service that receives notifications and/or sends requests to create, renew, and/or delete subscriptions.

Subscribing Event Sink
: An event sink that sends request to create, renew, and/or delete subscriptions, perhaps on behalf of another event sink.

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.

3. Subscription Messages

To create, renew, and delete subscriptions, subscribing event sinks send request messages to event sources.

When an event source accepts a request to create a subscription, it does so for a given amount of time. If the event source accepts a renewal request, it updates that amount of time. During that time, notifications are sent by the event source to the requested destination. An event source may support filtering to limit notifications that are sent 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 event source accepts an unsubscribe request for the subscription, the subscription expires without being renewed, or the event source cancels the subscription prematurely. Finally, 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 subscribing event sink sends a request message of the following form to an event source:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/01/eventing/Subscribe
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:FaultTo>endpoint-reference</wsa:FaultTo> ?
    <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wse:Subscribe ...>
      <wse:NotifyTo>endpoint-reference</wse:NotifyTo>
      <wse:EndTo>endpoint-reference</wse:EndTo> ?
      <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires> ?
      <wse:Filter Dialect="xs:anyURI"? > xs:string </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:NotifyTo
: Where to send notification messages. MUST be of type wsa:EndpointReferenceType.

/s:Envelope/s:Body/*/wse:EndTo
: Where to send a message containing Subscription End SOAP header block when subscription ends. (See 3.4 Subscription End.) MUST be of type wsa:EndpointReferenceType. Implied value is NotifyTo.

/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.

/s:Envelope/s:Body/*/wse:Filter
: A Boolean expression in some dialect. (See /s:Envelope/s:Body/*/wse:Filter/@Dialect.) If the expression is 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, the request MUST fail, and the event source MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "filtering not supported"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wse:FilteringNotSupported
- s12:Reason/s12:Text = e.g., "filtering 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 SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "cannot filter notifications as requested"
- [optional] detail/wse:SupportedDialect = repeating; one per filter dialect supported by the receiver

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wse:FilteringRequestedUnavailable
- s12:Reason/s12:Text = e.g., "cannot filter notifications as requested"
- [optional] s12:Detail/wse:SupportedDialect = repeating; one per filter dialect supported by the receiver

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

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

/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 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/2004/01/eventing/SubscribeResponse
      </wsa:Action>
      <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
      <wsa:To>xs:anyURI</wsa:To>
      ...
    </s:Header>
    <s:Body ...>
      <wse:SubscribeResponse ...>
        <wse:Id>xs:anyURI</wse:Id>
        <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:Id
: The identifier for this subscription. MUST be unique across all subscriptions created by this 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).

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 choose to generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Server
- faultstring = e.g., "event source has too many subscribers"

SOAP 1.2:

- s12:Code/s12:Value = s12:Receiver
- s12:Code/s12:Subcode/s12:Value = wse:EventSourceUnableToProcess
- s12:Reason/s12:Text = e.g., "event source has too many subscribers"

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/2003/03/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
```

```
    xmlns:ew='http://www.example.com/warnings' >
  <s12:Header>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/01/eventing/Subscribe
    </wsa:Action>
    <wsa:MessageID>
        uuid:e1886c5c-5e86-48d1-8c77-fc1c28d47180
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://www.example.com/MyEvEntsink</wsa:Address>
      <wsa:ReferenceProperties>
        <ew:MySubscription>2597</ew:MySubscription>
      </wsa:ReferenceProperties>
    </wsa:ReplyTo>
    <wsa:To>http://www.example.org/oceanwatch/EventSource</wsa:To>
  </s12:Header>
  <s12:Body>
    <wse:Subscribe>
      <wse:NotifyTo>
        <wsa:Address>
            http://www.other.example.com/OnStormWarning
        </wsa:Address>
        <wsa:ReferenceProperties>
          <ew:MySubscription>2597</ew:MySubscription>
        </wsa:ReferenceProperties>
      </wse:NotifyTo>
      <wse:EndTo>
        <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
        <wsa:ReferenceProperties>
          <ew:MySubscription>2597</ew:MySubscription>
        </wsa:ReferenceProperties>
      </wse:EndTo>
      <wse:Expires>2003-06-26T21:07:00.000-08:00</wse:Expires>
      <wse:Filter xmlns:ow='http://www.example.org/oceanwatch' >
        /s12:Envelope/s12:Body/ow:WindReport/ow:Speed > 60
      </wse:Filter>
```

