WS-Eventing Interoperability Scenario

Cetacean Tracking System

Version: 0.0.2
Date: October 29, 2010
Editor: Gilbert Pilz

Abstract

The following scenario is designed to provide a framework in which to test the interoperability of various WS-Eventing implementations. Because this scenario and the tests defined within it will be used to judge which features of WS-Eventing are implemented and which are not, the feature coverage is intended to be complete.

Timeline

Start	End	Activity

Table of Contents

1 Dependencies

1.1 Scope

The following specifications and technologies are in scope for this scenario:

- SOAP 1.1
- WS-Eventing
- WS-EventDescriptions
- WS-MakeConnection
- WS-Policy
- WSDL 1.1

1.2 Namespaces

The following table defines the namespaces used in this document:

Prefix	Namespace	Specification
xsd	http://www.w3.org/2001/XMLSchema	XML Schema
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL 1.1
soap11	http://schemas.xmlsoap.org/soap/envelope/	SOAP 1.1
wsoap11	http://schemas.xmlsoap.org/wsdl/soap/	WSDL 1.1
wsa	http://www.w3.org/2005/08/addressing	WS-Addressing 1.0
wse	http://www.w3.org/2010/08/ws-evt	WS-Eventing
gpx	http://www.topografix.com/GPX/1/1	GPS eXchange Format

1.3 Preconditions

2 Scenario Description

This scenario presupposes a cetacean tracking system in which a number of animals have been “tagged” with devices that track their location. These tags periodically communicate via satellite to a central system. External systems can consume this information by using WS-Eventing to subscribe to periodic notifications about the locations of the tags and, presumably, the animals they are attached to.

2.1 Event Description

The location of the tags is expressed in GPS coordinates using the GPS eXchange Format, an XML schema designed as a common GPS data format for software applications. In addition to the basic GPS information (latitude, longitude, elevation, and time), the notifications include an ID that uniquely identifies the tag and, by inference, the animal that the tag is attached to. An EventDescriptions document that describes the structure of the event information within the notifications can be found in Section 5.

2.2 Event Timing

While in the real world the frequency of notifications might be hourly or even daily, for the sake of feasibility we compress time by a scale of 1/120 so that one hour in “scenario time” is thirty seconds in real-world time. The time data contained in the notifications will reflect scenario time.

2.3 Tags

Again for the sake of feasibility, this scenario will only include three tags with the following IDs:

- 13c76450-de3d-11df-85ca-0800200c9a66 (Howard)
- 234b6840-de3d-11df-85ca-0800200c9a66 (Kerry)
- 32675b90-de3d-11df-85ca-0800200c9a66 (Oscar)

3 Tests

The following tests are designed to exercise all the mandatory and optional features of WS-Eventing except for those (such as the use of EventDescriptions or Notification WSDLs) that only affect the process of developing one or more of the components of a WS-Eventing-based system. Although the success criteria for some tests include the transmission of a specific fault, in general it is not feasible to test every fault defined by WS-Eventing.

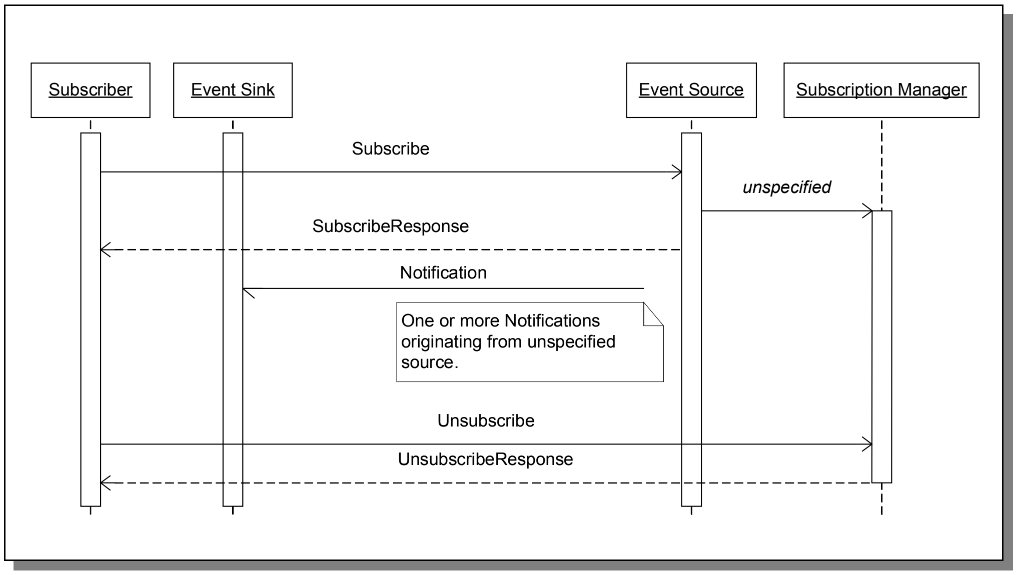
3.1 Basic Test

This test verifies the ability to subscribe and receive notifications. The initial subscription has the following features:

