HTNG Event Notification Specification
Version 3.0

24 April 2015

About HTNG

Hotel Technology Next Generation (HTNG) is a non-profit association with a mission to foster, through collaboration and partnership, the development of next-generation systems and solutions that will enable hoteliers and their technology vendors to do business globally in the 21st century. HTNG is recognized as the leading voice of the global hotel community, articulating the technology requirements of hotel companies of all sizes to the vendor community. HTNG facilitate the development of technology models for hospitality that will foster innovation, improve the guest experience, increase the effectiveness and efficiency of hotels, and create a healthy ecosystem of technology suppliers.

Table of contents

1 This Specification at a Glance

Many systems need to communicate key data to one another in order to support the business of hospitality. This technical specification provides a mechanism that allows systems to communicate in a more automated fashion.

An interested event consuming system can use the messages in this specification to:

- Determine which events have notification messages
- Subscribe to those notification messages
- Receive notification messages
- Determine which subscriptions are currently in place and modify the subscriptions as required

An interested event providing system can use the messages in this specification to:

- Publish the events that are available for subscription
- Receive requests for subscription
- Notify the appropriate systems when events take place
- Provide details on the events that are currently subscribed to

These messages allow the setup of the systems operating in the hospitality space to be far more collaborative. It also allows those designing software to move to decision for system interplay into the users control instead of the installer/administrator.

2 Document Information

2.1 Document History

Version	Date	Author	Comments
1.0	22 Apr 2011	Protocol & Message Transport Workgorup	Basic messages included in first version of document
1.0.01	05 May 2014	Kylene Reese	Promoted spec to updated template; included new scenarios/feedback from implementers
1.0.02-.05	May-Jun 2014	Event Notification (EN) Workgroup	Updated sample messages
1.0.06	10 Jul 2014	EN Workgroup	Updated the Get Subscriptions Available scenario
1.0.07	17 July 2014	Mark Jarman	Updated the Renew Subscriptions scenario
2.0	24 Oct 2014	Event Notification Workgroup	Updated the HTNG spec to be compatible with the W3C WS-Eventing standard
2.9	20 March	Event Notification Workgroup	Member Review Period

2.2 Document Purpose

Systems in hospitality exchange data are at an ever increasing rate. Effective collaboration between systems means that more automated mechanisms need to be in place, thereby allowing data to be exchanged when key events occur (i.e. guest check-in, guest check-out, room out of service).

This document provides details on a series of messages that simplify the process of publicizing the messages that are available for subscription, along with an automated subscription and notification process.

2.3 Scope

This document provides the detail on a number of messages along with an update to the HTNG Framework 2.1 header to provide a Publish and Subscribe methodology within workgroup messaging.

2.4 Relationship to Other Standards

This specification and its supporting schemas leverage the existing OpenTravel Alliance methodology for message construction and draws upon data definitions common to several HTNG specifications.

It is important to note that this specification supersedes the previously released "Protocol & Message Transport Event Notification 1.0" spec.

Related specifications:

- W3C Web Services Eventing standard – published 13 December 2011
- W3C WS-Addressing standard – published 9 May 2006
- OASIS WS-Security standard – published 1 February 2006
- OASIS WS-ReliableMessaging – published 2 February 2009

2.5 Useful Resources

- HTNG Discussion Board – currently available at http://www2.htng.org/discussion

2.6 Audience

The intended audiences of this document are development teams and system designers seeking to implement standardized interface specifications within their products. This document also provides business process flows that may used by hotel groups looking to standardize their interfaces within their hotel architectures.

2.7 Overview

HTNG created an Event Notification standard that met the needs of the hotel industry at the time the standard was created. W3C, the organization responsible for many web and web-services standards, also created an Event Notification Standard. The W3C WS-Eventing Standard provides a global standard for service-based subscriptions to event notifications.

These two standards have overlapping functionality but were not compatible with each other. This has led to implementation and integration issues and the need for each integration project to have to choose between the standards. The purpose of this HTNG effort was to create a new HTNG standard where the overlapping functions come from the broader W3C standard while preserving the extended functionality of the earlier HTNG standard.

2.8 Known Limitations

There are no known limitations as of the writing of this document.

2.9 Further Considerations

2.9.1 WS-Reliable Messaging

This HTNG Event Notification Specification does not specify a mechanism for guaranteed or reliable delivery of messages. It is assumed that if these features are required they will be implemented using the underlying delivery mechanism or a technology like WS-ReliableMessaging. Some implementers may find value in implementing WS-ReliableMessaging, but that is beyond the scope of this specification.

2.9.2 Use of Format@Name in the Subscribe Message

There are 2 standard defined format names that can be used to indicate the desired message type the Event Sink will receive. These are:

- `http://www.w3.org/2011/03/ws-evt/DeliveryFormats/Unwrap/`
- `http://www.w3.org/2011/03/ws-evt/DeliveryFormats/Wrap/`

See Section 4.1 in the W3C specification for further information.

2.9.3 Notification Responses

The W3C specification recognizes that SOAP messages are independent of the underlying transport mechanism and that a response is not required or may be returned over a separate channel. Many OpenTravel and HTNG standards assume that HTTP is used as the underlying protocol and use request/response message pairs. The working group suggests that if the sent notification is an HTNG standard request message, the Event Sink (receiving endpoint) should reply with the appropriate matching HTNG response message. If the request message does not have a response message, then acknowledgement is optional. If acknowledgement is required, the generic acknowledge message should be used for the reply.

2.9.4 Use of Faults

This specification leverages the W3C WS-Eventing specification and defines an appropriate set of SOAP Faults, which should be handled appropriately for the solution being deployed. In general, it is assumed that SOAP Faults are *not* ignored and are handled appropriately by the receiving endpoint or Event Sink.

2.9.5 Use of WS-Addressing and Reference Parameters

The WS-Eventing specification uses the WS-Addressing specification to define End Point References (EPR). An EPR is typically a URL but may optionally include reference-parameters to aid in managing state between multiple systems. For example in the sample Subscribe message the Subscriber uses a reference parameter named "SubcriberSubscriptionID:" as a part of the the EndTo and NotifyTo elements. These elements are End Point References that will be used to send notification and end requests to the Subscriber or Event Sink. The Event Sink or

Subscriber should always get these reference-parameters in the SOAP Header when a message is sent to the endpoint.

Likewise the response to the Subscribe request provides an EPR for the SubscriptionManager. This EPR includes a reference-parameter for the PublisherSubscriptionID. This parameter needs to be included in the header of the messages being sent by the Subscriber to the SubscriptionManager.

It should be noted that the W3C specification is flexible enough to support other means of accomplishing the same goals. For example if only one subscription is allowed per Sink endpoint then the content of messages like Unsubscribe could simply provide the end point of the Sink that is unsubscribing.

3 Scenarios

3.1 HTNG Get Subscriptions Available

3.1.1 Overview

This scenario provides a mechanism for a subscribing system to request a list of the available notification events to which it may subscribe. This takes into account that various interested groups/vendors may need specific modifications to particular messages to support unique business functions. In this scenario, a potential subscriber sends a Subscriptions Available request to an Event Source. The Event Source sends the set of events available for subscription.

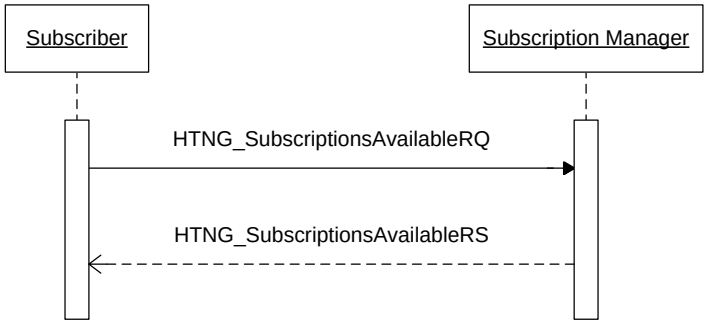
3.1.2 Roles

Role Name	Definition	Examples
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.	• Interface Gateway • Integration Platform • PMS
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.	• PMS • CRS • In-Room System

3.1.3 Use Case

Assumptions:	• Subscription capability exists on the source system. • Messages exist that may be subscribed to on a producer system. • Subscriber has working knowledge of messages available.
Pre-conditions:	The two systems that know about each other have the appropriate authentications in place and are able to communicate.
Trigger:	The Subscriber has some interest in the available messages from a producer system.
Basic Course of Events:	1. The Subscriber requests a list of all available event notifications from the Subscription Manager using the HTNG_SubscriptionsAvailableRQ. 2. The Subscription Manager returns a payload that includes all of the messages that may be subscribed to along with the URL where the subscription must be submitted.
Post-conditions:	None.
Exception Path:	If no subscriptions are available, then an empty set of data is returned.
Alternative Paths:	None.

