Sabre® Event Notification Services

Developer Guide

ENS Production release 2.4

February 2020

Sabre Corporation

3150 Sabre Drive, Southlake, TX 76092

www.sabre.com

Contents

Preface

This topic discusses what you need regarding hardware, communications, and software in order to use *Event Notification Services* (ENS).

Platform

Event Notification Services is a platform-neutral offering that adheres to cross-platform Web services standards. It does not restrict the environment for developing or deploying solutions. Solutions for ENS can be deployed in any environment that supports HTTP (Hypertext Transfer Protocol), including Microsoft Windows® and UNIX®, without requiring bridges or special interfaces.

A wide variety of languages and development toolkits can be used to design and deploy the solutions. It ascribes to the specifications developed by Web Services Eventing (WS-Eventing), using technologies backed by Sun Microsystems, Inc., IBM Corporation, and Microsoft Corporation.

Network and Communications

Your communications and connectivity must provide Internet access, both inbound and outbound, on port 443. Your firewall must allow ENS inbound and outbound traffic on the right port.

Notice: Ensure that your firewall is opened to allow traffic from the following ENS IP addresses. A high-speed Internet connection (DSL, cable, T1, etc.) is needed.

Certification Environment	Production Environment
151.193.58.254	151.193.255.254
151.193.119.48	151.193.119.1

Software

You can implement any solution or Web server that supports the following communications and messaging protocols that are compatible with your platform of choice:

- HTTP 1.1
- SOAP 1.1
- TLS 1.2
- XML 1.0

Resources

The following resources are needed to use ENS:

- Sabre Dev Studio – Sabre's developer portal with documentation for all components of Event Notification Services (*ENS*) including this developer's guide, sample Event Sync application and user guides (incl. videos) on how to use the ENS GUI: link.
- *ENS* GUI – Sabre's *ENS* GUI is available for users when they need to configure their subscriptions for ENS notifications in both Certification (CERT) and Production (PROD) environments: link.

Terminology

The following terms are related to *ENS*.

Term	Description
delivery mode*	The mechanism by which notifications are delivered from the source to the Event Sync. For *ENS*, this is push.
Event Engine	The "service provider" to USG for *Sabre* Event *Notification Services* subscription Web services. It updates the subscription database. This is a component of *ENS*.
Event Sync*	In this document, an Event Sync is an application that receives the notifications on a URL. It is an Event Sync listener on the system of the subscribing entity that waits for an Event Sync that will trigger the application code.
event source*	According to the WS-Eventing specification, it is a Web service that sends notifications and accepts requests to create subscriptions. In this document, an Event Sync source is the component of ENS that produces messages for subscriptions, once an Event Sync comes from the Sabre Pub/Sub system.
event topic	A type of Event for which notification messages are sent. Event topics for ENS are PNR changes, PSS Queue changes and Sabre Profile changes.
notification*	The WS-Eventing specification uses the term notification and describes it as follows: A one-way message sent to the Event Sync advising that an Event has occurred. This document uses the terms notification and notification message.
Pub/Sub	Sabre's enterprise Publish/Subscribe system for internal use only.
push mode*	The WS-Eventing specification describes this as a delivery mechanism where the source sends Event messages [notifications] to the Sync as individual, unsolicited, asynchronous SOAP messages.
Sabre Publishers	Producers of messages for PNR changes, Queue changes, and Profile changes.

*The source of this information was taken from http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/, submitted 15 March 2006.

Product Specifications and Standards

WS-Eventing, a public draft of the *Web Services Event Syncing Specification,* was released in August 2004. This is one of many "WS-"proposals. It was offered by BEA Systems, Microsoft Corporation, Tibco Software, Inc., Computer Associates International, Inc., Sun Microsystems, and IBM Corp. It is a unified protocol proposed for Java and Microsoft .NET Framework "Events."

The specification is published on the following sites:

- http://www.w3.org/Submission/WS-Eventing/ - The proposed submission to the W3C of the specification
- http://www-128.ibm.com/developerworks/library/specification/ws-eventing/ - From IBM's link, you can download the WS-Eventing specification, and the WSDL and XML schema definition.
- http://www.idevnews.com/IntegrationNews.asp?ID=99

Product Features and Benefits

Benefits

ENS benefits businesses in the following ways:

- Reduces scan costs – Eliminates excessive Sabre host system scans by alerting you of changes instead of you generating system inquiries
- Supports simplified and streamlined robotic activity
- Improves customer satisfaction with timely notifications to your customers
- Generates opportunities for increased revenue by providing value-added travel management services to your customers
- Gives you immediate access to booking information and changes to reservations to enforce policy compliance for corporate travel

