BOOK YOUR HOLIDAY hostelsclub

Hostelsclub.com

XML Web Service Integration Guide for Properties

Index

1.About this document....4
1.1.Who should read this document....4
1.2.Conventions used in this documentation....4
1.2.1.Prices....4
1.2.2.Capitalization....4
2.What does this XML Web Service actually do?....4
3.Open Travel Alliance Standards....6
4.Documentation version....6
4.1.What's new in this version?....6
5.Getting started....7
6.How to send requests....7
6.1.Note for PHP/curl users....8
6.2.Note for C# users....8
7.Basic request structure....8
8.Authentication....10
9.Basic response structure....10
9.1.Low-level errors....10
9.1.1.No data....10
9.1.2.Malformed data....10
9.1.3.Authentication Failure....11
9.1.4.Message not supported....11
9.2.Success/Error tags and main response element....11
9.2.1.Root element attributes for the response message....12
9.2.2.Request results....12
9.2.3.The <Errors> tag - insufficient data....13
9.2.4.The <Success/> and <Warnings> tags....13
9.2.5.The main response element....14
9.2.6.Errors vs. Warnings....15
10.Business logic....15
10.1.Obtaining a hotel's HotelCode....15
11.Messages....16
11.1.The OTA_PingRQ/RS....16
11.2.The OTA_ReadRQ/OTA_ResRetrieveRS....17
11.2.1.Retrieving a single reservation: The OTA_ReadRQ message....17
11.2.2.Retrieving a single reservation: The <OTA_ResRetrieveRS> message....18
11.2.2.1.The <HotelReservation> tag....21
11.2.2.1.1.The <RoomStays> tag....21
11.2.2.1.2.The <ResGuests> tag....23
11.2.2.1.3.The <ResGlobalInfo> tag....23
11.2.2.1.4.The <CancelPenalties> tag....24
11.2.2.1.5.The final <TPA_Extensions> tag....26
11.2.3.Retrieving a list of reservations: The OTA_ReadRQ message....27
11.2.3.1.The <Interval> tag....27
11.2.4.Retrieving a list of reservations: The <OTA_ResRetrieveRS> message....28
11.3.The OTA_HotelRateAmountNotifRQ / OTA_HotelRateAmountNotifRS....30
11.3.1.The OTA_HotelRateAmountNotifRQ message....30
11.3.2.The OTA_HotelRateAmountNotifRS message....33
11.4.The OTA_HotelAvailRQ / OTA_HotelAvailRS messages....33
11.4.1.The OTA_HotelAvailRQ message....34

11.4.1.1.The <StayDateRange> tag....34
11.4.2.The OTA_HotelAvailRS message....35
11.5.The OTA_HotelDescriptiveInfoRQ / OTA_HotelDescriptiveInfoRS messages....37
11.5.1.The OTA_HotelAvailRQ message....37
11.5.2.The OTA_HotelDescriptiveInfoRS message....38
12.The PUSH notification system for reservations and cancellations....40
13.Appendix....41
13.1.Error Types....41
13.2.Error Codes....42
13.3.Currency codes....42
13.4.Hotel Category codes....42
13.4.1.Property Class Type....42
13.4.2.Segment Category Code....42
13.5.OTA Room type codes....43
13.6.Hostelsclub Room type codes....43
13.7.Languages....43
13.8.Credit Card Codes....44
13.9.City and Country Codes....44

1. *About this document*

This document is the programmer's guide for the implementation of Hostelsclub.com's XML Web Service in a property management system. It will provide a basic documentation of the various XML requests and responses, a list of possible errors and error codes, as well as some suggestions on how to implement the system in your application.

1.1. Who should read this document

This document is meant for people who have technical experience of XML and programming in general. If you do not have this kind of experience, this document will probably provide you with some information on what can be achieved with our Web Service, but a good deal of it will look quite obscure.

1.2. Conventions used in this documentation

When we mention a tag in the XML messages, it will always be written like this: <tag>. When we mention a tag attribute, it will appear in *Italic*. Whenever a tag features a long series of attributes, they will be described in a table containing the attribute name in the left column, and the explanation on the right column.

1.2.1. Prices

All prices contained in the messages are output using a dot (.) for decimal places. Prices will have a maximum of two decimal places, regardless of the currency used. Prices might, in some instances, not have decimal places at all or have just one decimal place.

1.2.2. Capitalization

All the system, both for tag names and attribute values, is case-sensitive. This means that if you should send the <HotelReservations> tag and you send <hotelreservations> instead, our system will not recognize it. If an attribute value is "true" and you send "TRUE" or "True", it will not be recognized. If you send the *Target* attribute with a value of "test", instead of "Test", it will not be recognized and the system will switch to production mode.

Always use the same capitalization of the examples, copying and pasting from the documentation if necessary, or making reference to the OTA standards when in doubt.

2. *What does this XML Web Service actually do?*

This Web Service allows PMSs to make requests to our booking engine and receive an XML-formatted reply. The actions currently available are the following:

- Update of property availability
- Request of a list of reservations which have been created/cancelled within a certain time span
- Request of details for a single reservation
- Request of current availability for a property

- Our system allows PMS programmers to set up a notification URL where our system sends a notice whenever a reservation for the property is created or cancelled. Whereas all other calls are of PULL type, that is the PMS asks for information and our servers reply, this is a PUSH case, that is our servers notifies the PMS of an event. This system is described in a separate chapter at the end of the document (see Index).

3. *Open Travel Alliance Standards*

Our XML implementation follows the standards designed by the Open Travel Alliance[1] (OTA, http://www.opentravel.org/) standards 2006A.

This standard was developed by the OTA to make it easier for companies to implement Web Services and therefore to facilitate trade between companies. Refer to the web address above for more information on the OTA.

The OTA standards cover almost all possible business cases for each type of request, so our implementation is inevitably a subset of the complete standard, and won't feature all possible combination of elements and requests provided for by the OTA standard.

This guide you are reading should be sufficient to create a set of requests that are compatible with our implementation of the OTA standards. However, if you want to know more about the standards themselves, please visit the web address above.

4. *Documentation version*

As we add more features to our booking system, this documentation will be revised in order to illustrate changes and the new functionalities installed in our system. Please make sure you have the latest version. We will contact all affiliates who are using the system whenever there are changes to the available features, and provide them with the new documentation.

Every effort will be made to make changes to the system in a backwards-compatible way. If we will ever need to make changes that could disrupt the correct functioning of existing applications, we will warn you well in advance, so that you have time to make the appropriate changes to your system.

4.1. What's new in this version?

Version	***Changes log***
1.3.16	CancelDateTime attribute in list of reservations
1.3.15	Better description of cancellation policies in reservation details
1.3.14	Added chapter 6.2 for C# users
1.3.13	Information added: where to find your hotel's ID
1.3.12	Added infos about “Too many connections” error in Appendix
1.3.11	Extended <Fees> tag and added <RoomRates> tag to HotelReservation in OTA_ResRetrieveRS
1.3.10	Added note on connection problems when using PHP curl
1.3.9	<ArrivalTime> tag added to TPA extension in OTA_ResRetrieveRS
1.3.8	PUSH notification system for reservations, see chapter 12
1.3.7	Minor corrections of texts

1 All content related to the Open Travel Alliance is Copyright© Year 2006, OpenTravel™ Alliance, Inc.

1.3.6	Added OTA_HotelDescriptiveInfoRQ/OTA_HotelDescriptiveInfoRS messages
1.3.5	Added SiteID to TPA extension in OTA_ReadRQ
1.3.4	Added RoomName to RoomBreakdown TPA extension in OTA_ReadRQ
1.3.3	Documentation of RoomBreakdown TPA extension in OTA_ReadRQ
1.3.2	Room type codes used in availability insertion should be obtained from property backoffice, and not from attached spreadsheets. Please refer to .section 11.3.1, *The OTA_HotelRateAmountNotifRQ message*, for details
1.3.1	SpecialRequests tag added in OTA_ResRetrieveRS containing customer messages to the property
1.3.0	There is one important (backwards-compatible change in this documentation). It concerns the way room codes are treated in the room insertion process. Please refer to section 11.3.1, *The OTA_HotelRateAmountNotifRQ message*, for details

5. Getting started

Our XML Web Service cannot be accessed by unauthorized requests. You will first need to contact our Technical department to get a test account activated before you can make requests. It is advisable to start implementing the system on a test property (hotel), and check the results against the property's back office in order to see whether your programming does the desired effect.

After you've developed your implementation, a member of our staff will review it, and if your implementation passes the review, we will activate an account for the real properties you manage in our system.

If your implementation does not pass the review, our staff will suggest the changes to be made to correct the problems encountered.

6. How to send requests

The addresses to which to send requests are the following:

https://www.hostelspoint.com/xml/xml.php

The https secure protocol is required for all requests. More on this in the "Authentication" chapter below.

To receive a response, you must send your XML request as a POST variable called "OTA_request". This can be achieved through various programming methods, which will vary according to the programming language you're using. For example, in PHP the simplest way to send this kind of requests is using the CURL extension.