3.1.4 Message Flows



3.1.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">

  <soap:Header>
    <wsa:Action>http://www.htng.org/2014B/HTNG_SubscriptionsAvailable</wsa:Action>
    <wsa:MessageID>urn:uuid:1d8d20e4-33eb-4087-bd57-6ec6d24ba3ce</wsa:MessageID>
    <wsa:To>https://www.submgr.com/HTNG_SubscriptionsAvailable</wsa:To>
  </soap:Header>

  <soap:Body>
    <HTNG_SubscriptionsAvailableRQ xmlns="http://htng.org/2014B" />
  </soap:Body>

</soap:Envelope>
```

3.1.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.htng.org/2014B/HTNG_SubscriptionsAvailableRS</wsa:Action>
    <wsa:MessageID>urn:uuid:8e9cd5fe-1e6b-46d5-a606-3c80b3db08c2</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:1d8d20e4-33eb-4087-bd57-6ec6d24ba3ce</wsa:RelatesTo>
    <wsa:To>https://www.subscriber.com/subscription_responses</wsa:To>
  </soap:Header>

  <soap:Body>
    <HTNG_SubscriptionsAvailableRS xmlns="http://htng.org/2014B">
      <AvailableSubscriptions>
        <TypeOfEvent VendorVersionID="1.2" VendorID="resVendor"
EventID="urn:uuid:2515d557-04f4-46ae-81a4-bd3f1d07145b">

<MessageDef>http://www.opentravel.org/OTA/2003/05/OTA_HotelResNotifRQ</MessageDef>

<SendSubscribeTo>https://www.submgr.com/resVendor/OnResCreated</SendSubscribeTo>
          <Description>Notifies whenever a new reservation is created</Description>
          <FilterDialects>
            <Dialect>http://www.htng.org/2014B/HTNG_SimpleFilter</Dialect>
          </FilterDialects>
        </TypeOfEvent>
        <TypeOfEvent VendorVersionID="1.5" VendorID="resVendor"
EventID="urn:uuid:0fb99862-ce8e-4f51-b1aa-bd467243ee2d">
```

```
<MessageDef>http://www.opentravel.org/OTA/2003/05/OTA_HotelResNotifRQ</MessageDef>

<SendSubscribeTo>https://www.submgr.com/resVendor/OnResChanged</SendSubscribeTo>
          <Description>Notifies whenever a reservation is created or
modified</Description>
          <FilterDialects>
            <Dialect>http://www.w3.org/2011/03/ws-evt/Dialects/XPath10</Dialect>
            <Dialect>http://www.w3.org/1999/XSL/Transform</Dialect>
            <Dialect>http://www.htng.org/2014B/HTNG_SimpleFilter</Dialect>
          </FilterDialects>
        </TypeOfEvent>
        <TypeOfEvent VendorVersionID="1.2" VendorID="resVendor"
EventID="urn:uuid:09c6fce4-8b5f-476e-a3d7-3c41894f6df9">
          <MessageDef>http://www.opentravel.org/OTA/2003/05/OTA_CancelRQ</MessageDef>

<SendSubscribeTo>https://www.submgr.com/resVendor/OnResCancelled</SendSubscribeTo>
          <Description>Notifies whenever a reservation is canceled</Description>
          <FilterDialects>
            <Dialect>http://www.w3.org/2011/03/ws-evt/Dialects/XPath10</Dialect>
            <Dialect>http://www.w3.org/1999/XSL/Transform</Dialect>
            <Dialect>http://www.htng.org/2014B/HTNG_SimpleFilter</Dialect>
          </FilterDialects>
        </TypeOfEvent>
      </AvailableSubscriptions>
    </HTNG_SubscriptionsAvailableRS>
  </soap:Body>
</soap:Envelope>
```

3.2 Subscribe

3.2.1 Overview

Provides a mechanism for a system to subscribe to specific events. The subscription process includes a unique ID from the subscribing system called the ConsumerSubscriptionID. This provides a mechanism for the subscribed system to update its subscription request using an overlay methodology. In this scenario, a vendor system sends a subscription request to the notification producer to register interest for an event type. The notification system confirms that the subscription has been received and created within its notification register.

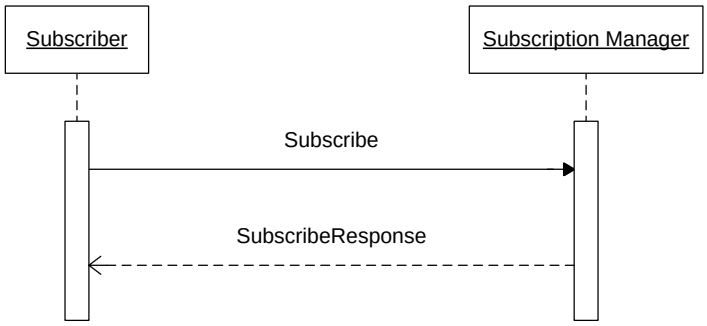
3.2.2 Roles

Role Name	Definition	Examples
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.	• Interface Gateway • Integration Platform • PMS
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.	• PMS • CRS • In-Room System

3.2.3 Use Case

Assumptions:	Subscription capability exists on the producer system. An HTNG_SubscriptionsAvailableRS message has provided some ProducerEventIDs that are available.
Pre-conditions:	The two systems that know about each other have the appropriate authentications in place and are able to communicate.
Trigger:	The Event Consumer wishes to subscribe to an event.
Basic Course of Events:	The use case begins when the interested system (Event Consumer) issues a request to subscribe to a particular event notification that was detailed in the HTNG_SubscriptionsAvailableRS message.
Post-conditions:	None.
Exception Path:	If the subscription request is not valid, then a fault will be returned (i.e. the subscription request was for a ProducerEventID that does not exist). Possible Faults: • DeliveryModeRequestedUnavailable • InvalidExpirationTime • UnsupportedExpirationType • FilteringNotSupported • FilteringRequestUnavailable • EventSourceUnableToProcess • InsufficientPermissions
Alternative Paths:	None.

3.2.4 Message Flows



3.2.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt"
```

```
  xmlns:wsse="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1">

  <soap:Header>
    <wsse:security>
      <wsse:UsernameToken>
        <wsse:Username>joe</wsse:Username>
        <wsse:Password>hiddenPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:security>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/Subscribe</wsa:Action>
    <wsa:MessageID>urn:uuid:d7c5726b-de29-4313-b4d4-b3425b200839</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.subscriber.com/subscription_responses</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscribedID>4321</SubscribedID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.submgr.com/resVendor/OnResChanged</wsa:To>
  </soap:Header>

  <soap:Body>
    <wse:Subscribe>
      <wse:EndTo>
        <wsa:Address>https://www.subscriber.com/subscription_end</wsa:Address>
        <wsa:ReferenceParameters>
          <SubscribedID>4321</SubscribedID>
        </wsa:ReferenceParameters>
      </wse:EndTo>
      <wse:Delivery>
        <wse:NotifyTo>
          <wsa:Address>https://www.subscriber.com/resChanged</wsa:Address>
          <wsa:ReferenceParameters>
            <SubscribedID>4321</SubscribedID>
          </wsa:ReferenceParameters>
        </wse:NotifyTo>
      </wse:Delivery>
      <wse:Format name="http://www.w3.org/2011/03/ws-evt/DeliveryFormats/Unwrap" />
      <wse:Expires BestEffort="true">P7D</wse:Expires>
      <wse:Filter Dialect="http://www.htng.org/2014B/HTNG_SimpleFilter">
        <htng:HTNG_SimpleFilter xmlns="http://www.htng.org/htngSimpleFilter">
          <htng:matchAny>
            <htng:name>HotelCode</htng:name>
            <htng:value>DCACY</htng:value>
            <htng:value>DCAFF</htng:value>
          </htng:matchAny>
        </htng:HTNG_SimpleFilter>
      </wse:Filter>
    </wse:Subscribe>
  </soap:Body>
</soap:Envelope>
```

3.2.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/SubscribeResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:dd41a736-756f-4827-a0d7-e39a4ac5050c</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:d7c5726b-de29-4313-b4d4-b3425b200839</wsa:RelatesTo>
    <wsa:To>https://www.subscriber.com/subscription_responses</wsa:To>
    <he:SubscribedID wsa:IsReferenceParameter="true">4321</he:SubscribedID>
  </soap:Header>

  <soap:Body>
    <wse:SubscribeResponse>
      <wse:SubscriptionManager>
        <wsa:Address>https://www.submgr.com/manage</wsa:Address>
```

```
        <wsa:ReferenceParameters>
          <SubscriptionID>41</SubscriptionID>
        </wsa:ReferenceParameters>
      </wse:SubscriptionManager>
      <wse:GrantedExpires>2015-03-01T13:30:00.0Z</wse:GrantedExpires>
    </wse:SubscribeResponse>
  </soap:Body>
</soap:Envelope>
```

