HotelBeds XML Interface Specification

developer's guide

v2.4.4

August 10

Hotelbeds Accommodation & Destination Services
Camí de Son Fangós, 100 - Torre A, 5ª Planta
07007 - Palma de Mallorca
Spain

I. Document Control

Version	Issue	Description	Operations
2.4.4	02/08/10	- Hotel Service document updated to 1.2.0 - New elements (HotelbedsCommonTypes.xsd): 1. Service/DirectPayment 2. Service/AcceptedCardTypes 3. Service/NetPrice 4. Service/Commission 5. SellingPrice 6. Service/SellingPrice 7. ServicePrice/NetPrice 8. ServicePrice/Commission 9. ServicePrice/SellingPrice 10. ExtraOverride 11. ServicePrice/ExtraOverride 12. AdditionalCostList/Currency 13. AdditionalCostList/PvpEquivalent 14. PaymentCard 15. ContactData 16. PaymentData/PaymentCard 17. PaymentData/ContactData 18. ConfirmationServiceData/PaymentData 19. ServiceHotel/PackageRate - Added type StringLength0to25 on HotelbedsSimpleTypes.xsd - New element (ServiceAddRQ.xsd): ServiceAddRQ/ShowNetPrice - Updated HotelbedsCommonTypes.html from xsddocs. - Updated HotelValuedAvailRQ.html from xsddocs. - Updated ServiceAddRQ.html from xsddocs. - Updated diagrams from xsddocs - Modified section II.3 Version Policy	8
2.4.3	02/08/10	- Car Service document updated to 1.0.1 - Hotel Service document updated to 1.0.3 - Transfer Service document updated to 1.0.1 - Ticket Service document updated to 1.0.2 - Product Update document updated to Product Update Nex Gen v.1.3.2 - Modified “Clients requirements” from Acceptance Test section. - Modified “Steps of certification process” section. - Added new section “Workflow” on Best Practices. - Deleted “Compression” section from Best Practices. - Deleted “Product content & product images locally stored on clients” section from Best Practices. - New optional elements (HotelbedsCommonTypes.xsd): 1. ServiceTicketModality/ContentSequence added. 2. ServiceTransfer/CancellationPolicy added. 3. CoreRequest/@version added. 4. ProductTicket/TicketClass added. 5. ServiceCarSpecialEquip/Amount added. 6. ServiceCarSpecialEquip/Currency added. 7. ServiceCar/FlightNumber added. - Modified ProductZone/HotelAddress and ConfirmationServiceDataTransfer/HotelAddress type definition. - Updated ServiceAddRS_Transfer.xml example from /xml - Updated HotelbedsCommonTypes.html from xsddocs - Updated HotelbedsSimpleTypes.html from xsddocs - Updated diagrams from xsddocs - Modified FAQ (16). - Modified section Request Observations from PurchaseConfirm operation.	8

		- Sections CancelProtectionAdd and CancelProtectionRemove deleted. - Deleted CancelProtectionAdd and CancelProtectionRemove from /xsddocs documentation files and /xml. - Deleted FAQ (8) What is a cancel protection? - Deleted FAQ (9) Cancel protection and service cancellation fees. - Gzip compression is mandatory. Modified section III.1.	
2.4.2	21/08/08	- Ticket Service document updated to 1.0.1 - Hotel Service document updated to 1.0.2 - Product Update document updated to 1.3.1 - Added new FAQ (18). - Fixed section Request Observations from PurchaseConfirm operation. - Fixed complexType_Customer.png from xsddocs/diagrams.	10
2.4.1	26/06/08	- Deleted removed operations from /xsddocs documentation files. - Hotel Service document updated to 1.0.1	10
2.4.0	04/06/08	- Product Update document updated to 1.3.0 - Modified section Request observations from TicketDetail operation. - Modified section Request observations from HotelValuedAvail operation. - Modified section Response Observations from TicketValuation operation. - Modified section Response Observations from ServiceAdd operation (only ticket services). - Modified section Request Business rules from PurchaseConfirm operation (only ticket services). - Modified section Response Observations from PurchaseDetail operation (only ticket services). - Modified section Response Observations from PurchaseConfirm operation (only ticket services). - Fixed section Business Rules from TransferValuedAvail operation. - Added new FAQ (17). - Added new FAQ (16). - Fixed ServiceAddRQ_Transfer_IN. xml and ServiceAddRQ_Transfer_OUT.xml examples. - Modified section Request Business Rules at HotelValuedAvail operation. - Deleted HotelRoomTypeGroupList operation. - Deleted HotelCategoryGroupList operation. - Deleted HotelBoardGroupList operation. - Deleted DestinationGroupList operation. - Deleted infants from specification. HotelbedsCommonTypes.xsd: ProductHotel/InfantAge deleted. ServiceOccupancy/InfantCount deleted. ServiceTicketModality/InfantAge deleted. HotelbedsSimpleTypes.xsd: HotelbedsCustomerType type IN deleted. Xml examples and documentation updated without infants. - New simple type: HotelbedsSimpleTypes.xsd: StringLength1to10. - New optional elements: HotelbedsCommonTypes.xsd: Purchase/CommentList added. ConfirmationPurchaseData/CommentList added. - Modified section Request Business Rules at PurchaseConfirm operation. - Modified section Request Important Notices at ServiceAdd and PurchaseConfirm. Added reference to HotelPostalCode. - New optional elements: HotelbedsCommonTypes.xsd: ProductZone/HotelPostalCode added. ConfirmationServiceDataTransfer/HotelPostalCode	10

		added. - Deleted InsuranceValuedAvail operation. - Added note about encoding in section 3 Technical Information.	
2.3.0	13/12/07	- Product Update document updated to 1.2.0 - Acceptance Test document updated to 1.1.0. - Modified Section Workflows. Duplicated serviceAdd request to transfer services, one to type IN and another one to type OUT. - Added new FAQ: “How can I add a complete transfer service to a purchase?”. - Added new xml examples ServiceAddRQ_Transfer_IN and ServiceAddRQ_Transfer_OUT. - Added new FAQ about the final price of ticket services. - Modified attribute ‘PaginationRequestData/@itemsPerPage’ annotation. - Modified section Technical Information. Added note about HTTP parameter. - Modified section 6.1. Fixed IncomingOfficeList operation name. - Fixed IncomingOfficeList request data diagram. - Modified section Request Important Notices at CountryList, CarCountryList, HotelCountryList, TicketCountryList and TransferCountryList operations. - Added ServiceAdd xml examples for Ticket and Transfer services. - Modified section 2.1 and 3. Added note about wsdl files. - Modified section Request Important Notices at ServiceAdd and PurchaseConfirm operations. Described the way to reserve transfer services when transfer origin or transfer destination is a zone. -New optional element: HotelbedsCommonTypes.xsd: ProductZone/HotelName added. ProductZone/HotelAddress added. ConfirmationServiceDataTransfer/HotelName added. ConfirmationServiceDataTransfer/HotelAddress added. - Section 8.20. Added note about hotel contract. - Modified section Request Important Notices at ServiceAdd and PurchaseConfirm operations. Described the way to indicate transfer service travel information. - Element length change: HotelbedsCommonTypes.xsd: Product/Code is now up to 15 characters. Zone/Code is now up to 15 characters. Reference/FileNumber is now up to 267 characters. Zone content is now up to 255 characters. -New optional element: HotelbedsCommonTypes.xsd: Purchase/CreationUser added. -Section 8.19 Fixed. -Section 3.1 Fixed the endpoint URLs.	33
2.2.0	07/07	-Request business rules for TransferValuedAvail fixed. -PurchaseConfirm response important notices fixed. -Namespace error fixed in WDSL. -Request business rules for CarValuedAvail fixed. -New optional attribute: HotelbedsCommonTypes.xsd: Zone/@serviceType. -Element length change: HotelbedsCommonTypes.xsd: AgencyIdentification/Code is now up to 99999999. -Product Update document updated to 1.1.0 -Request important notices for HotelValuedAvail fixed.	33

2.1.0	23/05/07	-General text mistakes fixed. -ServiceAdd: Request important notices fixed. -PurchaseConfirm: Request important notices fixed. -Troubleshooting section added. -Error codes added. -Live URL fixed. -Acceptance Test document updated to 1.0.1. -Product Update document updated to 1.0.2. -Added a list of temporal hotel issues to ProductHotel complexType. -New element: HotelbedsCommonTypes.xsd PaymentData -New element: HotelbedsCommonTypes.xsd Purchase/PaymentData -Element removed: HotelbedsCommonTypes.xsd Purchase/PaymentType -New element: Added two new elements to ServiceTicketModality. -New complex type: HotelbedsCommonTypes.xsd ServiceTypeModalityType -New complex type: HotelbedsCommonTypes.xsd ServiceTypeModalityMode	33
2.0.0	01/04/07	First release of the XML on-line reservation interface system.	33