```
(41)     </wse:Subscribe>
(42)   </s12:Body>
(43) </s12:Envelope>
(44)
```

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 notifications, and Lines (31-36) indicate where to send a message when the subscription is terminated; there is no requirement that any of these refer to the same endpoint; in this example, Lines (31-36) indicate to send the termination message to the same endpoint as the response to this request.

Line (37) indicates the event sink would prefer to have the subscription expire on 26 June 2003 at 9:07 PM Pacific time.

Lines (38-40) indicate the event sink only wants weather reports where the wind speed is over 60 miles per hour.

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/2003/03/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/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/2004/01/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:Id>uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa</wse:Id>
(20)       <wse:Expires>2003-07-01T00:00:00.000-00:00</wse:Expires>
(21)     </wse:SubscribeResponse>
```

```
(22)   </s12:Body>
(23) </s12:Envelope>
(24)
```

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. Line (19) lists the unique identifier for this subscription. Finally, Line (28) indicates the subscription will expire on 1 July 2003 unless renewed; there is no requirement that this time be necessarily longer or shorter than the requested expiration (Line (37) of Table 4).

3.2 Renew

To update an expiration for a subscription, a subscribing event sink MAY wait for the subscription to expire or delete the subscription (see 3.3 Unsubscribe) and then request a new one (see 3.1 Subscribe). However, to preserve subscription continuity, event sources SHOULD support requests to renew subscriptions.

To renew a subscription, a subscribing event sink sends a request of the following form to the event source that created the subscription:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/01/eventing/Renew
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:FaultTo>endpoint-reference</wsa:FaultTo> ?
    <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wse:Renew ...>
      <wse:Id>xs:anyURI</wse:Id>
      <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) with the following addition(s):

/s:Envelope/s:Body/*/wse:Id

Identifier of the subscription. If the value is the empty string, the request MUST fail, and the event source MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "bad identifier"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wse:InvalidIdentifier
- s12:Reason/s12:Text = e.g., "bad identifier"

If there is no subscription with this identifier, the request MUST fail, and the event source MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "unknown subscription"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wse:UnknownSubscription
- s12:Reason/s12:Text = e.g., "unknown subscription"

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

If the event source 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/2004/01/eventing/RenewResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </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):

If the event source chooses not to renew this subscription, the request MUST fail, and the event source MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Server
- faultstring = e.g., "event source can’t process renew"

SOAP 1.2:

- s12:Code/s12:Value = s12:Receiver
- s12:Code/s12:Subcode/s12:Value = wse:EventSourceUnableToProcess
- s12:Reason/s12:Text = e.g., "event source can't process renew"

If the event source does not support renewing subscriptions, the request MUST fail, and the event source MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "action not supported"
- detail/wsa:Action = Action URI for the request

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wse:ActionNotSupported
- s12:Reason/s12:Text = e.g., "action not supported"
- s12:Detail/wsa:Action = Action URI of the request

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

The corresponding WSDL [WSDL 1.1] for use with other bindings is below. (Refer to Appendix II - WSDL for the complete WSDL.)

```
<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:portType name='Eventing' >
...
    <!-- Support for Renew is optional -->
    <wsdl:operation name='RenewOp' >
      <wsdl:input message='wse:RenewMsg' />
      <wsdl:output message='wse:RenewResponseMsg' />
    </wsdl:operation>
...
</wsdl:portType>
```

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/2003/03/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/01/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>http://www.example.org/oceanwatch/EventSource</wsa:To>
(17)   </s12:Header>
(18)   <s12:Body>
(19)     <wse:Renew>
(20)       <wse:Id>uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa</wse:Id>
(21)       <wse:Expires>2004-06-26T21:07:00.000-08:00</wse:Expires>
(22)     </wse:Renew>
(23)   </s12:Body>
(24) </s12:Envelope>
(25)
```

Lines (07-09) indicate this is a request to renew a subscription. Line (20) indicates the subscription to be renewed is the one created in Table 5. Line (21) 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/2003/03/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
```

```
(06)   <s12:Header>
(07)     <wsa:Action>
(08)      http://schemas.xmlsoap.org/ws/2004/01/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>
(21)
```

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

3.3 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 event source that created the subscription:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/01/eventing/Unsubscribe
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:FaultTo>endpoint-reference</wsa:FaultTo> ?
    <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body>
    <wse:Unsubscribe ...>
```

```
      <wse:Id>xs:anyURI</wse:Id>
      ...
    </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 event source 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/2004/01/eventing/UnsubscribeResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body />
</s:Envelope>
```

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

The corresponding WSDL [WSDL 1.1] for use with other bindings is below. (Refer to Appendix II - WSDL for the complete WSDL.)

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

<wsdl:portType name='Eventing' >
...
  <wsdl:operation name='UnsubscribeOp' >
    <wsdl:input message='wse:UnsubscribeMsg' />
    <wsdl:output message='wse:UnsubscribeResponseMsg' />
  </wsdl:operation>
</wsdl:portType>
```

Table 8 lists a hypothetical request to delete the subscription created in Table 5.