3.3 Renew Subscription

3.3.1 Overview

When a Subscriber subscribes to an event a subscription expiration date is defined. To update subscription expiration, the Subscription Managers MUST support requests to renew subscriptions.

Notes: The W3C specification allows the subscriber to renew a subscription scheduled to expire. The W3C specification also allows the subscription manager to reject the subscription renewal request, based on factors such as the event no longer being available, or credentials expiring soon.

3.3.2 Roles

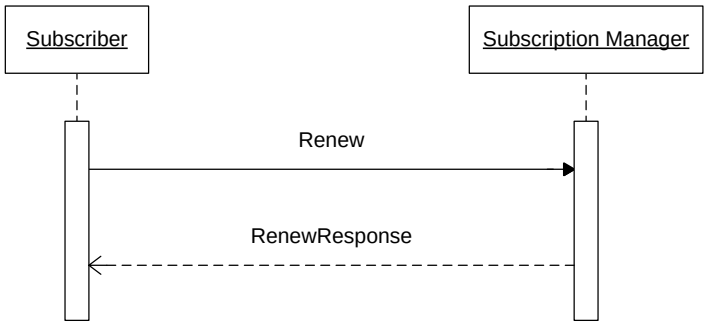
Role Name	Definition	Examples
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.	• Interface Gateway • Integration Platform • PMS
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.	• PMS • CRS • In-Room System

3.3.3 Use Case

Assumptions:	• Subscription capability exists. • Subscriber has working knowledge of messages available.
Pre-condition:	The Subscriber has registered a subscription in the Subscription Manager.
Trigger:	The Subscriber has a need to extend the subscription.
Basic Course of Events:	1. The Subscriber sends a request to the Subscription Manager to renew its existing subscription by updating the subscription expiration date. 2. The Subscription Manager accepts the renewal request and replies with either a success or failure response.

Post-condition:	The subscription expiration timestamp is updated to the value passed in the renewal request.
Exception Path:	If the Subscription Manager can not renew the subscription, a fault is returned to the Subscriber. If a retry after element is included, the Subscriber may retry after the specified duration has ended. Possible Faults: • InvalidExpirationTime • UnsupportedExpirationType • UnableToRenew • InsufficientPermissions
Alternative Path:	None

3.3.4 Message Flows



3.3.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/Renew</wsa:Action>
    <wsa:MessageID>urn:uuid:bd88b3df-5db4-4392-9621-aee9160721f6</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.subscriber.com/subscription_responses</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscribedID>4321</SubscribedID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.submgr.com/manage</wsa:To>
    <SubscriptionID wsa:IsReferenceParameter="true">41</SubscriptionID>
  </soap:Header>

  <soap:Body>
    <wse:Renew>
      <wse:Expires>P7D</wse:Expires>
    </wse:Renew>
  </soap:Body>
</soap:Envelope>
```

3.3.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/RenewResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:8dcd0566-e777-4c85-a39c-19432cc6c3eb</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:bd88b3df-5db4-4392-9621-aee9160721f6</wsa:RelatesTo>
    <wsa:To>https://www.subscriber.com/subscription_responses</wsa:To>
    <SubscribedID wsa:IsReferenceParameter="true">4321</SubscribedID>
  </soap:Header>

  <soap:Body>
    <wse:RenewResponse>
      <wse:GrantedExpires>2015-03-08T13:30:00.0-05:00</wse:GrantedExpires>
    </wse:RenewResponse>
  </soap:Body>
</soap:Envelope>
```

3.4 Unsubscribe

3.4.1 Overview

When a Subscriber no longer has a need to receive a given notification, it may unsubscribe.

3.4.2 Roles

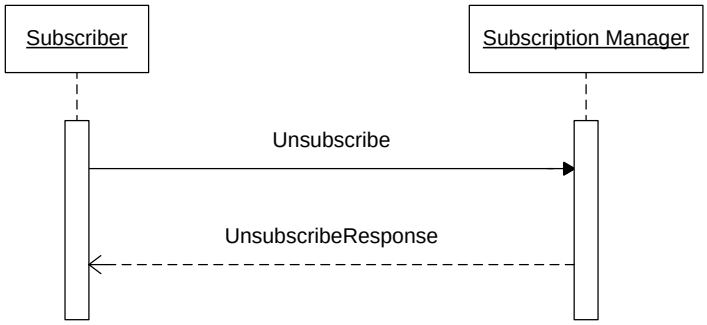
Role Name	Definition	Examples
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.	• Interface Gateway • Integration Platform • PMS
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.	• PMS • CRS • In-Room System

3.4.3 Use Case

Assumptions:	
Pre-conditions:	The Subscriber is currently subscribed to receive notifications.
Trigger:	The Subscriber no longer needs to receive notifications.
Basic Course of Events:	• The Subscriber sends unsubscribe request to Subscription Manager (or Event Source) • The Subscription Manager removes the subscription • The Subscription Manager returns response to Subscriber.
Post-conditions:	The Subscriber will no longer receive notifications for the unsubscribed event.

Exception Path:	If the subscription id supplied by the Subscriber is not valid, the Subscription Manager will return an UnknownSubscription fault. Possible Faults: • UnknownSubscription • InsufficientPermissions
Alternative Paths:	None.

3.4.4 Message Flows



3.4.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/Unsubscribe</wsa:Action>
    <wsa:MessageID>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.subscriber.com/subscription_responses</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscribedID>4321</SubscribedID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.submgr.com/manage</wsa:To>
    <SubscriptionID wsa:IsReferenceParameter="true">41</SubscriptionID>
  </soap:Header>

  <soap:Body>
    <Unsubscribe />
  </soap:Body>
</soap:Envelope>
```

3.4.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/UnsubscribeResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:dae306b5-0f3b-4363-a279-cc401e3a0f5d</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:RelatesTo>
    <wsa:To>https://www.subscriber.com/subscription_responses</wsa:To>
```

```
    <SubscribedID wsa:IsReferenceParameter="true">4321</SubscribedID>
  </soap:Header>

  <soap:Body>
    <wse:UnsubscribeResponse />
  </soap:Body>
</soap:Envelope>
```

3.5 Get Status

3.5.1 Overview

A Subscriber can determine the status of the subscription by requesting the subscription status from the Subscription Manager.

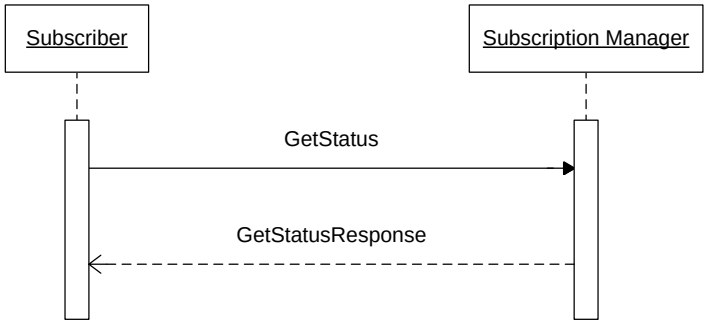
3.5.2 Roles

Role Name	Definition	Examples
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.	• Interface Gateway • Integration Platform • PMS
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.	• PMS • CRS • In-Room System

3.5.3 Use Case

Assumptions:	Subscription capability exists on the producer system Subscriber has previously subscribed to an event notification
Pre-conditions:	The two systems that know about each other have the appropriate authentications in place and are able to communicate.
Trigger:	The Subscriber wishes to know the status of an event it is subscribed to.
Basic Course of Events:	The use case begins when the Subscriber issues a request to determine if a subscription is active.
Post-conditions:	None.
Exception Path:	If the subscription does not exist, an UnknownSubscription fault will be returned. Possible Faults: • UnknownSubscription
Alternative Paths:	None.