In addition, using *ENS* reduces long-term IT costs due to the following:

- Elimination of dedicated network connections to the *Sabre®* system
- Reduction in application development and maintenance costs
- A shortened development life cycle
- Lower infrastructure costs because the infrastructure is reduced (Internet connectivity)
- Platform and technology independence (interoperability)
- Lower integration and deployment costs. Using open standards and Web Services Event Syncing standards enable the use of interoperable publish/subscribe systems

Features

High Reliability

ENS is designed to deliver messages with high reliability. The goal is for all components that comprise the infrastructure to be available 24x7, with the exception of scheduled maintenance. A window for maintenance activities will be established at a regular time and communicated to subscribing entities. When the need for maintenance arises, *ENS* will be unavailable during this maintenance window only, on an as needed basis.

Fast Delivery

The ENS infrastructure provides nearly real-time delivery of notifications. The notification is pushed when an Event Sync occurs.

Time to Live and Throttling

Time to Live

The system within Sabre that publishes the message determines the time to live for the message. Each message can have a different time to live value, and instances may arise when the time to live value expires before the message is delivered to the subscriber.

Throttling of Notification Delivery per Endpoint

While *ENS* starts a maximum of 6 TPS, the Event Sync does not always receive the maximum of 6 TPS. The delivery of 6 TPS depends on factors such as network speed, delays, or other reasons. For example, in the first second, suppose that *ENS* starts 6 TPS, and then a network delay occurs. In the next second, ENS starts another 4 transactions. During this time, the customer receives 10 transactions. The quantity of 10 TPS exceeds 6 TPS because the original 6 TPS in the first second were delayed until the next second.

Delivery Retry/Failure

Delivery Retry

If *ENS* cannot deliver the message, it retries delivery a maximum of 3 times as soon as possible.

If the system fails to deliver notifications after the maximum quantity of 10 retries, *ENS* goes into a suspended state and stops trying to send notification messages for 5 minutes. The subscriber URL is marked as failed or not alive and *ENS* stops attempting to deliver notifications.

Failed Endpoints

ENS attempts to deliver a specific notification a maximum of 3 times.

If *ENS* cannot deliver notifications 10 times, the endpoint is marked as failed. *ENS* discontinues sending notifications to the endpoint, and the subscribing entity receives an outage email notification saying that the endpoint is marked as failed.

After 5 minutes, *ENS* tries to reconnect to the failed endpoint by pinging the URL using the `GET` method. *ENS* continues to try to reconnect every 5 minutes to ascertain whether the endpoint is alive.

If *ENS* is unable to deliver notifications after 72 hours, *ENS* stops attempting to deliver notifications to the endpoint.

The failed notifications are stored for a period of 6 months for re-delivery.

When *ENS* tries to ping the failed URL at any point and discovers that the URL is alive, the delivery of notifications is resumed.

Outage Email Notifications

Whenever the endpoint of the subscribing entity experiences an outage, *ENS* sends an outage email notification to the email address of the administrator of the subscribing entity, as provided during registration.

- The first outage email notification is sent immediately.
- The second outage email notification is sent after 1 minute.
- The third outage email notification is sent after 5 minutes.
- A fourth outage email notification is sent after 10 minutes.
- A fifth outage email notification is sent after 30 minutes and then again after 1 hour.
- An outage email notification is sent every hour thereafter, for a maximum of 72 hours. If after 72 hours, ENS is still unable to deliver the notifications, the notifications will be stored for a period of 6 months for re-delivery.
- If one or more failed delivery attempts occur within 1 minute, *ENS* sends 1 outage email notification for all failed attempts within that 1 minute.
- When the endpoint is marked as failed, a final outage email notification is sent, warning that notifications will stop being delivered to the endpoint.

Subscribing entities may also receive outage email notifications for other circumstances - for example, when problems are encountered with the network or the Event Sync application.

Resume Delivery

When the endpoint of the Event Sync is alive again, delivery of new notifications resumes.

Access and Registration

Information and values you provided to your Sabre account manager during the product registration phase affects your subscriptions, Event Sync application behavior, and the behavior of *ENS*.

You supply or confirm the following:

- An iPCC for activating *ENS*. You will use this to create and manage subscriptions.
- The security credentials you want to activate for *ENS*. You will use these to create and manage subscriptions. Security credentials consist of username, password, organization, and domain.
- A valid URL for your Event Sync application.
- A contact email address for ENS to send notices about failed delivery of notifications and non-responding URLs.