- expiration time chose by Event Source/Subscription Manager
- no EndTo EPR
- no Filters
- unwrapped notifications

Sequence

The following diagram illustrates the sequence of messages for the Basic Test.



Success Criteria

1. Receipt of a valid Subscribe message by the Event Source.
2. Receipt of a valid SubscribeResponse message by Subscriber.
3. Receipt of one or more unwrapped Notifications by the Event Sink.
4. Receipt of a valid Unsubscribe message by the Subscription Manager.
5. Receipt of a valid UnsubscribeResponse message by the Subscriber.

3.2 Wrapped Notifications

This test verifies the simple ability to subscribe and receive wrapped notifications. The initial subscription has the following features:

- expiration time chosen by Event Source/Subscription Manager
- no EndTo EPR
- no Filters
- wrapped notifications

Sequence

The messaging sequence for this test is identical to that of the Basic Test.

Success Criteria

1. Receipt of a valid Subscribe message by the Event Source.
2. Receipt of a valid SubscribeResponse message by Subscriber.
3. Receipt of one or more wrapped Notifications by the Event Sink.
4. Receipt of a valid Unsubscribe message by the Subscription Manager.

5. Receipt of a valid UnsubscribeResponse message by the Subscriber.

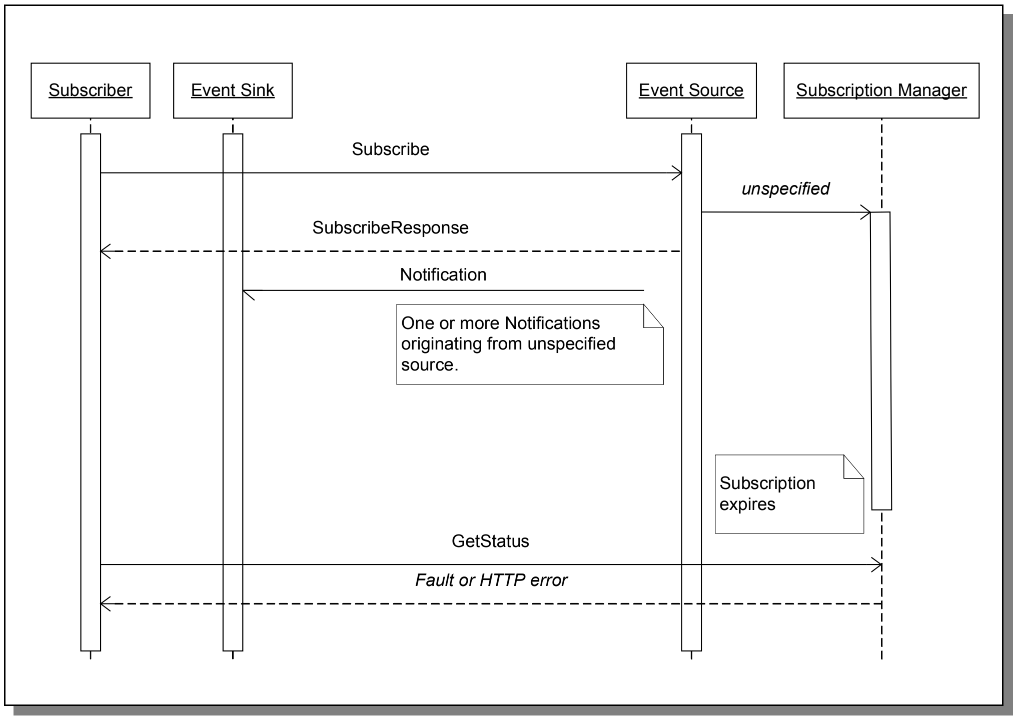
3.3 *Duration Expiration Test*

This test verifies the correct implementation of the expiration feature on the Event Source/Subscription Manager. The initial subscription has the following features:

- (short) expiration time chosen by Subscriber as `xs:duration`
- no EndTo EPR
- no Filters
- unwrapped notifications

Sequence

The following diagram illustrates the sequence of messages for the Duration Expiration Test. Note that the Subscriber waits until the expiration time has passed before sending the GetStatus request.



