# Web Services Eventing (WS-Eventing)

## W3C Working Draft 5 August 2010

**This version:**
http://www.w3.org/TR/2010/WD-ws-eventing-20100805
**Latest version:**
http://www.w3.org/TR/ws-eventing
**Previous version:**
http://www.w3.org/TR/2010/WD-ws-eventing-20100330
**Editors:**
Doug Davis, IBM
Ashok Malhotra, Oracle
Katy Warr, IBM
Wu Chou, Avaya



---

## Abstract

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

## Status of this Document

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.*

This document is second the Last Call Working Draft of the Web Services Eventing (WS-Eventing) specification; It is intended to be published and maintained as a W3C Recommendation after review and refinement.

The Web Services Resource Access Working Group (WSRA WG) believes to have addressed all issues brought forth through previous Working Draft iterations, including during the first Last Call period (See the [changelog]). The Working Group encourages

feedback about this document. The Last Call period extends through 17 September 2010.

It has been produced by the Web Services Resource Access Working Group (WG), which is part of the W3C Web Services Activity. The working Group home page provides additional information on this and the related specifications produced by this working group.

Please provide comments on this document to the public public-ws-resource-access-comments@w3.org mailing list ( public archive).

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

## Table of Contents

## Appendices



---

# 1 Composable Architecture

By using the XML, SOAP [SOAP11], [SOAP12], and WSDL [WSDL11] extensibility models, the Web service specifications (WS-*) are designed to be composed with each other to provide a rich set of tools for 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.

# 2 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"). The subscriber can manage the subscription by interacting with a Web service (called the "subscription manager") designated by the event source.

To improve robustness, a subscription can 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 notifications can be delivered to event sinks. This specification provides an extensible way for subscribers to identify the delivery mechanism they prefer.

## 2.1 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 notifications are to 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 [SOAP11] and SOAP 1.2 [SOAP12] Envelopes.

## 2.2 Delivery

This specification defines a method for transmitting notifications to the event sink through the use of the wse:NotifyTo element. Other methods or combination of methods MAY be defined through the use of delivery extensions.

When the wse:NotifyTo element is used within the Delivery element it specifies the endpoint to which notifications are sent. For delivery to addressable endpoints this is sufficient. However, for non-addressable endpoints some additional mechanisms are needed. A subscription manager MAY choose to support mechanisms, such as the [WS-MakeConnection] specification, to enable delivery of notifications and the SubscriptionEnd message to non-addressable endpoints. See the [WS-MakeConnection] specification for more information, and an example, of how this would work.

## 2.3 Notification Formats

This specification does not mandate how events are serialized into notification messages. Rather, within the Subscribe request message a subscriber can specify a "Format" that indicates the set of rules that MUST be followed when constructing notification messages.

This specification specifies two delivery formats: wrapped and unwrapped. Use of wrapped format indicates that notification messages MUST be contained in a wrapper element defined by this specification. Use of unwrapped format indicates that notification messages are not contained in a wrapper element.

Filtering occurs prior to any formatting of notification messages. This ensures that regardless of what type of formatting might occur, the same Filter dialect/expression can be used to subset the event stream.

## 2.4 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 might 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 can interact with to manage this subscription. This EPR MUST be the target for future requests to renew or cancel the subscription. It can address the same Web service (Address and ReferenceParameters) as the event source itself, or it can address some other Web service to which the event source has delegated management of this subscription.

The term "subscription manager" is used 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.

## 2.5 Example

Example 2-1 lists a hypothetical request to create a subscription for storm warnings.

**Example 2-1: Hypothetical request to create a subscription**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ew="http://www.example.com/warnings" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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:ReferenceParameters>
(26)             <ew:MySubscription>2597</ew:MySubscription>
(27)           </wsa:ReferenceParameters>
(28)         </wse:NotifyTo>
(29)       </wse:Delivery>
(30)     </wse:Subscribe>
(31)   </s12:Body>
(32) </s12:Envelope>
```

Lines (07-09) in Example 2-1 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 is sent, lines (20-29) indicate where and how notifications are to be delivered; there is no requirement that these match. The absence of any extensions to the wse:Delivery or wse:NotifyTo elements indicates that notifications are 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 parameter (line 26) that identifies the subscription and will be included in each notification.

Example 2-2 lists a hypothetical response to the request in Example 2-1.

**Example 2-2: Hypothetical response to a subscribe request**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ow="http://www.example.org/oceanwatch" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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:GrantedExpires>PT0S</wse:GrantedExpires>
(28)     </wse:SubscribeResponse>
(29)   </s12:Body>
(30) </s12:Envelope>
```

Lines (07-09) in Example 2-2 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 Example 2-1. Lines (17-26) provide the subscription manager EPR for this subscription, and line (27) indicates the subscription will not expire.

# 3 Notations and Terminology

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

## 3.1 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.
- The characters "[" and "]" are used to call out references and property names.
- Ellipsis (i.e. "...") indicate a point of extensibility.
- XML namespace prefixes (see Table 3-1) are used to indicate the namespace of the element being defined.

In addition to Message Information Header properties [WS-Addressing], this specification uses the following properties to define messages:

**[Headers]**

Unordered message headers.

**[Action]**

The value to be used for the wsa:Action IRI.

**[Body]**

A message body.

These properties bind to a SOAP Envelope as follows:

```
<s:Envelope>
  <s:Header>
    [Headers]
    <wsa:Action>[Action]</wsa:Action>
    ...
  </s:Header>
  <s:Body>[Body]</s:Body>
</s:Envelope>
```

This specification can be used in terms of XML Information Set (Infoset) [XML Infoset], even though the specification uses XML 1.0 terminology. Valid Infoset for this specification is the one serializable in XML 1.0, hence the use of XML 1.0.

The term "generate" is used in relation to the various faults defined by this specification to imply that a fault is produced and no further processing SHOULD be performed. In these cases the fault SHOULD be transmitted. However, there might be reasons when

a compliant implementation can choose not to transmit the fault - for example, security concerns - in these situations the service MAY choose not to transmit the fault.

## 3.2 Considerations on the Use of Extensibility Points

The elements defined in this specification MAY be extended at the points indicated by their outlines and schema. Implementations MAY add child elements and/or attributes 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 ignore that extension. Senders MAY indicate the presence of an extension that has to be understood through the use of a corresponding SOAP Header with a soap:mustUnderstand attribute with the value "1".

In cases where it is either desirable or necessary for the receiver of a request that has been extended to indicate that it has recognized and accepted the semantics associated with that extension, it is RECOMMENDED that the receiver add a corresponding extension to the response message. The definition of an extension SHOULD clearly specify how the extension that appears in the response correlates with that in the corresponding request.

Extension elements and attributes MUST NOT use the Web Services Eventing namespace URI.

## 3.3 Terminology

**Event Source**

> A Web service that accepts requests to create subscriptions. Notifications MAY be sent by any endpoint including the event source.

**Event Sink**

> A Web service that receives notifications.

**Notification**

> A message sent to indicate that an event, or events, have occurred.

**Subscription**

> A registration of interest in receiving notification messages from an event source. Subscriptions MAY be created, renewed, expired or cancelled. A subscription is "active" when it has been created but has not been expired or cancelled.

**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.

## 3.4 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 **3.5 XML Namespaces**) within SOAP Envelopes unless it is compliant with this specification.

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

All messages defined by this specification MUST conform to the WS-Addressing specifications and be sent to a Web service that is addressable by an EPR (see [WS-Addressing]).

Unless otherwise noted, all IRIs are absolute IRIs and IRI comparison MUST be performed according to [RFC 3987] section 5.3.1.

For any message defined by this specification, any OPTIONAL elements or attributes in the message MAY be used by senders of the message; however receivers of those messages MUST support those OPTIONAL elements and attributes, unless other behavior is explicitly defined by this specification.

Implementations are expected to support both UTF-8 and UTF-16 as described in XML 1.0.

## 3.5 XML Namespaces

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

```
http://www.w3.org/2010/08/ws-evt
```

Table 3-1 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 3-1: 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 [SOAP11] |
| s12 | http://www.w3.org/2003/05/soap-envelope | SOAP 1.2 [SOAP12] |
| wsdl | http://schemas.xmlsoap.org/wsdl/ | WSDL [WSDL11] |
| wsa | http://www.w3.org/2005/08/addressing | WS-Addressing [WS-Addressing] |
| wse | http://www.w3.org/2010/08/ws-evt | This specification |
| xs | http://www.w3.org/2001/XMLSchema | XML Schema [XMLSchema - Part 1], [XMLSchema - Part 2] |

The working group intends to update the value of the Web Services Eventing namespace URI each time a new version of this document is published until such time that the document reaches Candidate Recommendation status. Once it has reached Candidate Recommendation status, the working group intends to maintain the value of the Web Services Eventing namespace URI that was assigned in the Candidate Recommendation unless significant changes are made that impact the implementation or break post-CR implementations of the specification. Also see http://www.w3.org/2001/tag/doc/namespaceState.htmland http://www.w3.org/2005/07/13-nsuri.

# 4 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 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, only events that match the requested filter are sent. Notifications are sent 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, via a SubscriptionEnd message if an EndTo was present in the Subscribe message.

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.

A subscription can become invalid for any reason including:

1. Subscription deleted (via Unsubscribe)
2. Subscription expired
3. Subscription ended (via SubscriptionEnd)

In addition, the Event Source/Subscription Manager MAY cancel a subscription at any time, as necessary.

When processing a Renew, GetStatus or Unsubscribe operation, a Subscription Manager MUST generate an wse:UnknownSubscription fault if it determines that the subscription is invalid.

Once a Subscriber determines that a subscription is invalid, it MUST NOT issue any more WSEventing request messages using that subscription.