Make sure that you properly encode your XML request (URL encoding), otherwise ampersands (&s) which are present in your request will be interpreted by our system as the start of a new variable in POST and the XML will be considered invalid. If you don't have a tool that does this for you, you should basically substitute all ampersands with %26, which is the URL-equivalent of &. Make sure that all ampersands are still XML-entity encoded, so they look like this: &amp; before the conversion, and %26amp; after the conversion.

Our system will return an XML file.

6.1. Note for PHP/curl users

If you use php, curl and the POST content length is 1,025 or greater, then curl exploits a feature of http 1.1: 100 (Continue) Status.

See http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.2.3

* it adds a header, "Expect: 100-continue".

* it then sends the request head, waits for a 100 response code, then sends the content

Since our server are located behind a proxy, this can lead to a "INVALID REQUEST" error or "HTTP Error 417 – Expectation Failed". To workaround this problem you have to suppress the "Expect" header with this command:

```
<?php
...
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Expect:'));
...
?>
```

6.2. Note for C# users

Although we are not C# programmers ourselves, we are told by some users of the Web Service that the same problem mentioned in point 6.1 above applies to C# as well, and that you need to set the following option:

```
System.Net.ServicePointManager.Expect100Continue = false;
```

to get the queries working because the header is automatically added by the system.

7. Basic request structure

All requests must be sent in UTF-8 encoding.

A sample XML request will look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_ReadRQ xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_ReadRQ.xsd" EchoToken="3229853" TimeStamp="2003-11-
10T10:34:00" Target="Production"
Version="1.006" SequenceNmbr="1" PrimaryLangID="en-us"
ReturnListIndicator="0">
<POS>
<Source>
<RequestorID ID="login" MessagePassword="password" Type="10"/>
</Source>
</POS>
     <ReadRequests>
```

```
          <HotelReadRequest HotelCode="1">
            <TPA_Extensions>
             <Interval  EndDate="2007-01-19T05:07:30-05:00"
Duration="-PT02H00M" />
            </TPA_Extensions>
          </HotelReadRequest>
     </ReadRequests>
</OTA_ReadRQ>
```

This is a sample request for a list of reservations made for hotel number 1 in our system (there's no hotel number 1 in our system, by the way). Do not worry about details of the request. At this step we will only look at the **structure** of the request itself.

In the first row, you see the XML declaration. Depending on how you will produce your XML, this might be generated automatically for you when you export the document tree.

The first element in the XML, the *root element*, must have a name of one of the supported request types, in this case the *OTA_ReadRQ* . We will list all of these later on.

The root element may include several attributes: some of these are specific to one message type, others (such as xmlns) pertain to XML specifications, others are compulsory for all or most requests.

Here's a list of the main attributes supported by ALL calls to our system (all other optional, OTA-defined attributes that are not listed here and are not listed in specific request descriptions, such as AltLangID or SequenceNmbr, are ignored by our system – they can be sent anyway, though):

Attribute	***Notes***
EchoToken	This is valid for all messages. Although not compulsory, it is very important because it will be returned AS IT IS (as long as it does not exceed 128 chars length) by our system in an attribute of the same name in our response. It allows your system to pair your requests to our responses, especially in the cases in which the same program instance is sending more than one request at a time. It can be a string of up to 128 characters.
Target	This attribute can contain the values "Production" or "Test". If this attribute is not sent, our system will default to "Production". Every message sent with this attribute set to “Test” **will not produce changes in the database**. If this attribute is set to “Test”, our response XML will be indented, so it can be easily read by a person.
Version	This attribute refers to the OTA Version of the message being sent. Our system does not block requests that are earlier than the version we are supporting, and will try to reply to them if possible, but it will issue a warning about the fact that the version of the request is earlier than the one supported. If you're not familiar with OTA Message Versions, please use the ones in the sample requests contained in this document.
PrimaryLangID	This is expected to be a two-letter code identifying the language you want to receive customer-displayable data in. Currently, all messages concerning PMS integration will always be in English. All system error messages will still be issued in English. If the language you request is not supported or not recognized, the system will issue a warning and proceed returning data in English.
xmlns	This attribute is compulsory. It indicates the namespace of the xml document. It is better set exactly as you find it in each sample request in this document.

The inner content of each message (the tags contained in the root element) will vary from message to message, but authentication tags are common to most requests, so they will be discussed next.

8. Authentication

The only request that does not require authentication is, at the moment, the OTA_PingRQ, which is used for preliminary testing, or (where needed) to test whether our system is live and responding. All other requests require you to send authentication credentials using the POS tag as follows:

The POS tag (which must be positioned right after the root element) must contain a Source Tag, which in turn must contain a RequestorID tag. This last tag must contain an attribute ID containing **the login credentials to the property's back office**.

The RequestorID tag **must also** contain a MessagePassword attribute, which must contain the password to the property's back office. It must also contain a Type attribute set to 10 (it is needed by our system to know that the current request comes from a property). Please make sure you follow the correct authentication procedure: if you produce too many failed authentication attempts in too short an amount of time, your account may be blocked (blocking web back office access as well). If your think your account has been blocked, contact our staff to unblock it.

9. Basic response structure

Before describing the various requests and response types, we will describe the various error levels, then the elements that are in common to all or most response types.

9.1. Low-level errors

The OTA specifications provide several error levels: the most basic errors are the following:

9.1.1. No data

If there is no POST variable called OTA_request, our system will reply with an empty page.

9.1.2. Malformed data

In case the XML you sent in the OTA_request variable is malformed, our system will reply with a general OTA_ErrorRS response, which will look like this:

```
<?xml version="1.0" encoding="UTF-8" ?><OTA_ErrorRS
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/2006A/OTA_ErrorRS.xs
d"  TimeStamp="2007-01-09T11:15:31-05:00" Target="Production"
Version="1.004" PrimaryLangID="en" Status="NotProcessed"
ErrorCode="Malformed" ErrorMessage="Malformed XML - errors:
End tag : expected '>'
 - node name: EchoData - line 8 - column 1
Opening and ending tag mismatch: EchoData line 7 and Ech
 - node name: EchoData - line 8 - column 1
```

```
"></OTA_ErrorRS>
```

The ErrorCode attribute of the root element will contain the general error code, as per OTA specifications. The ErrorMessage will contain the error output of our parsing program, with details on errors in the received XML, in order to help you correct eventual errors.

9.1.3. Authentication Failure

In case your request could not be authenticated (either because the credentials are invalid, or you are trying to send a request that requires https through the http protocol), our system will reply with a general OTA_ErrorRS response, which will look like this:

```
<?xml version="1.0" encoding="UTF-8" ?><OTA_ErrorRS
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/2006A/OTA_ErrorRS.xs
d"  EchoToken="1"  TimeStamp="2007-01-09T11:19:46-05:00"
Target="Production" Version="1.004" PrimaryLangID="tw"
Status="NotProcessed" ErrorCode="AuthFail"
ErrorMessage="Authentication failure"></OTA_ErrorRS>
```

The ErrorCode and ErrorMessage here are quite self-explanatory.

Important: **please keep in mind** that the repeated failure to send the correct password for an account **can lead to it being locked by our system**. This means that you could also block human action in the property's backoffice. If you think an account is locked, please contact us to verify it. Repeated failures to login successfully can also lead to the blocking of your IP by our system.

It must also be noted that you must make sure that the property's staff keep the password updated in the system. If they request a password change, all PMS automatic tasks should be stopped before the newly received password has been inserted. We might create separate accounts for XML in the future, but currently this is the way our system behaves.

9.1.4. Message not supported

In case your request uses a root element (message type) which is incorrect, or not supported by our platform, the system will reply with a general OTA_ErrorRS response, which will look like this:

```
<?xml version="1.0" encoding="UTF-8" ?><OTA_ErrorRS
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/2006A/OTA_ErrorRS.xs
d"  EchoToken="1"  TimeStamp="2007-01-09T11:21:26-05:00"
Target="Production" Version="1.004" PrimaryLangID="tw"
Status="NotProcessed" ErrorCode="UnrecognizedRoot"
ErrorMessage="REQUEST TYPE NOT RECOGNIZED OR NOT
SUPPORTED"></OTA_ErrorRS>
```

9.2. Success/Error tags and main response element

If your request has passed the low-level error checks above, the root element of the response will be that of the correct response type, corresponding to the request type you sent (for ex., *OTA_HotelAvailRS* for the *OTA_HotelAvailRQ* request type).

9.2.1. Root element attributes for the response message

The root element of the response message will always contain, beside *xmlns* and other standard attributes, the following attributes:

Attribute	***Notes***
EchoToken	This returns the content of the attribute of the same name in your request (see above). It allows your system to pair your requests to our responses, especially in the cases in which the same program instance is sending more than one request at a time.
Target	This attribute can contain the values "Production" or "Test". If this attribute has not been sent in your request, our system will default to "Production". However, if your account is not authorized for production requests, our system will switch to testing mode. In all cases which require some action on our databases, this attribute will tell you if your request has actually been processed: if this attribute is set to "Test" in our response, it means that no actual reservation or cancellation has taken place, but they have just been simulated for testing purposes.
Version	This attribute refers to the OTA Version of the message being sent. Our system does not block requests that are earlier than the version we are supporting, and will try to reply to them if possible, but it will issue a warning about the fact that the version of the request is earlier than the one supported. If you're not familiar with OTA Message Versions, please use the ones in the sample requests contained in this document.
PrimaryLangID	If the system has recognized your language code, or part of it (for example, we generally parse only the first two letters of language codes like en-gb, with the exception of zh-tw and zh-cn for Traditional and Simplified Chinese), it will return it as you sent it. Most error/warning messages will still be in English, however.
TimeStamp	An attribute containing the time stamp of message processing according to the ISO 8601 standard: YYYY-MM-DDThh:mm:ssZ with time values using the 24 hours (e.g. 20 January 2006, 2:58:12 pm UTC becomes 2006-01-20T14:58:12Z). The Z in the pattern above will be substituted with the time zone indication of the responding server, taking into account daylight saving time.

9.2.2. Request results

If your request has passed the low-level error checks, there are three possible results of our processing it:

1. The message **contains insufficient data to process the request**: for example, some data is missing or not correctly formatted.
2. The message **contains the correct data**, but the **request itself failed**: for example, you have tried to insert availability in a room type that has not been inserted in your back office.
3. The message **contains the correct data**, and the **request is successful**: this is the case when everything goes as intended: your request is correct, we could process it and have a positive response.

The OTA Standards provide for two different ways of indicating that a message was not successful, one covering case 1. above, and the other covering case 2.

9.2.3. The <Errors> tag - insufficient data

In the first case, the response message contains, as the first child of the root element, an <Errors> tag containing one or more <Error> tags indicating the code for each error encountered and a brief textual explanation of the error itself. Each <Error> tag will have two attributes, *Code* indicating the actual error code according to OTA standards, and *Type* indicating the type of error (for ex. Required field missing, Authentication, Business Rules) according to OTA standards. The <Error> tag will have a text content which explains exactly what error took place.

You will find a list of error codes (and warning codes) used in the appendix of this document, with explanation of what they mean.

The response XML will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<OTA_HotelAvailRS xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelAvailRS.xsd" TimeStamp="2007-01-09T12:11:25-05:00"
Version="1.006" PrimaryLangID="de" Target="Test">
  <Errors>
    <Error Code="321" Type="3"> No means of identifying the city
requested found. You can pass a Hclub city id in the HotelRef tag
- HotelCityCode attribute or a city name with country and
optionally state in the Address tag</Error>
  </Errors>
</OTA_HotelAvailRS>
```

As you can see, here the root element is *OTA_HotelAvailRS* (which is the response for a request for availability, as opposed to the *OTA_ErrorRS* above). But the message's information was not sufficient to proceed, therefore the response only contaiins <Error> tags.

9.2.4. The <Success/> and <Warnings> tags

The second case is a bit trickier. The message has been successfully parsed, and so you will find a <Success/> tag (with no attributes or content) as first child of the root element. This is the indicator of the fact that the message was successfully processed.

However, this does not guarantee that there is data to respond to your request. This might be for any reason listed above at point 2. In this case, there will be a <Warnings> tag containing as many warnings as the number of "errors" encountered in processing your request. In the case of a **failed request**, the <Warnings> tag will follow the <Success/> tag, and will not be followed by other tags, as in the following example:

```
<?xml version="1.0" encoding="utf-8"?>
<OTA_HotelAvailRS xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelAvailRS.xsd" TimeStamp="2007-01-09T12:16:14-05:00"
Version="1.006" PrimaryLangID="de" Target="Test">
  <Success/>
  <Warnings>
    <Warning Code="357" Type="10"> City not found in
database</Warning>
  </Warnings>
```

```
</OTA_HotelAvailRS>
```

In the case above, there is one single <Warning> tag, which indicates that although you passed a sufficient amount of data to proceed with a city search, the search itself has returned no results. <Warning> tags also have a *Code* and *Type* attribute and a textual content, exactly like <Error> tags. The codes follow OTA standards.

9.2.5. The main response element

If a request is successful, it can still return warnings, usually "trivial" ones that are meant to point out small problems concerning your request, which could be ignored or corrected by our application.

The way to test that a request was *really successful*, you have to look for the *main response element*. According to the OTA Standards, every message has one or more main response elements (although our implementation usually uses just one per message use case), depending on the type of request.

Let's look at a simple example, using the *OTA_PingRQ*, which basically just tests that the system is live and that it correctly returns data:

```
<?xml version="1.0" encoding="utf-8"?>
<OTA_PingRS xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_PingRS.xsd" TimeStamp="2007-01-09T12:23:07-05:00"
Version="1.001" PrimaryLangID="en" Target="Test">
  <Success/>
  <Warnings>
    <Warning Code="184" Type="3"> Request language not supported -
switching to English</Warning>
    <Warning Code="999" Type="2"> Your request's version is
earlier than our supported version. We tried to process your
request in case the versions are compatible. We support version
1.001 for the OTA_PingRQ call.</Warning>
  </Warnings>
  <EchoData>Hello</EchoData>
</OTA_PingRS>
```

The main response element for this request type is the <EchoData> tag, which reflects exactly the content of the corresponding tag in the request message (apart from the fact that we will truncate its content if it is too long).

The response above tells that the message was successfully parsed, but two warnings were issued: one concerning the language of the request, which was not recognized (in these cases our system switches to English), and another concerning the fact that the message version of the request was lower than 1.001, which is the minimum request version for our implementation (but our system will try to parse it anyway in case the messages are compatible).

Besides that, the request was successful and the response returns the "Hello" content in the <EchoData> tag.

So, when you want to check that the request was successful, we suggest you to check if the expected main response node (or nodes) for your requests is actually present in the response. In that case, you should parse its contents and proceed with displaying data to the user. Otherwise, you should check Warnings first and Errors next. Warnings and

Errors exclude themselves mutually: you will not find a response containing both Errors and Warnings.

9.2.6. Errors vs. Warnings

Sometimes the difference between Errors and Warnings in the programming logic is a bit blurry: what must be considered an error and what must be considered a warning? We always followed the logic mentioned above: if the data is sufficient, we try to process the request and issue warnings if it fails. If the data is insufficient or incorrect in format, we issue an error. In any case, the text in the Error/Warning description will make clear what kind of problem was encountered.

10. Business logic

The OTA standards allow to implement almost any business logic with their messages: we (inevitably) adapted them to our business logic, trying to use the <TPA_Extensions> tag as little as possible; the <TPA_Extensions> tag is used to include XML data that cannot fit the standard request models. In some cases (both at request and response level) this could not be avoided.

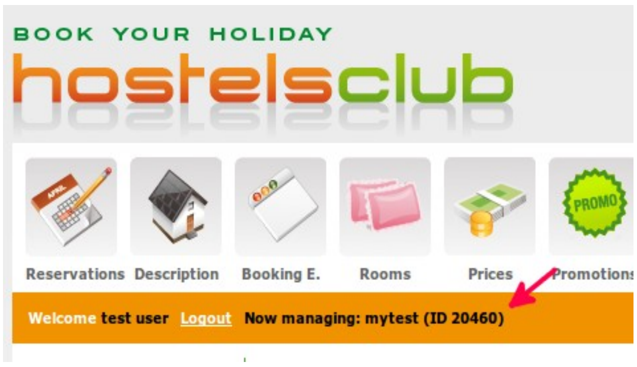
Our implementation currently allows the following actions:

1. **Searching reservations made or cancelled within a certain time interval** – you can provide start and end time, or a start/end date and a time interval
2. Getting **details about a single reservation**
3. **Inserting availability** by sending price, from and to date, room type and number of units to insert
4. **Checking current availability** for a property for a given time interval

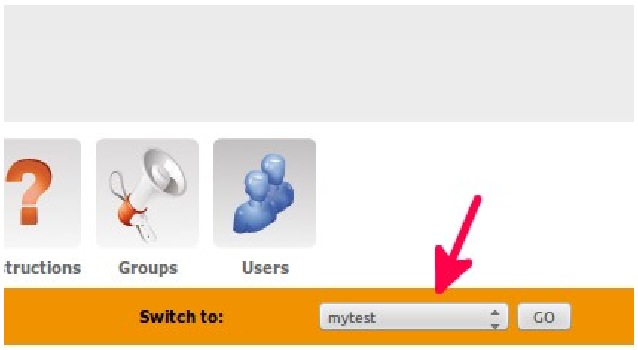
Procedure	***Request message***	***Response message***
Request for **details on a reservation**	OTA_ReadRQ	OTA_ResRetrieveRS
Searching reservations	OTA_ReadRQ	OTA_ResRetrieveRS
Inserting availability	OTA_HotelRateAmountNotifRQ	OTA_HotelRateAmountNotifRS
Checking current availability	OTA_HotelAvailRQ	OTA_HotelAvailRS
Checking room types that have been inserted, regardless of availability	OTA_HotelDescriptiveInfoRQ	OTA_HotelDescriptiveInfoRS
Ping	OTA_PingRQ	OTA_PingRS

10.1. Obtaining a hotel's *HotelCode*

The hotel's *HotelCode* which must be sent in many of the messages is a numeric code which identifies the hotel univocally in our system. It can be found in our property back office, on every page, on the top bar:



If your account manages more than one hotel (which is very likely if you work for a Property Management System), to find the *HotelCode* of each property you manage you just need to switch to the right property using the drop-down menu in the top bar:



You will be able to manage that property and see that property's *HotelCode* in the top bar.