Success Criteria

1. Receipt of a valid Subscribe message by the Event Source.
2. Receipt of a valid SubscribeResponse message by Subscriber.
3. Receipt of one or more unwrapped Notifications by the Event Sink.
4. Receipt of a valid GetStatus message by the Subscription Manager.

5. Receipt, by the Subscriber, of either the “UnknownSubscription” fault (defined by Section 6.10 of WS-Eventing), a SOAP fault that indicates that the Subscription Manager no longer exists, or an HTTP error (i.e. “404”) that indicates the Subscription Manager no longer exists

3.4 *Specific Time Expiration Test*

This test verifies the correct implementation of the expiration feature on the Event Source/Subscription Manager. The initial subscription has the following features:

- (short) expiration time chosen by Subscriber as `xs:dateTime`
- no EndTo EPR
- no Filters
- unwrapped notifications

Sequence

The messaging sequence for this test is identical to that of the Duration Expiration Test.

Success Criteria

1. Receipt of a valid Subscribe message by the Event Source.
2. Receipt of a valid SubscribeResponse message by Subscriber.
3. Receipt of one or more unwrapped Notifications by the Event Sink.
4. Receipt of a valid GetStatus message by the Subscription Manager.
5. Receipt, by the Subscriber, of either the “UnknownSubscription” fault (defined by Section 6.10 of WS-Eventing), a SOAP fault that indicates that the Subscription Manager no longer exists, or an HTTP error (i.e. “404”) that indicates the Subscription Manager no longer exists.

3.5 *Best Effort Expiration Test*

This test verifies the correct implementation of the “best effort” expiration feature on the Event Source/Subscription Manager. The initial subscription has the following features:

- expiration time chosen by Subscriber as `xs:duration` with @BestEffort ='true'
- no EndTo EPR
- no Filters
- unwrapped notifications

Sequence

The messaging sequence for this test is identical to that of the Duration Expiration Test.

Success Criteria

1. Receipt of a valid Subscribe message by the Event Source.
2. Receipt of a valid SubscribeResponse message by Subscriber.
3. Receipt of one or more unwrapped Notifications by the Event Sink.
4. Receipt of a valid GetStatus message by the Subscription Manager.
5. Receipt, by the Subscriber, of either the “UnknownSubscription” fault (defined by Section 6.10 of WS-Eventing), a SOAP fault that indicates that the Subscription Manager no longer exists, or an HTTP error (i.e. “404”) that indicates the Subscription Manager no longer exists.

3.6 Renew Test

This test verifies the ability of a Subscriber to update the expiration time of a Subscription via a Renew request. The initial subscription has the following features:

- (short) expiration time chosen by Subscriber as `xs:duration`
- no EndTo EPR
- no Filters
- unwrapped notifications

Sequence

The following diagram illustrates the sequence of messages for the Renew Test.

TBD

Success Criteria

3.7 SubscriptionEnd Test

This test verifies the correct implementation of the SubscriptionEnd feature for both the Subscription Manager and the target of the SubscriptionEnd message. The initial subscription has the following features:

- expiration time chosen by Event Source/Subscription Manager
- EndTo EPR
- no Filters
- unwrapped notifications

TBD

3.8 Filter Test

This test verifies the ability of the Event Source/Subscription Manager to correctly implement simple, XPATH filters. The initial subscription has the following features:

- expiration time chosen by Event Source/Subscription Manager
- no EndTo EPR
- Filter that selects only those events that apply to tag 234b6840-de3d-11df-85ca-0800200c9a66 (Helen)
- unwrapped notifications

TBD

3.9 Non-Addressable Event Sink Test

This test verifies the ability to subscribe and receive notifications in an environment in which the Event Sink cannot accept connections from systems outside its network (i.e. the Event Sink is non-addressable).

4 WSDLs

4.1 Event Source WSDL

TBD

4.2 Notification WSDL

TBD

5 EventDescriptions

TBD

6 Schemas

TBD

7 Change Log

Version	Date	Author	Changed
Initial	2010-10-26	Gilbert Pilz	Initial revision
0.0.1	2010-10-27	Gilbert Pilz	Fleshed out Basic, Wrapped, and Expiration tests; added sequence diagrams. Added stubs

Version	Date	Author	Changed
			for Renew and Non-Addressable Event Sink tests.
0.0.2	2010-10-28	Gilbert Pilz	Editorial fixes. Changed animal names in honor of the Irish light-bellied Brent Geese tracked by the WWT (http://www.wwt.org.uk/).