II. Introduction

II.1 Documentation organization

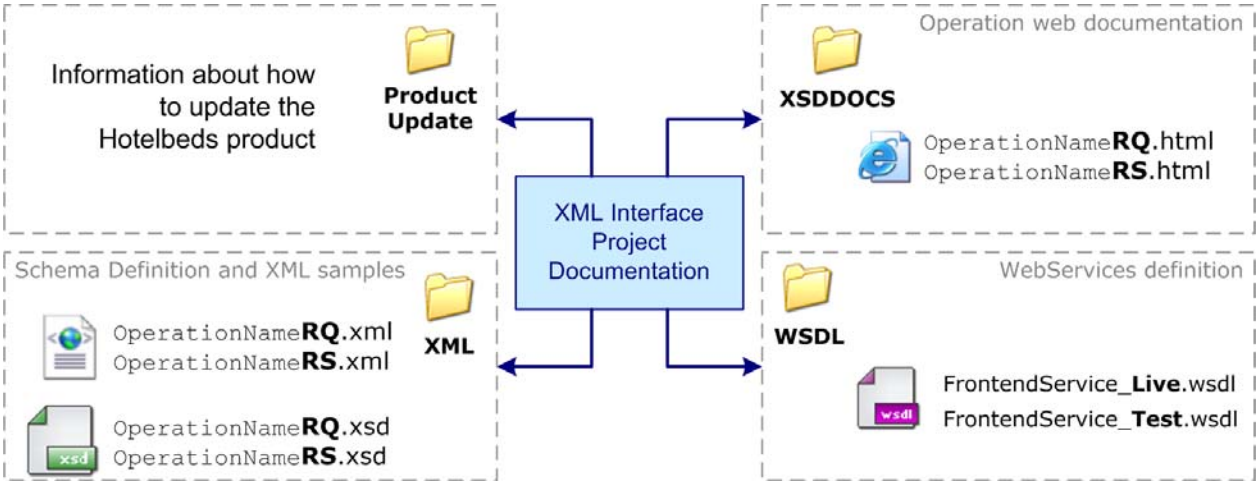


Figure 1 - Folder structure

II.2 XML Services integration documentation organization

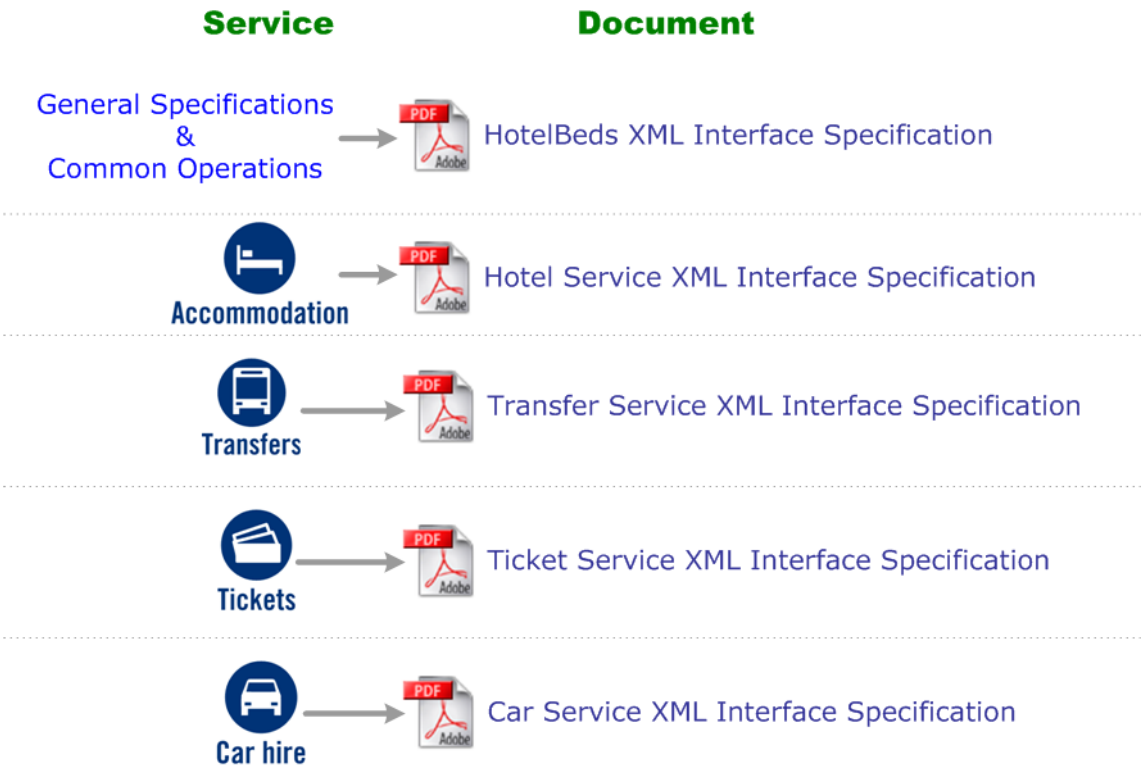


Figure 2 – Documentation files structure

II.3 Version Policy

II.3.1 DOCUMENT VERSION

The version number is divided in three different parts, major version, minor version and release. If the version of the document is 2.4.0 then:

2	4	0
Major version	Minor version	Release

- **Major version**
  Indicates the number of the document version, if this version number changes probably a brand new development in the interface is needed. There will not be backwards compatibility.

- **Minor version**
  Indicates minor changes in the interface specification like: new functionalities, or changes in the old functionalities. A minor version change can be backwards compatible or not. If new functionalities are added it will be backwards compatible, otherwise if the old functionalities change it will not be backwards compatible.

- **Release**
  Indicates a new release of the interface with minor changes like optional fields added to some types, adding a new connection method, but it will always be backwards compatible. It also indicates new documentation (best practices, faqs, changes in appendices, etc…).

II.3.2 INTERFACE XML VERSION

The version number is divided in two different parts, year release and month release.

2010	02
Year release	Month release

To use an specific XML interface version you have to provide an extra attribute in your requests: CoreRequest/@version.

```
<HotelValuedAvailRQ echoToken="DummyEchoToken"
                    sessionId="DummySessionId"
      xmlns="http://www.hotelbeds.com/schemas/2005/06/messages"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.hotelbeds.com/schemas/2005/06/messages
            HotelValuedAvailRQ.xsd" version="2010/02">
                                …
```

II.4 Setup Process

In this section you will find how to access the different environments in short and easy steps.

II.4.1 ACCESS TO TEST ENVIRONMENT

- Provide your xml support the ip/ips from which you are planning to access the test server (a maximum of 255 ips can be provided)

- Hotelbeds will configure the system to allow you to access from these ips to the test server
- Configure your interface client using the test server address, in the way that better suits you, web services using WSDL or direct HTTP connection.
- Use the credentials provided by your xml support. These credentials are the same for the live environment. You will not have access to the live environment until you have successfully passed the Acceptance Test (AT) (see Acceptance Test).

II.4.2 ACCESS TO LIVE ENVIRONMENT

- Prior to starting this process you must successfully pass the Acceptance Test (AT) (see Acceptance Test)
- Provide your xml support the ip/ips from which you are planning to access the live server (a maximum of 255 ips can be provided)
- Hotelbeds will grant you the access to the live server
- Configure your interface client using the live server address

III. Technical Information

The interface may be called by any web or classic application supporting standard web protocols like SOAP 1.1 and HTTP 1.1.

You can choose one of the following connection methods:

1. Use the WSDL descriptor and the XML Schema definitions to generate a web services proxy for the interface. A complete XML definition of the interface Web Services operations is available at `/wsdl/FrontendService_Live.wsdl` or `/wsdl/FrontendService_Test.wsdl`. These files are identical, the only thing that changes is the port address (live and test respectively). Remember that you must use **wsdl files** we provide at **documentation**, don't generate it.

2. Open a direct HTTP connection to the interface and POST valid xml requests according to the XML Schema definitions. To connect to the interface using HTTP direct connection you must use the following URL's:

TEST server:	`http://212.170.239.71/appservices/http/FrontendService`
LIVE server:	`http://212.170.239.18/appservices/http/FrontendService`

NOTE: If an HTTP request is used, it is assumed the request has a parameter named **xml_request** whose value is a string containing the XML document for the request. If this parameter is sent with another name, you will obtain the error "Unknown operation" (see Troubleshooting).

The requests must be in **UTF-8** (Unicode) encoding. The responses will be in UTF-8 too.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <hb:getHotelCategoryList
xmlns:hb="http://axis.frontend.hydra.hotelbeds.com"
xsi:type="xsd:string">
<HotelCategoryListRQ ... >
.
.
.
  </hb:getHotelCategoryList>
 </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
    <ns1:getHotelCategoryList xsi:type="xsd:string"
xmlns:ns1="http://axis.frontend.hydra.hotelbeds.com">
<HotelCategoryListRS ... >
.
.
.
  </ns1:getHotelCategoryList>
 </soapenv:Body>
</soapenv:Envelope>
```

III.1 Compression in responses

The interface accept compression of the responses. The use of this feature is **mandatory**. To use it please follow the next steps:

<table>
<tr><td>1</td><td>Invoke the URL
If you use web services:<table><tr><td>TEST server:</td><td>http://212.170.239.71/appservices/ws/FrontendService</td></tr><tr><td>LIVE server:</td><td>http://212.170.239.18/appservices/ws/FrontendService</td></tr></table>
If you use direct HTTP connection:<table><tr><td>TEST server:</td><td>http://212.170.239.71/appservices/http/FrontendService</td></tr><tr><td>LIVE server:</td><td>http://212.170.239.18/appservices/http/FrontendService</td></tr></table></td></tr>
<tr><td>2</td><td>The request MUST HAVE the header "Accept-Encoding:gzip, deflate".</td></tr>
<tr><td>3</td><td>If the server detects it then the response is compressed, and the client will receive our response with the header "Content-Encoding: gzip".</td></tr>
<tr><td>4</td><td>Response must be decompressed on the fly at client end before processing. This is part of the HTTP standard protocol.</td></tr>
</table>

III.2 SOAP messages

A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body.

- The Envelope is the top element of the XML document representing the message. The element name MUST be **ENVELOPE**
- The interface doesn't use the header element.
- The Body is a container for mandatory information intended for the interface. SOAP defines one element for the body, which is the Fault element used for reporting errors. The element name MUST be **BODY**

III.2.1 SOAP MESSAGE REQUEST EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Body>
    <hb:getHotelCategoryList
  xmlns:hb="http://axis.frontend.hydra.hotelbeds.com" x
  si:type="xsd:string">

  <HotelCategoryListRQ
    echoToken="DummyEchoToken"
    xmlns="http://www.hotelbeds.com/schemas/2005/06/messages"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.hotelbeds.com/schemas/2005/06/messages
        HotelCategoryListRQ.xsd">

    <Language>ENG</Language>
    <Credentials>
      <User>TEST</User>
      <Password>TEST</Password>
    </Credentials>
  </HotelCategoryListRQ>
    </hb:getHotelCategoryList>
  </soapenv:Body>
</soapenv:Envelope>
```