3.5.4 Message Flows



3.5.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/GetStatus</wsa:Action>
    <wsa:MessageID>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.subscriber.com/subscription_responses</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscribedID>4321</SubscribedID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.submgr.com/manage</wsa:To>
    <SubscriptionID wsa:IsReferenceParameter="true">41</SubscriptionID>
  </soap:Header>

  <soap:Body>
    <GetStatus />
  </soap:Body>
</soap:Envelope>
```

3.5.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/GetStatusResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:dae306b5-0f3b-4363-a279-cc401e3a0f5d</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:RelatesTo>
    <wsa:To>https://www.subscriber.com/subscription_responses</wsa:To>
    <SubscribedID wsa:IsReferenceParameter="true">4321</SubscribedID>
  </soap:Header>

  <soap:Body>
    <wse:GetStatusResponse>
      <wse:GrantedExpires>2015-03-07T13:30:00.0-05:00</wse:GrantedExpires>
    </wse:GetStatusResponse>
  </soap:Body>
</soap:Envelope>
```

3.6 HTNG Get Subscription Status

3.6.1 Overview

This service exposes a mechanism for a Subscriber to obtain the status of a single subscription based the parameters returned by the Subscribe message response or return the status of a list existing subscriptions based on the Subscriber identification. In the first scenario, the individual subscriber is interested in accessing the full description of the service to which it subscribed. In the second scenario, the subscribing system is interested in determining all of the types of events that have been subscribed. In this case the subscribing system sends a request with a Subscriber Identifier and receives the list of all of its currently active subscriptions. The Subscription Manager system replies with a list of subscribed event types.

Due to security concerns, this service is intended to be used only by the subscriber originally creating the subscription. However, it is understood that other applications may find value in leveraging this service. It is recommended a security role is established for these consuming applications and they are authenticated and authorized in that role.

3.6.2 Roles

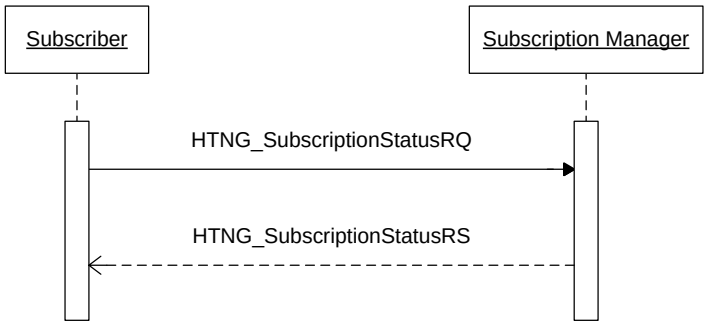
Role Name	Definition	Examples
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.	• Interface Gateway • Integration Platform • PMS
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.	• PMS • CRS • In-Room System

3.6.3 Use Case

Assumptions:	Subscription capability exists on the producer system The HTNG_SubscriptionStatusRQ/RS message pair is available and supported.
Pre-conditions:	The two systems that know about each other have the appropriate authentications in place and are able to communicate.
Trigger:	The Event Consumer wishes to know which events it is subscribed to.
Basic Course of Events:	The use case begins when the interested system (Subscriber) issues a request to provide the details for all of the subscriptions currently in place.
Post-conditions:	None.

Exception Path:	If no subscriptions currently exist, an empty set of data is returned. Possible Faults: • UnknownSubscriber
Alternative Paths:	None.

3.6.4 Message Flows



3.6.5 Sample Request – Request by SubscriberID

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt"
  xmlns:wsse="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1">

  <soap:Header>
    <wsse:security>
      <wsse:UsernameToken>
        <wsse:Username>joe</wsse:Username>
        <wsse:Password>hiddenPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:security>
    <wsa:Action>http://www.htng.org/2014B/HTNG_SubscriptionStatusRQ</wsa:Action>
    <wsa:MessageID>urn:uuid:1d8d20e4-33eb-4087-bd57-6ec6d24ba3ce</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.subscriber.com/recvStatus</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>https://www.submgr.com/manage</wsa:To>
  </soap:Header>

  <soap:Body>
    <HTNG_SubscriptionStatusRQ xmlns="http://htng.org/2014B">
      <SubscriberID>joe</SubscriberID>
    </HTNG_SubscriptionStatusRQ>
  </soap:Body>
</soap:Envelope>
```

3.6.6 Sample Request – Request for Single Subscription

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/GetStatus</wsa:Action>
    <wsa:MessageID>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:MessageID>
```

```
    <wsa:ReplyTo>
      <wsa:Address>https://www.subscriber.com/subscription_responses</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscribedID>4321</SubscribedID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.submgr.com/manage</wsa:To>
    <SubscriptionID wsa:IsReferenceParameter="true">41</SubscriptionID>
  </soap:Header>

  <soap:Body>
    <HTNG_SubscriptionStatusRQ/>
  </soap:Body>
</soap:Envelope>
```

3.6.7 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.htng.org/2014B/HTNG_SubscriptionStatusRS</wsa:Action>
    <wsa:MessageID>urn:uuid:dd41a736-756f-4827-a0d7-e39a4ac5050c</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:1d8d20e4-33eb-4087-bd57-6ec6d24ba3ce</wsa:RelatesTo>
    <wsa:To>https://www.subscriber.com/recvStatus</wsa:To>
  </soap:Header>

  <soap:Body>
    <HTNG_SubscriptionStatusRS xmlns="http://htng.org/2014B">
      <Subscriptions>
        <Subscription>
          <TypeOfEvent EventID="urn:uuid:0fb99862-ce8e-4f51-b1aa-bd467243ee2d"
VendorVersionID="1.2" VendorID="resVendor">

<MessageDef>http://www.opentravel.org/OTA/2003/05/OTA_HotelResNotifRQ</MessageDef>

<SendSubscribeTo>https://www.submgr.com/resVendor/resCreated</SendSubscribeTo>
            <Description>Notifies whenever a new reservation is created</Description>
          </TypeOfEvent>
          <SubscriptionManager>
            <wsa:Address>https://www.submgr.com/manage</wsa:Address>
            <wsa:ReferenceParameters>
              <SubscriptionID>41</SubscriptionID>
            </wsa:ReferenceParameters>
          </SubscriptionManager>
          <GrantedExpires>2015-03-01T13:30:00Z</GrantedExpires>
          <Endto>
            <wsa:Address>https://www.subscriber.com/subscription_end</wsa:Address>
            <wsa:ReferenceParameters>
              <SubscribedID>4321</SubscribedID>
            </wsa:ReferenceParameters>
          </Endto>
          <Delivery>
            <NotifyTo>
              <wsa:Address>https://www.subscriber.com/reservationChanged</wsa:Address>
              <wsa:ReferenceParameters>
                <SubscribedID>4321</SubscribedID>
              </wsa:ReferenceParameters>
            </NotifyTo>
          </Delivery>
          <Filter Dialect="http://www.htng.org/2014B/HTNG_SimpleFilter">
            <htng:HTNG_SimpleFilter xmlns:htng="http://www.htng.org/htngSimpleFilter">
              <htng:matchAny>
                <htng:name>HotelCode</htng:name>
                <htng:value>DCACY</htng:value>
                <htng:value>DCAFF</htng:value>
              </htng:matchAny>
            </htng:HTNG_SimpleFilter>
```