11. *Messages*

We'll start describing the usage of the simpler message types, then proceed to those that concern themselves with the reservation procedure.

In general, we won't describe details of messages that are self-explanatory.

Most of the request/reply messages you will find in this guide will have *Target* attributes set to "Test", not "Production". You must make sure that you request has the appropriate value, depending on whether you're making a test or a production request. Always check the value of the *Target* attribute in our response to make sure you've proceeded with the intended modality.

11.1. The OTA_PingRQ/RS

This message pair is the simplest of them all, and it is meant for applications to check if the other end is live and able to receive messages. This could easily be done by sending any other message and seeing if it gets a reply, but we've decided to implement this message because it's useful if you want to start implementing your application using the most basic of messages first, then proceeding to the more sophisticated ones.

The OTA_PingRQ looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_PingRQ xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
```

```
OTA_PingRQ.xsd" TimeStamp="2003-03-17T11:09:47-05:00"
Target="Production" Version="1.001" PrimaryLangID="en"
EchoToken="testtoken12">
<EchoData>Hello</EchoData>
</OTA_PingRQ>
```

This message contains just an <EchoData> tag with a string content. The reply will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<OTA_PingRS xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_PingRS.xsd" TimeStamp="2007-01-10T10:46:31-05:00"
Version="1.001" PrimaryLangID="en" EchoToken="testtoken12"
Target="Test">
  <Success/>
  <EchoData>Hello</EchoData>
</OTA_PingRS>
```

As you can see, the system returned the <EchoData> tag as it was, as well as the EchoToken contained in the request.

The OTA_PingRQ message is the only message that does not require authentication data to be passed along in the XML.

11.2. The OTA_ReadRQ/OTA_ResRetrieveRS

This message is used to retrieve data about an existing reservation or to retrieve a list of reservations. Note that the two messages (request and response) have a different base name. The request is called OTA_Read, and the response is OTA_ResRetrieve. If you implement a system that automatically recognizes the root element, it is advisable that you make pairs association request->response, instead of assuming that the response message will have a root element that is identical to the RQ message root element, with RS instead of RQ.

This message has two use cases: one for retrieving data about a specific reservation, of which the ID is known, and another for retrieving a list of reservations.

11.2.1. Retrieving a single reservation: The OTA_ReadRQ message

The message looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_ReadRQ xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_ReadRQ.xsd" EchoToken="5553429" TimeStamp="2003-03-
17T09:30:47-05:00" Target="Production"
Version="1.006" SequenceNmbr="1" PrimaryLangID="it">
<POS>
<Source>
<RequestorID ID="login" MessagePassword="password" Type="10"/>
```

```
</Source>
</POS>
<ReadRequests>
<ReadRequest>
<UniqueID Type="14" ID="249695" />
</ReadRequest>
</ReadRequests>
</OTA_ReadRQ>
```

This message contains just a <ReadRequests> tag, containing a <ReadRequest> tag, which in turn contains a <UniqueID> tag, with attributes *Type* set to 14 to indicate Reservation Confirmation Number, and *ID* containing our reservation ID, which is the one you received from the confirmation message.

Our system will allow you to read all requests made for a property managed by the back office user identified by the credentials of the <POS> node.

As usual, our system will accept one request at a time. If you want to retrieve multiple reservations, please make multiple calls.

11.2.2. Retrieving a single reservation: The <OTA_ResRetrieveRS> message

This message will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
    <OTA_ResRetrieveRS xmlns="http://www.opentravel.org/OTA/2003/05"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
    OTA_ResRetrieveRS.xsd" TimeStamp="2008-10-17T06:01:36-04:00"
    Version="1.006" PrimaryLangID="it" EchoToken="5553429"
    Target="Test">
      <Success/>
      <ReservationsList>
        <HotelReservation ResStatus="Reserved">
          <RoomStays>
            <RoomStay>
              <RoomTypes>
                <RoomType RoomTypeCode="2_20" NumberOfUnits="3">
                  <RoomDescription Language="it">
                    <Text Language="it">6 letto/i camerata mista con
    WC</Text>
                  </RoomDescription>
                </RoomType>
              </RoomTypes>
              <RoomRates>
                <RoomRate RoomTypeCode="2_20" NumberOfUnits="1">
                  <Rates>
                    <Rate UnitMultiplier="1" RateTimeUnit="Day"
    EffectiveDate="2008-11-12" ExpireDate="2008-11-14">
                      <Base AmountBeforeTax="60.00"
    CurrencyCode="EUR"/>
                      <Discount AppliesTo="Base" Percent="5.00">
                        <DiscountReason>
                          <Text>Discount description</Text>
                        </DiscountReason>
                      </Discount>
                      <Total AmountBeforeTax="57.00"
    CurrencyCode="EUR"/>
```

```
                  </Rate>
                </Rates>
              </RoomRate>
              <RoomRate RoomTypeCode="2_20" NumberOfUnits="1">
                <Rates>
                  <Rate UnitMultiplier="1" RateTimeUnit="Day"
EffectiveDate="2008-11-14" ExpireDate="2008-11-15">
                    <Base AmountBeforeTax="65.00"
CurrencyCode="EUR"/>
                    <Discount AppliesTo="Base" Percent="5.00">
                      <DiscountReason>
                        <Text>Discount description</Text>
                      </DiscountReason>
                    </Discount>
                    <Total AmountBeforeTax="61.75"
CurrencyCode="EUR"/>
                  </Rate>
                </Rates>
              </RoomRate>
              <RoomRate RoomTypeCode="2_20" NumberOfUnits="1">
                <Rates>
                  <Rate UnitMultiplier="1" RateTimeUnit="Day"
EffectiveDate="2008-11-15" ExpireDate="2008-11-16">
                    <Base AmountBeforeTax="70.00"
CurrencyCode="EUR"/>
                    <Discount AppliesTo="Base" Percent="5.00">
                      <DiscountReason>
                        <Text>Discount description</Text>
                      </DiscountReason>
                    </Discount>
                    <Total AmountBeforeTax="66.50"
CurrencyCode="EUR"/>
                  </Rate>
                </Rates>
              </RoomRate>
           </RoomRates>
           <BasicPropertyInfo HotelCode="2530" HotelName="Test
establishment" HotelCityCode="1193">
<Address>
<AddressLine>Test address</AddressLine>
<CityName>Zafferana Etnea</CityName>
<CountryName Code="IT">Italy</CountryName>
</Address>
            <ContactNumbers>
              <ContactNumber PhoneNumber="+39 - 00 -00 00 00
0000"/>
            </ContactNumbers>
          </BasicPropertyInfo>
          <TPA_Extensions>
            <HotelEmail>info@hostelsclub.com</HotelEmail>
            <Directions Language="it">
              <Text Language="it">HOW TO FIND US?by train...
</Text>
            </Directions>
          </TPA_Extensions>
        </RoomStay>
      </RoomStays>
      <ResGuests>
        <ResGuest>
```

```
          <Profiles>
            <ProfileInfo>
              <Profile>
                <Customer Gender="Female">
                  <PersonName>
                    <GivenName>francesco</GivenName>
                    <Surname>rossi</Surname>
                  </PersonName>
                  <Email>customer@hotmail.it</Email>
                  <CitizenCountryName Code="IT"/>
                </Customer>
              </Profile>
            </ProfileInfo>
          </Profiles>
        </ResGuest>
      </ResGuests>
      <ResGlobalInfo>
        <TimeSpan Start="2008-11-12" End="2008-11-16"/>
        <DepositPayments>
          <GuaranteePayment GuaranteeType="PrePay" NameInd="1"
PaymentCode="5" NonRefundableIndicator="true">
            <AmountPercent FeesInclusive="false" Amount="19.20"
CurrencyCode="EUR"/>
            <Description Language="it">
              <Text Language="it">Anticipo - 10% del prezzo
totale</Text>
            </Description>
          </GuaranteePayment>
        </DepositPayments>
        <CancelPenalties>
          <CancelPenalty>
            <Deadline OffsetDropTime="BeforeArrival"
OffsetTimeUnit="Day" OffsetUnitMultiplier="2"/>
            <AmountPercent NmbrOfNights="1"/>
          </CancelPenalty>
        </CancelPenalties>
        <Fees>
          <Fee ChargeFrequency="26" Amount="2" Code="14"
CurrencyCode="EUR" MandatoryIndicator="true" TaxInclusive="true"
Type="Inclusive">
            <Description Language="it">
              <Text Language="it">Costo fisso servizio di
prenotazione</Text>
            </Description>
          </Fee>
          <Fee Amount="2.00" Code="14" CurrencyCode="EUR"
TaxInclusive="true" Type="Inclusive">
            <Description Language="it">
              <Text Language="it">Internet</Text>
            </Description>
          </Fee>
        </Fees>
        <Total AmountBeforeTax="194" CurrencyCode="EUR"/>
        <HotelReservationIDs>
          <HotelReservationID ResID_Value="715965" ResID_Type="14"
ForGuest="true"/>
        </HotelReservationIDs>
      </ResGlobalInfo>
      <TPA_Extensions>
```

```
        <HotelPayment AmountBeforeTax="172.80"
CurrencyCode="EUR"/>
              <SiteID>125</SiteID>
              <ArrivalTime Time="10:00"/>
               <RoomBreakdown>
          <RoomDay Date="2008-11-12" RoomTypeCode="2_20"
Amount="48.00" Beds="3" InvBlockCode="18832" RoomName="Room 11"/>
          <RoomDay Date="2008-11-13" RoomTypeCode="2_20"
Amount="42.00" Beds="3" InvBlockCode="18832"  RoomName="Room 11"/>
          <RoomDay Date="2008-11-14" RoomTypeCode="2_20"
Amount="45.00" Beds="3" InvBlockCode="18859" RoomName="Room 12"/>
          <RoomDay Date="2008-11-15" RoomTypeCode="2_20"
Amount="57.00" Beds="3" InvBlockCode="18859" RoomName="Room 12"/>
        </RoomBreakdown>
      </TPA_Extensions>
    </HotelReservation>
  </ReservationsList>
</OTA_ResRetrieveRS>
```