  Note: All subscribing entities must manage their contact information on the Account Settings section of the subscription management GUI.
- Your PCCs for subscriptions to PNR/Queue change notifications.
- The configuration you want for PNR change notifications.
- Your Queue numbers for Queue Event notifications. A maximum of 512 queues per PCC per subscriber is allowable, however, not necessarily possible due to the pre-assigned Queues in the Sabre system. Subscribing entities must provide the range of Queue numbers during registration so those Queues can be included in the publishing system that monitors Queues.

Rules for Subscriptions

- Subscribing entities can create subscriptions to Event topics using a valid URL. Each subscription may specify a different URL. The receiving URLs must listen to *ENS* messages on HTTPS port 443.
- There is no maximum quantity of PCCs for a single subscription to PNR Change Event (the same PCC may be added to multiple subscriptions).
- Queue Event subscriptions can specify a maximum of 512 Queues for a single subscription. Queues are supported in the format *PCC.nnn*, where *PCC* is a PCC that the subscriber registered and *nnn* is a 3-digit numeric Queue number. The range of Queues must be from 0 – 512.
  Example: Queue Number Format Example: PCC.070
- You establish the dates and time to start and end every subscription. Delivery of notifications ceases as soon as the expiration date of a subscription is met.
- When creating multiple subscriptions to a single Event topic, each subscription must have unique parameters.
- A subscription becomes active on the *Sabre* Event Notification servers 5 minutes after it is created.

Requirements for Event Sync

- The *ENS* infrastructure supports HTTPS SSL connections on port 443. Your firewall must allow both inbound and outbound traffic.
- Your Event Sync must send an acknowledgement to *ENS* in the prescribed format within 10 seconds after the delivery of every notification message. You must acknowledge delivery whether you subscribe using the GUI or subscription services.
- When *ENS* calls the endpoint of your Event Sync using the GET method, the endpoint must respond with **HTTP 202 Accepted**. If your Event Sync fails to respond in this way, *ENS* interprets this as an inactive or failed endpoint and takes other action.
- You can design your Event Sync application to verify that the notification messages it receives are not duplicates of the same message.

Certification (CERT) and Production (PROD) Access

- Upon initial provisioning of *ENS* users are only granted access to Sabre's Certification (CERT) environment. This means that only changes in Sabre's CERT environment will trigger notifications.
- Production (PROD) activation will only occur once the customer has successfully completed the "Validate URL" step in the *ENS* GUI and saved their subscription in CERT. PROD activation will occur within 2 US business after this step is completed by the ENS user.

  Note: The above activation schedule may be affected by Sabre system freezes, typically around US holidays. Contact you Sabre Account Manager for additional details.
- The expectation is that users will test their event synch application in Sabre's CERT environment first, ensuring they can respond with the appropriate **HTTP 202 Accepted** to the messages sent by *ENS*.

Event Topic Notification Messages

This section describes Event notifications for the three Event topics currently available, a general description of the SOAP message header, and the Consolidated Notifications Management features.

PNR Change Notifications

This message notifies subscribers about changes to PNRs associated with one or more PCCs. When an end transaction on a PNR occurs or the PNR is saved otherwise, a notification message is delivered to the URL of the Event Sync soon thereafter.

Note: Two options for configuring PNR changes you want to be notified about are available—you must choose one option during registration.

- **No Filters**. (Default configuration). An Event is triggered for an end transaction on a PNR create ***and*** all subsequent end transactions on the PNR. If any of the updates are on monitored fields, the notification includes a change indicator.
- **Updates Filter**. An Event is triggered for an end transaction whenever any update is made to an **existing** PNR. If any of the updates are on monitored fields, the notification includes a change indicator.

Fourteen fields are monitored for changes if you subscribe to the PNR topic.

Change Indicator	Notification sent
Itinerary	When any itinerary segment data changes (added/deleted/modified)
Name	When the passenger name field is changed
Phone	When the Phone field (9) is changed
ReceivedFrom	When the received from field (6) is changed
Ticketing	When the ticketing fields (7 or 8) is changed
PassengerAddress	When the address field (W-) is changed.
PassengerDetail	When email field (PE) is changed
PreReservedSeats	When the Pre-Reserved Seat data is changed
HostFacts	When airline facts (SSR and OSI) data is changed
GeneralFacts	When general facts (SSR and OSI) data is changed
AgencyAccountingData	When accounting lines changed
Remarks	When the remarks field (5) is changed
FrequentTraveler	When the frequent traveler field (FF) is changed
MiscTicketing	When the MISC ticketing data (MCO, PTA etc.) is changed in the PNR

For notifications that include change indicators, the indicators communicate the types of changes that have been made to a PNR, but not the actual changes to the content. For example, a switch tells you that the Frequent Traveler information has been changed, but not what the change is.