III.2.2 HTTP SOAP REQUEST EXAMPLE

```
POST /appservices/ws/FrontendService HTTP/1.1
SOAPAction:
User-Agent: Java/1.4.2_12
Host: 127.0.0.1:8088
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: XXX
```

```
<soapenv:Envelope
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Body>
    <hb:getHotelCategoryList
  xmlns:hb="http://axis.frontend.hydra.hotelbeds.com" x
  si:type="xsd:string">

  <HotelCategoryListRQ
    echoToken="DummyEchoToken"
    xmlns="http://www.hotelbeds.com/schemas/2005/06/messages"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.hotelbeds.com/schemas/2005/06/messages
        HotelCategoryListRQ.xsd">

    <Language>ENG</Language>
    <Credentials>
      <User>TEST</User>
      <Password>TEST</Password>
    </Credentials>
  </HotelCategoryListRQ>
    </hb:getHotelCategoryList>
  </soapenv:Body>
</soapenv:Envelope>
```

III.2.3 SOAP MESSAGE RESPONSE EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getHotelCategoryList xsi:type="xsd:string"
xmlns:ns1="http://axis.frontend.hydra.hotelbeds.com">
      <HotelCategoryListRS
        xmlns="http://www.hotelbeds.com/schemas/2005/06/messages"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.hotelbeds.com/schemas/2005/06/messages
HotelCategoryListRS.xsd"
        totalItems="42"
        echoToken="DummyEchoToken">
        <AuditData>
          <ProcessTime>10</ProcessTime>
          <Timestamp>2006-07-17 12:07:11.444</Timestamp>
          <RequestHost>127.0.0.1</RequestHost>
          <ServerName>TEST</ServerName>
          <ServerId>TS</ServerId>
          <SchemaRelease>2005/06</SchemaRelease>
          <HydraCoreRelease>2.0.200607111140</HydraCoreRelease>
          <HydraEnumerationsRelease>1.0.200607111140</HydraEnumerationsRelease>
          <MerlinRelease>N/A</MerlinRelease>
        </AuditData>
        <Category type="SIMPLE" code="1LL" shortname="1K">1 KEY</Category>
        <Category type="SIMPLE" code="1EST" shortname="1*">1 STAR</Category>
        <Category type="SIMPLE" code="CAM1" shortname="C1">1ST CAMPING</Category>
      </HotelCategoryListRS>
    </ns1:getHotelCategoryList>
  </soapenv:Body>
</soapenv:Envelope>
```

III.2.4 HTTP SOAP MESSAGE RESPONSE EXAMPLE

The HTTP SOAP response example has been reduced for readability purposes. A real SOAP response should be longer, including more elements.

```
        HTTP/1.1 200 OK
        Server: Resin/2.1.17
        Content-Type: text/xml; charset=utf-8
        Transfer-Encoding: chunked
        Date: Mon, 17 Jul 2006 15:03:48 GMT