Table 8: 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/2003/03/addressing'
(04)     xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
(05)     xmlns:ow='http://www.example.org/oceanwatch' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/01/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>http://www.example.org/oceanwatch/EventSource</wsa:To>
(17)   </s12:Header>
(18)   <s12:Body>
(19)     <wse:Unsubscribe>
(20)       <wse:Id>uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa</wse:Id>
(21)     </wse:Unsubscribe>
(22)   </s12:Body>
(23) </s12:Envelope>
(24)
```

Lines (07-09) in Table 8 indicate the message is a request to delete a subscription. Line (20) indicates that is a request to delete the subscription created in Table 5.

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

Table 9: 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/2003/03/addressing' >
(04)   <s12:Header>
(05)     <wsa:Action>
(06)
http://schemas.xmlsoap.org/ws/2004/01/eventing/UnsubscribeResponse
(07)     </wsa:Action>
```

```
    <wsa:RelatesTo>
        uuid:2653f89f-25bc-4c2a-a7c4-620504f6b216
    </wsa:RelatesTo>
    <wsa:To>http://www.example.com/MyEventSink</wsa:To>
  </s12:Header>
  <s12:Body />
</s12:Envelope>

```

3.4 Subscription End

When a subscription ends, the event source SHOULD send a Subscription End SOAP header block in a message to the endpoint reference indicated when the subscription was created. (See 3.1 Subscribe.) The header block MAY be sent along with another message ("piggybacked") or MAY be sent with an otherwise empty message. The message MUST be of the following form:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/01/eventing/SubscriptionEnd
    </wsa:Action> ?
    <wse:SubscriptionEnd s11:mustUnderstand="xs:boolean" ?
                         s12:mustUnderstand="xs:boolean" ?
                         ...>
      <wse:Id>xs:anyURI</wse:Id>
      <wse:Code>
        [wse:Unsubscribed | wse:Expired | wse:NotifyToFailure |
         wse:SourceCanceling]
      </wse:Code>
      <wse:Reason xml:lang="language identifier" >xs:string</wse:Reason> ?
      ...
    </wsa:SubscriptionEnd>
    ...
  </s:Header>
  <s:Body ...>
    ...
  </s:Body>
</s:Envelope>
```

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

/s:Envelope/s:Header/wsa:Action

If count(/s:Envelope/s:Body/*) = 0, then MUST have the indicated value. If a SOAP Action URI is used in the binding for SOAP, then the indicated value MUST be used for that URI.

/s:Envelope/s:Header/wse:SubscriptionEnd/wse:Code = "wse:Unsubscribed"

This value MUST be used if the event source accepted an unsubscribe request.

/s:Envelope/s:Header/wse:SubscriptionEnd/wse:Code = "wse:Expired"

This value MUST be used if the subscription expired without being renewed.

/s:Envelope/s:Header/wse:SubscriptionEnd/wse:Code = "wse:NotifyToFailure"

This value MUST be used if the event source terminated the subscription because of problems delivering notifications to the NotifyTo in the subscribe request.

/s:Envelope/s:Header/wse:SubscriptionEnd/wse:Code = "wse:SourceCanceling"

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

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

Table 10 lists a hypothetical message containing a Subscription End header block corresponding to an early termination of the subscription created in Table 5.

Table 10: Hypothetical subscription end header block.