Let's look at its components. The reservation requested will be contained in a <ReservationsList> tag, which will contain only one <HotelReservation> element. The *ResStatus* attribute of the <HotelReservation> tag can have two values: "Cancelled", if the reservation has been cancelled, or "Reserved" in all other cases. If the reservation has been cancelled, the room breakdown data is not present any more.

Please note that this message will only allow the retrieval of a reservation made for one of the hotels which are administered by the login/password pair provided.

The response message will **not** contain details of the customer's credit card. These must still be retrieved using the system in our back office.

11.2.2.1. The <HotelReservation> tag

The <HotelReservation> tag contains the following information:

11.2.2.1.1. The <RoomStays> tag

```
     <RoomStays>
        <RoomStay>
          <RoomTypes>
            <RoomType RoomTypeCode="2_20" NumberOfUnits="3">
              <RoomDescription Language="it">
                <Text Language="it">6 letto/i camerata mista con
WC</Text>
              </RoomDescription>
            </RoomType>
          </RoomTypes>
         <BasicPropertyInfo HotelCode="2530" HotelName="Test
establishment" HotelCityCode="1193">
<Address>
<AddressLine>Test address</AddressLine>
<CityName>Zafferana Etnea</CityName>
<CountryName Code="IT">Italy</CountryName>
</Address>
            <ContactNumbers>
              <ContactNumber PhoneNumber="+39 - 00 -00 00 00
0000"/>
```

```
              </ContactNumbers>
            </BasicPropertyInfo>
            <TPA_Extensions>
              <HotelEmail>info@hostelsclub.com</HotelEmail>
              <Directions Language="it">
                <Text Language="it">HOW TO FIND US?by train...
</Text>
              </Directions>
            </TPA_Extensions>
          </RoomStay>
        </RoomStays>
```

This tag contains several subtags.

The <RoomTypes> tag contains the global room breakdown: for each room type booked, there will be a <RoomType> node, containing the room code, the number of units booked (beds or rooms, depending on room type), and the room description in the request language. **IMPORTANT:** since it is possible in our system to book reservations with change of room types between different days (one day a double with WC, one day a double without WC), this might not be the correct breakdown of rooms in the reservation, but just an indication of how many beds were booked per day. Please refer to the <RoomRates> tag.

The <RoomRates> tag *[starting from version 1.3.11]*

```
<RoomRates>
  <RoomRate RoomTypeCode="2_20" NumberOfUnits="1">
     <Rates>
       <Rate UnitMultiplier="1" RateTimeUnit="Day"
EffectiveDate="2008-11-12" ExpireDate="2008-11-14">
           <Base AmountBeforeTax="60.00" CurrencyCode="EUR"/>
             <Discount AppliesTo="Base" Percent="5.00">
               <DiscountReason>
                 <Text>Discount description</Text>
               </DiscountReason>
             </Discount>
           <Total AmountBeforeTax="57.00" CurrencyCode="EUR"/>
       </Rate>
     </Rates>
   </RoomRate>
   [...]
</RoomRates>
```

The <RoomRates> subtree contains informations about booked room rates, with a per-day-per-room-type basis. Each <RoomRate> tag contains the *RoomTypeCode* relative to the rate, the *NumberOfUnits* that indicates how many rooms of this type were booked.

The <Rate> tag contains the *time unit* that the Base rate refers (*UnitMultiplier* and *RateTimeUnit* are fixed to “1” and “Day” values respectively, because the Base rate is always per-Day and per-RoomType) and the rate validity period: *EffectiveDate* indicates the beginning date and *ExpireDate* indicates the end of the *validity period* (*ExpireDate* value is always the first day after the applicable period).

If some discount will be applied to the Base rate, the <Discount> tag shows the discount *Percent*, and the <DiscountReason><Text> tag shows the discount description.

Finally the <Total> tag show the real daily rate applied to the room type for the rate validity period (including discounts).

The <BasicPropertyInfo> tag

```
<BasicPropertyInfo HotelCode="2530" HotelName="Test establishment"
HotelCityCode="1193">
<Address>
<AddressLine>Test address</AddressLine>
<CityName>Zafferana Etnea</CityName>
<CountryName Code="IT">Italy</CountryName>
</Address>
            <ContactNumbers>
              <ContactNumber PhoneNumber="+39 - 00 -00 00 00
0000"/>
            </ContactNumbers>
          </BasicPropertyInfo>
```

This tag contains all information about the property. Name, code and address, as well as the property's phone number.

It will be followed by a <TPA_Extensions> tag:

```
<TPA_Extensions>
            <HotelEmail>info@hostelsclub.com</HotelEmail>
            <Directions Language="it">
              <Text Language="it">HOW TO FIND US?by train...
</Text>
            </Directions>
          </TPA_Extensions>
```

This tag contains further info about the property: the email, and the text directions on how to get to the property. The text directions will be in the request language, if available, otherwise they will be in English. The *Language* attribute of the <Directions> and <Text> tag will tell you what language the information is in.

11.2.2.1.2. The <ResGuests> tag

```
 <ResGuests>
        <ResGuest>
          <Profiles>
            <ProfileInfo>
              <Profile>
                <Customer Gender="Female">
                  <PersonName>
                    <GivenName>francesco</GivenName>
                    <Surname>rossi</Surname>
                  </PersonName>
                  <Email>customer@hotmail.it</Email>
                  <CitizenCountryName Code="IT"/>
                </Customer>
              </Profile>
            </ProfileInfo>
          </Profiles>
        </ResGuest>
      </ResGuests>
```

This tag will contain one <ResGuest> tag, with all the information about the customer: gender, name, telephone number (if provided at the time of booking), email and nationality code. Our system will always return one <ResGuest> tag, because we always request the information for one of the customers booking the room/s, not for all of them.

11.2.2.1.3. The <ResGlobalInfo> tag

```
<ResGlobalInfo>
```

```
        <TimeSpan Start="2008-11-12" End="2008-11-16"/>
        <DepositPayments>
          <GuaranteePayment GuaranteeType="PrePay" NameInd="1"
PaymentCode="5" NonRefundableIndicator="true">
            <AmountPercent FeesInclusive="false" Amount="19.20"
CurrencyCode="EUR"/>
            <Description Language="it">
              <Text Language="it">Anticipo - 10% del prezzo
totale</Text>
            </Description>
          </GuaranteePayment>
        </DepositPayments>
        <CancelPenalties>
          <CancelPenalty>
            <Deadline OffsetDropTime="BeforeArrival"
OffsetTimeUnit="Day" OffsetUnitMultiplier="2"/>
            <AmountPercent NmbrOfNights="1"/>
          </CancelPenalty>
        </CancelPenalties>
        <Fees>
          <Fee ChargeFrequency="26" Amount="2" Code="14"
CurrencyCode="EUR" MandatoryIndicator="true" TaxInclusive="true"
Type="Inclusive">
            <Description Language="it">
              <Text Language="it">Costo fisso servizio di
prenotazione</Text>
            </Description>
          </Fee>
        </Fees>
        <Total AmountBeforeTax="194" CurrencyCode="EUR"/>
        <HotelReservationIDs>
          <HotelReservationID ResID_Value="715965" ResID_Type="14"
ForGuest="true"/>
        </HotelReservationIDs>
      </ResGlobalInfo>
```

This tag contains the general information about the reservation. Start and End dates (remember that *End* is the check-out date), special messages from the user, the type of deposit and fees paid, cancel penalties, and the <Total> tag containing the total amount of the reservation, fees included (this is the total amount of all rooms plus all specified fees, not the sum of the deposit plus fees).

There is a <HotelReservationIDs> tag containing a <HotelReservationID> tag. This has the attribute *ResID_Value* which contains the reservation number, which is the Confirmation number (as indicated by the *ResID_Type* set to 14, the OTA code for Reservation). The *ForGuest* attribute indicates that this number should be shown to the end customer. The same reservation number goes to the property, as we do not use separate reservation IDs for internal and public use.