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <ns1:getHotelCategoryList xsi:type="xsd:string"
xmlns:ns1="http://axis.frontend.hydra.hotelbeds.com">&lt;HotelCategoryListRS
xmlns=&quot;http://www.hotelbeds.com/schemas/2005/06/messages&quot;
xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
xsi:schemaLocation=&quot;http://www.hotelbeds.com/schemas/2005/06/messages
HotelCategoryListRS.xsd&quot; totalItems=&quot;42&quot;
echoToken=&quot;DummyEchoToken&quot;&gt;&lt;AuditData&gt;&lt;ProcessTime&gt;16&lt;/ProcessTime&gt;&l
t;Timestamp&gt;2006-07-17
17:03:49.187&lt;/Timestamp&gt;&lt;RequestHost&gt;127.0.0.1&lt;/RequestHost&gt;&lt;ServerName&gt;TEST
&lt;/ServerName&gt;&lt;ServerId&gt;TS&lt;/ServerId&gt;&lt;SchemaRelease&gt;2005/06&lt;/SchemaRelease
&gt;&lt;HydraCoreRelease&gt;2.0.200607111140&lt;/HydraCoreRelease&gt;&lt;HydraEnumerationsRelease&gt
;1.0.200607111140&lt;/HydraEnumerationsRelease&gt;&lt;MerlinRelease&gt;N/A&lt;/MerlinRelease&gt;&lt;
/AuditData&gt;&lt;Category type=&quot;SIMPLE&quot; code=&quot;1LL&quot;
shortname=&quot;1K&quot;&gt;1 KEY&lt;/Category&gt;&lt;Category type=&quot;SIMPLE&quot;
code=&quot;1EST&quot; shortname=&quot;1*&quot;&gt;1 STAR&lt;/Category&gt;
&lt;/HotelCategoryListRS&gt;</ns1:getHotelCategoryList>
  </soapenv:Body>
</soapenv:Envelope>
```

IV. Basic Concepts

Three main concepts are used in the interface. Understanding of these three concepts is very important in making a good integration.

IV.1 Product

A product is any available item in the system, it holds descriptive information, and **MAY** be booked. Products will never have information about dates, price, customer, policies, contracts…

- Hotel product
  This item will contain descriptive information related to a hotel such as name, chain, category, address, …

- Car product
  This item will contain descriptive information related to a car such as description, features (transmission type, ABS, power steering, seat count…), images, …

- Ticket product
  This item will contain descriptive information related to a ticket such as description, features, …

- Transfer product
  This item will contain descriptive information related to a transfer such as type of transfer (in, out, in/out), transfer vehicle type, …

IV.2 Service

A service is a product containing information about dates, price, customers, policies, contracts,… and is the minimum booking entity.

e.g.:

- Double hotel room with full board, that has a price of 560€ from August 12 th to August 25 th.
- Car hire from July 12 th to July 15th , economy type, and a price of 135€.

IV.3 Purchase

A purchase is a service container with a status (`SHOPPING_CART`, `BOOKING`, `MODIFIED`, `CANCELLED`), global supplements information, global discount information, global additional costs, a reference number if it is confirmed, currency information and a total price of contained services.

To add a service to a purchase it MUST be valued. A purchase will be valid for 30 minutes after you add the first service or can be obtained using `PurchaseDetail` operation. Note that `Purchase/@timeToExpiration` attribute shows the purchase remaining life time in milliseconds. Adding or removing a service to or from a purchase when it is close to becoming invalid, extends purchase's life by five extra minutes.

V. Acceptance Test

V.1 What is the Acceptance Test (AT)?

The acceptance test is the way Hotelbeds ensures that an integration will not compromise the system stability, this is achieved by making some test before allow the client go live. If any problem or issue is detected during the Acceptance Test process the client will be notified.

V.2 Client requirements

- Hotelbeds will need access to the client test server to perform the Acceptance Test for an estimated period of two weeks. You can use either a VPN or a public web site.
- The Acceptance Test will be performed running all the operations developed on the client test server.
- It is mandatory to pass successfully the Acceptance Test before going live.
- In order to do the request data verification you must point your test server to the following urls depending on the frontend you are using:

HTTP frontend:	`http://212.170.239.71/appservices/http/AcceptanceTest`
WS frontend:	`http://212.170.239.71/appservices/ws/AcceptanceTest`

V.3 Acceptance test considerations

V.3.1 DATA CONSISTENCY

The data consistency test prevents you to have inconsistent data inside the Hotelbeds backoffice system and omit mandatory data in the vouchers. All the sensible data shown in your test server (prices, dates,...) during the purchase process will be compared with the data stored in the Hotelbeds backoffice for each service.

Purchase cancellations and modifications (if developed) will also be audited.
The vouchers for all successfully finished purchases will be audited.

V.3.2 REQUEST DATA VERIFICATION

The request data verification test prevents you of sending useless and/or redundant XML data to our system. To avoid massive request or a bad use of operations all the XML requests sent to our system will be monitorized and thoroughly audited during the Acceptance Test.

V.3.3 NET TRAFFIC

In order to finalize the acceptance test you must perform an stress-test. This stress test will be set with at least 30% of your average production server load. It will be performed on every request, and always targeting our live server. Before launching the stress-test you must schedule it with our technical support at xml@hotelbeds.com.

V.4 Steps of certification process

The certification process has 4 steps:

1. We make sure the site will confirm proper bookings. By proper we mean, right prices as well as number of rooms, dates, hotels, type of board, type of room… Customers must be also aware of what hoteliers demand us to show them. In this step you should provide us with a link to your beta site to make some test reservations. We pay special attention to:

   1.1. Child ages must be asked on the searching page. They cannot be changed at any step of the booking flow.

   1.2. The correct implementation and display of pagination (<PaginationData itemsPerPage="999" pageNumber="1"/>)

   1.3. The contract remarks (<Comment type="CONTRACT">) must be indicated on the booking summary just before booking and also on the voucher. This is critical information from the hotelier for the client and must be indicated together with the room information. The information is provided through the "ServiceAdd" and "PurchaseConfirm" responses. Most hotels in test destination AND (Andorra) have contract remarks.

   1.4. In the voucher the supplier information such as supplier name and VAT must be indicated ("Payable through…"). The information is provided through the "PurchaseConfirm" and "PurchaseDetail" responses. Hotelbeds booking reference has to be highlighted. The guests' names (at least one per room), children ages, destination, check in/out dates, room type, board type, hotel details (address, phone and category). For an example, please check voucher.doc.

   1.5. Although it's not mandatory, we also recommend showing before confirmation the cancellation policies ("ServiceAdd") and the hotel issues (ExportCSV folder, HotelIssues.csv). Displaying this information will prevent further issues with clients.

   1.6. Prices have to be checked in all steps (ValuedAvail, ServiceAdd and PurchaseConfirm). Have in mind that our system has a small percentage (less than 0,1%) of returning a miscalculated price in the search (ValuedAvail) when rates, discounts, supplements or any other condition have been recently changed. Note that the price to be confirmed will be always the one returned in the ServiceAdd response.

   1.7. Due to the high volume of requests we experience day by day we have been obliged to ask our clients at developing stage to implement the compression. We also believe you could profit from this feature as it will speed up the communication between your system and ours.
   Please note that some destinations offer a lot of products (more than 400). Although time for calculating the availability may be quite fast, the system will spend some extra time sending this information through the Internet connection (700KB-800KB). The system is designed to reduce the amount of time in the transmission step through the compression tool, by reducing the transmission time and response size by about 95%. The compression is part of the standard HTTP protocol (Gzip Algorithm) and you will find a lot of examples on the Internet.

2. Once we are sure customers will be well informed, we analyze the structure of the XML petitions. In order to make this test, you will have to point your system to a certification url that we will provide you, confirm few bookings (we will also provide you the parameters) and email us the reference numbers, in order for us to analyze their code.

HotelBeds IT department will then crosscheck each provided reference number and its details with the logged XML requests. At this point we will review each step of the booking process, each missing or unnecessary parameter or request, and all the recommendations of any type will be underlined to the client. The IT department will be able to request any modification they consider necessary to assure a correct behaviour of the system once the client goes live, such as deleting and adding steps (XML requests) or ask for more tests. A customer's website will not have live access until they receive the good to go from IT, after they are sure that the Acceptance Test has been totally successful.

Some frequent critical errors are:

2.1. There are duplicated requests that contain the same information and are sent within a short period of time.

2.2. Some steps such as ServiceAddRQ or PurchaseConfirmRQ are repeated unnecessarily. Everytime a ServiceAddRQ is launched without a PurchaseToken this will create a temporary shopping cart that will be stored in memory and the massive creation of them may fill up the memory. On the other hand a duplicated PurchaseConfirmRQ will result in an error once the booking has been confirmed.

2.3. Not all the features the website offers to its users are checked with the run test case.

a) There are no multiple service bookings (ie. more than one room with the same occupancy, with different occupancy, etc.)

b) PurchaseFlush request is not implemented and generates an excessive amount of temporary purchase carts.

c) PurchaseCancel is not thrown though it might represent a key action for users.