```
<s12:Envelope
    xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
    xmlns:wsa='http://schemas.xmlsoap.org/ws/2003/03/addressing'
    xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
    xmlns:ew='http://www.example.com/warnings' >
  <s12:Header>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/01/eventing/SubscriptionEnd
    </wsa:Action>
    <wsa:To>http://www.example.com/MyEventSink</wsa:To>
    <ew:MySubscription>2597</ew:MySubscription>
    <wse:SubscriptionEnd>
      <wse:Id>uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa</wse:Id>
      <wse:Code>wse:SourceCanceling</wse:Code>
      <wse:Reason xml:lang='en-US' >
        Event source going off line.
      </wse:Reason>
    </wse:SubscriptionEnd>
  </s12:Header>
  <s12:Body />
</s12:Envelope>
```

```
(22)
```

Line (08) is the action URI for Subscription End since the header block is not piggybacked on another message and the Body has no children (Line 20). Lines (10-11) indicate the message is sent to the EndTo of the subscribe request (Lines (31-36) in Table 4.) Lines (12-18) contain the Subscription End SOAP header block; Line (14) indicates the subscription was cancelled early, and Lines (15-17) indicate the event source was going off line.

3.5 Other Faults

If a request message does not comply with the corresponding outline listed above, the request MUST fail, and the event source MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "message is invalid"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wse:InvalidRequest
- s12:Reason/s12:Text = e.g., "message is invalid"

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 /s:Envelope/s:Body/wse:Subscribe/wse:NotifyTo/wsa:ReferenceProperties; per WS-Addressing [WS-Addressing], the event source MUST include each such literal SOAP header block in every notification sent to the /s:Envelope/s:Body/wse:Subscribe/wse:NotifyTo.

Table 11 lists a hypothetical notification message sent as part of the subscription created by the subscribe request in Table 4.

Table 11: 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/2003/03/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)   </s12:Header>
(16)   <s12:Body>
(17)     <ow:WindReport>
(18)       <ow:Date>030701</ow:Date>
(19)       <ow:Time>0041</ow:Time>
(20)       <ow:Speed>65</ow:Speed>
(21)       <ow:Location>BRADENTON BEACH</ow:Location>
(22)       <ow:County>MANATEE</ow:County>
(23)       <ow:State>FL</ow:State>
(24)       <ow:Lat>2746</ow:Lat>
(25)       <ow:Long>8270</ow:Long>
(26)       <ow:Comments xml:lang='en-US' >
(27)         WINDS 55 WITH GUSTS TO 65. ROOF TORN OFF BOAT HOUSE. REPORTED
(28)         BY STORM SPOTTER. (TBW)
(29)       </ow:Comments>
(30)     </ow:WindReport>
(31)   </s12:Body>
(32) </s12:Envelope>
(33)
```

Lines (13-14) indicate the message is sent to the endpoint indicated by the subscribe request (Lines (23-30) in Table 4). Lines (01, 16, 17, 20) match the filter in the subscribe request (Lines (38-40) in Table 4).

5. Security Considerations

It is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [WS-Security, Addendum]. 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. 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.

7. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including: Don Box (Microsoft), Josh Cohen (Microsoft), Geary Eppley (Microsoft), Omri Gazitt (Microsoft), Peter Jarvis (Microsoft), Chris Kaler (Microsoft), Amy Lewis (TIBCO Software), Toby Nixon (Microsoft), Denny Page (TIBCO Software), Anders Vinberg (Microsoft), Alex Weinert (Microsoft).

8. 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]
A. Bosworth, et al, "Web Services Addressing (WS-Addressing)," March 2003.

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

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

[WS-ReliableMessaging]
R. Bilorusets, et al, "Web Services Reliable Messaging Protocol (WS-ReliableMessaging)," March 2003.

[WS-SecureConversation]
G. Della-Libera, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," December 2002.

[WS-Security]
B. Atkinson, et al, "Web Services Security (WS-Security)," April 2002.

[WS-Security Addendum]
G. Della-Libera, et al, "Web Services Security Addendum," August 2002.

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

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

[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 – XML Schema

A normative copy of the XML Schema [XML Schema Part 1, Part 2] and WSDL [WSDL 1.1] descriptions 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 description is listed below for convenience.