```
          </Filter>
        </Subscription>
      </Subscriptions>
    </HTNG_SubscriptionStatusRS>
  </soap:Body>
</soap:Envelope>
```

3.7 Subscription End

3.7.1 Overview

The W3C Standard provides a mechanism for the Subscription Manager to terminate a subscription before the expiration of the subscription and notify the Subscriber of the unexpected termination. This message is sent to the EndTo EPR provided by the Subscriber in the Subscribe message. The SubscriptionEnd message does not require a response message, however HTNG has provided an optional response message in case it is required for specific implementations. It should be noted that the Subscription Manager may no longer be available to accept the response.

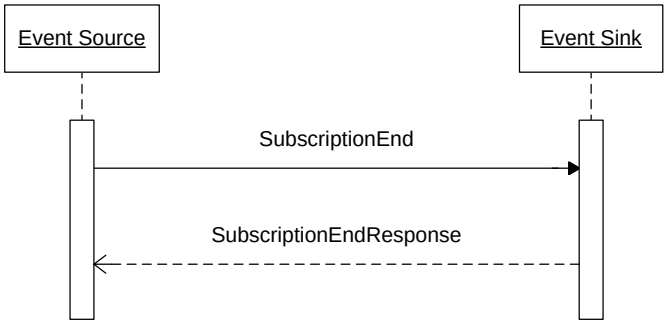
3.7.2 Roles

Role Name	Definition	Examples
Event Source	A producer of event information that wishes to keep all interested systems updated on subscribed events.	• PMS • CRS • In-Room System
Event Sink	A system that wishes to consume information relating to events on a regular basis.	• Interface Gateway • Integration Platform • PMS

3.7.3 Use Case

Assumptions:	The Subscription Manager or Event Source needs to end a subscription early. A Subscription End endpoint must have been previously provided by the Subscriber.
Pre-conditions:	A Subscriber is subscribed to a notification.
Trigger:	The Subscription Manager or Event Source needs to end a subscription early.
Basic Course of Events:	The Subscription Manager sends the SubscriptionEnd message to the Subscription End endpoint.
Post-conditions:	There is no longer a subscription for the event.
Exception Path:	None
Alternative Paths:	None

3.7.4 Message Flows



3.7.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/SubscriptionEnd</wsa:Action>
    <wsa:MessageID>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.submgr.com/endReply</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscribedID>41</SubscribedID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.subscriber.com/subscription_end</wsa:To>
    <SubscribedID wsa:IsReferenceParameter="true">4321</SubscribedID>
  </soap:Header>

  <soap:Body>
    <wse:SubscriptionEnd>
      <wse:Status>wse:SourceShuttingDown</wse:Status>
      <wse:Reason xml:lang="en-us">Event Source is going off-line</wse:Reason>
    </wse:SubscriptionEnd>
  </soap:Body>
</soap:Envelope>
```

3.7.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <!-- This response message is optional and is not part of the W3C specification -->
  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/SubscriptionEndResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:dae306b5-0f3b-4363-a279-cc401e3a0f5d</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:997087d7-375d-4f8c-a872-972e46c7ba9a</wsa:RelatesTo>
    <wsa:To>https://www.submgr.com/endReply</wsa:To>
    <SubscriptionID wsa:IsReferenceParameter="true">41</SubscriptionID>
  </soap:Header>

  <soap:Body>
    <HTNG_AcknowledgeReceipt xmlns=http://htng.org/2014B />
  </soap:Body>
</soap:Envelope>
```

3.8 Event Notification

3.8.1 Overview

When an event occurs the Event Source sends the Event Notification message to the Event Sinks that have subscribed to receive the notifications for the event. The W3C does not require a response to a notification message. HTNG has provided an optional response message in case it is required for specific implementations that do not already have a defined response.

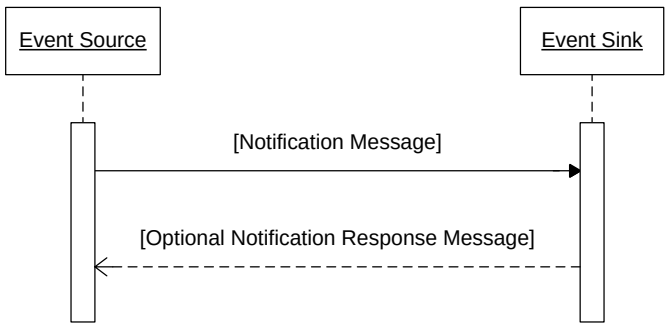
3.8.2 Roles

Role Name	Definition	Examples
Event Source	A producer of event information that wishes to keep all interested systems updated on subscribed events.	• PMS • CRS • In-Room System
Event Sink	A system that wishes to consume information relating to events on a regular basis.	• Interface Gateway • Integration Platform • PMS

3.8.3 Use Case

Assumptions:	Subscription capability exists on the Event Source system.
Pre-conditions:	The two systems that know about each other have the appropriate authentications in place and are able to communicate.
Trigger:	The Event Source wishes to send out an Event Notification.
Basic Course of Events:	The use case begins when the Event Source determines that an event that has occurred is of particular interest to an Event Sink.
Post-conditions:	None.
Exception Path:	If the Event Sink is no longer available at the nominated endpoint, the Event Source should optionally send a SubscriptionEnd message.
Alternative Paths:	None.

3.8.4 Message Flows



3.8.5 Sample Request

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">

  <soap:Header>
    <wsa:Action>http://www.subscriber.com/OnResChanged</wsa:Action>
    <wsa:MessageID>urn:uuid:568b4ff2-5bc1-4512-957c-0fa545fd8d7f</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>https://www.submgr.com/notifyReply</wsa:Address>
      <wsa:ReferenceParameters>
        <SubscriptionID>41</SubscriptionID>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:To>https://www.subscriber.com/resChanged</wsa:To>
    <SubscribedID wsa:IsReferenceParameter="true">4321</SubscribedID>
  </soap:Header>

  <soap:Body>
    <ota:OTA_HotelResNotifRQ xmlns:ota="http://www.opentravel.org/OTA/2003/05"
Version="1.003" EchoToken="879791878" ResStatus="Commit" TimeStamp=" 2005-10-
09T18:51:45">

      <!-- Lots of message specific content here -->

    </ota:OTA_HotelResNotifRQ>
  </soap:Body>
</soap:Envelope>
```

3.8.6 Sample Response

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wse="http://www.w3.org/2011/03/ws-evt">

  <!-- This response message is optional and is not part of the W3C specification -->
  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/SubscriptionEndResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:dae306b5-0f3b-4363-a279-cc401e3a0f5d</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:568b4ff2-5bc1-4512-957c-0fa545fd8d7f</wsa:RelatesTo>
    <wsa:To>https://www.submgr.com/notifyReply</wsa:To>
    <SubscriptionID wsa:IsReferenceParameter="true">41</SubscriptionID>
  </soap:Header>

