

# ***Sabre® Web Services***



## ***Guide to Accessing and Consuming Orchestrated Sabre® Web Services***

March 1, 2013 v2.1

*Sabre® Web Services: Guide to Accessing and Consuming Orchestrated Sabre® Web Services,  
March 1, 2013 v2.1*

© 2003-2005, 2007-2013 Sabre Holdings Inc. All rights reserved.

This documentation is the confidential and proprietary information of Sabre Inc. Any unauthorized use, reproduction, preparation of derivative works, performance, or display of this document, or software represented by this document, without the express written permission of Sabre Inc., is strictly prohibited.

Sabre, Sabre Holdings, Sabre Travel Network, and Sabre Web Services are trademarks and/or service marks of an affiliate of Sabre Holdings Corporation. All other trademarks, service marks, and trade names are the property of their respective owners.

**Disclaimer of Warranty and Limitation of Liability**

This software and any compiled programs created using this software are furnished “as is” without warranty of any kind, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. No oral or written information or advice given by Sabre, its agents or employees shall create a warranty or in any way increase the scope of this warranty, and you may not rely on any such information or advice.

Sabre does not warrant, guarantee, or make any representations regarding the use, or the results of the use, of this software, compiled programs created using this software, or written materials in terms of correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of this software and any compiled applications created using this software is assumed by you. Neither Sabre nor anyone else who has been involved in the creation, production or delivery of this software shall be liable for any direct, indirect, consequential, or incidental damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use of or inability to use such product even if Sabre has been advised of the possibility of such damages.

Sabre Holdings Inc.

3150 Sabre Drive, Southlake, TX 76092

Tel: 682 605 1000

[www.sabre-holdings.com](http://www.sabre-holdings.com)

## Table of Contents

Introduction to Orchestrated <i>Sabre® Web Services</i> .....	4
Overview .....	4
Benefits of utilizing Orchestrated Sabre Web Services .....	5
Orchestrated <i>Sabre Web Services</i> Overview .....	6
Sessioning Requirements .....	7
Delineating Successful & Failing Transactions .....	8
Message Processing .....	10
HaltOnError Flags .....	10
HaltOnError Flag .....	10
IgnoreOnError Flag .....	10
Fatal Errors .....	11
Orchestrated Services Description .....	12
PassengerDetailsRQ .....	12
Segment Status Handling .....	14
Tools and Artifacts .....	17
WSDL and Schema Documentation .....	17
Supporting Documentation .....	17
Technical Support .....	18

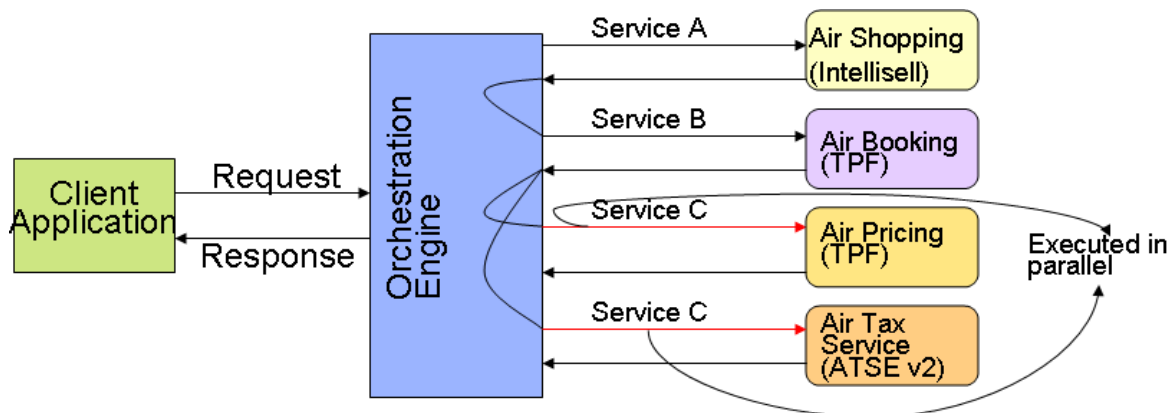
# Introduction to Orchestrated *Sabre® Web Services*