2.4. Unnecessary parameters in any requests that will not be processed. We will inform about those considered useless parameters and suggest to remove them.

2.5. The workflow is too confusing because of dozens of Availability requests, unused carts or any other mixed operations that make the Acceptance Test impossible to be reviewed. In this case IT will need all the tests to be repeated again with a simplified process.

2.6. Requests that are set to be sent automatically after another one. A common case is a website that sends a PurchaseDetailRQ automatically once the PurchaseConfirmRS has been received while both responses display the same information.

2.7. RoomCount element is ignored on Customer list. (Please read FAQ "*How can I search multiple occupancy for a hotel service?*" from Hotel Service document.)

3. In the third step we will simulate the behaviour your site will have once it is live. In other words, you will post a 30% of the number of XML petitions per second you estimate your system will send our production servers, once live, for 10 minutes. In order to do so, you can create a script which posts XML petitions (they can be repeated) in the frequency you think your system will. However, if the amount of XML petitions per minutes that you estimate is low (our IT department decides where the limit is), you might not have to take this test, and you will go straight to live.

4. At this point, you will be already live. However, in order to remain live you will have to confirm a reservation (dates six months in advance) and send us both voucher and price. Bear in mind that you will have to cancel the booking afterwards, otherwise you would be charged. Check with us before cancelling.

Please check the FTP for future revisions of the certification process (document *Certification Process.doc*).

VI. Best Practices

In this section you will find some best practices recommended by our technical staff to make a good integration.

VI.1 SessionId

The different service availabilities are session oriented. The responsibility of generating the session ids relies on the client. For each new request you must generate a different session id. If you are using pagination and retrieving next pages from a previous initial search you must use the same session id, this is because the results are cached using the session id.

VI.2 Search by geographical criteria

Our system is designed to search all the services available in one specific destination for the dates and the occupancies requested. This model allows you to get in a single request all the available services from our inventory. If you need just to check availability for one single product, you can add this information in the availability request, so the system will return you just this product if available. This single product request is useful when you want to recheck availability or when you wish to focus on one particular product.

VI.3 What you should never do

You should NOT use the individual product search to launch multithreaded requests for many individual products.

VI.4 Workflow

To confirm a booking you should send the following requests:

1. Availability
2. ServiceAdd
3. PurchaseConfirm

This is the optimal workflow and we expect you to send it in that way.

There are some inappropiate workflows we will not accept. Here you are some examples:

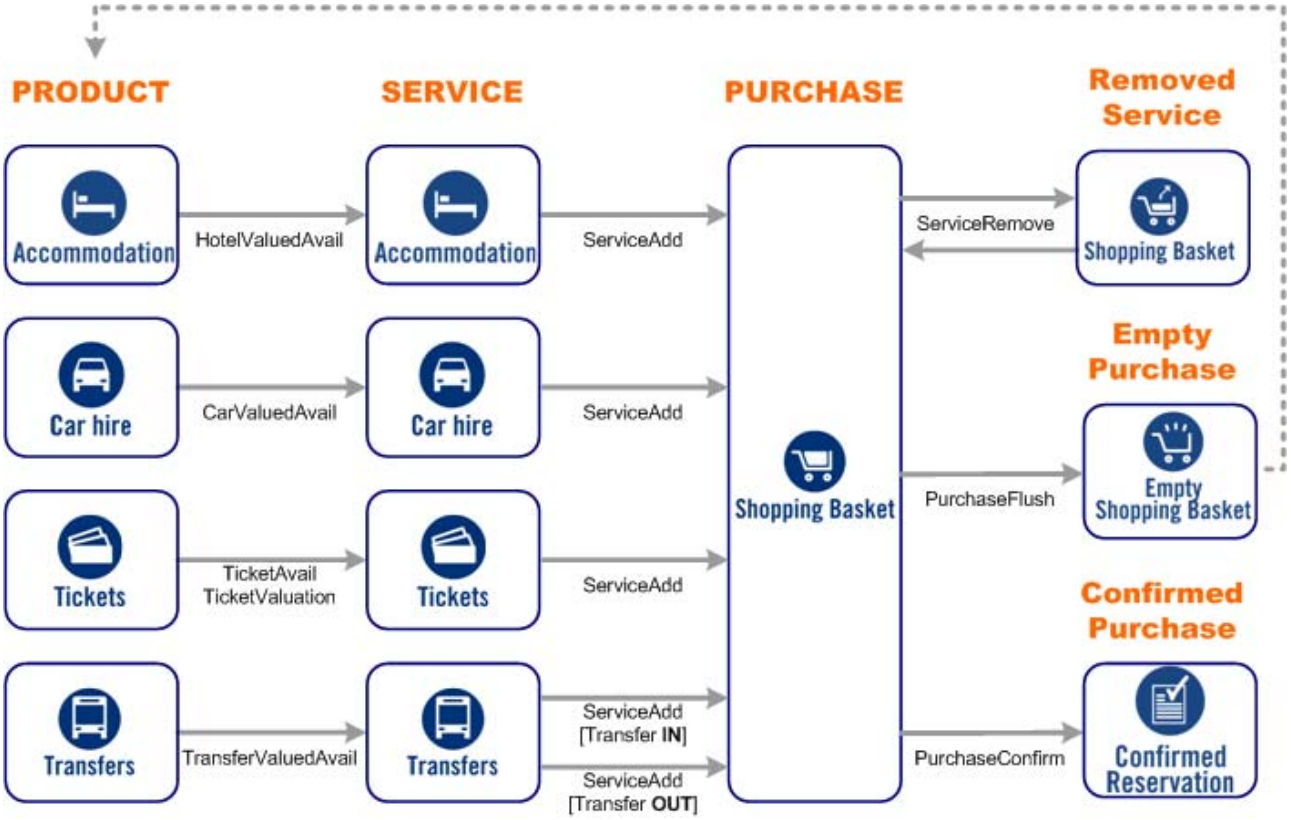
1. Availability
2. ServiceAdd
3. Availability (same as 1)
4. ServiceAdd (same as 2)
5. PurchaseConfirm

1. Availability
2. ServiceAdd
3. ServiceAdd (same as 2)
4. PurchaseConfirm

1. Availability
2. Availability (same as 1)
3. Availability (same as 1)
4. ServiceAdd
5. PurchaseConfirm

VII. Workflows

VII.1Service confirmation workflow



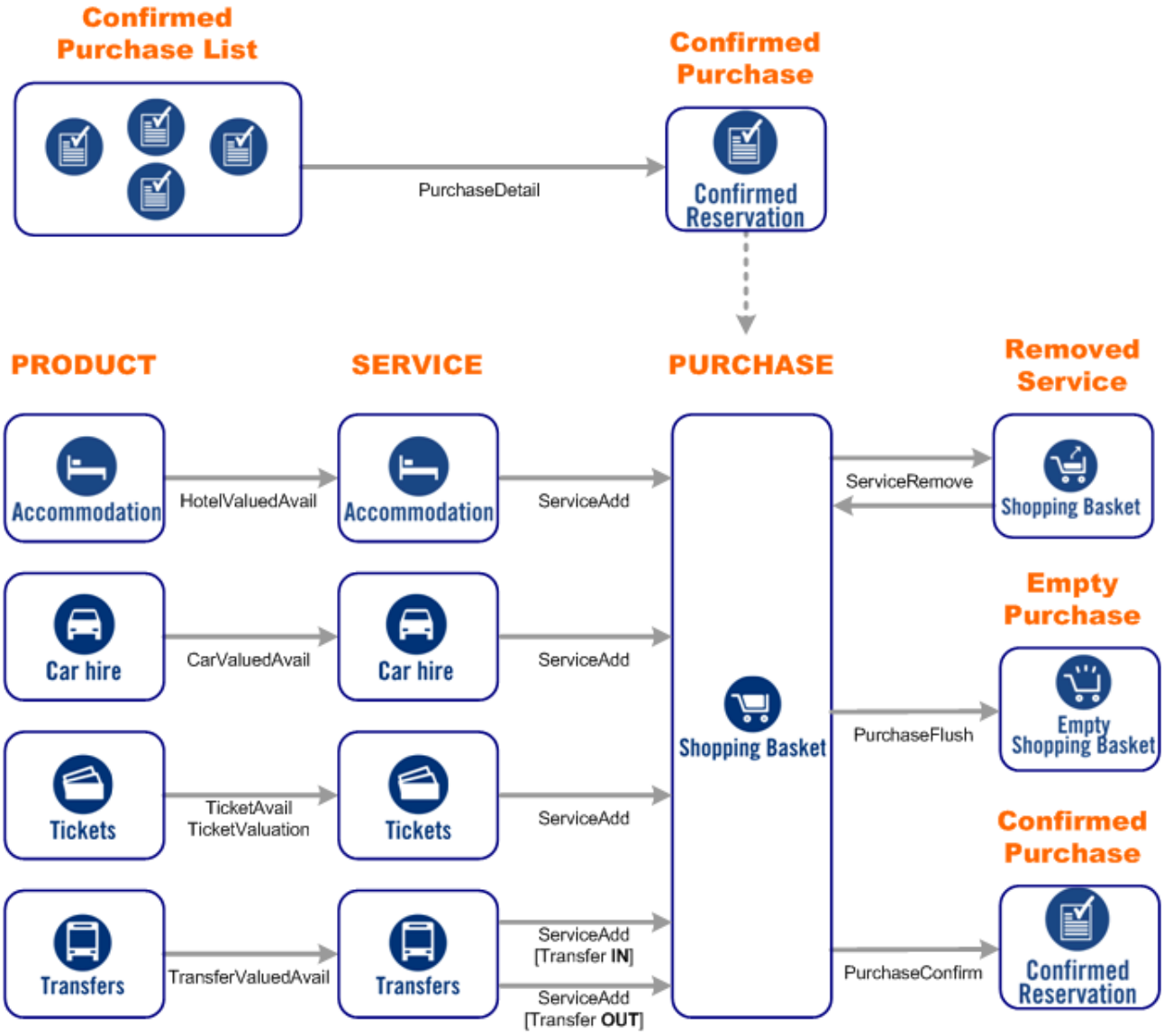
VII.1.1 DESCRIPTION

To confirm a purchase containing multiple or single services you should add new services to the purchase at any time using `ServiceAdd` operation subsequent to new service availability (`CarValuedAvail`, `HotelValuedAvail`, `TicketAvail` - `TicketValuation`, `TransferValuedAvail`).

- Remove an existing service at any time using `ServiceRemove` operation.
- Empty the purchase at any time using `PurchaseFlush` operation.
- Once you have the purchase containing all the desired services you can confirm it using `PurchaseConfirm` operation.

Please note that you can only add to a purchase valued services, the `TicketAvail` operation **does not** return valued ticket services. `TicketValuation` operation **must** be called before adding a ticket service to the purchase.

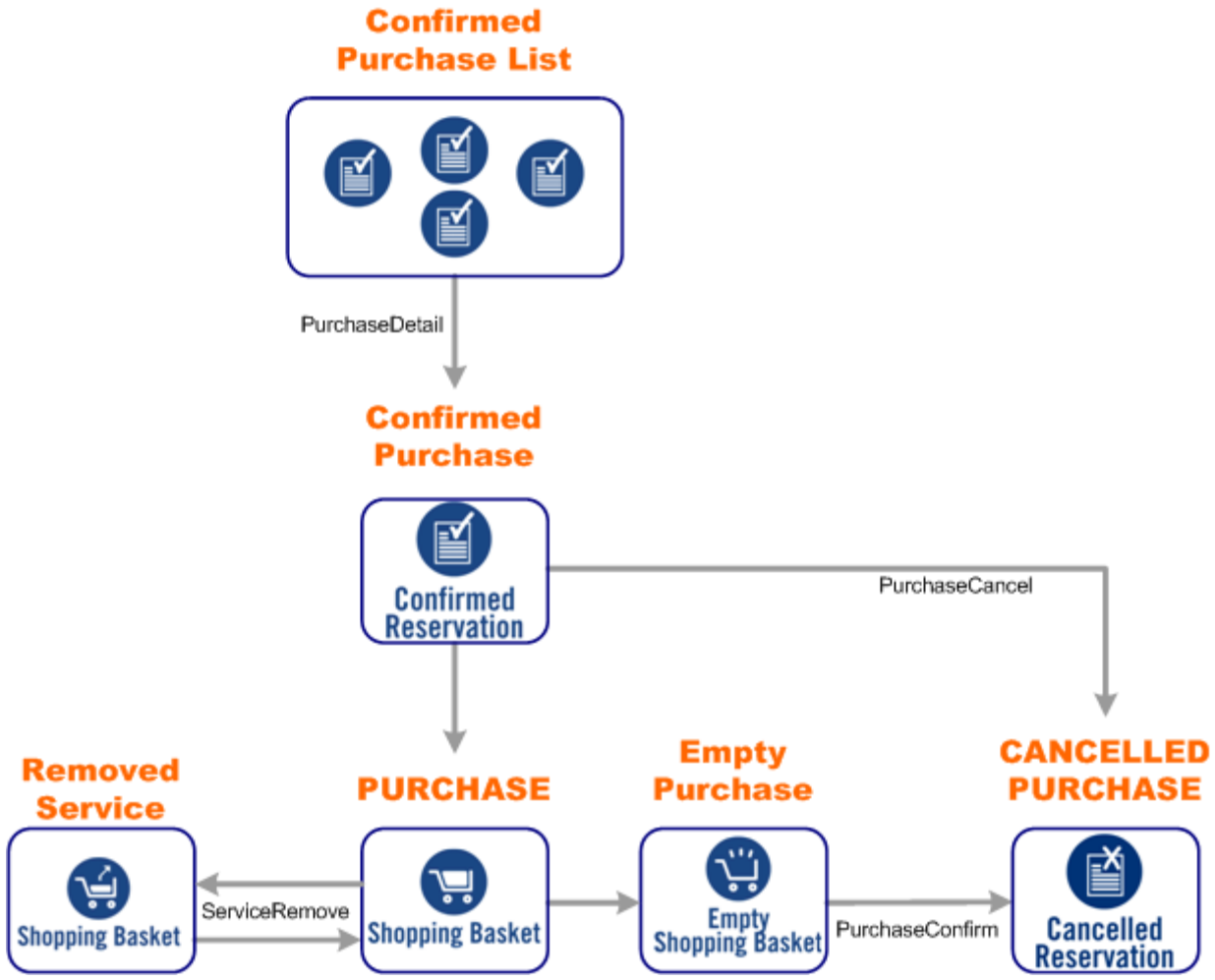
VII.2Purchase modification (add or remove services) workflow



VII.2.1 DESCRIPTION

To modify a purchase, first of all, you should know which purchase you want to modify. Use `PurchaseList` and `PurchaseDetail` operations to get the desired purchase data. Once you have the purchase data, add new services individually using `ServiceAdd` operation subsequent to new service availability (`CarValuedAvail`, `HotelValuedAvail`, `TicketAvail`, `TicketValuation`, `TransferValuedAvail`). Remove an existing service at any time using `ServiceRemove` operation. Once you have the purchase containing all the desired services you can confirm the purchase using `PurchaseConfirm` operation. Please note that you can only add to a purchase valued services, the `TicketAvail` operation **does not** return valued ticket services. `TicketValuation` operation **must** be called before adding a ticket service to the purchase.

VII.3Purchase cancellation workflow



VII.3.1 DESCRIPTION

To cancel a purchase, first of all, you should know which purchase you want to cancel. Use `PurchaseList` and `PurchaseDetail` operations to get the desired purchase data. Once you have the purchase data you want to cancel you have two options:

- Remove all the services individually using `ServiceRemove` operation. Confirm the modification using `PurchaseConfirm` operation.
- Cancel the whole purchase using `PurchaseCancel` operation.

VIII. Common Operations

This section show the common operations in use for all services (hotel, transfers, tickets and cars).

VIII.1 ServiceAdd

Use this operation to:

- Add a new service to a purchase:
  - If the Purchase has the BOOKING status you can add new services or modify existing services. Note that every service has a modification policy list defining all allowed modifications to this service. You can find this information at Purchase/ServiceList/Service/ModificationPolicyList element. You can modify a service only when it has a MODIFICATION value at ModificationPolicy element. At the moment only some hotel services can be modified. The allowed modifications for a 'modifiable' hotel service are: board modification, add, modify or remove a room (including paxes) and change dates.

  - To add the first service to a purchase use this operation without specifying a purchaseToken. This will create a new purchase and will generate a brand new purchaseToken.

- Add a new room to a confirmed hotel service:
  - Set the ServiceAddRQ/Service/@SPUI attribute with the SPUI of the purchase.
  - Set the ServiceAddRQ/Service/AvailableRoom with the new selected room.

- Remove only a room from a confirmed hotel service:
  - Set the ServiceAddRQ/Service/@SPUI attribute with the SPUI of the purchase.
  - Set the ServiceAddRQ/Service/AvailableRoom element you want to remove.
  - Set the ServiceAddRQ/Service/AvailableRoom/HotelRoom/@cancelled attribute to Y.
  - Note that removing the last room from a hotel service is equivalent to calling the ServiceRemove operation, it will remove the service from the purchase. If the service is the only service contained in the purchase it is equivalent to removing the last service of the purchase and you will get a purchase containing no services but with the cancellation fees (when applicable).

- Modify a room from a confirmed hotel service:
  - Set the ServiceAddRQ/Service/@SPUI attribute with the SPUI of the purchase.
  - Set the ServiceAddRQ/Service/AvailableRoom element you want to modify.
  - Set the ServiceAddRQ/Service/AvailableRoom/HotelRoom/@SHRUI obtained in the availability.
  - Set the ServiceAddRQ/Service/AvailableRoom/HotelRoom/@modifiesSHRUI attribute with the SHRUI you want to modify.

ServiceAdd

	Request	Response
Details:	xsddocs/ServiceAddRQ.html	xsddocs/ServiceAddRS.html
Schema:	xml/ServiceAddRQ.xsd	xml/ServiceAddRS.xsd
Example:	*see xml/ServiceAddRQ_serviceName*	*see xml/ServiceAddRS_serviceName*

VIII.1.1 REQUEST BUSINESS RULES

Element / attribute	Rule
Language	Mandatory
Service/ContractList/Contract	Mandatory. Only the first element will be taken in consideration

VIII.1.2 REQUEST OBSERVATIONS

At the end of the purchase booking process all customer name and last name is mandatory for all services. If you don't provide this information in this step you must provide it in the PurchaseConfirm request operation call.

VIII.1.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- ServiceAddRQ/ExtraParamList
- ServiceAddRQ/Service/Reference
- Only first ServiceAddRQ/Service/ContractList/Contract is taken in consideration
- ServiceAddRQ/Service/ContractList/Contract/IncomingOffice/Description
- ServiceAddRQ/Service/ContractList/Contract/IncomingOffice/ContactInfo
- ServiceAddRQ/Service/ContractList/Contract/IncomingOffice/FiscalNumber
- ServiceAddRQ/Service/ContractList/Contract/Sequence
- ServiceAddRQ/Service/ContractList/Contract/Classification
- ServiceAddRQ/Service/ContractList/Contract/CommentList
- ServiceAddRQ/Service/Supplier
- ServiceAddRQ/Service/CommentList
- ServiceAddRQ/Service/DateFrom/@time
- ServiceAddRQ/Service/Currency
- ServiceAddRQ/Service/TotalAmount
- ServiceAddRQ/Service/SupplementList
- ServiceAddRQ/Service/DiscountList
- ServiceAddRQ/Service/AdditionalCostList
- ServiceAddRQ/Service/ErrorList
- ServiceAddRQ/Service/ModificationPolicyList

VIII.1.4 RESPONSE BUSINESS RULES

If ServiceAddRS/Purchase element is present the following elements and attributes must be always present:

- ServiceAddRS/Purchase/@timeToExpiration
- ServiceAddRS/Purchase/Status
- ServiceAddRS/Purchase/Agency
- ServiceAddRS/Purchase/Language
- ServiceAddRS/Purchase/Currency
- ServiceAddRS/Purchase/TotalPrice

VIII.1.5 RESPONSE OBSERVATIONS

Note that if customer data is not specified in the service add request, the system will generate and return empty customer data with a customer id. You must use the customer ids to specify customer data in the purchase confirm request.

VIII.2 ServiceRemove

Remove a service from a purchase.