11.2.2.1.4. The <CancelPenalties> tag

We've implemented a more flexible cancellation policy system, which can be controlled from the property's backoffice. This reflects on the XML messages as follows. The property must choose a general cancellation policy for all bookings on our system, so what they see from the Web Service should not come as a surprise to them.

```
      <CancelPenalties>
            <CancelPenalty>
                  <Deadline OffsetDropTime="BeforeArrival"
      OffsetTimeUnit="Day" OffsetUnitMultiplier="1" />
                  <AmountPercent NmbrOfNights="1" />
                  <Description Language="en">
                        <Text Language="en">Free cancellation at least 24
      hours before arrival date, first night of stay fee for late
      cancellation or no show</Text>
                  </Description>
            </CancelPenalty>
      </CancelPenalties>
```

This is the OTA-compatible description of our standard cancellation policy: the reservation must be cancelled 1 day in advance (until midnight, server time, of the day before the day before the reservation starts), otherwise the establishment will be allowed to charge one room/night on the credit card provided for the reservation. **IMPORTANT**: previously we were expressing this concept with a value of 2 for *OffsetUnitMultiplier.* This was meant as a cautionary measure due to the fact that users might not be in the same time zone as our servers. Due to the new, extended, cancellation policies, this concept must now be expressed with a value of 1.

Some properties might have a different cancellation policy:

```
      <CancelPenalties>
            <CancelPenalty>
                  <Deadline OffsetDropTime="BeforeArrival"
      OffsetTimeUnit="Day" OffsetUnitMultiplier="4" />
                  <AmountPercent Percent="50" />
                  <Description Language="en">
                        <Text Language="en">We charge the 50% of BALANCE
      if cancelled less than 4 days before the arrival date</Text>
                  </Description>
            </CancelPenalty>
      </CancelPenalties>
```

in this case, the cancellation must take place 4 days before the reservation starts, and in case of a failure to cancel earlier than that, the property will charge 15% of the reservation value.
Therefore, the <AmountPercent> tag can have two possible attributes, which determine what kind of penalty must be paid: *Percent* if the amount is a percent of the total, *NmbrOfNights* if the penalty is equal to the cost of one or more nights' stay at the property, according to the prices in the reservation.

Some properties also have a different policy between a late cancellation and a “no show” (the customer does not arrive at the hotel at all, without cancelling in any way):

```
      <CancelPenalties>
            <CancelPenalty>
                  <Deadline OffsetDropTime="BeforeArrival"
      OffsetTimeUnit="Day" OffsetUnitMultiplier="4" />
                  <AmountPercent Percent="50" />
                  <Description Language="en">
                        <Text Language="en">We charge the 50% of BALANCE
      if cancelled less than 4 days before the arrival date</Text>
                  </Description>
            </CancelPenalty>
            <CancelPenalty>
                  <Deadline OffsetDropTime="BeforeArrival"
      OffsetTimeUnit="Day" OffsetUnitMultiplier="0" />
                  <AmountPercent Percent="50" />
                  <Description Language="en">
                        <Text Language="en">In the event of a no show we
      charge the 50% of BALANCE on customer's credit card</Text>
                  </Description>
            </CancelPenalty>
      </CancelPenalties>
```

In this case, the penalty is indicated by a <Deadline> of 0 Days, that is, there has been no cancellation before the user was supposed to arrive at the hotel.

In some cases, the cancellation policy can be text-only.

```
            <CancelPenalties>
                  <CancelPenalty>
                        <Description Language="en">
                              <Text Language="en">Custom textual
cancellation policy</Text>
                        </Description>
                  </CancelPenalty>
            </CancelPenalties>
```

We strongly suggest to print the text version of all the CancelPenalty nodes to your end-users, as they will be explanatory and will be translated in the request's language.

11.2.2.1.5. The final <TPA_Extensions> tag

There is another <TPA_Extensions> tag, which is a direct child of <HotelReservation> (instead of being a child of the <RoomStay> tag like the previous one):

```
<TPA_Extensions>
       <HotelPayment AmountBeforeTax="172.80" CurrencyCode="EUR"/>
       <SiteID>125</SiteID>
       <ArrivalTime Time="10:00"/>
        <RoomBreakdown>
          <RoomDay Date="2008-11-12" RoomTypeCode="2_20"
Amount="48.00" Beds="3" InvBlockCode="18832" RoomName="Room 11"/>
          <RoomDay Date="2008-11-13" RoomTypeCode="2_20"
Amount="42.00" Beds="3" InvBlockCode="18832"  RoomName="Room 11"/>
          <RoomDay Date="2008-11-14" RoomTypeCode="2_20"
Amount="45.00" Beds="3" InvBlockCode="18859" RoomName="Room 12"/>
          <RoomDay Date="2008-11-15" RoomTypeCode="2_20"
Amount="57.00" Beds="3" InvBlockCode="18859" RoomName="Room 12"/>
        </RoomBreakdown>
      </TPA_Extensions>
```

It contains the following tags:

1. the <HotelPayment> tag, which contains the amount to be paid to the establishment upon arrival, with the corresponding currency code (which can be different from both the request currency and Euro).
2. The <SiteID> tag, which contains the ID of the site that has originated the reservation. The ID is our internal ID for the relative site. If you think you need to make use of this value, please contact our technical department for more information on this.

3. The <ArrivalTime> tag, which contains the expected arrival time at the hotel, as indicated by the customer at the time of booking. It is in 24h format.
4. The <RoomBreakdown> tag, which will contain a breakdown of all the exact rooms that have been booked each day, one <RoomDay> tag per room/day. It also contains the room name in our system. **IMPORTANT:** this tag has been kept for the sake of backwards compatibility, and the tag <RoomRates>, which displays the same information, should be implemented instead.

11.2.3. Retrieving a list of reservations: The OTA_ReadRQ message

In this case, the request is for a list of reservations which have been created or cancelled within a certain timespan. More reservation search features will be added in future versions of the system.

The message looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_ReadRQ xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_ReadRQ.xsd" EchoToken="3229853" TimeStamp="2003-11-
10T10:34:00" Target="Production"
Version="1.006" SequenceNmbr="1" PrimaryLangID="en-us"
ReturnListIndicator="0">
<POS>
<Source>
<RequestorID ID="username" MessagePassword="password" Type="10"/>
</Source>
</POS>
     <ReadRequests>
        <HotelReadRequest HotelCode="1">
          <TPA_Extensions>
           <Interval  StartDate="2007-01-19T03:00:00-05:00"
EndDate="2007-01-19T05:07:30-05:00" Duration="-PT02H00M" />
          </TPA_Extensions>
        </HotelReadRequest>
     </ReadRequests>
</OTA_ReadRQ>
```

Please note that this message will only allow the retrieval of reservations made for one of the hotels which are administered by the login/password pair provided.

To identify the Hotel for which reservations are being searched, you must provide the hotel code in the HotelCode *attribute* of the <HotelReadRequest> tag.

The time interval in which to search must be provided in the <Interval> tag of the <TPA_Extensions> tag. Please note that the example contains three attributes, *StartDate, EndDate* and *Duration*. The real call will use just two of these attributes (if three are sent, the *Duration* attribute will be ignored).

11.2.3.1. The <Interval> tag

This tag is used here to provide a time interval. It consists of three possible attributes: at least two of them must be sent.

Attribute	*Notes*
StartDate	This is an ISO datetime, with format yyyy '-' mm '-' dd 'T' hh ':' mm ':' ss (zzzzzz)? For example, 2002-10-10T12:00:00-05:00 means noon on 10 October 2002, Central Daylight Savings Time as well as Eastern Standard Time in the U.S. If a time zone is omitted, the system defaults to UTC
EndDate	This is an ISO datetime, with format yyyy '-' mm '-' dd 'T' hh ':' mm ':' ss (zzzzzz)? For example, 2002-10-10T12:00:00-05:00 means noon on 10 October 2002, Central Daylight Savings Time as well as Eastern Standard Time in the U.S. If a time zone is omitted, the system defaults to UTC
Duration	This must be an ISO duration. Example values can be: P1Y2M3DT10H30M, which means 1 year, 2 months, 3 days, 10 hours and 30 minutes. -PT02H00M, which means minus 2 hours.

The possible uses of these attributes are the following:

1. Sending a *StartDate* and an *EndDate.* The *EndDate* must be later than the *StartDate*. Both can contain just a date indication, without time (in which case also the time zone will be ignored)
2. Sending a *StartDate* and a **positive** *Duration* value. In this case, the duration will be added to the start date to create an end date/time.
3. Sending an End*Date* and a **negative** *Duration* value. In this case, the duration will be subtracted to the end date to create a start date/time.

For example, if your system automatically retrieves reservations every three hours, you can send your **local** date/time (with time zone included) *as EndDate* and a **negative** *Duration* value of three hours (-PT03H00M).
All reservations which fall into the resulting interval will be returned.

11.2.4. Retrieving a list of reservations: The <OTA_ResRetrieveRS> message

The resulting response message will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<OTA_ResRetrieveRS xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_ResRetrieveRS.xsd" TimeStamp="2007-05-24T06:48:32-04:00"
Version="1.006" PrimaryLangID="en-us" EchoToken="3229853"
Target="Production">
  <Success/>
  <ReservationsList>
    <HotelReservation ResStatus="Cancelled" CreateDateTime="2007-
01-18T05:23:06-04:00" CancelDateTime="2007-01-19T04:39:14-04:00">
      <RoomStays>
        <RoomStay>
          <BasicPropertyInfo HotelCode="2530"/>
        </RoomStay>
      </RoomStays>
      <ResGlobalInfo>
        <TimeSpan Start="2009-01-19" End="2009-01-19"/>
        <HotelReservationIDs>
          <HotelReservationID ResID_Value="249695" Type="14"
ForGuest="true"/>
```

```
          </HotelReservationIDs>
        </ResGlobalInfo>
      </HotelReservation>
      <HotelReservation ResStatus="Reserved" CreateDateTime="2007-
01-17T10:54:51-04:00">
        <RoomStays>
          <RoomStay>
            <BasicPropertyInfo HotelCode="2530"/>
          </RoomStay>
        </RoomStays>
        <ResGlobalInfo>
          <TimeSpan Start="2009-01-17" End="2009-01-17"/>
          <HotelReservationIDs>
            <HotelReservationID ResID_Value="249358" Type="14"
ForGuest="true"/>
          </HotelReservationIDs>
        </ResGlobalInfo>
      </HotelReservation>
      <HotelReservation ResStatus="Reserved" CreateDateTime="2007-
01-18T04:30:31-04:00">
        <RoomStays>
          <RoomStay>
            <BasicPropertyInfo HotelCode="2530"/>
          </RoomStay>
        </RoomStays>
        <ResGlobalInfo>
          <TimeSpan Start="2009-01-18" End="2009-01-18"/>
          <HotelReservationIDs>
            <HotelReservationID ResID_Value="249680" Type="14"
ForGuest="true"/>
          </HotelReservationIDs>
        </ResGlobalInfo>
      </HotelReservation>
    </ReservationsList>
</OTA_ResRetrieveRS>
```