The travel industry has seen a steady growth in ecommerce during the last few years. At the same time, on-line travel agencies, travel suppliers, and traditional agencies with on-line presence are continuing to face several challenges, such as building brand loyalty, high look-to-book ratios, and unsatisfactory customer experience. Web services technology has emerged as a key enabler for application developers in travel agencies, airlines, and providers of related content to deliver products and services to overcome some of these business challenges. *Sabre Web Services* now offers Orchestrated *Sabre Web Services* to help agencies, airlines, and other customers reduce their overall travel content integration costs.

## Overview

An orchestrated Web service easily automates the process of a commonly performed business function or workflow. Orchestrated *Sabre Web Services* bundle several functions/operations into a single Web service call. This type of service consists of transactions that are mapped to individual Web services, and executed either sequentially, in parallel, or both.

An example of how a single orchestrated service request is fulfilled by executing multiple Web service requests is illustrated in the following execution model of an orchestrated service. A single service request call will provide you with a response that retrieves your content and performs the processing for you seamlessly. By using orchestrated services, customers can develop client applications faster and may see an improvement in the overall response time to offer a better customer experience.



## ***Benefits of utilizing Orchestrated Sabre Web Services***

- Applications can be deployed to production sooner. By using pre-built workflow functions that have been architected and designed to work together, development time and quality assurance are reduced, shortening the time to place an application into production.
- Integration costs are lowered. Orchestrated services help reduce development complexity, letting clients accomplish more with a single service call.
- Learning curve is shortened. Newcomers to the travel industry or Web services can use these services more easily than the low level services.
- Gain access to content and business logic in multiple Sabre data sources. One orchestrated service may obtain content from multiple sources and orchestrate business logic, shielding clients from making multiple points of entry to various systems.
- Improved response time. By avoiding the round trip latency inherent with multiple individual low level service calls to perform a function, responses are faster.
- Improved bandwidth utilization. Because the HTTP overhead associated with multiple low level Web service calls is avoided, use of bandwidth is improved.
- Improved end-to-end performance. An orchestrated Web service gives clients an opportunity to optimize existing applications.
- Ability to adapt to changing market needs quickly. Clients can build new products and services faster by integrating an orchestrated service versus multiple service calls.
- The majority of the operations available inside an orchestrated service are optional. As such, the client application has the flexibility to control which operations are executed, allowing it to mix and match functions to meet its specific needs.
- The client application that is consuming orchestrated services has the flexibility to control the error/exception processing during the execution of the low level services within the orchestrated service. Client applications have the ability to halt the execution of subsequent service operations by passing in the appropriate "HaltOnError" attribute at the individual transaction level.
- Each 2.x orchestrated request and response is designed so that individual calls are delineated by an "stl:element" element that can be used to associate particular XML nodes to specific low level calls. Client applications can easily identify the offending node using the "stl:element" element returned in the error message.

## Orchestrated Sabre Web Services Overview

There are presently two 2.x-based orchestrated *Sabre Web Services* available for consumption: EnhancedAirBookRQ and PassengerDetailsRQ.

In terms of feature/function:

- EnhancedAirBookRQ is used to book and price an air itinerary, and retrieve applicable tax-related information.
- PassengerDetailsRQ is used to create a basic Passenger Name Record (PNR).

These services can be utilized together, and they can also be used with existing TPF Connector-based low level services.

### Example 1:

If a client application has the desired air itinerary along with all of the relevant passenger information, the client can invoke:

- EnhancedAirBookRQ – to book and price the air itinerary.
- PassengerDetailsRQ – to add the passenger-related information, associate the passenger-related information to the pricing related information, and end the record.