## 4.1 Subscribe

To create a subscription, a subscriber sends a Subscribe request message to an event source. This operation MUST be supported by compliant event sources. The message MUST be of the following form:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/Subscribe

[Body]
  <wse:Subscribe ...>
    <wse:EndTo> endpoint-reference </wse:EndTo> ?
    <wse:Delivery ...> xs:any* </wse:Delivery>
    <wse:Format Name="xs:anyURI"? > xs:any* </wse:Format> ?
    <wse:Expires BestEffort="xs:boolean"? ...>
      (xs:dateTime | xs:duration)
    </wse:Expires> ?
    <wse:Filter Dialect="xs:anyURI"? ...> xs:any* </wse:Filter> ?
    xs:any*
  </wse:Subscribe>
```

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

**[Body]/wse:Subscribe/wse:EndTo**

> Where to send a SubscriptionEnd message if the subscription is terminated unexpectedly. (See **4.5 SubscriptionEnd**.) If present, this element MUST be of type wsa:EndpointReferenceType. Default is not to send this message. The endpoint to which the EndTo EPR refers MUST support the SubcriptionEndPortType portType. Unless some mechanism is used to indicate otherwise, the messages sent to this endpoint MUST use the same version of SOAP that was used for the Subscribe message.

If the event source does not support the use of the EndTo EPR, the event source MUST generate a wse:EndToNotSupported fault.

**[Body]/wse:Subscribe/wse:Delivery**

This element contains the information necessary to convey notification messages to the event sink in a manner REQUIRED by the subscriber. This element MUST contain at least one child element. If no delivery mechanism is specified then the event source MUST generate a wse:NoDeliveryMechanismEstablished fault.

**[Body]/wse:Subscribe/wse:Delivery/wse:NotifyTo**

This is an OPTIONAL element. When present, this element indicates that notifications MUST be sent to the EndpointReference identified by this element. Unless some mechanism is used to indicate otherwise, the messages sent to this endpoint MUST use the same version of SOAP that was used for the Subscribe message.

**[Body]/wse:Subscribe/wse:Format**

This OPTIONAL element contains the delivery format to be used for notification messages sent in relation to this subscription. Implied value is "http://www.w3.org/2010/08/ws-evt/DeliveryFormats/Unwrap", which indicates that unwrapped delivery MUST be used. See Section **2.3 Notification Formats** for details.

If the event source does not support the requested delivery format, the request MUST generate a wse:DeliveryFormatRequestedUnavailable fault indicating that the requested delivery format is not supported.

**[Body]/wse:Subscribe/wse:Format@Name="http://www.w3.org/2010/08/ws-evt/DeliveryFormats/Unwrap"**

Indicate the unwrapped event delivery format.

**[Body]/wse:Subscribe/wse:Format@Name="http://www.w3.org/2010/08/ws-evt/DeliveryFormats/Wrap"**

Indicate the wrapped event delivery format.

**[Body]/wse:Subscribe/wse:Expires**

This OPTIONAL element can be used by the subscriber to indicate the expiration time of the requested subscription. If this element is not present, the implied default expiration time is indefinite. A value of PT0S indicates a request for an indefinite expiration time.

If the event source does not support the wse:Expires element being present in a Subscribe request message then a wse:ExpiresNotSupported fault MUST be generated.

If the wse:Expires element is present and the event source is not able to grant an expiration time that matches the specified value then it MUST generate a wse:UnsupporedExpirationValue fault.

The value of the wse:Expires element MAY be either a duration (xs:duration) or a specific time (xs:dateTime). Event sources and subscription managers MUST accept duration values and MAY accept specific time values. Upon receiving a request that contains specific time values, an event source or subscription manager that does not support such value types MUST fail the request and generate a wse:UnsupportedExpirationType fault.

If a subscriber chooses to use specific time values in a request, it is RECOMMENDED that these values include a time zone component. Specific time values that lack a time zone MUST be interpreted in the local time zone of the receiver.

**[Body]/wse:Subscribe/wse:Expires@BestEffort**

This OPTIONAL attribute, when present with a value of 'true', indicates that the event source MUST grant an expiration time that is its best effort to match the requested expiration time. Default value of this attribute is "false" in which case the event source MUST grant the requested expiration time or fault the subscription request as defined above.

**[Body]/wse:Subscribe/wse:Filter**

A Boolean expression in some dialect, either as a string or as an XML fragment (see **[http://www.w3.org/TR/ws-eventing/#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 MUST 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 MUST generate a wse:FilteringRequestedUnavailable fault indicating that the requested filter dialect is not supported.

It is possible for a Subscribe request to contain a filter that will never evaluate to true for the lifetime of the subscription. If an event source detects this condition it

MUST generate a wse:EmptyFilter fault in response to the Subscribe request message.

**[Body]/wse:Subscribe/wse:Filter/@Dialect**

Implied value is "http://www.w3.org/2010/08/ws-evt/Dialects/XPath10".

If filtering is supported, then support for the XPath 1.0 dialect (described below) is RECOMMENDED. Alternate filter dialects can be defined. Such dialect definitions MUST include sufficient information for proper application. For example, it would need to define the context (which data) over which the filter operates.

**[Body]/wse:Subscribe/wse:Filter/@Dialect=" http://www.w3.org/2010/08/ws-evt/Dialects/XPath10 "**

Value of **[Body]**/wse:Subscribe/wse:Filter is an XPath [XPath1.0] predicate expression (PredicateExpr); the context of the expression is:

- Context Node: the root of the event XML.
- Context Position: 1.
- Context Size: 1.
- Variable Bindings: None.
- Function Libraries: Core Function Library [XPath1.0].
- Namespace Declarations: The [in-scope namespaces] property [XML Infoset] of /s:Envelope/s:Body/*/wse:Filter.

The namespace bindings are evaluated against any namespace declarations that are in scope where the XPath expression appears within the SOAP message. Note that the evaluation of expressions that rely on such context dependent bindings is fragile in the face of transformations that alter namespace prefixes. Such transformations might occur during the transmission, processing, storage, or retrieval of a request. Clients that wish to isolate expressions from the effects of any changes to the namespace prefixes in the containing SOAP message are advised to construct expressions in a manner that avoids the use of namespace prefixes. For example, use an expression such as "/a[namespace-uri()='http://www.example.com']" not "/ns1:a".

**[Body]/wse:Subscribe/wse:Filter/@Dialect=" http://www.w3.org/2010/08/ws-evt/Dialects/XPath20 "**

This filter dialect is the same as the XPath 1.0 filter dialect except that it uses [XPath2.0] instead of XPath 1.0 as the expression language.

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

If included within the Subscribe request message, the wse:NotifyTo and wse:EndTo SHOULD have some cursory validity checking performed before the Subscribe response is returned. While not all errors can be detected prior to sending a message to those EPRs, some obvious ones can be detected. For example, an unsupported transport specified within the wsa:Address IRI. Detecting these errors during Subscribe processing will lessen the chances of the subscriber creating an unusable subscription. If this check is performed and a problem is detected then the event source MUST generate a wse:UnusableEPR fault rather than returning the SubscribeResponse message.

If the event source chooses not to accept a subscription due to the content of the Subscribe message, then the event source MUST generate a SOAP 1.1 Client fault or a SOAP 1.2 Sender fault. If the event source does not accept this subscription due to an internal processing reason and not due to the specific content of the Subscribe message, then the event source MUST generate a SOAP 1.1 Server fault or a SOAP 1.2 Receiver fault which MAY contain the RetryAfter fault detail element.

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

```
[Action]
  http://www.w3.org/2010/08/ws-evt/SubscribeResponse

[Body]
  <wse:SubscribeResponse ...>
    <wse:SubscriptionManager>
      wsa:EndpointReferenceType
    </wse:SubscriptionManager>
    <wse:GrantedExpires ...>
      (xs:dateTime | xs:duration)
    </wse:GrantedExpires> ?
    xs:any*
  </wse:SubscribeResponse>
```

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

**[Body]/wse:SubscribeResponse/wse:SubscriptionManager**

The EPR of the subscription manager for this subscription.

**[Body]/wse:SubscribeResponse/wse:GrantedExpires**

The expiration time assigned by the event source. The expiration time MAY be either a specific time or a duration but MUST be of the same type as the wse:Expires element of the corresponding request. If the corresponding request did not contain a wse:Expires element, this element MUST be a duration (xs:duration).

When expressed as a duration, the wse:GrantedExpires element designates a time interval that began at the moment the subscription is created. Although this specification cannot dictate when, during the processing of a Subscribe request, a subscription is created, the event source MUST start the expiration interval at or before it transmits the wse:SubscribeResponse message.

If this element does not appear, then the subscription will not expire. That is, the subscription has an indefinite lifetime. Likewise, a value of PT0S indicates that the subscription will not expire.

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

Example 4-1 lists another hypothetical request to create a subscription.

**Example 4-1: 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://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ew="http://www.example.com/warnings" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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:ReferenceParameters>
(16)        <ew:MySubscription>2597</ew:MySubscription>
(17)      </wsa:ReferenceParameters>
(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:ReferenceParameters>
(28)           <ew:MySubscription>2597</ew:MySubscription>
(29)         </wsa:ReferenceParameters>
(30)       </wse:EndTo>
(31)       <wse:Delivery>
(32)         <wse:NotifyTo>
(33)           <wsa:Address>
(34)             http://www.other.example.com/OnStormWarning
(35)           </wsa:Address>
(36)           <wsa:ReferenceParameters>
(37)             <ew:MySubscription>2597</ew:MySubscription>
```

```
(38)           </wsa:ReferenceParameters>
(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)         /*/ow:Speed &gt; 50
(44)       </wse:Filter>
(45)     </wse:Subscribe>
(46)   </s12:Body>
(47) </s12:Envelope>
```

Like the request in Example 2-1, lines (07-09) of Example 4-1 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 (42-44) indicate the event sink only wants weather reports where the speed of wind is greater than 50.

Example 4-2 lists a hypothetical response to the request in Example 4-1.

**Example 4-2: Hypothetical response to second subscribe request**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ew="http://www.example.com/warnings"
(06)     xmlns:ow="http://www.example.org/oceanwatch" >
(07)   <s12:Header>
(08)     <wsa:Action>
(09)       http://www.w3.org/2010/08/ws-evt/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 wsa:IsReferenceParameter="true">
(16)       2597
(17)     </ew:MySubscription>
(18)   </s12:Header>
(19)   <s12:Body>
(20)     <wse:SubscribeResponse>
(21)       <wse:SubscriptionManager>
(22)         <wsa:Address>
```

```
(23)           http://www.example.org/oceanwatch/SubscriptionManager
(24)         </wsa:Address>
(25)         <wsa:ReferenceParameters>
(26)           <x:SubID xmlns:x="http://example.com">
(27)             uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(28)           </x:SubID>
(29)         </wsa:ReferenceParameters>
(30)       </wse:SubscriptionManager>
(31)       <wse:Expires>2004-06-26T21:07:00.000-08:00</wse:Expires>
(32)     </wse:SubscribeResponse>
(33)   </s12:Body>
(34) </s12:Envelope>
```

Like the response in Example 2-2, lines (08-10) of Example 4-2 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 Example 4-1 . Lines (14-17) indicate the response is sent to the event sink indicated in lines (13-18) of Example 4-1. Lines (21-30) provide the address of the subscription manager for this subscription; note that this particular response uses the x:SubID element as a reference parameter to distinguish this subscription EPR from other subscription EPRs. Finally, line (31) indicates the subscription will expire on 26 June 2004 at 9:07 PM Pacific time unless renewed.

## 4.2 Renew

To update, or renew, the expiration for a subscription, a subscriber sends a Renew request message to the subscription manager. This operation MUST be supported by compliant subscription managers. The Renew request message message MUST be of the following form:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/Renew

[Body]
  <wse:Renew ...>
    <wse:Expires BestEffort="xs:boolean"? ...>
      (xs:dateTime | xs:duration)
    </wse:Expires> ?
    xs:any*
  </wse:Renew>
```

Components of the outline listed above are additionally constrained as for a request to create a subscription (see **4.1 Subscribe**). Other components of the outline above are not further constrained by this specification.