Normative text within this specification takes precedence over the normative outlines that take precedence over the XML Schema and WSDL descriptions.

```
<xs:schema
```

```
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/01/eventing'
    xmlns:tns='http://schemas.xmlsoap.org/ws/2004/01/eventing'
    xmlns:s11='http://schemas.xmlsoap.org/soap/envelope/'
    xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
    xmlns:wsa='http://schemas.xmlsoap.org/ws/2003/03/addressing'
    xmlns:wsp='http://schemas.xmlsoap.org/ws/2002/12/policy'
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    elementFormDefault='qualified'
    blockDefault='#all' >

  <xs:import
      namespace='http://schemas.xmlsoap.org/soap/envelope/'
      schemaLocation='http://schemas.xmlsoap.org/soap/envelope/' />

  <xs:import
      namespace='http://www.w3.org/2003/05/soap-envelope'
      schemaLocation='http://www.w3.org/2003/05/soap-envelope' />

  <xs:import
      namespace='http://schemas.xmlsoap.org/ws/2003/03/addressing'
      schemaLocation='http://schemas.xmlsoap.org/ws/2003/03/addressing' />

  <xs:import
      namespace='http://schemas.xmlsoap.org/ws/2002/12/policy'
      schemaLocation='http://schemas.xmlsoap.org/ws/2002/12/policy' />

<!--
  <xs:import namespace='http://www.w3.org/XML/1998/namespace' />
-->

  <xs:element name='Subscribe' >
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='tns:NotifyTo' />
        <xs:element ref='tns:EndTo' minOccurs='0' />
        <xs:element ref='tns:Expires' minOccurs='0' />
      <xs:element ref='tns:Filter' 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='SubscribeTo' type='wsa:EndpointReferenceType' />

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

<xs:element name='EndTo' type='wsa:EndpointReferenceType' />

<xs:element name='Expires' >
  <xs:simpleType>
    <xs:union
      memberTypes='xs:dateTime tns:NonNegativeDurationType' />
  </xs:simpleType>
</xs:element>

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

<xs:element name='Filter' type='wsp:MessagePredicateAssertion' />

<xs:element name='SupportedDialect' type='xs:anyURI' />

<xs:element name='SubscribeResponse' >
  <xs:complexType>
    <xs:sequence>
```

```
        <xs:element ref='tns:Id' />
        <xs:element ref='tns:Expires' />
        <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='Id' type='xs:anyURI' />

  <xs:element name='Renew' >
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='tns:Id' />
        <xs:element ref='tns:Expires' 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='RenewResponse' >
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='tns:Expires' />
        <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='Unsubscribe' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='tns:Id' />
      <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 -->

<xs:element name='SubscriptionEnd' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='tns:Id' />
      <xs:element ref='tns:Code' />
      <xs:element ref='tns:Reason' minOccurs='0' />
      <xs:any namespace='##other'
              processContents='lax'
              minOccurs='0'
              maxOccurs='unbounded' />
    </xs:sequence>
    <xs:attribute ref='s11:actor' />
    <xs:attribute ref='s11:mustUnderstand' />
    <xs:attribute ref='s12:relay' />
    <xs:attribute ref='s12:role' />
    <xs:attribute ref='s12:mustUnderstand' />
    <xs:anyAttribute namespace='##other' processContents='lax' />
    </xs:complexType>
  </xs:element>

  <xs:element name='Code' >
    <xs:simpleType>
      <xs:restriction base='xs:QName' >
        <xs:enumeration value='tns:Unsubscribed' />
        <xs:enumeration value='tns:Expired' />
        <xs:enumeration value='tns:NotifyToFailure' />
        <xs:enumeration value='tns:SourceCanceling' />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name='Reason' >
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string' >
<!-- should be defined here but omitted because my validator complains
          <xs:attribute ref='xml:lang' use='required' />
-->
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <!-- No constraint on s:Envelope/s:Body/ for SubscriptionEnd -->

</xs:schema>
```

Appendix II – WSDL

A non-normative copy of the WSDL [WSDL 1.1] description for this specification is listed below for convenience.

```
<wsdl:definitions
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/01/eventing'
    xmlns:wse='http://schemas.xmlsoap.org/ws/2004/01/eventing'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns:xs='http://www.w3.org/2001/XMLSchema' >

<wsdl:types>
  <xs:schema>
    <xs:import
        namespace='http://schemas.xmlsoap.org/ws/2004/01/eventing'
        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='UnsubscribeMsg' >
  <wsdl:part name='body' element='wse:Unsubscribe' />
</wsdl:message>
<wsdl:message name='UnsubscribeResponseMsg' />

<wsdl:portType name='Eventing' >
  <wsdl:operation name='SubscribeOp' >
    <wsdl:input message='wse:SubscribeMsg' />
    <wsdl:output message='wse:SubscribeResponseMsg' />
  </wsdl:operation>
  <!-- Support for Renew is optional -->
  <wsdl:operation name='RenewOp' >
```

```
      <wsdl:input message='wse:RenewMsg' />
      <wsdl:output message='wse:RenewResponseMsg' />
    </wsdl:operation>
    <wsdl:operation name='UnsubscribeOp' >
      <wsdl:input message='wse:UnsubscribeMsg' />
      <wsdl:output message='wse:UnsubscribeResponseMsg' />
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```