### Example 2:

If a client application has the relevant passenger information, and wants to add it while the customer shops the client can invoke:

- BargainFinderMaxRQ, or OTA\_AirLowFareSearchLLSRQ – to shop for an air itinerary.
- PassengerDetailsRQ – to add the passenger-related information.
- EnhancedAirBookRQ – to book and price the air itinerary.
- PassengerDetailsRQ – to add any additional passenger-related information, i.e. seats, SSRs, etc, associate the passenger-related information to the pricing related information, and end the record.

## ***Sessioning Requirements***

Orchestrated *Sabre Web Services* assumes that the requesting client application has already authenticated into the *Sabre Web Services* infrastructure via a SessionCreateRQ message, and obtained a valid BinarySecurityToken prior to invoking any orchestrated calls.

The consuming application must also be aware that it cannot reuse the BinarySecurityToken for any other request until the orchestrated Web service call completes. If the client application disregards this point and sends requests while the orchestrated service is processing it runs the risk of corrupting the orchestrated transaction and generating errors.

Once the orchestrated call completes the BinarySecurityToken is free to be reused.

## ***Delineating Successful & Failing Transactions***

Each 2.x orchestrated response message contains several ".../stl:ApplicationResults/STL:Element" occurrences that are used to tie a particular operation's response status back to the associated request XML node that caused it to be invoked. This is useful for troubleshooting purposes. If warnings or errors are generated they can quickly be associated back to particular request operations.

### **Example 1:**

```
<PassengerDetailsRS xmlns="http://webservices.sabre.com/sabreXML/2011/10"
Version="2.0.0">
  <stl:ApplicationResults xmlns:stl="http://services.sabre.com/STL/v01"
status="Complete">
    <stl:Success timeStamp="2012-05-14T07:00:31.238-05:00">
      <stl:SystemSpecificResults>
        <stl:Element>TravellitineraryAddInfoRQ</stl:Element>
      </stl:SystemSpecificResults>
    </stl:Success>
    <stl:Success timeStamp="2012-05-14T07:00:31.341-05:00">
      <stl:SystemSpecificResults>
        <stl:Element>MiscSegmentSellRQ</stl:Element>
      </stl:SystemSpecificResults>
    </stl:Success>
    <stl:Success timeStamp="2012-05-14T07:00:31.409-05:00">
      <stl:SystemSpecificResults>
        <stl:Element>AddRemarkRQ</stl:Element>
      </stl:SystemSpecificResults>
    </stl:Success>
    <stl:Success timeStamp="2012-05-14T07:00:31.456-05:00">
      <stl:SystemSpecificResults>
        <stl:Element>EndTransactionRQ</stl:Element>
      </stl:SystemSpecificResults>
    </stl:Success>
    <stl:Success timeStamp="2012-05-14T07:00:31.669-05:00">
      <stl:SystemSpecificResults>
        <stl:Element>RedisplayReservation</stl:Element>
      </stl:SystemSpecificResults>
    </stl:Success>
  </stl:ApplicationResults>
  ...
</PassengerDetailsRS>
```

In this example all of the designated request operations succeed.

### **Example 2:**

```
<PassengerDetailsRS xmlns="http://webservices.sabre.com/sabreXML/2011/10"
Version="2.0.0">
  <stl:ApplicationResults xmlns:stl="http://services.sabre.com/STL/v01"
status="Incomplete">
    <stl>Error type="BusinessLogic" timeStamp="2012-05-14T07:03:47.166-05:00">
      <stl:SystemSpecificResults>
```



```

                                <stl:Message>.RESTRICTED.NOT ENT BGNG
WITH</stl:Message>

    <stl:Message>ERR.SWS.HOST.ERROR_IN_RESPONSE</stl:Message>

    <stl:ShortText>ERR.SWS.ORCH.PROVIDER_ERROR</stl:ShortText>
        <stl:Element>TravelltnitaryAddInfoRQ</stl:Element>
        </stl:SystemSpecificResults>
    </stl>Error>
</stl:ApplicationResults>
</PassengerDetailsRS>

```

In this example the request operation failed, and an error was generated during during the TravelltnitaryAddInfoRQ operation.