If the subscription manager chooses not to renew this subscription, the request MUST fail, and the subscription manager MUST generate a SOAP 1.1 Server fault or a SOAP 1.2 Receiver fault indicating that the renewal was not accepted. This fault MAY contain the RetryAfter fault detail element.

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

```
[Action]
  http://www.w3.org/2010/08/ws-evt/RenewResponse

[Body]
  <wse:RenewResponse ...>
    <wse:GrantedExpires ...>
      (xs:dateTime | xs:duration)
    </wse:GrantedExpires> ?
    xs:any*
  </wse:RenewResponse>
```

Components of the outline listed above are constrained as for a response to a subscribe request (see **4.1 Subscribe**). Other components of the outline above are not further constrained by this specification.

Example 4-3 lists a hypothetical request to renew the subscription created in Example 4-2.

**Example 4-3: Hypothetical request to renew second subscription**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ow="http://www.example.org/oceanwatch" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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)     <x:SubID wsa:IsReferenceParameter="true"
xmlns:x="http://example.com">
(20)       uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(21)     </x:SubID>
(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 Example 4-2. Line (25) in Example 4-3 indicates the request is to extend the subscription until 26 June 2004 at 9:07 PM Pacific.

Example 4-4 lists a hypothetical response to the request in Example 4-3.

**Example 4-4: Hypothetical response to renew request**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ow="http://www.example.org/oceanwatch" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)      http://www.w3.org/2010/08/ws-evt/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>2004-06-26T21:07:00.000-08:00</wse:Expires>
(18)     </wse:RenewResponse>
(19)   </s12:Body>
(20) </s12:Envelope>
```

Line (17) in Example 4-4 indicates the subscription has been extended only until 26 June 2004 at noon.

## 4.3 GetStatus

To get the status of a subscription, the subscriber sends a GetStatus request message to the subscription manager. This operation MUST be supported by compliant subscription managers. The GetStatus request MUST be of the following form:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/GetStatus

[Body]
  <wse:GetStatus ...>
    xs:any*
  </wse:GetStatus>
```

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

If the subscription is active, the subscription manager MUST reply with a response of the following form:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/GetStatusResponse

[Body]
  <wse:GetStatusResponse ...>
    <wse:GrantedExpires ...>
      (xs:dateTime | xs:duration)
    </wse:GrantedExpires> ?
    xs:any*
  </wse:GetStatusResponse>
```

Components of the outline listed above are constrained as for a response to a renew request (see **4.2 Renew**). Other components of the outline above are not further constrained by this specification.

This operation is safe; it will not result in any side effect imputable to the requester. This means that in case of an underlying protocol error that might get unnoticed, resending the same request can be done automatically.

Example 4-5 lists a hypothetical request to get the status of the subscription created in Example 4-2.

**Example 4-5: 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://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ow="http://www.example.org/oceanwatch" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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)     <x:SubID wsa:IsReferenceParameter="true"
xmlns:x="http://example.com">
(20)       uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(21)     </x:SubID>
(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 Example 4-2.

Example 4-6 lists a hypothetical response to the request in Example 4-5.

**Example 4-6: Hypothetical response to get status request**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ow="http://www.example.org/oceanwatch" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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 Example 4-6 indicates the subscription will expire on 26 June 2004 at noon.

## 4.4 Unsubscribe

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

To explicitly delete a subscription, a subscriber sends an Unsubscribe request message to the subscription manager. This operation MUST be supported by compliant subscription managers. The Unsubscribe request message MUST be of the following form:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/Unsubscribe

[Body]
```

```
  <wse:Unsubscribe ...>
    xs:any*
  </wse:Unsubscribe>
```

Components of the outline above are additionally constrained only as for a request to renew a subscription (see **4.2 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:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/UnsubscribeResponse

[Body]
  <wse:UnsubscribeResponse ...>
    xs:any*
  </wse:UnsubscribeResponse>
```

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

Example 4-7 lists a hypothetical request to delete the subscription created in Example 4-2.

**Example 4-7: 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://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ow="http://www.example.org/oceanwatch" >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)       http://www.w3.org/2010/08/ws-evt/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)     <x:SubID wsa:IsReferenceParameter="true"
xmlns:x="http://example.com">
(20)       uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
(21)     </x:SubID>
(22)   </s12:Header>
(23)   <s12:Body>
(24)     <wse:Unsubscribe />
```

```
(25)   </s12:Body>
(26) </s12:Envelope>
```

Lines (07-09) in Example 4-7 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 Example 4-2.

Example 4-8 lists a hypothetical response to the request in Example 4-7.

**Example 4-8: Hypothetical response to unsubscribe request**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing" >
(04)   <s12:Header>
(05)     <wsa:Action>
(06)       http://www.w3.org/2010/08/ws-evt/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)     <wse:UnsubscribeResponse/>
(15)   </s12:Body>
(16) </s12:Envelope>
```

## 4.5 SubscriptionEnd

If the event source terminates a subscription unexpectedly, and it supports the use of the EndTo EPR, and the EndTo EPR was present in the Subscribe message for that subscription (see **4.1 Subscribe**), the SubscriptionEnd message MUST be sent to the endpoint reference indicated by that EPR, if possible.

Note, a subscription expiring as expected is not considered to be an unexpected termination, therefore a SubscriptionEnd message MUST NOT be sent in this case.

The message MUST be of the following form:

```
[Action]
  http://www.w3.org/2010/08/ws-evt/SubscriptionEnd

[Body]
  <wse:SubscriptionEnd ...>
    <wse:Status>
      ( http://www.w3.org/2010/08/ws-evt/DeliveryFailure |
        http://www.w3.org/2010/08/ws-evt/SourceShuttingDown |
        http://www.w3.org/2010/08/ws-evt/SourceCancelling |
```

```
        xs:anyURI )
    </wse:Status>
    <wse:Reason xml:lang="language identifier" >xs:string</wse:Reason> ?
    xs:any*
  </wse:SubscriptionEnd>
```

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

**[Body]/wse:SubscriptionEnd/wse:Status = "http://www.w3.org/2010/08/ws-evt/DeliveryFailure"**

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

**[Body]/wse:SubscriptionEnd/wse:Status = "http://www.w3.org/2010/08/ws-evt/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.

**[Body]/wse:SubscriptionEnd/wse:Status = "http://www.w3.org/2010/08/ws-evt/SourceCancelling"**

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

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

Example 4-9 lists a hypothetical SubscriptionEnd message corresponding to an early termination of the subscription created in Example 4-1.

**Example 4-9: Hypothetical subscription end message**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)     xmlns:wse="http://www.w3.org/2010/08/ws-evt"
(05)     xmlns:ew="http://www.example.com/warnings" >
(06)   <s12:Header>
(07)     <wsa:Action>
```

```
(08)       http://www.w3.org/2010/08/ws-evt/SubscriptionEnd
(09)     </wsa:Action>
(10)     <wsa:To>http://www.example.com/MyEventSink</wsa:To>
(11)     <ew:MySubscription wsa:IsReferenceParameter="true">
(12)       2597
(13)     </ew:MySubscription>
(14)   </s12:Header>
(15)   <s12:Body>
(16)     <wse:SubscriptionEnd>
(17)       <wse:Status>wse:SourceShuttingDown</wse:Status>
(18)       <wse:Reason xml:lang="en-US" >
(19)         Event source going off line.
(20)       </wse:Reason>
(21)     </wse:SubscriptionEnd>
(22)   </s12:Body>
(23) </s12:Envelope>
```

Line (08) is the action IRI for SubscriptionEnd. Lines (10-13) indicate the message is sent to the EndTo of the subscribe request (lines (23-30) in Example 4-1). Line (17) indicates the event source is shutting down, and lines (18-20) indicate that the event source was going off line.

# 5 Notifications

Notifications are SOAP messages that are transmitted to the event sink as the result of a successful Subscribe operation.

Example 5-1 lists a hypothetical notification message sent as part of the subscription created by the subscribe request in Example 4-1.

**Example 5-1: Hypothetical notification message**

```
(01) <s12:Envelope
(02)     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
(03)     xmlns:wsa="http://www.w3.org/2005/08/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 wsa:IsReferenceParameter="true">
(15)       2597
(16)     </ew:MySubscription>
(17)   </s12:Header>
(18)   <s12:Body>
(19)     <ow:WindReport>
(20)       <ow:Date>030701</ow:Date>
```

```
(21)       <ow:Time>0041</ow:Time>
(22)       <ow:Speed>65</ow:Speed>
(23)       <ow:Location>BRADENTON BEACH</ow:Location>
(24)       <ow:County>MANATEE</ow:County>
(25)       <ow:State>FL</ow:State>
(26)       <ow:Lat>27.46</ow:Lat>
(27)       <ow:Long>82.70</ow:Long>
(28)       <ow:Comments xml:lang="en-US" >
(29)         WINDS 55 WITH GUSTS TO 65. ROOF TORN OFF BOAT HOUSE. REPORTED
(30)         BY STORM SPOTTER. (TBW)
(31)       </ow:Comments>
(32)     </ow:WindReport>
(33)   </s12:Body>
(34) </s12:Envelope>
```

Lines (13-16) indicate the message is sent to the endpoint indicated by the subscribe request (lines (32-39) in Example 4-1). Line (22) matches the filter in the subscribe request (lines (42-44) in Example 4-1).