Sample Payload: PNR Creation

An example of the SOAP body with payload for a PNR creation notification is shown as follows:

```
  <soap-env:Body>
    <swse:CCC.PNRCHNG>
      <swse:OWNPCC>IPCC</swse:OWNPCC>
      <swse:HOMEPCC>IPCC</swse:HOMEPCC>
      <swse:Locator>SVRDXH</swse:Locator>
      <swse:EventTimeStamp format="yyyy-MM-dd hh:mm:ss.ffffff">2019-04-25
03:15:06.000320</swse:EventTimeStamp>
      <swse:ChangeIndicators>
        <swse:Indicator name="First Transaction">
          <swse:hasChanged>Y</swse:hasChanged>
        </swse:Indicator>
      </swse:ChangeIndicators>
    </swse:CCC.PNRCHNG>
  </soap-env:Body>
```

Sample Payload: PNR Update

An example of the ENS XML message (full SOAP envelope) for a PNR change notification is shown as follows:

```
  <soap-env:Body>
    <swse:CCC.PNRCHNG>
      <swse:OWNPCC>IPCC</swse:OWNPCC>
      <swse:HOMEPCC>IPCC</swse:HOMEPCC>
      <swse:Locator>SVRDXH</swse:Locator>
      <swse:EventTimeStamp format="yyyy-MM-dd hh:mm:ss.ffffff">2019-04-25
03:16:51.000380</swse:EventTimeStamp>
      <swse:ChangeIndicators>
        <swse:Indicator name="Itinerary">
          <swse:hasChanged>Y</swse:hasChanged>
        </swse:Indicator>
        <swse:Indicator name="Ticketing">
          <swse:hasChanged>Y</swse:hasChanged>
        </swse:Indicator>
        <swse:Indicator name="General Facts">
          <swse:hasChanged>Y</swse:hasChanged>
        </swse:Indicator>
        <swse:Indicator name="Remarks">
          <swse:hasChanged>Y</swse:hasChanged>
        </swse:Indicator>
      </swse:ChangeIndicators>
    </swse:CCC.PNRCHNG>
  </soap-env:Body>
```

This example shows that four changes have been made to monitored fields in record locator SVRDXH: itinerary, ticketing, passenger details and remarks.

PSS Queue Event Notifications

This notification informs you about changes to Sabre Queues that are associated with one or more PCCs and message Queues. The notification provides the Queue depth of all Queues that have changed since the previous Queue change notification was sent.

When a Queue placement action occurs in Sabre, the following happens:

- The count is updated for the corresponding Sabre Queue.
- A notification message is sent immediately, unless a notification message was sent within the preceding minute. In this case, no message is sent.

Notice: Queue Event Notifications are only delivered once a minute. For example, if there are continuous placements on a queue then the notification will be triggered only for the 1st queue placement of the minute. Subsequent placements in that minute will not trigger a notification. The 1st queue placement in the next minute will trigger a notification and subsequent queue placements in that next minute will not trigger a notification and so on. This is a restriction put in place to reduce the number of notifications generated that could impact system performance.

Event notifications are driven only when actual Sabre Queue activity occurs.

Queues are polled at one-minute intervals. If no changes have been made within the one-minute window, a notification is not generated.

Sample Payload: Queue Event Notification

An example of the SOAP body with payload for a Sabre Queue Event notification is shown as follows. For complete information about the notification message for Queue Event topic, consult the Queue Event Notification description and related documents on Sabre Dev Studio.

```
<soap-env:Body>
    <swse:QueueElem>
        <swse:UniqueId>HIJKLM</swse:UniqueId>
        <swse:PartitionCode>AA</swse:PartitionCode>
        <swse:PCC>PCC3</swse:PCC>
        <swse:QueueNo>060</swse:QueueNo>
        <swse:PrefactoryCode></swse:PrefactoryCode>
        <swse:QueueDepth>0025</swse:QueueDepth>
    </swse:QueueElem>
</soap-env:Body>
```

The preceding notification is for message Queue 060 belonging to PCC3. Notifications for message Queues send an empty `PrefactoryCode` element; the `UniqueId` may or may not be blank.