## Message Processing

Orchestrated *Sabre Web Services* offer clients several options for controlling what happens when errors are encountered:

1. .../HaltOnError flags
2. .../IgnoreOnError flag

### HaltOnError Flags

The HaltOnError flags take the form of ".../HaltOnError" attributes located throughout the 2.x-based Orchestrated *Sabre Web Services* message set. This attribute pair controls whether or not the processing of the orchestrated service is stopped if an encountered while an operation processes.

### HaltOnError Flag

**Request example (Child):**

```
<PassengerDetailsRQ Version="2.0.0" xmlns="http://webservices.sabre.com/sabreXML/2011/10"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
  <PostProcessing HaltOnError="true" IgnoreAfter="false" RedisplayReservation="true">
    <EndTransactionRQ HaltOnError="true">
      <EndTransaction Ind="true">
        <Source ReceivedFrom="SWS TESTING"/>
      </EndTransactionRQ>
    </PostProcessing>
  ...
</PassengerDetailsRQ>
```

In this request, the client application opted to set the .../EndTransactionRQ child ".../HaltOnError" attribute to "true." This means that if this particular operation fails during invocation the orchestration engine will recognize the failure and halt any subsequent requests/processing.

### IgnoreOnError Flag

The client application can utilize the ".../IgnoreOnError" element/attribute pair to ignore the entire transaction if an error is encountered during processing.

```
<EnhancedAirBook Version="2.0.0" xmlns="http://webservices.sabre.com/sabreXML/2011/10"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <HaltOnError Ind="true"/>
  <IgnoreOnError Ind="true"/>
  ...
</EnhancedAirBookRQ>
```

Please note that the ".../IgnoreOnError" flag works only when the transaction is stopped due to an error, so it usually is combined with the ".../HaltOnError" attribute.

Note, it is possible that the ignore operation could also fail, so the client application needs to check the response to ensure that it succeeded.

## Fatal Errors

There are two kinds of fatal errors which stop the processing regardless of the values set via the “.../HaltOnError” attribute or the “.../IgnoreOnError@Ind” element/attribute:

1. *Timeout:* Each Orchestrated Sabre Web Service is set to timeout at two minutes. The timeout can be decreased by defining, “soap-env:Envelope/soap-env:Header/eb:MessageHeader/eb:MessageData/eb:Timeout” value in the request (in seconds). However, we do not recommend that clients utilize this functionality, because it could cause premature errors if the back-end content systems are slow to respond.
2. *Internal error:* When an error occurs outside of the operations specified in the Orchestrated *Sabre Web Services* message it is considered to be an internal error. A good example of this sort of error would be a connection refused error encountered when the orchestration engine is communicating with the back-end content systems.

# Orchestrated Services Description

## ***PassengerDetailsRQ***

The PassengerDetailsRQ service allows client applications to create shell PNRs containing names, phone numbers, email addresses, customer numbers, passenger types, address information, remarks, and retention segments. The client application has the ability to end the transaction once processing is complete, or to leave the transaction open in the AAA for subsequent processing. If this option is chosen the client application can add additional information into the PNR via existing TPF Connector-based *Sabre Web Service* calls, or other orchestrated service calls before ending the transaction.