- When you remove the las service from a purchase that has not already been confirmed, you get a purchase containing no services. You can add new services to this purchase.
- When you remove the last service from a confirmed purchase, you get a purchase containing no services with the cancellation fees (when applicable). After that you must call the PurchaseConfirm operation to cancel the booking. This is equivalent to calling the PurchaseCancel operation.

ServiceRemove

	Request	Response
Details:	xsddocs/ServiceRemoveRQ.html	xsddocs/ServiceRemoveRS.html
Schema:	xml/ServiceRemoveRQ.xsd	xml/ServiceRemoveRS.xsd
Example:	*see xml/ServiceRemoveRQ_serviceName*	*see xml/ServiceRemoveRS_serviceName*

VIII.2.1 REQUEST BUSINESS RULES

Element/attribute	Rule
Language	Mandatory

VIII.2.2 REQUEST OBSERVATIONS

To get the purchase token for a purchase with the 'BOOKING' status please use the PurchaseDetail operation. Use the ServiceRemoveRQ/@purchaseToken attribute to select the purchase to remove the service. Use the ServiceRemoveRQ/@SPUI attribute to select the service to remove from the purchase.

VIII.2.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- ServiceRemoveRQ/ExtraParamList

VIII.2.4 RESPONSE BUSINESS RULES

If ServiceRemoveRS/Purchase element is present the following elements must be present:

- ServiceRemoveRS/Purchase/@timeToExpiration
- ServiceRemoveRS/Purchase/Status
- ServiceRemoveRS/Purchase/Agency
- ServiceRemoveRS/Purchase/Language
- ServiceRemoveRS/Purchase/Currency
- ServiceRemoveRS/Purchase/TotalPrice

VIII.2.5 RESPONSE OBSERVATIONS

VIII.3 PurchaseFlush

Empty a purchase. This operation is only valid when the purchase has a SHOPPING_CART status.

PurchaseFlush

	Request	Response
Details:	xsddocs/PurchaseFlushRQ.html	xsddocs/PurchaseFlushRS.html
Schema:	xml/PurchaseFlushRQ.xsd	xml/PurchaseFlushRS.xsd
Example:	xml/PurchaseFlushRQ.xml	xml/PurchaseFlushRS.xml

VIII.3.1 REQUEST BUSINESS RULES

VIII.3.2 REQUEST OBSERVATIONS

Use PurchaseFlushRQ/@purchaseToken attribute to select the purchase to flush.

VIII.3.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- PurchaseFlushRQ/ExtraParamList

VIII.3.4 RESPONSE BUSINESS RULES

VIII.3.5 RESPONSE OBSERVATIONS

VIII.4 PurchaseConfirm

Confirm a purchase. If the purchase has already been confirmed, the purchase data is updated with the provided information.

PurchaseConfirm

	Request	Response
Details:	xsddocs/PurchaseConfirmRQ.html	xsddocs/PurchaseConfirmRS.html
Schema:	xml/PurchaseConfirmRQ.xsd	xml/PurchaseConfirmRS.xsd
Example:	xml/PurchaseConfirmRQ.xml	xml/PurchaseConfirmRS.xml

VIII.4.1 REQUEST BUSINESS RULES

You can only provide AGENCY comments at `PurchaseConfirmRQ/ConfirmationData/CommentList` (Ex: telephone number, email …).

VIII.4.2 REQUEST OBSERVATIONS

- Holder name and last name are mandatory for new purchase confirmation.
- Agency reference are mandatory for new purchase confirmation.
- Holder and agency reference data cannot be modified. So, you should not send elements Holder and AgencyReference to confirm a modification.
- You can only provide comments for hotel and transfer services.
- You cannot add new or modify comments to a previously confirmed service.
  To provide the customer data you must specify the same customer id returned in the ServiceAdd response. If you didn't provide the customer name and last name for all customer and services in the ServiceAdd request you must provide it now.
- Age is mandatory for children.

VIII.4.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- `PurchaseConfirmRQ/ExtraParamList`

VIII.4.4 RESPONSE BUSINESS RULES

If `PurchaseConfirmRS/Purchase` element is present the following elements and attributes must be always present:

- `PurchaseConfirmRS/Purchase/@timeToExpiration`
- `PurchaseConfirmRS/Purchase/Reference`
- `PurchaseConfirmRS/Purchase/Reference/IncomingOffice`
- `PurchaseConfirmRS/Purchase/Status`
- `PurchaseConfirmRS/Purchase/Agency`
- `PurchaseConfirmRS/Purchase/Language`
- `PurchaseConfirmRS/Purchase/CreationDate`
- `PurchaseConfirmRS/Purchase/Holder`
- `PurchaseConfirmRS/Purchase/Currency`
- `PurchaseConfirmRS/Purchase/TotalPrice`

VIII.4.5 RESPONSE OBSERVATIONS

Note that if you find a `PurchaseConfirmRS/Purchase/ServiceList/Service/ErrorList/Error` element it means that the confirmation of this service has not been

successful, this implies the purchase has not been confirmed. Note that two different services with the same hotel and dates, after perform the PurchaseConfirm operation will appear grouped in one single service.

VIII.5 PurchaseDetail

Get detailed information from a purchase.

PurchaseDetail

	Request	Response
Details:	xsddocs/PurchaseDetailRQ.html	xsddocs/PurchaseDetailRS.html
Schema:	xml/PurchaseDetailRQ.xsd	xml/PurchaseDetailRS.xsd
Example:	xml/PurchaseDetailRQ.xml	xml/PurchaseDetailRS.xml

VIII.5.1 REQUEST BUSINESS RULES

Element / attribute	Rule
Language	Mandatory
PurchaseReference/IncomingOffice/@code	Mandatory if PurchaseReference is present

VIII.5.2 REQUEST OBSERVATIONS

PurchaseDetailRQ/PurchaseReference element must be provided if the purchase status is BOOKING. PurchaseDetailRQ/PurchaseToken element must be provided if the purchase status is SHOPPING_CART.

VIII.5.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- PurchaseDetailRQ/ExtraParamList
- PurchaseDetailRQ/PurchaseReference/IncomingOffice/Description
- PurchaseDetailRQ/PurchaseReference/IncomingOffice/ContactInfo
- PurchaseDetailRQ/PurchaseReference/IncomingOffice/FiscalNumber

VIII.5.4 RESPONSE BUSINESS RULES

If PurchaseDetailRS/Purchase element is present the following elements and attributes must be always present:

- PurchaseDetailRS/Purchase/@timeToExpiration
- PurchaseDetailRS/Purchase/Status
- PurchaseDetailRS/Purchase/Agency
- PurchaseDetailRS/Purchase/Language
- PurchaseDetailRS/Purchase/Currency
- PurchaseDetailRS/Purchase/TotalPrice

VIII.5.5 RESPONSE OBSERVATIONS

VIII.6 PurchaseList

Get a list of purchases according to the requested parameters.

PurchaseList

	Request	Response
Details:	xsddocs/PurchaseListRQ.html	xsddocs/PurchaseListRS.html
Schema:	xml/PurchaseListRQ.xsd	xml/PurchaseListRS.xsd
Example:	xml/PurchaseListRQ.xml	xml/PurchaseListRS.xml

VIII.6.1 REQUEST BUSINESS RULES

Element/attribute	Rule
Holder/Name	Mandatory if Holder is provided
Holder/LastName	Mandatory if Holder is provided

VIII.6.2 REQUEST OBSERVATIONS

VIII.6.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- PurchaseFlushRQ/ExtraParamList
- PurchaseListRQ/IncomingOffice/Description
- PurchaseListRQ/IncomingOffice/ContactInfo
- PurchaseListRQ/IncomingOffice/FiscalNumber
- PurchaseListRQ/Destination/Name
- PurchaseListRQ/Destination/ZoneList
- PurchaseListRQ/Reference/IncomingOffice
- PurchaseListRQ/Holder/CustomerId
- PurchaseListRQ/Holder/Age
- PurchaseListRQ/Holder/AdditionalInfo
- PurchaseListRQ/Holder/Document
- PurchaseListRQ/Holder/BirthDate
- PurchaseListRQ/Holder/CountryCode

VIII.6.4 RESPONSE BUSINESS RULES

One of PurchaseListRS/Purchase or PurchaseListRS/ErrorList element must be present unless the PurchaseListRS/@totalItems attribute is 0. If PurchaseListRS/Purchase element is present the following elements and attributes must be present:

- PurchaseListRS/Purchase/@timeToExpiration
- PurchaseListRS/Purchase/Status
- PurchaseListRS/Purchase/Agency
- PurchaseListRS/Purchase/Language
- PurchaseListRS/Purchase/Currency
- PurchaseListRS/Purchase/TotalPrice

VIII.6.5 RESPONSE OBSERVATIONS

If one or more PurchaseListRS/Purchase elements are present PurchaseListRS/@totalItems attribute contains the total number of purchases returned in the response. If no PurchaseListRS/Purchase element is present and PurchaseListRS/@totalItems attribute is 0, no purchases were found matching the search criteria.

VIII.7 PurchaseCancel

Cancel a purchase or valuate a purchase cancellation. This operation is only valid when the purchase has a BOOKING status.

PurchaseCancel

	Request	Response
Details:	xsddocs/PurchaseCancelRQ.html	xsddocs/PurchaseCancelRS.html
Schema:	xml/PurchaseCancelRQ.xsd	xml/PurchaseCancelRS.xsd
Example:	xml/PurchaseCancelRQ.xml	xml/PurchaseCancelRS.xml

VIII.7.1 REQUEST BUSINESS RULES

Element/attribute	Rule
Language	Mandatory
PurchaseReference/IncomingOffice	Mandatory

VIII.7.2 REQUEST OBSERVATIONS

Set PurchaseCancelRQ/@type attribute value to V to get the cost of the purchase cancellation. Using this value does not cancel the purchase, only returns information about the cost of cancellation.

Set PurchaseCancelRQ/@type attribute value to C to cancel the purchase in the system, note that cancellation fees may apply.

VIII.7.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- PurchaseCancelRQ/ExtraParamList
- PurchaseCancelRQ/PurchaseReference/IncomingOffice/Description
- PurchaseCancelRQ/PurchaseReference/IncomingOffice/ContactInfo
- PurchaseCancelRQ/PurchaseReference/IncomingOffice/FiscalNumber

VIII.7.4 RESPONSE BUSINESS RULES

If PurchaseCancelRS/Purchase element is present the following elements and attributes must be always present:

- PurchaseCancelRS/Currency
- PurchaseCancelRS/Amount
- PurchaseCancelRS/Purchase/@timeToExpiration
- PurchaseCancelRS/Purchase/Reference
- PurchaseCancelRS/Purchase/Reference/IncomingOffice
- PurchaseCancelRS/Purchase/Status
- PurchaseCancelRS/Purchase/Agency
- PurchaseCancelRS/Purchase/Language
- PurchaseCancelRS/Purchase/CreationDate
- PurchaseCancelRS/Purchase/Holder
- PurchaseCancelRS/Purchase/Currency
- PurchaseCancelRS/Purchase/TotalPrice

VIII.7.5 RESPONSE OBSERVATIONS

If `PurchaseCancelRS/@type` attribute value is `V` the response shows the cost of the purchase cancellation. If you get this value the purchase has not been cancelled, only information about the cost of cancellation is provided.

If `PurchaseCancelRS/@type` attribute value is `C` the purchase cancellation has been done, note that cancellation fees may apply. Cancellation cost is shown `PurchaseCancelRS/@type` in element.

VIII.8 IncomingOfficeList

Get the information for each incoming office code provided. If no incoming office codes are provided all incoming office information will be returned.

IncomingOfficeList

	Request	Response
Details:	xsddocs/IncomingOfficeListRQ.html	xsddocs/IncomingOfficeListRS.html
Schema:	xml/IncomingOfficeListRQ.xsd	xml/IncomingOfficeListRS.xsd
Example:	xml/IncomingOfficeListRQ.xml	xml/IncomingOfficeListRS.xml

VIII.8.1 REQUEST BUSINESS RULES

Element/attribute	Rule
Language	Mandatory

VIII.8.2 REQUEST OBSERVATIONS

Provide a list of `IncomingOfficeListRQ/IncomingOfficeCode` elements to get only the selected incoming offices.