Sabre Profiles Event Notifications

Sabre Profiles is Sabre Travel Network's profile system. It is a relational database that:

- Improves an agency's ability to use traveler, corporation, supplier and agency profile information.
- Streamlines the process of creating and managing traveler information in a structured manner.
- Makes critical information available at all transaction touch points.

Sabre Profiles APIs provide easy and flexible access to the *Sabre Profiles* database, enabling integration with other applications.

Customers that subscribe to Event Notification Services for *Sabre Profiles* updates will be notified about changes to Profiles residing within a PCC. Notifications are generated for the following Event types:

- Create Profile
- Update Profile
- Delete Profile
- Restore Profile

Sample Payload: Profile Create

```
  <soap-env:Body>
    <swse:Sabre_OTA_EventNotification xmlns="http://www.sabre.com/eps/schemas">
      <swse:ProfileEvent>
        <swse:Action Timestamp="2011-06-15T18:43:38.728Z" Type="CREATE"/>
        <swse:TPA_Identity ClientCode="TN" ClientContextCode="MYS" DomainID="XXXX"
ProfileName="Sample Profile Name" ProfileTypeCode="TVL" UniqueID="123456789"/>
      </swse:ProfileEvent>
    </swse:Sabre_OTA_EventNotification>
  </soap-env:Body>
```

Profile consists of Subject Areas like: Email, Address, and Payment Form.

Payloads for Profile Update and Profile Move contain information about Subject Areas which were changed.

Sample Payload: Profile Update

```
  <soap-env:Body>
    <swse:Sabre_OTA_EventNotification xmlns="http://www.sabre.com/eps/schemas">
      <swse:ProfileEvent>
        <swse:Action Timestamp="2011-06-15T18:43:38.728Z" Type="UPDATE"/>
        <swse:TPA_Identity ClientCode="TN" ClientContextCode="MYS" DomainID="XXXX"
ProfileName="Sample Profile Name" ProfileTypeCode="TVL" UniqueID="123456789"/>
        <swse:SubjectArea Name="PersonName"/>
        <swse:SubjectArea Name="Telephone"/>
        <swse:SubjectArea Name="Email"/>
        <swse:SubjectArea Name="Address" OrderSequenceNo="1"/>
        </swse:ProfileEvent>
      </swse:Sabre_OTA_EventNotification>
    </soap-env:Body>
```

SOAP Header Format for Notification Messages

ENS obtains content from Certification & Production systems. Notifications are delivered using the HTTP POST method.

Notification messages are formatted as XML and are OTA XML compliant. They are delivered as SOAP envelopes with a header and body. The notification portion of the message is wrapped inside the body in a root element that is specific to the Event topic.

The Event header delivers information about the Event topic, subscription, and notification message.

The SOAP body includes the root element and payload, that is, the actual notification content.

Sample Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
  xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:swse="http://wse.sabre.com/eventing"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
  <soap-env:Header>
    <eb:MessageHeader eb:version="1.0" soap-env:mustUnderstand="1">
      <wse:MySubscription>IPCC</wse:MySubscription>
      <swse:EventTopic>WSE.QUEUE.CCC.PNRCHNG</swse:EventTopic>
    </eb:MessageHeader>
    <wsa:Action>http://wse.sabre.com/EventSource/notification</wsa:Action>
    <wsa:MessageID>wsevlc201fb4266cb-018a-4278-8593-07246b37acaa</wsa:MessageID>
    <wse:Identifer>03915295-fb61-47b8-95f4-1b87d361d44d</wse:Identifer>
    <wsa:To>https://myeventsynch</wsa:To>
  </soap-env:Header>
  <soap-env:Body>
  ...
  ...
  </soap-env:Body>
</soap-env:Envelope>
```

Common Elements of a SOAP Header for a Notification

Header elements which are especially helpful are described as follows:

Element	Description
<wse:MySubscription>	The PCC that the subscribing organization registered
<swse:EventTopic>	The value that identifies the Event topic for this notification, shown as follows: PNR Change Events - WSE.QUEUE.CCC.PNRCHNG Queue Events - WSE.QUEUE.PSS.SABREQUEUE Profiles Events - WSE.QUEUE.PPP.PROFILECHNG
<wsa:MessageID>	The unique identification number of the message
<wse:Identifer>	The unique subscription identification number
<wsa:To>	The URL or IP address and port of the Event Sync, where the notification is delivered

Consolidated Notifications Management

The Consolidated Notifications Management feature allows ENS customers to select by frequency and/or by number the PNR, Queue and Profile notifications. Host messages are compared to each ENS customer profile. If an Event Notification Management option is associated with the message, ENS will hold the message until the notification condition has been met and will “batch” all the messages as a single notification.

Sample Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
  xmlns:soap- env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:swse="http://wse.sabre.com/eventing"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
  <soap-env:Header>
    <eb:MessageHeader eb:version="1.0" soap- env:mustUnderstand="1">
      <wse:MySubscription>IPCC</wse:MySubscription>
      <swse:EventTopic>WSE.QUEUE.CCC.PNRCHNG</swse:EventTopic>
    </eb:MessageHeader>
    <wsa:Action>http://wse.sabre.com/EventSource/notification</wsa:Action>
    <wsa:MessageID>3ce859c4-5e4c-4b2f-9a08- f656beacd883</wsa:MessageID>
    <wse:Identifer>ea2737cb-00e3-4ba2-aa99- 8970cee70e42</wse:Identifer>
    <wsa:To>https://myeventsynch</wsa:To>
  </soap-env:Header>
  <soap-env:Body>
    <swse:BATCH.NOTIFICATION TYPE="PNRCHNG">
```

```
      <swse:CCC.PNRCHNG>
        <swse:OWNPCC>IPCC</swse:OWNPCC>
        <swse:HOMEPCC>PCC3</swse:HOMEPCC>
        <swse:Locator>KDNIRN</swse:Locator>
        <swse:EventTimeStamp format="yyyy-MM-dd hh:mm:ss.ffffff">2019-05-30
09:49:52.000493</swse:EventTimeStamp>
        <swse:ChangeIndicators>
          <swse:Indicator name="Itinerary">
            <swse:hasChanged>Y</swse:hasChanged>
          </swse:Indicator>
        </swse:ChangeIndicators>
      </swse:CCC.PNRCHNG>
    </swse:BATCH.NOTIFICATION>
  </soap-env:Body>
</soap-env:Envelope>
```