# 6 Faults

All fault messages defined in this specification MUST be sent according to the rules and usage described in [WS-Addressing 1.0 SOAP Binding] Section 6 for encoding SOAP 1.1 and SOAP 1.2 faults. The **[Action]** property below MUST be used for faults defined in this specification:

```
http://www.w3.org/2010/08/ws-evt/fault
```

The definitions of faults in this section 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.

For SOAP 1.2, the **[Code]** property MUST be either "Sender" or "Receiver". These properties are serialized into text XML as follows:

| SOAP Version | Sender | Receiver |
|---|---|---|
| SOAP 1.2 | s12:Sender | s12:Receiver |

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

```
<s12:Envelope>
  <s12:Header>
    <wsa:Action> [Action] </wsa:Action>
```

```
    <!-- Headers elided for brevity. -->
  </s12:Header>
  <s12:Body>
    <s12:Fault>
      <s12:Code>
        <s12:Value>[Code]</s12:Value>
        <s12:Subcode>
          <s12:Value>[Subcode]</s12:Value>
        </s12:Subcode>
      </s12:Code>
      <s12:Reason>
        <s12:Text xml:lang="en">[Reason]</s12:Text>
      </s12:Reason>
      <s12:Detail>
        [Detail]
        ...
      </s12:Detail>
    </s12:Fault>
  </s12:Body>
</s12:Envelope>
```

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>
      <detail>
        [Detail]
        ...
      </detail>
    </s11:Fault>
  </s11:Body>
</s11:Envelope>
```

## 6.1 Fault Detail RetryAfter Element

The following element MAY be used to convey additional information in the detail element of a fault.

**/wse:RetryAfter**

This element (whose content is of type xs:unsignedLong) is a suggested minimum duration in milliseconds to wait before retransmitting the message. Omission of this element indicates that a retry is never likely to succeed.

## 6.2 ExpiresNotSupported

This fault MUST be generated when a request specifies an expiration but the event source does not support Expires.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:ExpiresNotSupported |
| **[Reason]** | The specification of an Expires element is not allowed. |
| **[Detail]** | *none* |

## 6.3 UnsupporedExpirationValue

This fault MUST be generated when a request specifies an expiration that is not within the min/max range.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:UnsupporedExpirationValue |
| **[Reason]** | The expiration time requested is not within the min/max range. |
| **[Detail]** | *none* |

## 6.4 UnsupportedExpirationType

This fault MUST be generated 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* |

## 6.5 FilteringNotSupported

This fault MUST be generated 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* |

## 6.6 FilteringRequestedUnavailable

This fault MUST be generated when a Subscribe request specifies a filter dialect that the event source does not support. This fault MAY contain a list of supported filter dialect IRIs in the Detail property.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:FilteringRequestedUnavailable |
| **[Reason]** | The requested filter dialect is not supported. |
| **[Detail]** | <wse:SupportedDialect> +<br>*OPTIONAL; one per filter dialect supported by the receiver* |

## 6.7 DeliveryFormatRequestUnavailable

This fault MUST be generated when a Subscribe request specifies a delivery format that is not supported by the event source. This fault MAY contain a list of supported delivery format IRIs in the Detail property.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:DeliveryFormatRequestedUnavailable |
| **[Reason]** | The requested delivery format is not supported. |
| **[Detail]** | <wse:SupportedDeliveryFormat> +<br>*OPTIONAL, one per delivery format supported by the receiver.* |

## 6.8 EmptyFilter

This fault MUST be generated when an event source detects a wse:Subscribe request containing a filter that, for whatever reason, will never evaluate to true.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:EmptyFilter |
| **[Reason]** | The wse:Filter would result in zero notifications. |
| **[Detail]** | *The wse:Filter value.* |

## 6.9 UnusableEPR

This fault MUST be generated when an event source detects that the wse:NotifyTo or wse:EndTo EPR is unusable.

| | |
|---|---|
| **[Code]** | s12:Sender |

| | |
|---|---|
| **[Subcode]** | wse:UnusableEPR |
| **[Reason]** | An EPR in the Subscribe request message is unusable. |
| **[Detail]** | *The specific EPR that generated the error and why.* |

### 6.10 UnknownSubscription

This fault MUST be generated when a request specifies a subscription that is not known.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:UnknownSubscription |
| **[Reason]** | The subscription is not known. |
| **[Detail]** | *none* |

### 6.11 EndToNotSupported

This fault MUST be generated by an event source that does not support /wse:Subscribe/wse:EndTo semantics, in response to a subscription request that contains a wse:EndTo element.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:EndToNotSupported |
| **[Reason]** | wse:EndTo semantics is not supported. |
| **[Detail]** | *none* |

### 6.12 NoDeliveryMechanismEstablished

This fault MUST be generated by an event source when the Subscribe request message did not specify a delivery mechanism.

| | |
|---|---|
| **[Code]** | s12:Sender |
| **[Subcode]** | wse:NoDeliveryMechanismEstablished |
| **[Reason]** | No delivery mechanism specified. |
| **[Detail]** | *none* |

## 7 Security Considerations

This specification considers two sets of security requirements, those of the applications that use the WS-Eventing protocol and those of the protocol itself.

This specification makes no assumptions about the security requirements of the applications that use WS-Eventing. However, once those requirements have been satisfied within a given operational context, the addition of WS-Eventing to this operational context cannot undermine the fulfillment of those requirements; the use of WS-Eventing SHOULD NOT create additional attack vectors within an otherwise secure system.

The material below is not a "check list". There are many other security concerns that need to be considered when implementing or using this protocol. Implementers and users of this protocol are urged to perform a security analysis to determine their particular threat profile and the appropriate responses to those threats.

## 7.1 Notifications

The information contained in Notifications might be sensitive. In such cases it is advisable to authenticate and authorize subscribers as part of the processing of the Subscribe request. Note that an Event Source may authorize the delivery of Notifications on a per-message basis after the subscription has been created. This might be necessary in cases where the sensitivity of the Notification information is not known at Subscribe time or varies over the lifetime of a subscription.

To protect the Notifications sent over a network, Notifications ought to have the proper authenticity, integrity and confidentiality protections applied.

The ability to subscribe on behalf of a third-party Event Sink could be misused by a malicious Subscriber to create a denial-of-service attack. While it does not remove the ability for such misuse, authenticating Subscribers provides a way to deter and trace the origin of such attempts. Additionally, the authorization of Subscribers reduces the pool of potential attackers.

## 7.2 Subscriptions

Once created, subscriptions ought to be treated as protected resources. Renew, GetStatus, and Unsubscribe requests ought to be authenticated and authorized (for example, the identity of the requester ought to be checked against the identity of the entity that performed the original Subscribe request). Likewise SubscriptionEnd messages ought to be authenticated and authorized (for example, the identity of the sender ought to be checked against the identity of the entity that sent the original SubscribeResponse message). Note that authentication and authorization policies (i.e. the rules that define which entities are allowed to perform which requests and the mechanisms by which the identities of these entities are discovered and verified) are particular to individual deployments.

## 7.3 Endpoint Verification

Implementations that perform validity checks on the EPRs used in WS-Eventing (wse:NotifyTo, wse:EndTo) are advised that such checks can be misused to obtain information about a target network. For example, suppose an Event Source implementation verifies the address of NotifyTo EPRs by attempting to create a connection to the EPR's address and faulting the Subscribe request if a connection cannot be created. When deployed within a DMZ, such an Event Source could be exploited by a malicious Subscriber to probe for other, non-visible hosts by guessing target addresses and using them in Subscribe requests. Note that, even if the returned fault does not provide connection information, the time the Event Source spends processing the Subscribe request might reveal if there is a host with the target address.

Implementations that perform validity checks on the EPRs used in WS-Eventing are advised to provide a means to disable such checks in environments where these types of attacks are an issue.

# 8 Implementation Considerations

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.

# 9 WS-Eventing Metadata

An endpoint MAY indicate its support of WS-Eventing, or its features, by including the WS-Eventing EventSource or SubscriptionManager Policy assertions within its WSDL. By doing so the endpoint is indicating that the corresponding WS-Eventing operations are supported by that endpoint even though they are implicit and do not explicitly appear in its WSDL (ie. the WS-Eventing operations do not appear in the WSDL that MAY be retrievable by using WS-MetadataExchange GetWSDL to that endpoint).

The WS-Eventing WSDL containing the operations indicated by the EventSource or SubscriptionManager assertions MAY be exposed by including the WSDL as a child of the appropriate Policy assertion or by including a reference to it using the mex:Location or mex:Reference element ( as described in WS-MetadataExchange [WS-MetadataExchange] Section 9).

This WS-Eventing WSDL can be annotated to indicate any endpoint specific metadata that might be needed by clients interacting with the WS-Eventing operations. For example, the WSDL MAY have policy assertions that indicate a particular security mechanism used to protect the WS-Eventing operations supported by this endpoint.

## 9.1 EventSource Assertion

Services indicate support for the WS-Eventing's definition of an event source through the use of the Web Services Policy - Framework [WS-Policy] and Web Services Policy - Attachment [WS-Policy Attachment] specifications.

This specification defines a policy assertion (wse:EventSource). The normative outline of this assertion is:

```
<wse:EventSource ...>
  <wse:FilterDialect URI="xs:anyURI" ...>
    xs:any*
  </wse:FilterDialect> *
  <wse:FormatName URI="xs:anyURI" ...>
    <!-- Notification WSDL can go here - see A Advertising Event
Information -->
    xs:any*
  </wse:FormatName> *
  <wse:ExpiresSupported .../> ?
  <wse:DateTimeSupported .../> ?
  <wse:MaxExpires ...> xs:duration </wse:MaxExpires> ?
  <wse:EndToSupported .../> ?
  <wse:NotificationPolicy ...>
    xs:any
  </wse:NotificationPolicy> ?
  <!-- EventDescription data can go here - see A Advertising Event
Information -->
  xs:any*
</wse:EventSource>
```

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

**/wse:EventSource**

This policy assertion has Endpoint Policy Subject. When present in a policy alternative, it indicates that the subject is an event source and the WS-Eventing protocol MUST be used when communicating with this endpoint.

**/wse:EventSource/wse:FilterDialect**

When present, this OPTIONAL parameter indicates support for the specified filter dialect IRI.

**/wse:EventSource/wse:FilterDialect/xs:any**

This extensibility point allows for additional FilterDialect specific metadata to be included within the policy assertion. Any metadata that appears is scoped to the use of the specified FilterDialect URI.

**/wse:EventSource/wse:FormatName**

When present, this OPTIONAL parameter indicates support for the specified event delivery format name URI.

**/wse:EventSource/wse:FormatName/xs:any**

This extensibility point allows for additional FormatName specific metadata to be included within the policy assertion. Any metadata that appears is scoped to the use of the specified FormatName URI. If the Event Source advertises Notification WSDL then it MUST appear as a child of the FormatName element.

**/wse:EventSource/wse:ExpiresSupported**

When present, this OPTIONAL parameter indicates support for Expires element on the Subscribe operations.

**/wse:EventSource/wse:DateTimeSupported**

When present, this OPTIONAL parameter indicates support for expiration time expressed as specific time (rather than duration).

**/wse:EventSource/wse:MaxExpires**

When present, this OPTIONAL parameter indicates the maximum subscription expiration time that this endpoint will support. The implied default value is indefinite. Note: a value of "PT0S" indicates that this endpoint supports subscriptions with an infinite lifetime.

**/wse:EventSource/wse:EndToSupported**

When present, this OPTIONAL parameter indicates support for the /wse:Subscribe/wse:EndTo semantics. That is, when a subscription request contains a wse:EndTo element, a SubscriptionEnd message will be sent to the EPR contained in the wse:EndTo element, if the subscription terminates unexpectedly.

**/wse:EventSource/wse:NotificationPolicy**

When present, this OPTIONAL parameter includes the Endpoint Subject Policy that the event source supports for sending notifications. This element MUST have one child element - typically a wsp:Policy element or a wsp:PolicyReference element. A subscriber can use this information to discover the valid set of Policy Alternatives that MAY be used within a wse:NotifyTo EPR which will be used for any Notification message sent from the event source to the event sink. Any policy alternatives included within this parameter SHOULD be compatible with those Policy Alternatives available from the Notification WSDLs that the event source advertises.

**/wse:EventSource/xs:any**

This extensibility point allows for additional WS-Eventing specific metadata to be included within the policy assertion - e.g. WS-Eventing WSDL, or nested policy assertions related to the WS-Eventing message exchanges. Any metadata that appears is scoped to the operations and features of the WS-Eventing specification. If the Event Source advertises EventDescription data then it MUST appear as a child of the EventSource element.

The following is an example of a wse:EventSource policy assertion:

```
(01) <wse:EventSource ...>
(02)   <wse:DateTimeSupported />
(03)   <wse:FilterDialect URI="http://www.w3.org/2010/08/ws-
evt/Dialects/XPath10"/>
(04)   <wse:FormatName URI="http://www.w3.org/2010/08/ws-
evt/DeliveryFormats/Wrap">
(05)     <wsdl:definitions>
(06)       <!-- WSDL for Wrapped Notifications -->
(07)     </wsdl:definitions>
(08)   </wse:FormatName>
(09)   <wse:EndToSupported/>
(10)   <wsevd:EventDescriptions>
(11)     <!-- EventDescription data -->
(12)   </wsevd:EventDescriptions>
(13) </wse:EventSource>
```

In the above example line (02) indicates that this event source supports expiration times as specific times. Line (03) indicates that Filtering is supported and that it supports the XPath 1.0 filter dialect. Lines (04) - (08) indicates that this event source supports sending notifications in the Wrapped format and the Notification WSDL defining the messages is included on lines (05) - (07). Line (09) indicates that the EndTo features is supported. And finally, lines (10) - (12) contains the EventDescription data that describes the syntax of the events that this event source might send.

## 9.2 SubscriptionManager Assertion

Services indicate support for the WS-Eventing's definition of a subscription manager through the use of the Web Services Policy - Framework [WS-Policy] and Web Services Policy - Attachment [WS-Policy Attachment] specifications.

This specification defines a policy assertion (wse:SubscriptionManager). The normative outline of this assertion is:

```
<wse:SubscriptionManager ...>
  <wse:ExpiresTimeSupported .../> ?
  <wse:DateTimeSupported .../> ?
  <wse:MaxExpires ...> xs:duration </wse:MaxExpires> ?
```

```
  xs:any*
</wse:SubscriptionManager>
```

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

**/wse:SubscriptionManager**

This policy assertion has Endpoint Policy Subject. When present in a policy alternative, it indicates that the subject is an subscription manager and the WS-Eventing protocol MUST be used when communicating with this endpoint.

**/wse:SubscriptionManager/wse:ExpiresTimeSupported**

When present, this OPTIONAL parameter indicates support for Expires element on the Renew operation.

**/wse:SubscriptionManager/wse:DateTimeSupported**

When present, this OPTIONAL parameter indicates support for expiration time expressed as specific time (rather than duration).

**/wse:SubscriptionManager/wse:MaxExpires**

When present, this OPTIONAL parameter indicates the maximum subscriptions expiration time that this endpoint will support. The implied default is indefinite. Note: a value of "PT0S" indicates that this endpoint supports subscriptions with an infinite lifetime.

**/wse:SubscriptionManager/xs:any**

This extensibility point allows for additional WS-Eventing specific metadata to be included within the policy assertion - e.g. WS-Eventing WSDL, or nested policy assertions related to the WS-Eventing message exchanges. Any metadata that appears is scoped to the operations and features of the WS-Eventing specification.

# 10 Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including: Alessio Soldano (Red Hat), Ashok Malhotra (Oracle Corp.), Asir Vedamuthu (Microsoft Corp.), Bob Freund (Hitachi, Ltd.), Bob Natale (MITRE Corp.), David Snelling (Fujitsu, Ltd.), Doug Davis (IBM), Fred Maciel (Hitachi, Ltd.), Geoff Bullen (Microsoft Corp.), Gilbert Pilz (Oracle Corp.), Greg Carpenter (Microsoft Corp.), Jeff Mischkinsky (Oracle Corp.), Katy Warr (IBM), Li Li (Avaya Communications), Mark Little (Red Hat), Martin Chapman (Oracle Corp.), Paul Fremantle (WSO2), Paul Nolan

(IBM), Prasad Yendluri (Software AG), Ram Jeyaraman (Microsoft Corp.), Sreedhara Narayanaswamy (CA), Sumeet Vij (Software AG), Tom Rutt (Fujitsu, Ltd.), Vikas Varma (Software AG), Wu Chou (Avaya Communications), Yves Lafon (W3C/ERCIM).

# 11 References

## 11.1 Normative References


**RFC 2119**
*Key words for use in RFCs to Indicate Requirement Levels* , S. Bradner, Author. Internet Engineering Task Force, March 1997. (See http://www.ietf.org/rfc/rfc2119.txt.)

**RFC 3986**
*Uniform Resource Identifier (URI): Generic Syntax* , T. Berners-Lee, R. Fields and L. Masinter, Authors. Network Working Group, January 2005. (See http://www.ietf.org/rfc/rfc3986.txt.)

**RFC 3987**
*Internationalized Resource Identifiers (IRIs)* , M. Duerst and M. Suignard, Authors. Internet Engineering Task Force, January 2005. (See http://www.ietf.org/rfc/rfc3987.txt.)

**SOAP11**
*W3C Note, "Simple Object Access Protocol (SOAP) 1.1"* , D. Box, et al, Editors. World Wide Web Consortium (W3C), 8 May 2000. (See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.)

**SOAP12**
*W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework"* , M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielson, Editors. World Wide Web Consortium (W3C), 27 April 2007. (See http://www.w3.org/TR/soap12-part1/.)

**WS-Addressing**
*W3C Recommendation, "Web Services Addressing 1.0 (WS-Addressing)"* , M. Gudgin, M. Hadley, T. Rogers, Editors. World Wide Web Consortium (W3C), 9 May 2006. (See http://www.w3.org/TR/ws-addr-core.)

**WS-Addressing 1.0 SOAP Binding**
*W3C Recommendation, "Web Services Addressing 1.0 - SOAP Binding"* , M. Gudgin, M. Hadley, T. Rogers, Editors. World Wide Web Consortium (W3C), 9 May 2006. (See http://www.w3.org/TR/ws-addr-soap.)

**WS-EventDescriptions**
*W3C Working Group Draft, "Web Services Event Descriptions (WS-EventDescriptions) 1.0"* , D. Davis, et al., Editors. World Wide Web Consortium (W3C), 15 September 2009. (See http://www.w3.org/TR/ws-event-descriptions.)

**WS-Policy**
*W3C Recommendation, "Web Services Policy (WS-Policy) 1.5 - Framework"* , A. Vedamuthu, et al., Editors. World Wide Web Consortium (W3C), 4 September 2007. (See http://www.w3.org/TR/ws-policy/.)

**WS-Policy Attachment**

*W3C Recommendation, "Web Services Policy (WS-Policy) 1.5 - Attachment"* , A. Vedamuthu, et al., Editors. World Wide Web Consortium (W3C), 4 September 2007. (See http://www.w3.org/TR/ws-policy-attach.)

**WSDL11**

*W3C Note, "Web Services Description Language (WSDL) 1.1"* , E. Christensen, et al., Editors. World Wide Web Consortium (W3C), 15 March 2001 (See http://www.w3.org/TR/2001/NOTE-wsdl-20010315.)

## 11.2 Informative References

**WS-MakeConnection**

*OASIS Standard, "Web Services Make Connection (WS-MakeConnection) 1.1" Web Services Make Connection (WS-MakeConnection)* , D. Davis, et al., Editors. Organization for the Advancement of Structured Information Standards (OASIS), 2 February 2009. (See http://docs.oasis-open.org/ws-rx/wsmc/v1.1/wsmc.doc.)

**WS-MetadataExchange**

*W3C Working Group Draft, "Web Services Metadata Exchange (WS-MetadataExchange) 1.1"* , D. Davis, et al., Editors. World Wide Web Consortium (W3C), 15 September 2009. (See http://www.w3.org/TR/ws-metadata-exchange.)

**WS-ReliableMessaging**

*OASIS Standard, "Web Services Reliable Messaging Protocol (WS-ReliableMessaging) 1.2"* , Davis, et al., Editors. Organization for the Advancement of Structured Information Standards (OASIS), 2 February 2009. (See http://docs.oasis-open.org/ws-rx/wsrm/v1.2/wsrm.doc.)

**WS-SecureConversation**

*OASIS Standard, "Web Services Secure Conversation (WS-SecureConversation) 1.4"* , A. Nadalin, et al., Editors. Organization for the Advancement of Structured Information Standards (OASIS), 2 February 2009. (See http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc.)

**WS-Security**

*OASIS Standard, "Web Services Security: SOAP Message Security 1.1"* , K. Lawrence, C. Kaler, Editors. Organization for the Advancement of Structured Information Standards (OASIS), 1 February 2006. (See http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.)

**WS-SecurityPolicy**

*OASIS Standard, "Web Services Security Policy (WS-SecurityPolicy) 1.3, Version 1.1"* , K. Lawrence, C. Kaler, Editors. Organization for the Advancement of Structured Information Standards (OASIS), 2 February 2009. (See http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.doc.)

**WS-Trust**

*OASIS Standard, "Web Services Trust (WS-Trust) 1.4"* , K. Lawrence, C. Kaler, Editors. Organization for the Advancement of Structured Information Standards

(OASIS), 2 February 2009. (See http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc.)

**XML Infoset**

*W3C Recommendation, "XML Information Set (Second Edition)"* , J. Cowan, R. Tobin, Editors. World Wide Web Consortium (W3C), 4 February 2004. (See http://www.w3.org/TR/xml-infoset.)

**XMLSchema - Part 1**

*W3C Recommendation, "XML Schema Part 1: Structures (Second Edition)"* , H. Thompson, et al., Editors. World Wide Web Consortium (W3C), 28 October 2004. (See http://www.w3.org/TR/xmlschema-1/.)

**XMLSchema - Part 2**

*W3C Recommendation, "XML Schema Part 2: Datatypes (Second Edition)"* , P. Biron, A. Malhotra, Editors. World Wide Web Consortium (W3C), 28 October 2004. (See http://www.w3.org/TR/xmlschema-2/.)

**XPath1.0**

*W3C Recommendation, "XML Path Language (XPath) Version 1.0"* , J. Clark, S. DeRose, Editors. World Wide Web Consortium (W3C), 16 November 1999. (See http://www.w3.org/TR/xpath.)

**XPath2.0**

*W3C Recommendation, "XML Path Language (XPath) 2.0"* , A. Berglund, et al., Editors. World Wide Web Consortium (W3C), 23 January 2007. (See http://www.w3.org/TR/xpath20/.)

# A Advertising Event Information

There are many use cases for WS-Eventing in which it is necessary for the subscriber and the event sink to know the structure and contents of the notifications that can result from a successful Subscribe request. For example, a developer might wish to use WSDL-based tools to generate service stubs capable of unmarshalling and dispatching notifications. In addition to this, the effective use of filters (including those in the XPath dialect defined in **4.1 Subscribe** as well as other dialects not defined in this specification) requires some knowledge of the schema of the events over which the filter will be applied.

There are many ways in which an event source could describe and advertise the structure of the events for which it will issue notifications. To provide a basic level of interoperability, this specification uses the following two OPTIONAL mechanisms, in **A.1 Event Types & Event Descriptions** and **A.2 Notification WSDLs**, for describing and advertising event information. If an implementation of a WS-Eventing event source chooses to describe the structure of its events and advertise this description, it is RECOMMENDED that at least one of these mechanisms be used. Mechanisms other than these MAY be used to describe and advertise the structure of events, but the definition of such mechanisms is out of the scope of this specification.

## A.1 Event Types & Event Descriptions

An event source MAY advertise the events that are generated by making available an Event Description document as defined by WS-EventDescriptions [WS-EventDescriptions].

The following is an example of an EventDescriptions element that could serve as a description of the Event Type used in Example 2-1.

**Example 1-1: EventDescriptions**