VIII.8.3 REQUEST ACCEPTANCE TEST

The following elements and attributes are useless:

- `IncomingOfficeListRQ/ExtraParamList`

VIII.8.4 RESPONSE BUSINESS RULES

One of `IncomingOfficeListRS/IncomingOffice` or `IncomingOfficeListRS/ErrorList` elements must be present in the response unless the `@totalItems` attribute is `0`.

If `IncomingOfficeListRS/IncomingOffice` element is present the following elements and attributes must be always present:

- `IncomingOfficeListRS/IncomingOffice/Description`
- `IncomingOfficeListRS/IncomingOffice/FiscalNumber`

VIII.8.5 RESPONSE OBSERVATIONS

If no `IncomingOfficeListRS/IncomingOffice` element is present and `/@totalItems` attribute is `0`, no incoming offices were found matching the search criteria.

If one or more `IncomingOfficeListRS/IncomingOffice` elements are present:

- `/@totalItems` attribute contains the total number of incoming offices returned in the response.

IX. FAQ

1. Which encoding type should I use in XML requests and responses?

The requests must be encoded in UTF-8. Responses are also encoded in UTF-8.

2. What is the availToken?

The availToken is a temporary availability identifier. Use this token to add a service to a purchase.

3. What is the SPUI?

The SPUI is the Service Purchase Unique Identifier. It identifies a service inside a purchase.

4. What is the purchaseToken?

The purchaseToken is a temporary purchase identifier. This identifier is needed to make operations with purchases.

The following rules are applied to the purchaseToken:

- Adding the first service to a purchase will generate a brand new purchaseToken.
- Getting the purchase detail from a confirmed purchase (using PurchaseDetail operation) will generate a brand new purchaseToken.
- PurchaseList operation will NOT generate usable purchase tokens.
- Confirming a purchase will invalidate its purchaseToken.
- After 30 minutes of a purchaseToken creation it will become invalid. Note that Purchase/@timeToExpiration attribute shows the purchaseToken's remaining life time in milliseconds. Adding or removing a service to or from a purchase when it is near to becoming invalid, extends the purchaseToken's life by five extra minutes.

5. How can I add the first service to a purchase?

To add the first service to a purchase use the ServiceAdd operation without specifying a purchaseToken. This will create a new purchase and will generate a brand new purchaseToken.

6. How can I get a purchaseToken to start working with a purchase?

If you haven't created a purchase, you need to add the first service to the purchase (see above). If you already have a confirmed purchase get its detail using PurchaseDetail operation. Note that you can get a list of confirmed purchases using the PurchaseList operation, this operation does NOT generate usable purchase tokens.

7. How long will a purchaseToken be valid without being confirmed?

The purchaseToken will be valid for 30 minutes subsequent to you adding the first service or getting it using PurchaseDetail operation. Note that Purchase/@timeToExpiration attribute shows the purchaseToken's remaining life time in milliseconds. Adding or removing a service to or from a purchase when it is near to becoming invalid, extends the purchaseToken's life by five extra minutes.

8. Can a service availability return different services with the same product code but different contracts?

Yes, a service availability can return different services with the same product code but different contract code. The system works with contracts, this means that a service availability can return the same product more than once but with different contracts, this usually happens when different prices are provided for the same product and different contracts.

9. How can I specify the customer data of a service?

Prior to adding customer data to a service it is necessary to understand what `CustomerId` tag means. Each tag representing a customer has a `CustomerId` tag, this id is generated and provided by our system. You can use this id to identify a customer inside a purchase.
The following rules are applied to specify the customer data of a service when you are using the `ServiceAdd` operation:

- The customer data is optional. If you send it, it will be stored in the system and returned in the response with the corresponding `CustomerId`. You can modify this data later when you call the `PurchaseConfirm` operation.
- In the case of children the age is mandatory.
- If no customer data is provided, empty customer data will be generated by the system and returned in the response with the corresponding `CustomerId` and empty data.
- Customer ids sent in the `ServiceAdd` operation will be ignored.
- If you are using the `ServiceAdd` operation to modify an existing purchase you cannot reuse the old customer data, you need to provide new data.
- To specify the customer data of a service when you are using the `PurchaseConfirm` operation:
  - You should provide the customer data you want to add or modify specifying the `CustomerId` provided in the `ServiceAdd` operation response.
  - At least the name of one adult customer must be provided for each service, thus you cannot confirm a service containing only children.

10. Which modifications are allowed for a confirmed purchase?

Only the Hotel service allows modifications for a confirmed purchase.

11. How can I add a new service to a confirmed purchase?

To add a new service to a confirmed purchase follow the steps below:

- Make the desired service availability. Note that you can only add to a purchase valued services, the `TicketAvail` operation does not return valued ticket services. `TicketValuation` operation MUST be called before adding a ticket service to the purchase.
- Call the `ServiceAdd` operation with the following considerations:
  - Set the `ServiceAddRQ/@purchaseToken` attribute with the purchase token of the purchase you want to add the service.
  - Set the `ServiceAddRQ/Service` element with the selected service data.
- Call `PurchaseConfirm` operation to commit the changes to the system.

Note that you need the purchase detailed information to get a valid purchase token. (see `PurchaseDetail` operation).

12. How can I remove a service from a confirmed purchase?

To remove a service from a confirmed purchase follow the steps below:

- Call the `ServiceRemove` operation with the following considerations:
  - Set the `ServiceRemoveRQ/@purchaseToken` attribute with the purchase token of the purchase you want to remove from the service.
  - Set the `ServiceRemoveRQ/@SPUI` attribute with the `SPUI` of the service you want to remove.
- Call `PurchaseConfirm` operation to commit the changes to the system.

Note that you need the purchase detailed information to get a valid purchase token. (see `PurchaseDetail` operation).

13. What is the system behaviour when I remove the last service of a purchase?

You can find two possible situations:

- When you remove the last service from a purchase that has not already been confirmed, you get a purchase containing no services. You can add new services to this purchase.
- When you remove the last service from a confirmed purchase, you get a purchase containing no services but with the cancellation fees (when applicable). After that you MUST call the `PurchaseConfirm` operation to cancel the booking. This is equivalent to calling the `PurchaseCancel` operation.

14. What is an XSD ?

XML Schema, published as a W3C recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve Recommendation status by the W3C.

Like all XML schema languages, XML Schema can be used to express a schema: a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. However, unlike most other schema languages, XML Schema was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types. Such a post-validation infoset can be useful in the development of XML document processing software, but the schema language's dependence on specific data types has provoked criticism.

An XML Schema instance is an XML Schema Definition (XSD) and typically has the filename extension ".xsd". The language itself is sometimes informally referenced as XSD. It has been suggested that WXS (for W3C XML Schema) is a more appropriate initialism though this acronym has not been in a widespread use and W3C working group rejected it. XSD is also an initialism for XML Schema Datatypes, the datatype portion of XML Schema.

15. Why is defined the type of elements like ‘anyType’ at wsdl’s?

Hotelbeds xml specification is not a closed language. This means that objects can be modified including new elements. In order to avoid issues to clients’ integrations, we define the type of requests and responses like ‘anyType’. A proxy generator will create them as Object types and in that way new elements in specification will not cause validation errors.

16. Which currency is returned by the system?

Price of services can be returned in different currencies. There are three default currencies available:

Currency code	Description	Location
GBP	United Kingdom Pound	UK
USD	US Dollar	America and Caribbean
EUR	Euro	Rest of countries

The currencies returned will depend as well on the bussiness model.

X. Troubleshooting

In this section you will find possible problems and solutions that may help you to make an integration with the interface.

X.1 I get the HTTP Status code 500 with the following error message: "IndexOutOfBoundsException: Index: 0, Size: 0" error.

Problem: When I send a soap request to the Web Services frontend I get an "IndexOutOfBoundsException: Index: 0, Size: 0" error.

Cause: This error is usually caused when the hotelbeds namespace is not provided in the soap request.

Solution: You need to include the hotelbeds namespace declaration: http://axis.frontend.hydra.hotelbeds.com.
For instance, a correct hotel category list XML request for the Web services frontend is:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Body>
    <hb:getHotelCategoryList
      xmlns:hb="http://axis.frontend.hydra.hotelbeds.com"
      xsi:type="xsd:string">

      <HotelCategoryListRQ
        echoToken="DummyEchoToken"
        xmlns="http://www.hotelbeds.com/schemas/2005/06/messages"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.hotelbeds.com/schemas/2005/06/messages
HotelCategoryListRQ.xsd">

        <Language>ENG</Language>
        <Credentials>
          <User>TEST</User>
          <Password>TEST</Password>
        </Credentials>
      </HotelCategoryListRQ>
    </hb:getHotelCategoryList>
  </soapenv:Body>
</soapenv:Envelope>
```