Event Sync Application Design

When an Event occurs that meets the criteria you specified in the subscription process, you automatically receive notification as a structured XML message on the URL you supplied in the subscription. Polling is not needed.

Some of the facets of your design for the Event Sync application are discussed below:

- Event Sync acknowledgements
- Duplicate delivery of a notification message
- Responses to pings by *ENS*

Acknowledgements by the Event Sync

This information is applicable to all subscribing entities. Your Event Sync must acknowledge receipt of all notifications. The purpose of the acknowledgement is to tell *ENS* that a specific notification was received.

You are advised to set confirmation to occur immediately after delivery, and to respond within 1 second.

ENS requires your Event Sync to comply with either of two formats or types of acknowledgements, discussed in the topics that follow.

HTTP Header Format

The first option is sending an HTTP header without data.

Example

```
HTTP/1.1 202 Accepted
```

For information about HTTP status codes, see the following URL:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html.

SOAP Message Format

The second option is sending a SOAP message as shown in the below sample.

Sample Payload

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soap-env:Envelope xmlns:soap- env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:eb="http://www.ebxml.org/namespaces/messageHeader"
  xmlns:swse="http://wse.sabre.com/eventing"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing">
  <soap-env:Header>
    <eb:MessageHeader eb:version="1.0" soap-env:mustUnderstand="0">
      <wsa:MessageID>56d6b85a-467e-4da2-a6ac-3860f244912a</wsa:MessageID>
      <wse:Identifer>7e183baf-9f61-46c5-b9af-4e3006811550</wse:Identifer>
    </soap-env:Header>
    <soap-env:Body>
      <swse:Response>
        <swse:Status>OK</swse:Status>
        <swse:Received>1184603384290</swse:Received>
        <swse:Processed>48</swse:Processed>
      </swse:Response>
    </soap-env:Body>
  </soap-env:Envelope>
```

All elements must be sent with the values as shown. Variable values are described as follows.

Element	Description
`<wsa:MessageID>`	Send the message ID parsed from wsa:MessageID in thenotification you are acknowledging.
`<wse:Identifer>`	Send the value parsed from wse:Identifer in the notification youare acknowledging.
`<swse:Status>`	You must pass one of the following values: OK or ERROR If ERROR is sent, *ENS* continues to deliver notifications. This data is collected internally.
`<swse:Received>`	Send the time when your Event Sync received the notification, in milliseconds. The Java method currentTimeMillis() calculates this value. For the method, see the following URL: https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/System.html#currentTimeMillis()
`<swse:Processed>`	(Optional) This is the duration of time that it took your Event Sync to process the notification, in milliseconds.

Duplicate Delivery of Notifications

If *ENS* fails to receive an acknowledgement from an Event Sync, *ENS* tries to send the notification a second or third time, as usual. Because of this, it is possible that the Event Sync will receive duplicates of the same notification.

This can happen if the notification message is delivered to the Event Sync and the Event Sync sends an acknowledgement, but the acknowledgement is not delivered. This could be due either to a network outage, a connection time-out, or a component of the *ENS* architecture being unavailable. Because the acknowledgement is not received, *ENS* sends the same message a second time.

Delivery failures are usually the result of malfunctions, most often due to network problems. Subscribing entities are responsible for their network connections. If their networks are unreliable, they may receive duplicate messages often.

ENS counts duplicate notifications as 1 notification, therefore, a count of 1 notification is provided to the billing system in the case of duplicate deliveries. Consequently, subscribing entities will only be billed for 1 notification.

Sample Scenario

In this scenario, *ENS* sends a notification twice.

1. *ENS* sends a notification to the Event Sync.
2. The Event Sync receives the notification and sends an acknowledgement.
3. A brief network outage occurs.
4. *ENS* does not receive the acknowledgement in the format `HTTP 202 Accepted`
5. *ENS* sends the same notification to the Event Sync.
6. The Event Sync receives the notification and sends a second acknowledgement.
7. *ENS* receives the acknowledgement from the Event Sync.
8. *ENS* provides a count of 1 notification to the billing system.

Verifying the Uniqueness of a Message ID

When you design your Event Sync application, you can verify whether the message ID in notification messages has been received, and thereby avoid handling duplicate data. Test or parse the value of `<wsa:MessageID>` in the notification's header. Every notification has a unique message ID. When the same message is sent multiple times, the message ID is the same. The following line shows the message ID.