PassengerDetailsRQ orchestrates the following operations:

1. SabreCommandLLSRQ (N\*(Profile Name)(end-item)NM)
2. TravelltineraryAddInfoLLSRQ (-, 9, PE, DK, PD, W-, 7)
3. MiscSegmentSellLLSRQ (0OTH, 0MCO, 0INS, 0PTA)
4. SpecialServiceLLSRQ (3, 4)
5. AddRemarkLLSRQ (5)
6. AirSeatLLSRQ (4G)
7. ARUNK\_LLSRQ (0AA)
8. SabreCommandLLSRQ (PQL(record number)(\* or -)(name number)
9. EndTransactionLLSRQ (6, E)
10. QueuePlaceLLSRQ (QP)
11. TravelltineraryReadLLSRQ (JX PNR)
12. IgnoreTransactionLLSRQ (I)

Client applications have the ability to pass any of the services contained within the workflow outlined previously, and in terms of responses they can opt to only receive the record locator generated as a result of the process, or to receive the entire PNR generated as a result of the process via the TravelltineraryReadLLSRQ response message.

## ***EnhancedAirBookRQ***

The EnhancedAirBookRQ service allows client applications to book and price flight segments via a single Web services call. This service also provides the ability to request air tax information. This is useful for clients that operate their own negotiated or private fares databases since most of the time all that they require in regards to pricing is the relevant tax-related information.

EnhancedAirBookRQ orchestrates the following operations:

1. IgnoreTransactionLLSRQ (I)
2. OTA\_AirTaxRQ
3. OTA\_AirBookLLSRQ (JA)
4. TravelltineraryReadLLSRQ (JX PNR)
5. OTA\_AirPriceLLSRQ (WP)
6. TravelltineraryReadLLSRQ (JX PNR)
7. IgnoreTransactionLLSRQ (I)

EnhancedAirBookRQ allows a client application to perform an ignore transaction prior to booking to ensure that the AAA is clear. This service also has the ability to control what needs to be done when UC segments are encountered. This service allows client applications to specify a wait interval after booking in order to give carriers a chance to respond with updated segment status. In conjunction with this wait interval client applications can also specify for the system to redisplay the itinerary looking for UC segments up to ten times. In the event that a UC segment is encountered, the client application can specify that the system halt processing for further action from the client application, i.e. a new request utilizing different marriage connection logic, etc...

During the subsequent pricing step, the orchestration engine will also make note of the value contained in ".../OTA\_AirPriceRQ/PriceComparison@AmountSpecified" which can then be used to by client applications to compare the actual price being stored during PNR creation against the price gathered during shopping.

After a successful OTA\_AirPriceLLSRQ response is received the orchestration engine will extract the value contained in "OTA\_AirPriceRS/PricedItineraries/PricedItinerary/AirItineraryPricingInfo/ItinTotalFare/TotalFare@Amount," and return that value along with the initial specified fare amount in the response to allow customers to determine if there was a fare increase between the shopping and booking transaction.

Finally, the client application has the ability to ignore the transaction upon successful processing.

In terms of responses client applications can opt to only receive the flight segments generated as a result of the OTA\_AirBookLLSRQ message, or to receive the entire PNR generated as a result of the process via the TravelltineraryRS message.

## Segment Status Handling

As mentioned previously, the EnhancedAirBookRQ service has a provision for checking segment status after initial booking to ensure that the air itinerary can be successfully priced. Air segments that have “UC,” or “NN” status cannot be priced.

To successfully utilize this functionality client applications need to:

1. Set the appropriate segment status codes, i.e. UC, NN, to halt processing via “.../OTA\_AirBookRQ/**HaltOnStatus@Code**.”
2. Set the appropriate number of times, 1-10, to redisplay the reservation via “.../OTA\_AirBookRQ/**RedisplayReservation@NumAttempts**,” so that the segment status can be checked.
3. Set the appropriate wait interval, 0-10000 milliseconds, between redispays via “.../OTA\_AirBookRQ/**RedisplayReservation@WaitInterval**,” in order to give the carrier an opportunity to respond to the sell message. Some carriers can actually take up to seven seconds to respond to a sell message.