```
(01) <wsevd:EventDescriptions
(02)
targetNamespace="http://www.example.org/oceanwatch/notifications"
(03)        xmlns:wsevd="http://www.w3.org/2010/08/ws-evt"
(04)        xmlns:ow="http://www.example.org/oceanwatch">
(05)   <wsevd:types>
(06)     <xs:schema targetNamespace="http://www.example.org/oceanwatch">
(07)       <xs:include
schemaLocation="http://www.example.org/schemas/oceanwatch.xsd"/>
(08)       <xs:element name="WindReport" type="ow:WindReportType"/>
(09)     </xs:schema>
(10)   </wsevd:types>
(11)
(12)   <wsevd:eventType name="WindReportEvent"
(13)                 element="ow:WindReport"
(14)
actionURI="http://www.example.org/oceanwatch/2003/WindReport"/>
(15) </wsevd:EventDescriptions>
```

Lines (12-14) describe an Event Type with a QName of "{http://www.example.org/oceanwatch/notifications}:WindReportEvent". The GED for this Event Type is defined on line (08) as being of type "{http://www.example.org/oceanwatch}:WindReportType".

### A.1.1 Retrieving Event Descriptions

Although there are many ways in which an event source can make its EventDescriptions available, this specification RECOMMENDS the use of the mechanisms described in WS-MetadataExchange [WS-MetadataExchange]. In particular, this specification RECOMMENDS that the Event Description metadata be made available through the EventSource Policy assertion. This MAY be done by either embedding the Event Description metadata directly within the assertion, or by including a MetadataExchange reference to the data.

The value of the @Identifier attribute, if present, for this MetadataSection MUST be equal to the value of its wsevd:EventDescriptions/@targetNamespace. An event source MUST NOT have more than one EventDescriptions document.

The following examples show how Event Description metadata might appear within a WS-Eventing EventSource Policy assertion.

**Example 1-2: Sample Embedded EventDescription Metadata**

```
(01) <wse:EventSource ...>
(02)   <wse:FormatName URI="..."/>
(03)   <wsevd:EventDescriptions ...>
(04)     ...
(05)   </wsevd:EventDescriptions>
(06) </wse:EventSource>
```

Example 1-2 shows how the Event Description metadata might be embedded directly within a WS-Eventing EventSource Policy assertion.

**Example 1-3: Sample Reference to EventDescription Metadata**

```
(01) <wse:EventSource ...>
(02)   <wse:FormatName URI="..."/>
(03)   <mex:Location
(04)     Type="evd:EventDescriptions"
(05)     URI="http://example.com/EVD_Metadata" />
(06) </wse:EventSource>
```

Example 1-3 shows the same policy assertion from Example 1-2 except the embedded Event Description metadata is replaced with a reference to an HTTP resource whose representation is the Event Description metadata. The data can be retrieved via an HTTP GET to the specified URL.

### A.1.2 Bindings for Event Descriptions

For any Notification Format it MUST be possible to determine how a given wsevd:eventType will appear on the wire as a notification in a subscription created with that format. The following sections define how wsevd:eventTypes bind to notifications for the two Notification Formats defined in this specification; Unwrapped and Wrapped. Specifications or profiles that define additional Notification Formats MUST define how wsevd:eventTypes bind to the notifications for those formats. In the absence of a mapping for a particular Notification Format, implementations MAY provide a Notification WSDL (see below) that explicitly describes the notification operations.

#### *A.1.2.1 Binding for Unwrapped Notifications*

The information about an Event Type contained in the wsevd:eventType element binds to a Unwrapped Notification for that type as follows:

- The **[Action]** property of the notification has the value of the @actionURI attribute of the wsevd:eventType element corresponding to the type of the event being transmitted.
- The **[Body]** property of the notification has a single child element. This child element is an instance of the Global Element Declaration referenced by the @element attribute of the wsevd:eventType element corresponding to the type of

the event being transmitted. If the @element attribute is absent then the **[Body]** property has no children.

#### *A.1.2.2 Binding for Wrapped Notifications*

The information about an Event Type contained in the eventType element binds to a Wrapped Notification for that type as follows:

- The /soap:Envelope/soap:Body/wse:Notify/@actionURI attribute of the Wrapped Notification has the value of the @actionURI attribute of the eventType element corresponding to the type of the event being transmitted.
- The /soap:Envelope/soap:Body/wse:Notify element has a single child element. This child element is an instance of the Global Element Declaration referenced by the @element attribute of the eventType element corresponding to the type of the event being transmitted. If the @element attribute is absent then the wse:Notify element has no children.

## A.2 Notification WSDLs

As described previously, events are transmitted to event sinks as SOAP messages called "Notifications". These notifications MAY be described via a Web Services Description Language [WSDL11] definitions element termed a "Notification WSDL". A Notification WSDL describes the interface that the event sink is REQUIRED to implement to receive and process the notifications that might result from a successful Subscribe request that used a particular Format IRI. The following is an example of a Notification WSDL:

**Example 1-4: Notification WSDL**

```
(01) <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
(02)
targetNamespace="http://www.example.org/oceanwatch/notifications"
(03)                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
(04)                   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
(05)
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
(06)                   xmlns:ow="http://www.example.org/oceanwatch"
(07)
xmlns:tns="http://www.example.org/oceanwatch/notifications">
(08)   <wsdl:types>
(09)     <xs:schema targetNamespace="http://www.example.org/oceanwatch">
(10)       <xs:include
schemaLocation="http://www.example.org/schemas/oceanwatch.xsd"/>
(11)       <xs:element name="WindReport" type="ow:WindReportType"/>
(12)     </xs:schema>
(13)   </wsdl:types>
(14)
(15)   <wsdl:message name="WindReportNotificationMsg">
(16)     <wsdl:part name="event" element="ow:WindReport"/>
(17)   </wsdl:message>
```

```
(18)
(19)   <wsdl:portType name="WindReportPortType">
(20)     <wsdl:operation name="WindReportNotificationOp">
(21)       <wsdl:input message="tns:WindReportNotificationMsg"
(22)
wsam:Action="http://www.example.org/oceanwatch/2003/WindReport"/>
(23)     </wsdl:operation>
(24)   </wsdl:portType>
(25)
(26)   <wsdl:binding name="WindReportBinding"
type="tns:WindReportPortType">
(27)     <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
(28)     <wsdl:operation name="WindReportNotificationOp">
(29)       <soap:operation soapAction=""/>
(30)       <wsdl:input>
(31)         <soap:body use="literal"/>
(32)       </wsdl:input>
(33)     </wsdl:operation>
(34)   </wsdl:binding>
(35) </wsdl:definitions>
```

### A.2.1 Retrieving Notification WSDLs

Although there are many ways in which an event source can make Notification WSDLs available, this specification RECOMMENDS the use of the mechanisms described in WS-MetadataExchange [WS-MetadataExchange]. In particular, if an event source has Notification WSDL then it SHOULD be referenced from the wse:EventSource policy assertion.

## B XML Schema

A normative copy of the XML Schema [XMLSchema - Part 1], [XMLSchema - Part 2] description for this specification can be retrieved from the following address:

```
http://www.w3.org/2010/08/ws-evt/eventing.xsd
```

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

```
<xs:schema
  targetNamespace='http://www.w3.org/2010/08/ws-evt'
  xmlns:tns='http://www.w3.org/2010/08/ws-evt'
  xmlns:wsa='http://www.w3.org/2005/08/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://www.w3.org/2005/08/addressing'
    schemaLocation='http://www.w3.org/2005/08/addressing/ws-addr.xsd' />

  <!-- Types and global elements -->
  <xs:complexType name='DeliveryType' mixed='true'>
    <xs:sequence>
      <xs:element ref='tns:NotifyTo' minOccurs='0' maxOccurs='1' />
      <xs:any namespace='##other' processContents='lax'
              minOccurs='0' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:anyAttribute namespace='##other' processContents='lax' />
  </xs:complexType>

  <xs:complexType name='FormatType' mixed='true'>
    <xs:sequence>
      <xs:any namespace='##any' processContents='lax'
              minOccurs='0' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:attribute name='Name' type='xs:anyURI' use='optional'
      default='http://www.w3.org/2010/08/ws-evt/DeliveryFormats/Unwrap' />
    <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='DurationDateTime'>
      <xs:union memberTypes='xs:dateTime tns:NonNegativeDurationType' />
  </xs:simpleType>

  <xs:complexType name='MiniExpirationType'>
    <xs:simpleContent>
      <xs:extension base='tns:DurationDateTime'>
        <xs:anyAttribute namespace='##other' processContents='lax'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name='ExpirationType'>
    <xs:simpleContent>
      <xs:extension base='tns:MiniExpirationType'>
        <xs:attribute name='BestEffort' type='xs:boolean' use='optional'/>
        <xs:anyAttribute namespace='##other' processContents='lax'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <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'
      default='http://www.w3.org/2010/08/ws-evt/Dialects/XPath10' />
    <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' />

  <xs:complexType name='NotificationPolicy'>
    <xs:sequence>
      <xs:any namespace='##other' processContents='lax'
              maxOccurs='unbounded'/>
    </xs:sequence>
  </xs:complexType>

  <!-- 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='Format' type='tns:FormatType'
                    minOccurs='0' />
        <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>

  <!-- Subscribe response -->
  <xs:element name='SubscribeResponse'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='SubscriptionManager'
                    type='wsa:EndpointReferenceType' />
        <xs:element name='GrantedExpires' type='tns:MiniExpirationType' />
        <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 format name -->
  <xs:element name='SupportedDeliveryFormat' 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='GrantedExpires' type='tns:MiniExpirationType'
                    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='GrantedExpires' type='tns:MiniExpirationType'
                    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>

  <!-- Unsubscribe response -->
  <xs:element name='UnsubscribeResponse'>
    <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>

  <!-- SubscriptionEnd message -->
  <xs:element name='SubscriptionEnd'>
    <xs:complexType>
      <xs:sequence>
        <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://www.w3.org/2010/08/ws-evt/DeliveryFailure' />
      <xs:enumeration value=
  'http://www.w3.org/2010/08/ws-evt/SourceShuttingDown' />
      <xs:enumeration value=
  'http://www.w3.org/2010/08/ws-evt/SourceCancelling' />
    </xs:restriction>
  </xs:simpleType>

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

  <!-- RetryAfter Fault Detail Element -->
```

```
  <xs:element name='RetryAfter'  type='tns:RetryAfterType'/>
  <xs:complexType name='RetryAfterType'>
    <xs:simpleContent>
      <xs:extension base='xs:nonNegativeInteger'>
        <xs:anyAttribute namespace='##other' processContents='lax' />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <!-- Wrapped Events -->
  <xs:complexType name='EventType' mixed='true'>
    <xs:sequence>
      <xs:any namespace='##any' processContents='lax' minOccurs='0'
              maxOccurs='unbounded'/>
    </xs:sequence>
    <xs:attribute name='actionURI' type='xs:anyURI' use='optional' />
    <xs:anyAttribute namespace='##other' processContents='lax' />
  </xs:complexType>

  <xs:element name='Notify' type='tns:EventType' />

  <!-- Policy -->
  <xs:complexType name='Duration'>
    <xs:simpleContent>
      <xs:extension base='tns:NonNegativeDurationType'>
       <xs:anyAttribute namespace='##other' processContents='lax'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name='URI'>
    <xs:simpleContent>
      <xs:extension base='xs:anyURI'>
        <xs:anyAttribute namespace='##other' processContents='lax'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name='Empty'>
    <xs:sequence/>
    <xs:anyAttribute namespace='##other' processContents='lax'/>
  </xs:complexType>

  <xs:element name='EventSource'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='FilterDialect' minOccurs='0'
maxOccurs='unbounded'>
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace='##other' processContents='lax'
minOccurs='0'
                                          maxOccurs='0'/>
            </xs:sequence>
            <xs:attribute name='URI' type='xs:anyURI' use='required' />
            <xs:anyAttribute namespace="##other" processContents='lax'/>
```

```
          </xs:complexType>
        </xs:element>
        <xs:element name='FormatName' minOccurs='0' maxOccurs='unbounded'>
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace='##other' processContents='lax'
minOccurs='0'
                                          maxOccurs='0'/>
            </xs:sequence>
            <xs:attribute name='URI' type='xs:anyURI' use='required' />
            <xs:anyAttribute namespace="##other" processContents='lax'/>
          </xs:complexType>
        </xs:element>
        <xs:element name='ExpiresSupported' type='tns:Empty'
minOccurs='0'/>
        <xs:element name='DateTimeSupported' type='tns:Empty'
minOccurs='0'/>
        <xs:element name='MaxExpires' type='tns:Duration' minOccurs='0'/>
        <xs:element name='EndToSupported' type='tns:Empty' minOccurs='0'/>
        <xs:element name='NotificationPolicy' type='tns:NotificationPolicy'
                                              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='SubscriptionManager'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='ExpiresSupported' type='tns:Empty'
minOccurs='0'/>
        <xs:element name='DateTimeSupported' type='tns:Empty'
minOccurs='0'/>
        <xs:element name='MaxExpires' type='tns:Duration' 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:schema>
```

## C WSDL

A normative copy of the WSDL [WSDL11] description can be retrieved from the following address:

```
http://www.w3.org/2010/08/ws-evt/eventing.wsdl
```

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

```
<wsdl:definitions
  targetNamespace='http://www.w3.org/2010/08/ws-evt'
  xmlns:wsa='http://www.w3.org/2005/08/addressing'
  xmlns:wsam='http://www.w3.org/2007/05/addressing/metadata'
  xmlns:wse='http://www.w3.org/2010/08/ws-evt'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns:xs='http://www.w3.org/2001/XMLSchema' >

  <wsdl:types>
    <xs:schema>
       <xs:import
         namespace='http://www.w3.org/2010/08/ws-evt'
         schemaLocation=
  'http://www.w3.org/2010/08/ws-evt/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:part name='body' element='wse:UnsubscribeResponse' />
  </wsdl:message>

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

  <wsdl:message name='notifyEvent'>
    <wsdl:part name='parameter' element='wse:Notify'/>
  </wsdl:message>

  <wsdl:portType name='EventSource' >
    <wsdl:operation name='SubscribeOp' >
      <wsdl:input
```

```
        message='wse:SubscribeMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/Subscribe'/>
      <wsdl:output
        message='wse:SubscribeResponseMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/SubscribeResponse'/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:portType name='SubscriptionEndPortType' >
    <wsdl:operation name='SubscriptionEnd' >
      <wsdl:input
        message='wse:SubscriptionEnd'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/SubscriptionEnd'/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:portType name='SubscriptionManager' >
    <wsdl:operation name='RenewOp' >
      <wsdl:input
        message='wse:RenewMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/Renew'/>
      <wsdl:output
        message='wse:RenewResponseMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/RenewResponse'/>
    </wsdl:operation>
    <wsdl:operation name='GetStatusOp' >
      <wsdl:input
        message='wse:GetStatusMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/GetStatus'/>
      <wsdl:output
        message='wse:GetStatusResponseMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/GetStatusResponse'/>
    </wsdl:operation>
    <wsdl:operation name='UnsubscribeOp' >
      <wsdl:input
        message='wse:UnsubscribeMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-evt/Unsubscribe'/>
      <wsdl:output
        message='wse:UnsubscribeResponseMsg'
        wsam:Action='http://www.w3.org/2010/08/ws-
evt/UnsubscribeResponse'/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:portType name='WrappedSinkPortType'>
    <wsdl:operation name='NotifyEvent'>
      <wsdl:input message='wse:notifyEvent' name='NotifyEvent'
        wsam:Action='http://www.w3.org/2010/08/ws-
evt/WrappedSinkPortType/NotifyEvent'/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

## D WSDL for Standard Wrapped Delivery

If an event subscriber specifies the wrapped event delivery format http://www.w3.org/2010/08/ws-evt/DeliveryFormats/Wrap in the Subscribe request message, then the event sink MUST implement the following abstract WSDL and notification messages MUST be wrapped in the element defined in the WSDL.

```
<definitions
  xmlns='http://schemas.xmlsoap.org/wsdl/'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  xmlns:wsam='http://www.w3.org/2007/05/addressing/metadata'
  xmlns:wsa='http://www.w3.org/2005/08/addressing/'
  xmlns:tns='http://www.w3.org/2010/08/ws-evt'
  targetNamespace='http://www.w3.org/2010/08/ws-evt'>

  <types>
    <xs:schema
        targetNamespace='http://www.w3.org/2010/08/ws-evt'>

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

      <xs:element name='Notify' type='tns:EventType' />
    </xs:schema>
  </types>

  <message name='notifyEvent'>
    <part name='parameter' element='tns:Notify'/>
  </message>

  <portType name='WrappedSinkPortType'>
    <operation name='NotifyEvent'>
      <input message='tns:notifyEvent' name='NotifyEvent'
        wsam:Action='http://www.w3.org/2010/08/ws-
evt/WrappedSinkPortType/NotifyEvent'/>
    </operation>
  </portType>
</definitions>
```

# E Action Tables

The purpose of the action tables is to illustrate, via a separate means from the normative text, the allowable order and interactions of various messages and activities. The action tables are not intended to constrain implementations beyond those necessary to insure this ordering.