  <soap:Body>
    <HTNG_AcknowledgeReceipt xmlns="http://htng.org/2014B" />
  </soap:Body>
</soap:Envelope>
```

4 Messages

4.1 Get Subscriptions Available

4.1.1 Data Element Table – Request

Element | @Attribute	Num	Description/Contents
HTNG_SubscriptionsAvailableRQ	1	Root element of the message. This message initiates the request for the subscriptions available from the producer system.

4.1.2 Data Element Table – Response

Element | @Attribute	Num	Description/Contents
HTNG_SubscriptionsAvailableRS	1	Root element of the message. This message provides all of the available subscriptions from the producer system.
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions	0..1	Provides the Event Types that a system offers as possible subscriptions
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions / TypeOfEvent	1..n	Each individual Event Type the publishing system is making available for subscription.
@VendorVersionID	0..1	Provides a sequential number of the revision to this particular message when an event is updated along with a VendorProfileID. This allows for different subscriptions/content for different vendors. Vendors are strongly encouraged to formalize this subscription scenario in future HTNG schema releases.
@VendorID	0..1	A unique identifier that represents the source system that is publishing a given event notification.
@EventID	1	A unique Identifier provided by the Producer to define a specific message that can be subscribed to. For the same MessageEventType, there may be many ProducerEventIDs due to generational and specific changes made for certain interested groups/vendors.
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions / TypeOfEvent / MessageDef	1	Defines the Event Type, typically the message name (i.e. http://www.opentravel.org/OTA/2003/05/OTA_HotelResNotifRQ). This should translate directly to the messages defined by workgroups.
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions / TypeOfEvent / SendSubscribeTo	1	Provides the endpoint that should be used when subscribing to this message.
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions / TypeOfEvent / Description	0..1	Brief description of the Event Type. More detail should be available from the various message specifications within other workgroups.
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions / TypeOfEvent / FilterDialects	0..1	Describes the filters that may be available for a specific Event Type.

Element | @Attribute	Num	Description/Contents
HTNG_SubscriptionsAvailableRS / AvailableSubscriptions / TypeOfEvent / FilterDialects / Dialect	1..n	A named filter expression negotiated between trading partners. Valid values are: • http://www.w3.org/2011/03/ws-evt/Dialects/XPath10 • http://www.w3.org/1999/XSL/Transform • http://www.htng.org/2014B/HTNG_SimpleFilter

4.2 Get Subscription Status

4.2.1 Data Element Table – Request

Element | @Attribute	Num	Description/Contents
HTNG_SubscriptionStatusRQ	1	Root element of the message. Requests the existing subscriptions currently in place for a particular Event Consumer.
HTNG_SubscriptionStatusRQ / SubscriberID	0..1	Used to retrieve existing subscriptions for a specific system. This is the value of the WS-Security username used when the Subscribe request was sent.

4.2.2 Data Element Table – Response

Element | @Attribute	Num	Description/Contents
HTNG_SubscriptionStatusRS	1	Root element of the message. Describes the current subscriptions in place with the Event Producer.
HTNG_SubscriptionStatusRS / Subscriptions	0..1	A collection of the events the Event Consumer is currently subscribed to.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription	1..n	Child element describing each subscription currently in place.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / TypeOfEvent	1..n	Each individual Event Type the publishing system is making available for subscription.
@VendorVersionID	0..1	Provides a sequential number of the revision to this particular message when an event is updated along with a VendorProfileID. This allows for different subscriptions/content for different vendors. Vendors are strongly encouraged to formalize this subscription scenario in future HTNG schema releases.
@VendorID	0..1	A unique identifier that represents the source system that is publishing a given event notification.
@EventID	1	A unique Identifier provided by the Producer to define a specific message that can be subscribed to. For the same MessageEventType, there may be many ProducerEventIDs due to generational and specific changes made for certain interested groups/vendors.
HTNG_SubscriptionStatusRS /	1	Defines the Event Type, typically the message name (i.e.

Element | @Attribute	Num	Description/Contents
Subscriptions / Subscription / TypeOfEvent / MessageDef		http://www.opentravel.org/OTA/2003/05/OTA_HotelResNotifRQ). This should translate directly to the messages defined by workgroups.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / TypeOfEvent / SendSubscribeTo	1	Provides the endpoint that should be used when subscribing to this message.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / TypeOfEvent / Description	0..1	Brief description of the Event Type. More detail should be available from the various message specifications within other workgroups.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / SubscriptionManager	0..1	A container element describing the subscription manager.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / SubscriptionManager / Address	1	The endpoint reference of the subscription manager for this subscription.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / SubscriptionManager / ReferenceParameters	1	A container element holding the necessary reference parameters used to create the subscription.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / SubscriptionManager / ReferenceParameters / SubscriptionID	1	The unique identifier representing the subscription as issued by the subscription manager.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / GrantedExpires	1	The expiration time assigned by the event source.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / EndTo	0..1	The element that represents the necessary data that will be used in the event that a subscription needs to be terminated.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / EndTo / Address	1	The endpoint reference for where the SubscriptionEnd message will be sent.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / EndTo / ReferenceParameters	1	A container element holding the necessary reference parameters to manage the termination of the subscription.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / EbdTo / ReferenceParameters / SubscribedID	1	The unique identifier representing the subscription as issued by the subscriber.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / DeliveryTo /	1	This element contains the information necessary to convey notification messages to the event sink in a manner required by the subscriber.

Element | @Attribute	Num	Description/Contents
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / DeliveryTo / NotifyTo	1	This element indicates that notifications must be sent to the EndpointReference identified by this element.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / DeliveryTo / NotifyTo / Address	1	The endpoint reference for where the event notification messages will be sent.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / DeliveryTo / NotifyTo / ReferenceParameters	1	A container element holding the necessary reference parameters to be sent with the event notification.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / DeliveryTo / NotifyTo / ReferenceParameters / SubscribedID	1	The unique identifier representing the subscription as issued by the subscriber.
HTNG_SubscriptionStatusRS / Subscriptions / Subscription / Filter	0..1	See Section 5 on Simple Filters

4.3 HTNG_AcknowledgeReceipt

4.3.1 Data Element Table – Response

HTNG_AcknowledgeReceipt	1	Root element of the message. This is used to generically acknowledge any message where a response has no payload but must be acknowledged.

5 Simple Filters

One of the features gained by using the W3C specification is the support for filters to determine if a specific instance of a notification should be sent to a Sink. Filters can be created using existing XML technologies like XPath and XML Stylesheets (XSLT). An XPath filter might for example allow a message to be sent only if the provided XPath expression returns true or a non-empty set when evaluated against a particular notification. An XSLT filter might return "Accept" or "Reject" after processing the notification with a provided Stylesheet.

While XPath and XSLT can be used to process any arbitrarily complex filters there is a desire within the HTNG to provide a simpler filtering solution to easily address the more common types of filters encountered without the need to understand XPath or XSLT. To this end the HTNG_SimpleFilter is defined within this standard as an implementer option. This standard introduces the HTNG_SimpleFilter type and a schema to define a simple filter. The elements of this schema are described in this section.

5.1 Simple Filter Schema

The schema for the HTNG_SimpleFilter is provided here:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.htng.org/htngSimpleFilter"
  xmlns="http://www.htng.org/htngSimpleFilter"
  elementFormDefault="qualified">

  <!-- defines a type for the name element with two attributes rule and type -->
  <xsd:complexType name="NameType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute
          name="rule"
          type="xsd:string" />
        <xsd:attribute
          name="type"
          type="xsd:string" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <!--
    MatchType defines a type for match elements.
    A match element may have either a name-value pair or
    a list of match elements
  -->
  <xsd:complexType name="MatchType">
    <xsd:choice>
      <xsd:sequence>
        <xsd:element
          name="name"
          type="NameType" />
        <xsd:element
          name="value"
          type="xsd:string" />
      </xsd:sequence>
      <xsd:sequence>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element
            name="matchAny"
```

```
            type="MatchType" />
          <xsd:element
            name="matchAll"
            type="MatchType" />
          <xsd:element
            name="matchOne"
            type="MatchType" />
          <xsd:element
            name="matchNone"
            type="MatchType" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:choice>
  </xsd:complexType>

  <!-- An HTNG_SimpleFilter must have a single element of any matchType element -->
  <xsd:complexType name="HTNG_SimpleFilterType">
    <xsd:choice>
      <xsd:element
        name="matchAny"
        type="MatchType" />
      <xsd:element
        name="matchAll"
        type="MatchType" />
      <xsd:element
        name="matchOne"
        type="MatchType" />
      <xsd:element
        name="matchNone"
        type="MatchType" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element
    name="HTNG_SimpleFilter"
    type="HTNG_SimpleFilterType" />

</xsd:schema>
```

5.2 Element HTNG_SimpleFilter

The HTNG_SimpleFilter element is the root element for all simple filters. The filter contains a single matchAll, matchAny, matchOne, or matchNone element and sends the message if the match element matches its content or rejects the message if there is not a match.

5.3 Element matchAll

A matchAll element matches if all of the elements it contains match their contents. The element can contain a mixed list of matchAll, matchAny, matchNone, or matchOne elements or a single name element and a list of value elements. A matchAll element may complete processing immediately after it encounters a failed match.

5.4 Element matchAny

A matchAny element matches if any one (or more) of the elements it contains match their content. The element can contain a mixed list of matchAll, matchAny, matchOne, or matchNone elements or a single name element and a list of value elements. A matchAny element may complete processing immediately upon the first successful match.

5.5 Element matchNone

A matchNone element matches if none of the elements it contains match their contents. The element can contain a mixed list of matchAll, matchAny, or matchNone elements or a single name element and a list of value elements. When this element contains a single matchAll or matchAny element it essentially negates the result of the contained match. A matchNone element may complete processing immediately upon the first successful match.

5.6 Element matchOne

A matchOne element matches only if exactly one of the elements it contains matches its contents. The element can contain a mixed list of matchAll, matchAny, matchOne, or matchNone elements or a single name element and a list of value elements. A matchOne element may complete processing immediately upon the second successful match.

5.7 Element name

Only one name element is allowed in a match element. The name element contains a simple character string that identifies or references a value to be tested or compared. If a value element does not follow a name element then the message succeeds if the name exists within the evaluation context. How the name is interpreted as a reference or is processed is left up to the implementation. If the name does not exist or is not valid within the evaluation context then the subscription request with the filter should be rejected by the source.