Note: if the carrier responds with “SS” immediately upon initial booking the orchestration engine will override any values set via “.../HaltOnStatus,” and “.../RedisplayReservation” and move onto the subsequent operations specified in the request message since SS segments can be priced.

### Example:

```
<EnhancedAirBookRQ Version="2.0.0" xmlns="http://webservices.sabre.com/sabreXML/2011/10"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <OTA_AirBookRQ HaltOnError="true">
    <OriginDestinationInformation>
      <HaltOnStatus Code="UC"/>
      <HaltOnStatus Code="NN"/>
      <FlightSegment ...>
        ...
      </FlightSegment>
      <RedisplayReservation NumAttempts="4" WaitInterval="1500"/>
    </OriginDestinationInformation>
  </OTA_AirBookRQ>
  ...
</Enhanced_AirBookRQ>
```

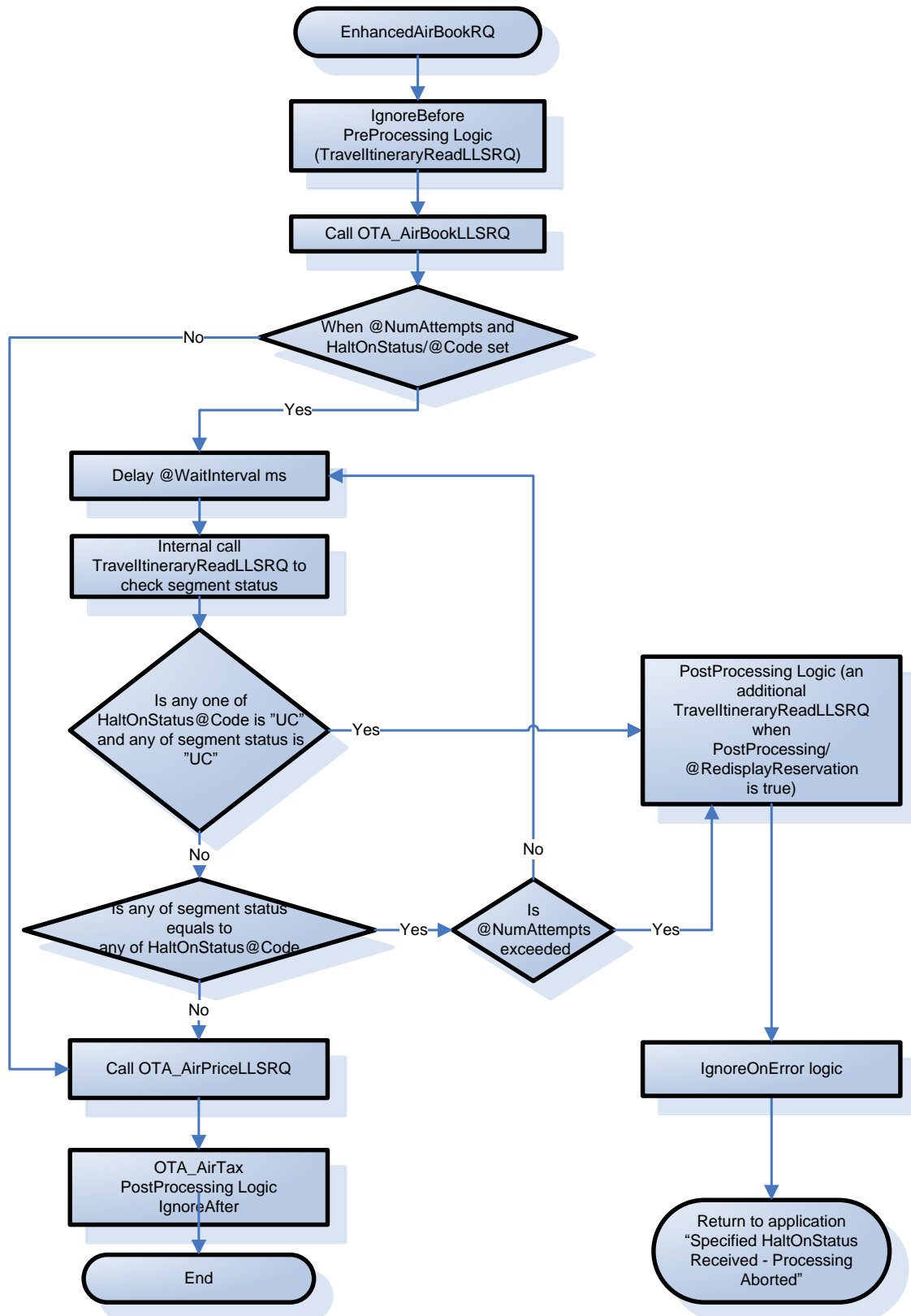
In this example the client application has specified to halt subsequent processing if a carrier returns “UC,” or “NN” status. The client application has also specified for the system to redisplay the reservation up to four times over the course of six seconds (1500\*4) to check segment status.