- Actions are represented as columns.

- Triggers (messages, application actions, timer events) are represented as rows. Triggers are annotated by their type; "[app]" - represents an application action (e.g. a user selecting a "Subscribe" menu item); "[msg]" - represents an incoming, WS-Eventing defined, message; "[timer]" - represents an internal timer event.
- Each cell describes the appropriate action for a given state and trigger. Where the action is dependent upon other factors than the state and trigger (e.g. the value of a fault message), the activity is described in pseudo-code. The section of the specification that describes these activities is displayed in curly brackets (e.g. "{4.2}").
- Empty box indicates that the spec is silent for the specified trigger/action pair.

Table 5-1: Event Source Action Table

| Trigger | Action |
|---|---|
| Subscribe Request [msg] | Create subscription<br>Send SubscribeResponse + Subscription Manager EPR<br>{**4.1 Subscribe**} |
| Event [app] | Send notification to all pertinent subscriptions<br>{**5 Notifications**} |

Table 5-2: Subscription Manager Action Table

| Trigger | Action | |
|---|---|---|
| | **Trigger refers to a valid subscription** | **Trigger refers to an invalid subscription** |
| Renew Request [msg] | Update expiration timer<br>Send RenewResponse<br>{**4.2 Renew**} | Generate UnknownSubscription Fault<br>{**4.2 Renew**} |
| GetStatus Request [msg] | Send GetStatusResponse<br>{**4.3 GetStatus**} | Generate UnknownSubscription Fault<br>{**4.3 GetStatus**} |
| Unsubscribe Request [msg] | Send UnsubscribeResponse<br>{**4.4 Unsubscribe**} | Generate UnknownSubscription Fault<br>{**4.4 Unsubscribe**} |
| Expiration [timer] | If (EndTo engaged)<br>  Send SubscriptionEnd<br>Invalidate subscription<br>{**4.1 Subscribe**} | |
| Shutdown/Error [app] | If (EndTo engaged)<br>  Send SubscriptionEnd<br>Invalidate subscription<br>{**4.1 Subscribe**} | |

Table 5-3: Event Sink Action Table

| Trigger | Action |
|---|---|
| Expiration [timer] | Invalidate subscription<br>{**4.1 Subscribe**} |
| SubscriptionEnd [msg] | Invalidate subscription<br>{**4.5 SubscriptionEnd**} |

# F Change Log

| Data | Author | Description |
|---|---|---|
| 2009/03/04 | DD | Added resolution of issue 6391 |
| 2009/03/04 | DD | Added resolution of issue 6519 |
| 2009/03/04 | DD | Added resolution of issue 6427 |
| 2009/03/04 | DD | Added resolution of issue 6459 |
| 2009/03/09 | DD | Added resolution of issue 6397 |
| 2009/03/09 | DD | Added resolution of issue 6426 |
| 2009/03/11 | DD | Added change log |
| 2009/03/11 | DD | Added resolution of issue 6641 |
| 2009/03/11 | DD | Added resolution of issue 6498 |
| 2009/03/11 | DD | Added resolution of issue 6425 |
| 2009/03/16 | KW | Added resolution of issue 6587 |
| 2009/03/17 | KW | Added resolution of issue 6400 |
| 2009/03/17 | DD | Added resolution of issue 6428 |
| 2009/03/17 | DD | Added resolution of issue 6687 |
| 2009/03/23 | DD | Added resolution of issue 6666 |
| 2009/03/23 | DD | Added resolution of issue 6681 |
| 2009/03/24 | DD | Added resolution of issue 6595 |
| 2009/03/24 | DD | Added resolution of issue 6648 |
| 2009/04/07 | DD | Added resolution of issue 6727 |
| 2009/04/07 | DD | Added resolution of issue 6725 |
| 2009/04/07 | DD | Added resolution of issue 6715 |
| 2009/04/22 | KW | Added resolution of issue 6739 |
| 2009/04/28 | DD | Added resolution of issue 6787 |
| 2009/04/28 | DD | Added resolution of issue 6788 |
| 2009/05/13 | KW | Added resolution of issue 6472 |
| 2009/05/13 | DD | Added resolution of issue 6850 |
| 2009/05/13 | DD | Added resolution of issue 6696 |
| 2009/05/19 | DD | Added resolution of issue 6907 |
| 2009/05/19 | DD | Added resolution of issue 6429 |
| 2009/05/21 | DD | Added resolution of issue 6674 |
| 2009/05/27 | DD | Added resolution of issue 6906 |
| 2009/06/04 | DD | Added resolution of issue 6955 |

| | | |
|---|---|---|
| 2009/06/04 | DD | Added resolution of issue 6916 |
| 2009/06/04 | DD | Added resolution of issue 6986 |
| 2009/07/07 | DD | Added resolution of issue 7039 |
| 2009/07/21 | DD | Added resolution of issue 6980 |
| 2009/07/28 | DD | Added resolution of issue 6692 |
| 2009/08/04 | DD | Added resolution of issue 7136 |
| 2009/08/05 | DD | Added resolution of issue 6432 |
| 2009/08/05 | DD | Added resolution of issue 7159 |
| 2009/08/05 | DD | Added resolution of issue 7205 |
| 2009/08/18 | DD | Added resolution of issue 7206 |
| 2009/08/25 | DD | Added resolution of issue 7365 |
| 2009/08/25 | DD | Added resolution of issue 7270 |
| 2009/08/25 | DD | Added resolution of issue 7235 |
| 2009/08/26 | DD | Added resolution of issue 7160 |
| 2009/09/01 | DD | Added resolution of issue 6700 |
| 2009/09/02 | DD | Added resolution of issue 6694 |
| 2009/09/02 | DD | Added resolution of issue 6401 |
| 2009/09/02 | DD | Added resolution of issue 6533 |
| 2009/09/22 | DD | Added resolution of issue 7698 |
| 2009/09/23 | DD | Added resolution of issue 6569 |
| 2009/10/01 | DD | Added resolution of issue 7589 |
| 2009/10/02 | DD | Added resolution of issue 7426 |
| 2009/10/02 | DD | Added resolution of issue 7554 |
| 2009/10/05 | DD | Added resolution of issues 6402, 6721 |
| 2009/10/06 | DD | Added resolution of issue 7478 |
| 2009/10/13 | DD | Added resolution of issue 7827 |
| 2009/10/20 | DD | Added resolution of issue 7982 |
| 2009/10/20 | DD | Added resolution of issue 7068 |
| 2009/10/20 | DD | Added resolution of issue 7553 |
| 2009/10/20 | DD | Added resolution of issue 7207 |
| 2009/10/27 | DD | Added resolution of issues 7586, 7588, 7828 |
| 2009/11/05 | DD | Added resolution of issue 8076 |
| 2009/11/05 | DD | Added resolution of issue 7912 |
| 2009/11/06 | DD | Added resolution of issue 8172 |
| 2009/11/06 | DD | Added resolution of issue 8168 |

| | | |
|---|---|---|
| 2009/11/06 | DD | Added resolution of issue 7970 |
| 2009/11/06 | DD | Added resolution of issue 8213 |
| 2009/11/06 | DD | Added resolution of issue 8170 |
| 2009/11/06 | DD | Added resolution of issue 8162 |
| 2009/11/06 | DD | Added resolution of issue 8163 |
| 2009/11/06 | DD | Added resolution of issue 8166 |
| 2009/11/06 | DD | Added resolution of issue 8167 |
| 2009/11/06 | DD | Added resolution of issue 8169 |
| 2009/11/06 | DD | Added resolution of issue 8124 |
| 2009/11/17 | DD | Added resolution of issue 8280 |
| 2009/11/17 | DD | Added resolution of issue 8276 |
| 2009/11/17 | DD | Added resolution of issue 8277 |
| 2009/11/17 | DD | Added resolution of issue 8285 |
| 2009/12/01 | DD | Added resolution of issue 8201 |
| 2009/12/08 | DD | Added resolution of issue 8287 |
| 2010/01/05 | DD | Added resolution of issue 8165 |
| 2010/01/12 | DD | Added resolution of issue 8164 |
| 2010/01/12 | DD | Added resolution of issue 8281 |
| 2010/01/12 | DD | Added resolution of issue 8176 |
| 2010/01/19 | DD | Added resolution of issue 8068 |
| 2010/01/19 | DD | Added resolution of issue 8286 |
| 2010/01/19 | DD | Added resolution of issue 8283 |
| 2010/01/26 | DD | Added resolution of issue 8288 |
| 2010/01/26 | DD | Added resolution of issue 8275 |
| 2010/01/27 | DD | Added resolution of issue 7986 |
| 2010/01/28 | DD | Added resolution of issue 8196 |
| 2010/02/05 | DD | Added resolution of issue 6435 |
| 2010/02/08 | DD | Added resolution of issue 8901 |
| 2010/02/09 | DD | Added resolution of issue 8160 |
| 2010/02/09 | DD | Added resolution of issue 8271 |
| 2010/03/09 | DD | Added resolution of issue 6463, 8031, 8198 |
| 2010/03/16 | DD | Added resolution of issue 8886 |
| 2010/03/30 | DD | Added resolution of issue 9320 |
| 2010/03/30 | DD | Added resolution of issue 9031 |
| 2010/03/30 | DD | Added resolution of issue 9266 |

| | | |
|---|---|---|
| 2010/04/20 | DD | Added resolution of issue 9321 |
| 2010/04/20 | DD | Added resolution of issue 9543 |
| 2010/04/27 | DD | Added resolution of issue 8832 |
| 2010/05/04 | DD | Added resolution of issue 9588 |
| 2010/05/11 | DD | Added resolution of issue 9567 |
| 2010/05/11 | DD | Added resolution of issue 9613 |
| 2010/05/11 | DD | Added resolution of issue 9699 |
| 2010/05/11 | DD | Added resolution of issue 9676 |
| 2010/05/12 | DD | Added resolution of issue 9712 |
| 2010/05/12 | DD | Added resolution of issue 9713 |
| 2010/05/12 | DD | Added resolution of issue 9675 |
| 2010/05/12 | DD | Added resolution of issue 9717 |
| 2010/05/12 | DD | Added resolution of issue 9701 |
| 2010/05/13 | DD | Added resolution of issue 9702 |
| 2010/05/13 | DD | Added resolution of issue 9610 |
| 2010/06/15 | DD | Added resolution of issue 9610 |