```
<wsa:MessageID>17aa9b55-7cac-4327-80d9-e5f2b98967b3</wsa:MessageID>
```

Responding to Pings

When *ENS* pings the endpoint of your Event Sync, you are required to respond. All Event Sync must support this. It is used to verify that the Event Sync is operational.

The flow is as follows:

- *ENS* calls the URL of your Event Sync using the HTTP GET method.
- Your HTTP server responds. This can be an HTTP header followed by the string `OK`.
- *ENS* accepts the response, and ensures that the response includes data.

Notifications Driven by Automated Applications

If you have robotic or other automated applications whose activities trigger Event with *ENS*, your Event Sync will receive notifications for the affected Event topics.

You must maintain an awareness of your automated actions that trigger Events, and design your Event Sync to process notifications that are triggered by your own actions. Several sample scenarios that illustrate this follow.

Sample Scenario 1

- Within *Sabre* systems, an end transaction occurs on a PNR for one of your PCCs.
- A PNR change notification is delivered to your Event Sync.
- Your robotic application processes the PNR change notification by adding remarks to the PNR and ending the transaction.
- A PNR change Event occurs due to the addition of remarks.
- *ENS* delivers a PNR change notification to your Event Sync related to the change your robotic application made.

The situation may be further complicated if your robotic application places the PNR on queue, driving a PNR change notification that your robotic application triggers.

Sample Scenario 2

- A PNR is created and an end transaction occurs for one of your PCCs.
- You receive a PNR change notification. The PNR is created and waiting to be ticketed.
- Your robotic application tickets the PNR and places it on Queue.
- You receive a Queue Event notification for an Event triggered by your own robotic.

Subscription Management Sample Flows

An overview of the flow for using the subscription management ENS GUI follows.

Note: Dev Studio reference on how to use the ENS GUI: link.

Subscription Management ENS GUI Flow

When you use the subscription ENS GUI, you subscribe to Event topics one at a time. For each subscription, you choose the criteria or parameters you want.

Event Notification Flow

Your Event Sync application receives the notification messages and parses the content. Other applications or processes can re-use the content any way you want.

- When an Event is triggered that matches the parameters you specified, the notification is delivered in structured XML format via HTTPS on port 443 of the IP address or URL provided in the subscription request. These messages are easily received like any browser page posted to your Web server, using any good HTTP or Web server.
- Upon delivery of every notification, your application or server acknowledges receipt of every notification message to *ENS*. Only a normal HTTP acknowledgement is needed.
- Your Web application can perform any business logic you want to manipulate the content in the notification message, for example, it can parse the content and either request additional content using other systems or re-use the content in other ways.

Errors

SOAP Faults and System Errors

When *ENS* cannot process a request successfully, client errors are returned to the *endpoint* as SOAP faults in the SOAP Body of the SOAP Envelope.

SOAP Fault Message Format

Sample Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP- ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>WSE.AuthorizationFailed</faultcode>
      <detail>
        <ErrorCode>10044</ErrorCode>
        <ErrorID>1184792966907-955213</ErrorID>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The message includes the following error-related elements:

- `<faultcode>` returns faults.
- `<faultstring>` returns an accompanying description of the fault code.
- `<ErrorCode>` and `<ErrorID>` indicate internal errors. `ErrorID` is a unique ID that corresponds to the exception. Please provide technical support with values for both of these elements.

Common <Fault> Messages

A partial list of faults is shown in the following table.

<faultcode> and <faultstring>	Reason
<faultcode>WSE.AuthorizationFailed</faultcode>	The security credentials provided are any of the following: Invalid Not authorized to subscribe to the specified topic Not authorized to use the selected subscription parameters
<faultcode>Duplicate.Subscription</faultcode> <faultstring>Subscription already exists</faultstring>	A subscription with the same parameters, (event topic, dates, parameters, and Event Sync URL) is already present.
<faultcode>Invalid.Request</faultcode> <faultstring>Incoming SOAP Request is not valid</faultstring>	The SOAP message has missing or invalid parameters. The data format is incorrect.
<faultcode>Invalid.Subscription.Identifier</faultcode>	The subscription identifier passed in the SOAP message request is invalid.
<faultcode>Invalid.Customer.Identifier</faultcode>	The value passed in <MySubscription> is not authorized to use *ENS.*
<faultcode>Throttle.Error</faultcode>	Event Notification Services Subscription Manager is at its peak value.

Error Format for a Duplicate Subscription

An example of a SOAP message with a fault response is shown below.

Sample Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope">
  <soap-env:Header/>
  <soap-env:Body>
    <soap-env:Fault>
      <faultcode>soap-env:Duplicate.Subscription</faultcode>
      <faultstring>Subscription already exists</faultstring>
      <detail>
        <StackTrace>errors.general.DUPLICATE_SUBSCRIPTION</StackTrace>
      </detail>
    </soap-env:Fault>
  </soap-env:Body>
</soap-env:Envelope>
```