X.2 I get “Envelope is not a valid tag name.” error.

Problem: When I send any request to the HTTP frontend I get “Envelope is not a valid tag name” error.

Cause: This error is caused when you try to access the HTTP frontend using a SOAP request.

Solution: Remove the SOAP envelope and body from the XML request.
For instance, a correct hotel category list XML request for the HTTP frontend is:

```
<?xml version="1.0" encoding="UTF-8"?>
<HotelCategoryListRQ
  echoToken="DummyEchoToken"
  xmlns="http://www.hotelbeds.com/schemas/2005/06/messages"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.hotelbeds.com/schemas/2005/06/messages
HotelCategoryListRQ.xsd">

  <Language>ENG</Language>
  <Credentials>
    <User>TEST</User>
    <Password>TEST</Password>
  </Credentials>
</HotelCategoryListRQ>
```

X.3 I get the HTTP Status code 500 with the following error message: “Unknown operation”

Problem: When I send a XML request to the HTTP frontend I get the `Unknown operation` error.

Cause: This error may be caused by any combination of the following resasons:

1. You use the `GET` method.
2. You didn’t provide the `xml_request` param name.
3. You didn’t send a valid XML request.

Solution: Ensure you are using `POST` method to access the HTTP frontend, sending the XML request with a parameter named `xml_request` and is a valid XML operation.

X.4 I get the HTTP Status code 500 with the following error message: “Method ‘GET’ not allowed, please use ‘POST’ method”

Problem: When I send a XML request to the HTTP frontend I get the `Method GET not allowed, please use POST method` error.

Cause: This error is caused when you use the `GET` method instead of `POST`.

Solution: Use `POST` method to access the HTTP frontend.

XI. Conventions

XI.1 Language Codes

One more characteristic of the interface is the capability of returning all the information in the requested language. The valid languages codes are not available through interface operations and are shown in the table below.

Language	Code
Spanish	CAS
French	FRA
English	ENG
German	ALE
Portuguese	POR
Italian	ITA
Dutch	HOL

XI.2 Error Codes

In this appendix you will find the error codes returned by the system and their meanings. Error codes are dynamically generated by the system, thus a complete list of error codes cannot be provided. You can find three different error categories.

XI.2.1 APPLICATION ERRORS

Format:	AXX-XX-XXX-XX where X can be any character between [A-Z] or [0-9]
Cause:	These errors are caused by an internal application error. You can't do anything to fix them but send it to hotelbeds technical support. Although there is a set of application error codes that depends on the XML request you sent. The A12-02-XXX-XX errors are caused by a XML Schema restriction violation. A description message in the requested language will be provided to allow you to correct the error.
Example:	A12-02-A3R-05

XI.2.2 BUSSINES LOGIC ERRORS

Format:	BXX-XX-XXX-XX where X can be any character between [A-Z] or [0-9]
Cause:	These errors are caused by a bussines logic restriction violation. A description message in the requested language will be provided to allow you to correct the error.
Example:	B01-01-123-07

XI.2.3 COMMUNICATION ERRORS

Format:	CXX-XX-XXX-XX where X can be any character between [A-Z] or [0-9]
Cause:	These errors are caused by a grant error. They occur when you are accesing from an unknown ip, you are accessing to the wrong environment or your credential details are wrong.
Example:	C01-01-007-11