The name element has two optional attributes; rule and type. The rule attribute identifies how the name and value will be compared or matched. If the rule attribute is not provided or supported then the match will be a regular expression match using the values as regular expressions.

The type attribute identifies the data type of the item referenced by the name and is also used as the data type for interpreting the value elements. If the type attribute is not provided then the item is treated as a string.

5.7.1 Attribute rule

The rule attribute on the name element is optional. If a rule attribute is not provided then the default behavior should be to use the content of the value element as regular expression and match if the regular expression matches the data reference by the name element.

When a rule attribute is provided it is to be treated as a reference to comparison operation. This operation must return a boolean value, true corresponding to match and false corresponding to doesn't match using the information referenced by the name element as one argument and the value element as the second argument. If multiple value elements exist then the rule is applied once for each value until the conditions of the match are met or all values are exhausted.

If rules are supported then it is recommended that the following rules be supported:

- regex – the value is treated as a regular expression and applied against the named item
- isGreater – the named item is greater than the value
- isLess – the named item is less than the value
- isGreaterOrEqual – then named item is greater or equal to the value
- isLessOrEqual – then named item is less or equal to the value
- isEqual – the named item is equal to the value
- isNotEqual – the named item does not equal the value

The regex rule is the default if no rule attribute is given. In the other rules if the value is a number the comparison will be numeric, otherwise it will be lexical.

5.7.2 Attribute type

The optional type attribute of the name element describes the data type of the item referenced by the name element. The value elements should all be evaluated as this same type. In the absence of a type the named items and all values will be treated as strings. The following types should be recognized:

- boolean – true or false
- double – any number with a decimal point
- date – a date or date-time
- integer – an integer
- string – a string
- time – a time
- duration – a duration of time

Dates, Times, and Durations should all be expressed in ISO8601standard formats.

5.8 Element value

A MatchType (matchAll, matchAny, matchOne, matchNone) element may contain zero or more value elements. The value element holds a value to be compared to the item referenced by the name element. By default value will be treated as a regular expression and the match is made if the regular expression is matched against the named item. If multiple values are provided then each element is evaluated according to the rules of its matching container.

If the name element has a rule and a type attribute then the value should be evaluated as a value of the same type. If the name element does not have a type attribute then the value should be treated as a string. Each value should be checked as being a valid instance in its type and context prior to accepting the filter in a subscribe message.

5.9 Filter Examples

The following sections provide some examples demonstrating how the HTNG_SimpleFilters might be applied.

5.9.1 Match a single hotel

This filter will only pass messages where the hotelCode matches dcacy;

```
<HTNG_SimpleFilter>
  <matchAny>
     <name>hotelCode</name>
     <value>dcacy</value>
  </matchAny>
</HTNG_SimpleFilter>
```

5.9.2 Match Multiple Hotels

This filter will accept any message for any of the hotel codes listed in the value elements.

```
<HTNG_SimpleFilter>
  <matchAny>
    <name>hotelCode</name>
    <value>dcacy</value>
    <value>dcafi</value>
    <value>dcass</value>
  </matchAny>
</HTNG_SimpleFilter>
```

This could also have been written as:

```
<HTNG_SimpleFilter>
  <matchOne>
    <name>hotelCode</name>
    <value>dcacy</value>
  </matchOne>
  <matchOne>
    <name>hotelCode</name>
    <value>dcafi</value>
  </matchOne>
  <matchOne>
    <name>hotelCode</name>
    <value>dcass</value>
  </matchOne>
</HTNG_SimpleFilter>
```

5.9.3 Match Multiple Hotels using a Regular Expression

This example uses a regular expression to match any hotel whose 5 letter code begins with dca.

```
<HTNG_SimpleFilter>
  <matchAny>
    <name>hotelCode</name>
    <value>^dca..</value>
  </matchAny>
</HTNG_SimpleFilter>
```

Note: in this particular case matchAny, matchAll, and matchOne will all have the same result. If you wanted to match everything but hotels whose letter code begins with dca use the following;

```
<HTNG_SimpleFilter>
  <matchNone>
    <name>hotelCode</name>
    <value>^dca..</value>
  </matchNone>
</HTNG_SimpleFilter>
```

5.9.4 Match for Arrival Dates

This example shows a rule that matches a single hotel and an arrival date within 14 days. It uses a special rule "occursBefore". This rule looks at the arrival date and matches if it is within value=14 days of the current date.

```
<HTNG_SimpleFilter>
  <matchAll>
    <matchAny>
      <name>hotelCode</name>
      <value>dcacy</value>
    </matchAny>
    <matchAny>
      <name rule="occursBefore" type="integer">arrivalDate</name>
      <value>14</value>
    </matchAny>
  </matchAll>
</HTNG_SimpleFilter>
```

Using an ISO 8601 Interval type for the value element would allow the creation of a rule like occursWithin which would match any date or time within the provided time interval. This example matches any date between 01 January 2015 and 15 January 2015.

```
<HTNG_SimpleFilter>
  <matchAll>
    <matchAny>
       <name>hotelCode</name>
       <value>dcacy</value>
    </matchAny>
    <matchAny>
       <name rule="occursWithin" type="interval">arrivalDate</name>
       <value>2015-01-01/2015-01-31</value>
    </matchAny>
  </matchAll>
</HTNG_SimpleFilter>
```

5.9.5 Match room status change and floor

This message will match on any room status change to vacated or ready for a specific floor:

```
<HTNG_SimpleFilter>
  <matchAll>
    <matchAll>
      <name>floor</name>
      <value>concierge</value>
    </matchAll>
    <matchAny>
     <name>roomStatus</name>
     <value>vacated</value>
     <value>ready</value>
    </matchAny>
```

```
  </matchAll>
</HTNG_SimpleFilter>
```

6 Appendices

6.1 Glossary of Terms

For the purpose of this document, the following terms have been defined as follows:

Term	Definition
Subscription Manager	A web service that accepts requests to manage, get the status of, renew and/or cancel subscriptions on behalf of an event source.
Subscriber	A system that sends requests to create, renew and/or cancel subscriptions.
Event	An action or occurrence detected by a program that may be handled by the program and subsequently delivered to an event Subscriber.
Event Source	A producer of event information that wishes to keep all interested systems updated on subscribed events.
Event Sink	A system that wishes to consume information relating to events on a regular basis.
Notification	A message sent to indicate that an event, or events, has occurred.

6.2 Implementation Notes

It is strongly advised all programmers and system designers working on an integration using Event Notification thoroughly read and understand all of the concepts in the HTNG and W3C Eventing specifications before writing code. Both the subscribing system and event source system must have services compatible to send and receive messages.

6.2.1 Event Sinks are Message End Points

While it may be obvious, an Event Sink must able to receive, accept, and process a notification message when sent by the source. This means that subscribing systems need to implement and host a web service. This service must be accessible to the Event Source and may require proxies or ports to be opened in firewalls when notifications are coming from an external source.

6.2.2 Message Flexibility

An Event Source may be flexible in the implementation of the notification messages. It is possible within this framework to support SOAP messages, XML messages, or JSON messages)for example). It is also possible for the subscriber to identify the preferred format for the messages if this feature is supported by the implementation.

6.2.3 Separation of Event Source and Subscription Manager

A Subscription Manager is not required to distribute events generated from an Event Source; it may simply notify the Event Source of the new Subscription and allow each source to send the notifications directly to the subscribers. Alternatively an Subscription Manager may be designed to redistribute the notifications from the Event Sources to the subscribers.

6.2.4 Event Redistribution

An Event Redistribution system can be used to share events between multiple systems. Redistribution can effectively obscure how events generated by the partner are distributed within the client organization. This allows changes within the client organization without the need to communicate those changes to the partner.

6.3 Links

W3C Web Services Eventing standard – published 13 December 2011
W3C WS-Addressing standard – published 9 May 2006
OASIS WS-Security standard – published 1 February 2006
OASIS WS-ReliableMessaging – published 2 February 2009

6.4 Referenced Documents

The following table shows the documents upon which this document depends:

Document Title	Location/URL
OpenTravel Alliance Specifications	http://www.opentravel.org/Specifications/Default.aspx
W3C WS-Eventing Specification	http://www.w3.org/TR/ws-eventing/
W3C WS-Addressing	http://www.w3.org/TR/ws-addr-core/

6.5 Security

Messages may contain information that require protection and should not be exposed between the source and destination systems.