### Notes:

- If a “UC” is encountered at any point during the six seconds the orchestration engine will immediately halt processing.
- If the carrier responds with “SS” at any point during the six seconds the orchestration engine will override any values set via “.../HaltOnStatus,” and

“.../RedisplayReservation” and move onto the subsequent operations specified in the request message.

- If the segment status remains “NN” at the end of the six second interval the orchestration engine will halt processing.





## ***Tools and Artifacts***

---

### **WSDL and Schema Documentation**

Each Sabre Web service has a WSDL, schema, and design documents available on the Developer Resource Center (DRC).

Customers can download the latest WSDL and schema documents via the DRC, located at <https://drc.sabre.com> or directly from the environment URL. Please note that a login is required to access the DRC.

### **Supporting Documentation**

Additional supporting documentation on the DRC includes Web service descriptions, sample request and response design XML documents, sample request and response payloads, as well as other service-specific documents.

When a new TPF Connector-based Web services release is deployed to certification or production, updated supporting documentation is also made available via the DRC.

During the customer acceptance testing phase, customers can refer to these documents to make the necessary client code updates to take advantage of the new services/enhancements.

## ***Technical Support***

---

If you have any questions or need assistance, please contact our *Sabre Web Services* Global Customer Support Center.

### **Telephone:**

When reporting production or other critical/time sensitive issues, please contact us via the telephone:

- **USA:** 800-678-9460
- **Canada:** 682-605-5570
- **International:** 598-2-518-6020, or your regional Sabre Software help desk.

### **Email:**

Email is monitored 24 x 7 with a response within 24 hours or less:

- [webservices.support@sabre.com](mailto:webservices.support@sabre.com)

Providing the support desk with the necessary files at the time of initial contact improves our ability to troubleshoot and provide a timely resolution.

In order to better serve you please note the following:

- Please include the Sabre Pseudo City Code (PCC) where the issue is occurring.
- When reporting an issue with Sabre Web Services, input and output payloads are required. Please attach the payloads as separate files, and name them clearly.
- To help ensure that our environment is free of viruses, our policy mandates that all messages received by Sabre from external sources follow special file name guidelines. File names must end in ".sabre.zip" or the zipped attachment will be removed by the e-mail server (for example, "docs.zip" would need to be renamed to "docs.sabre.zip").
- If your correspondence is regarding a previously reported issue, please include the service incident ("SI") number in the subject line of your message.

© 2012 Sabre Inc. All rights reserved. This documentation is the confidential and proprietary information of Sabre Inc. Any unauthorized use, reproduction, preparation of derivative works, performance, or display of this document, or software represented by this document, without the express written permission of Sabre Inc., is strictly prohibited.

Sabre, Sabre Holdings, and Sabre Web Services are trademarks and/or service marks of an affiliate of Sabre Holdings Corporation. All other trademarks, service marks, and trade names are the property of their respective owners.