StackTrace Errors

If you receive a StackTrace similar to the following example, verify that the SOAP request and parameters are correct, and send the request again. If the errors continue, provide the SOAP request you are sending and the faultcode, including StackTrace, to Technical Support.

Sample Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Invalid.Request</faultcode>
      <faultstring>Incoming SOAP Request is not valid</faultstring>
      <faultactor>UNREADABLE_MESSAGE</faultactor>
      <detail>
        <StackTrace>errors.general.INVALID_SOAP_REQUEST</StackTrace>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Outage Email Notifications

ENS sends outage email notifications to administrators of the subscribing organization when it is unable to deliver notifications.

The outage email notifications contain the following information:

- The system that generated the information, which is the WS-Eventing engine, ENS.
- The host where the outage occurred.
- The time when the notification email message was sent.
- A description of the outage, which includes Event Sync endpoint, Event topic, and variable information, as explained in the following examples.

Examples

Example 1: Outage Notification Providing Number of Undelivered Notifications

Subject:	ENS Delivery Outage Notification
Message:	The Event Notification Service is unable to deliver notifications to the following end point **Endpoint URL:** https://www.test.com/wse_event_sync Following Queue messages are undelivered since last notification Queue UniqueId Time XXXX.244 ETPILX 2019-02-13 22:51:16.759 XXXX.244 GHIJKL 2019-02-13 23:53:42.060 Notification topic: WSE.QUEUE.PSS.SABREQUEUE Number of undelivered notifications: 198

In Example 1, *ENS* tried to connect to the endpoint URL in the outage email message 3 times. After 3 times, Event Notification Services stopped trying to deliver the notification. In Number of undelivered notifications, the count = 1 undelivered notification because Event Notification Services makes 3 attempts for a single notification.

Example 2: Outage Notification with Limit for Undelivered Notifications Exceeded

Subject:	ENS Delivery Outage Notification
Message:	The Event Notification Service is unable to deliver notifications to the following end point **Endpoint URL:** https://www.test.com/wse_event_sync Following Queue messages are undelivered since last notification Queue UniqueId Time XXXX ABCDEF 2019-02-18 04:48:32.646 XXXX GHIJKL 2019-02-18 04:49:48.929 XXXX MNOPQR 2019-02-18 04:50:05.705 The limit for undelivered notifications has been exceeded! *** Specified endpoint is marked as failed! *** Please take appropriate actions immediately to activate the end point.

The limit of undelivered notifications is 10. When *ENS* cannot deliver 10 notifications, the endpoint is marked as failed.