In order to protect this information it is important to be able to protect the contents of the messages and to verify the identities of the systems in the roles of Subscription Manager, Subscriber, and Event Source. In some cases it may also be important to verify the Event Sink or target destination of the message.

In an Event-based system there are a number of potential security threats. Spoofing is one concern. An Event Sink is spoofed when a malevolent system pretends to be an Event Sink to capture messages that it is not authorized to receive. An Event Source is spoofed to send falsified data to Event Sinks in order change system behavior. Another concern is the man-in-

the-middle attack where a malevolent system sits between a source and a sink. In this middle position the malevolent system captures, observes, and potentially modifies the messages that are exchanged.

To protect against these and other threats the implementer should consider the following needs;

- A Subscription Manager needs to validate that the Subscriber is authorized to subscribe to receive the event notification
- A Subscription Manager also needs to validate that the requestor is authorized to make; Renew, Unsubscribe, GetStatus, and HTNGGetStatus requests for a Subscriber with an existing subscription.
- An Event Sink needs to validate that a notification message was generated by a legitimate Event Source
- Notification Messages may need protection from exposure or alteration between the source and destination
- Subscription End messages need be authenticated to ensure the message originated from the proper system

To summarize; each destination system should have a means to confirm the source system and each source system may need to confirm the destination if the message and the message needs to be protected between the two systems.

In most cases the implementer should consider exchanging all messages over a secure transport layer like HTTPS using Transport Layer Security (TLS) and using Mutual Authentication to verify both the sender and the receiver. WS-Security provides alternative solutions supporting message integrity and confidentiality from source to final destination.

6.5.1 Notification Security

The Event Sinks should verify the origin of each notification to prevent notifications from being spoofed by false event sources. If Notifications contain sensitive information then appropriate protections need to be applied. This normally implies using a secure transport channel or other means of protecting and hiding the message content during transit.

There are a number of ways to verify the identity of an Event Source including authentication of the source when establishing a secure communications channel. An alternative is for the Event Source to provide identity credentials or a secure identity token with each message. These credentials should be verified with each message.

The Event Source may also want to verify the Event Sink to protect against man-in-the-middle or spoofed-receiver attacks. When using web protocols this is often accomplished by using Mutual Authentication when establishing a secure Transport Layer Security (TLS) session.

6.5.2 Subscription Security

Once a subscription has been created it needs to be handled as a protected resource. It is recommended that TLS or another secure transport layer be used for Renew, GetStatus, Unsubscribe, and HTNGGetStatus messages. The Subscription Manager or other End Point receiving these messages should authenticate and verify the requestor validating that the requestor is authorized to perform the operation.

If the SubscriptionID is used to verify the message sender then the SubscriptionID needs to be protected to prevent its use by a malevolent sender.

6.5.3 Subscription End Messages

Subscription End messages are sent from an Event Source to a Subscriber to notify that the Subscription is ending early. The source of the Subscription End message should be authenticated to ensure the message originated from the proper system. This normally means that some method must be provided for the Subscriber to validate the source of the message. Subscription End messages should be carried over a secure transport layer allowing the source to be validated using Mutual Authentication when establishing the TLS session. A protected identity or protected SubscriptionID token or source system credentials could alternatively be used to verify the sender's identity.

6.6 Faults

The following table outlines the possible faults that can be thrown by various messages outlined in this specification.

Spec	Subcode	Reason	Methods	Description
W3C	DeliveryModeRequestedUnavailable	The requested delivery mode is not supported	Subscribe	This fault is sent when a Subscribe request specifies a delivery mode that is not supported by the event source. Optionally, this fault may contain a list of supported delivery mode URIs in the Detail property.
W3C	InvalidExpirationTime	The expiration time requested is invalid.	Subscribe Renew	This fault is sent when a Subscribe request specifies an expiration time that is in the past or an expiration duration of zero.
W3C	UnsupportedExpirationType	Only expiration durations are supported.	Subscribe Renew	This fault is sent when a Subscribe request specifies an expiration time and the event source is only capable of accepting expiration

Spec	Subcode	Reason	Methods	Description
				durations; for instance, if the event source does not have access to absolute time.
W3C	FilteringNotSupported	Filtering is not supported.	Subscribe	This fault is sent when a Subscribe request contains a filter and the event source does not support filtering.
W3C	FilteringRequestedUnavailable	The requested filter dialect is not supported.	Subscribe	This fault is sent when a Subscribe request specifies a filter dialect that the event source does not support. Optionally, this fault may contain a list of supported filter dialect URIs in the Detail property.
W3C	EventSourceUnableToProcess	Text explaining the failure; e.g., "The event source has too many subscribers".	Subscribe	This fault is sent when the event source is not capable of fulfilling a Subscribe request for local reasons unrelated to the specific request.
W3C	UnableToRenew	Text explaining the failure; e.g., "The event source has too many subscribers".	Renew	This fault is sent when the event source is not capable of fulfilling a Renew request for local reasons unrelated to the specific request.
W3C	InvalidMessage	The message is not valid and cannot be processed.		If a request message does not comply with the corresponding outline listed above, the request MUST fail and the event source or subscription manager MAY generate the following fault indicating that the request is invalid
HTNG	InsufficientPermissions	The subscriber does not	Subscribe Renew Unsubscri	This fault is sent when the subscriber soap header credentials do not have

Spec	Subcode	Reason	Methods	Description
		have permissions to complete the operation.	be	permissions to complete the requested operation.
HTNG	UnknownSubscriber	The specified subscriber is not known by the event source.	HTNG_GetSubscriptionStatus	This fault is sent when the subscriber id is not found in the subscription manager.

6.6.1 Sample Fault Response

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">

  <soap:Header>
    <wsa:Action>http://www.w3.org/2011/03/ws-evt/Renew</wsa:Action>
    <wsa:MessageID>urn:uuid:bd88b3df-5db4-4392-9621-aee9160721f6</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://www.example.com/MyEventSink</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://www.example.org/oceanwatch/SubscriptionManager</wsa:To>
  </soap:Header>

  <soap:Body>
    <!--Below is an example of the HTNG fault code raised
    when the subscriber id is not found in the subscription manager.-->
    <soap:Fault>
      <!-- The faultcode element below is the code uniquely identifying the error
      raised. It is intended for use by software to provide an algorithmic mechanism
      for identifying the fault.-->

      <faultcode>s12:UnknownSubscriber</faultcode>

      <!-- The faultstring element below is a text description of the error raised.
      It is intended to provide a human readable explaination of the error.-->

      <faultstring>The subscriber isn't known by the event source</faultstring>

      <!-- The faultactor element below is intended to provide information about which
      SOAP node on the SOAP message path caused the fault to happen. -->

      <faultactor>Renew</faultactor> <!-- Is this correct? -->

      <!-- The detail element information item is intended for carrying application
      specific error information related to the SOAP Body.  The absence of the detail
      element information item indicates that a SOAP Fault is not related to the
      processing of the SOAP Body. This can be used to find out whether the SOAP Body
      was at least partially processed by the ultimate SOAP receiver before the fault
      occurred, or not. -->
      <detail>
        <Error xmlns="WSSoapException">
```

```
          <ErrorNumber>48187</ErrorNumber>
          <ErrorMessage>Subscriber 58090 was not renewed because the subscriber is not
known</ErrorMessage>
          <ErrorSource>Hotel Application Server </ErrorSource>
        </Error>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

6.6.2 Client Fault Handling Pseudo-code

```
private void function renewSubscription(String subscriptionID) {

  RenewSubscriptionRequest request = new RenewSubscriptionRequest()
  Request.setSubscriptionID(subscriptionID);

  RenewSubscriptionResponse response = null;

  try {
    response = service.Renew(request);
  }
  catch (SOAPException e) {
    handleSOAPException e);
  }
}

private void handleSOAPException(SOAPException e) {
  switch (e.faultsubcode) {
    CASE “InvalidMessage”:
      SendAlert(323);
      break;
    CASE “InsufficientPermissions”:
      SendAlert(512);
      break;
    CASE  “UnknownSubscriber”:
      SendAlert(253);
      break;
  }
}
```