This message contains a <HotelReservation> tag for each of the reservations matching the request. Each tag contains the following attributes:

Attribute	***Notes***
ResStatus	Can be “Reserved” or “Cancelled”
CreateDateTime	An ISO datetime representing the creation date (the date and time the reservation was made)
CancelDateTime	An ISO datetime representing the last the date and time the reservation was cancelled, if applicable

The <HotelReservation> tag contains two other tag:

- the <RoomStays> tag, which contains a <RoomStay> tag containing a <BasicPropertyInfo> tag, which has a *HotelCode* attribute with a code corresponding to the property the reservation was made for.
- the <ResGlobalInfo> tag, which contains (within some other tags) the start and end date of the reservation and the reservation ID.

Further details about the reservation can be collected using the same request message, but passing only the resevation ID to get details about a single reservation (see above).

Please note that in the event where **there are no reservations in the time period requested**, our system will reply with a single <Success> tag, and no <ReservationsList>. Our system sends <Success> because your call is correct, but since there are no reservations, it does not send any data.

11.3. The OTA_HotelRateAmountNotifRQ / OTA_HotelRateAmountNotifRS

This message couple is used to update availability on our system.

11.3.1. The OTA_HotelRateAmountNotifRQ message

This message looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelRateAmountNotifRQ
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelRateAmountNotifRQ.xsd"
TimeStamp="2005-02-02T12:00:00" Target="Production"
Version="2.002" PrimaryLangID="en">
<POS>
<Source>
<RequestorID ID="login" MessagePassword="password" Type="10"/>
</Source>
</POS>
     <RateAmountMessages HotelCode="1">
        <RateAmountMessage>
            <StatusApplicationControl Start="2007-08-10"
End="2007-08-15" InvTypeCode="2_12" />
            <Rates>
                <Rate NumberOfUnits="16">
                    <BaseByGuestAmts>
                       <BaseByGuestAmt AmountBeforeTax="19" />
                    </BaseByGuestAmts>
                </Rate>
                <Rate Sat="true">
                 <BaseByGuestAmts>
                       <BaseByGuestAmt AmountBeforeTax="21" />
                    </BaseByGuestAmts>
                </Rate>
            </Rates>
        </RateAmountMessage>
         <RateAmountMessage>
            <StatusApplicationControl Start="2007-08-16"
End="2007-08-16"  InvTypeCode="1_23" />
            <Rates>
                <Rate NumberOfUnits="16">
                    <BaseByGuestAmts>
                       <BaseByGuestAmt AmountBeforeTax="19" />
                    </BaseByGuestAmts>
                </Rate>
            </Rates>
        </RateAmountMessage>
        <RateAmountMessage>
            <StatusApplicationControl Start="2007-08-17"
End="2007-08-17" InvTypeCode="1_4" />
```

```
                <Rates>
                    <Rate NumberOfUnits="0">
                        <BaseByGuestAmts>
                           <BaseByGuestAmt AmountBeforeTax="0" />
                        </BaseByGuestAmts>
                    </Rate>
                </Rates>
            </RateAmountMessage>
         </RateAmountMessages>
      </OTA_HotelRateAmountNotifRQ>
```

This message is composed by a <RateAmountMessages> tag, which contains *a HotelCode* attribute, which is used to send the ID of the hotel for which availability must be updated. It must also contain a series of <RateAmountMessage> tags. Each of the <RateAmountMessage> represents an indication to insert/delete availability for a certain room type in a certain time interval.

The **time interval** is passed as two dates, contained in the *Start* and *End* attributes of a <StatusApplicationControl> tag, contained in the <RateAmountMessage> tag. The dates must be in the future, the end date must be the same or later than the start date. To insert availability for a single day, you send the same date with both attributes. The *InvTypeCode* attribute of the <StatusApplicationControl> indicates the **category of rooms** the request applies to. This code is Hostelsclub's internal code, and appears in the **property backoffice** in the list of rooms, in the table field called "Room type code".

IMPORTANT: This is now the **Full Code** field which you find in the room_type_codes table of our documentation. (for ex. 1_2 for double bed private with WC) In previous versions of our web service, the system required one to send only the second part of the code (ex. 2 for double bed private with WC). The new system is backward-compatible not to break existing implementation, but we strongly encourage you to use the full code.

IMPORTANT – 2: While this code could previously be obtained from the spreadsheet containing room codes, this system is now obsolete (although all room codes published there are still valid), and the correct code should be obtained through the OTA_HotelDescriptiveInfoRS call. This is because hostels have many different room types and we generate new room codes dynamically whenever they're required.

PLEASE NOTE: for the system to work, you **must insert the appropriate number of rooms** of the desired type using your back office. If no rooms of the type sent are present in our system, you will receive an error.

The <RateAmountMessage> must contain another tag, called <Rates>. This must contain at least one <Rate> tag, and a maximum of two.

The first <Rate> tag **must** contain a *NumberOfUnits* attribute, indicating the number of units you want to insert for the given interval:

- 0 (zero) means remove availability for this room type for the period
- a positive number indicates the number of rooms (for private rooms) or the number of beds (for shared rooms) you want inserted

The <Rate> must contain a <BaseByGuestAmts> tag, which in turn must contain a <BaseByGuestAmt> tag, with an attribute *AmountBeforeTax* set to the price you want to set for that period. The price is **per room** for private rooms, and **per bed** for shared rooms. The price must be in the property's default operating currency in our system. Prices can be integers, or can contain one or two decimals separated with a dot (.). Examples: 10, 10.1, 10.52

Please note that, for the sake of OTA compatibility, the <BaseByGuestAmts> must be sent even if you're removing availability. If either the *NumberOfUnits* or the *AmountBeforeTax* attributes are set to 0, our system will remove availability. The best practice is to send both a number and a price set to 0.

If you're sending availability for more than one day at a time, you can also send a second <Rate> tag with the attribute *Sat* set to “true”. The price contained in that tag will be used as a weekend price for the period sent. If this tag is not sent, the weekend price inserted will be the same as the week price. The weekend price policy (whether a weekend price applies to Fri, Sat and Sun, or to Fri and Sat, or to Sat and Sun depends on your standard setup in our backoffice. This setup can be changed by contacting our staff).

The example above contains three <RateAmountMessage> tags:

The first:

```
           <RateAmountMessage>
                 <StatusApplicationControl Start="2007-08-10"
      End="2007-08-15" InvTypeCode="2_12" />
                 <Rates>
                     <Rate NumberOfUnits="16">
                         <BaseByGuestAmts>
                            <BaseByGuestAmt AmountBeforeTax="19" />
                         </BaseByGuestAmts>
                     </Rate>
                     <Rate Sat="true">
                      <BaseByGuestAmts>
                            <BaseByGuestAmt AmountBeforeTax="21" />
                         </BaseByGuestAmts>
                     </Rate>
                 </Rates>
             </RateAmountMessage>
```

inserts availability from the 10th of August, 2007 to the 15th of August. It inserts 16 beds in room code type 2_12 (8 beds mixed shared), at price 19 for the weekdays and 21 for the weekend.

The second:

```
           <RateAmountMessage>
                 <StatusApplicationControl Start="2007-08-16"
      End="2007-08-16"  InvTypeCode="1_23" />
                 <Rates>
                     <Rate NumberOfUnits="16">
                         <BaseByGuestAmts>
                            <BaseByGuestAmt AmountBeforeTax="19" />
                         </BaseByGuestAmts>
                     </Rate>
                 </Rates>
             </RateAmountMessage>
```

Inserts 16 rooms of the type 1_23 (4 beds private) at a price of 19 for the night of the 16th of August, 2007. Since this update sends price and availability for one day only, there's no need to send a weekend price. If the date sent is part of the property's weekend setting, the price will still be set to 19.

The third:

```
           <RateAmountMessage>
                 <StatusApplicationControl Start="2007-08-17"
      End="2007-08-17" InvTypeCode="1_4" />
                 <Rates>
```

```
                    <Rate NumberOfUnits="0">
                        <BaseByGuestAmts>
                           <BaseByGuestAmt AmountBeforeTax="0" />
                        </BaseByGuestAmts>
                    </Rate>
                </Rates>
            </RateAmountMessage>
         </RateAmountMessages>
```
removes availability for the night of the 17th of August, 2007 for all rooms of code 1_4 (2 beds private).

You can send a maximum of 100 <RateAmountMessage> tags with each request. We suggest to send fewer when you're inserting a big number of beds/rooms, for performance reasons. Any tag exceeding the limit will be ignored and will issue a warning.

There is another limit: each request you send to our server cannot set availability for a total period greater than 6 months. If you need to insert availability for a longer period, split your request in more separate messages. If the total period for which you're inserting availability exceeds 6 months, our system will issue an error and will not insert any availability.

PLEASE NOTE: for performance reasons, our system can sometimes insert a little less availability than requested on days where a lot of the rooms/beds of the type indicated have already been booked. Obviously, our system will never insert more availability than requested.

11.3.2. The OTA_HotelRateAmountNotifRS message

The standard response to this message is quite straightforward:

```
<OTA_HotelRateAmountNotifRS
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/2006A/OTA_HotelRateA
mountNotifRS.xsd" TimeStamp="2007-05-25T04:39:17-04:00"
Version="2.002" PrimaryLangID="en" EchoToken=""
Target="Production">
     <Success/>
</OTA_HotelRateAmountNotifRS>
```
If there are errors in any of the dates or if the room types sent are invalid (they do not exist in our database), or your calls do not respect the limitations explained above, no availability will be inserted at all. In this case the system will issue an error.

If you send a room type that is valid, but no rooms of that type have been inserted for that property, the system will issue a warning and will try to insert all other valid availability in the messages.

If you receive a response which only contains a <Success/> tag, this means your availability has been successfully updated. If there are any warnings, you should check what they are to avoid reproducing errors in future calls.

11.4. The OTA_HotelAvailRQ / OTA_HotelAvailRS messages

This message couple is used to read current availability for a property on our system.

11.4.1. The OTA_HotelAvailRQ message

This is a sample message:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelAvailRQ xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelAvailRQ.xsd" Version="1.006" PrimaryLangID="it"
Target="Test">
<POS>
<Source>
<RequestorID ID="username" MessagePassword="password" Type="10" />
</Source>
</POS>
<AvailRequestSegments>
<AvailRequestSegment>
<HotelSearchCriteria>
<Criterion>
<StayDateRange Start="2008-02-26" End="2008-02-27"/>
<HotelRef HotelCode="1"/>
</Criterion>
</HotelSearchCriteria>
</AvailRequestSegment>
</AvailRequestSegments>
</OTA_HotelAvailRQ>
```

The root element is an <AvailRequestSegments> tag, containing an <AvailRequestSegment> tag, a <HotelSearchCriteria> tag and a <Criterion> tag. We will process only the first <Criterion> tag and ignore the others, if there are any.

You must identify what property you want information for by inserting a <HotelRef> tag and its *HotelCode* attribute. The Hotel Code must be our internal code.

We will return only the details about the rooms which are available, in the property's default currency.

Our system will return all rooms available in the interval indicated by the StayDateRange. Please note that our system currently refuses requests for a period longer than 31 days, and we reserve the right to make the accepted period shorter in the future, depending on usage and load caused by this message's requests.

11.4.1.1. The <StayDateRange> tag

This tag is used to indicate the period in which you want to perform a search.

It must contain a *Start* attribute with the start date of the search in ISO format, which is yyyy-mm-ddThh:mmZ - we will only take into consideration the date part of the data, so you can omit everything from the T onwards, although our system will parse the full date format correctly.

The **End** of the period can be indicated in two ways: either by sending an end (departure /check-out date) date in the *End* attribute of the <StayDateRange> tag, or by sending the number of nights of the interval in the *Duration* attribute of the <StayDateRange> tag. According to the OTA Standards, the duration must be expressed as follows: P([0-9]{1,3})D, which means that for a duration of 3 nights you must send P3D (you can alternatively use P3N, which is a non-iso notation, to express 3 nights).

If you send the *End* attribute, remember that our system considers that the **last day of the**

request period date, so a request from Feb 12th to Feb 16th will show availability of a room from the nights of the 12th to the night of the 16th (a total of 5 nights). Due to our internal parsing of the duration values, a duration of 4N/4D will result in an interval of 5 days to be covered in the response, hence we recommend to use the *End* attribute to avoid misinterpretation.

If you send both the *Duration* and the *End* attributes, the *End* attribute will have precedence over the *Duration* attribute. So if you send a tag that looks like this:

```
<StayDateRange Start="2007-02-12" End="2007-02-16"
Duration="P7N"/>
```

the system will consider it a search for 5 nights (from the 12th to the 16th included).

Important: currently, searches for property availability in our system are restricted to **a maximum of 31 nights per request**. Any search for a longer interval **will issue an error.**

11.4.2. The OTA_HotelAvailRS message

Here is a sample response message:

```
<OTA_HotelAvailRS xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelAvailRS.xsd" TimeStamp="2007-11-30T17:46:52-05:00"
Version="1.006" PrimaryLangID="it" EchoToken="" Target="Test">
  <Success/>
  <RoomStays>
    <RoomStay>
      <RoomTypes>
        <RoomType RoomTypeCode="1_1" NumberOfUnits="7"/>
      </RoomTypes>
      <RoomRates>
        <RoomRate>
          <Rates>
            <Rate>
              <Base AmountBeforeTax="67.00" CurrencyCode="EUR"/>
              <TimeSpan Start="2008-02-26" Duration="P1N"/>
            </Rate>
          </Rates>
        </RoomRate>
      </RoomRates>
      <BasicPropertyInfo HotelCode="1"/>
    </RoomStay>
    <RoomStay>
      <RoomTypes>
        <RoomType RoomTypeCode="1_1" NumberOfUnits="6"/>
      </RoomTypes>
      <RoomRates>
        <RoomRate>
          <Rates>
            <Rate>
              <Base AmountBeforeTax="69.00" CurrencyCode="EUR"/>
              <TimeSpan Start="2008-02-27" Duration="P1N"/>
            </Rate>
          </Rates>
        </RoomRate>
      </RoomRates>
```

```
        <BasicPropertyInfo HotelCode="1"/>
      </RoomStay>
      <RoomStay>
        <RoomTypes>
          <RoomType RoomTypeCode="2_12" NumberOfUnits="72"/>
        </RoomTypes>
        <RoomRates>
          <RoomRate>
            <Rates>
              <Rate>
                <Base AmountBeforeTax="11.00" CurrencyCode="EUR"/>
                <TimeSpan Start="2008-02-26" Duration="P1N"/>
              </Rate>
            </Rates>
          </RoomRate>
        </RoomRates>
        <BasicPropertyInfo HotelCode="1"/>
      </RoomStay>
      <RoomStay>
        <RoomTypes>
          <RoomType RoomTypeCode="2_12" NumberOfUnits="40"/>
        </RoomTypes>
        <RoomRates>
          <RoomRate>
            <Rates>
              <Rate>
                <Base AmountBeforeTax="13.00" CurrencyCode="EUR"/>
                <TimeSpan Start="2008-02-27" Duration="P1N"/>
              </Rate>
            </Rates>
          </RoomRate>
        </RoomRates>
        <BasicPropertyInfo HotelCode="1"/>
      </RoomStay>
    </RoomStays>
  </OTA_HotelAvailRS>
```

The response will contain one <RoomStay> node per room type per day per price. This means that if all rooms of a certain type have the same price (which is the only possibility if you're inserting availability using the current Web Service messages), you will get one node per room type per day. Each <RoomStay> node will contain:

- a <RoomTypes> tag, which will include one and only one <RoomType> tag with the actual room type and the number of units available for that day. The number of units indicates number of **rooms** for private rooms, number of **beds** for shared rooms (so 40 units for a shared 8-bed dormitory means that there are 40 beds available, regardless of whether they represent a total of 5 empty rooms or, for example, 5 beds free in a total of 8 rooms).
- a <RoomRates> tag, which will contain one and only one <Rate> tag (which is itself included in a few wrapper tags, as you can see above). The <Rate> tag will contain a <Base> tag which includes the price (in the *AmountBeforeTax* attribute) and the *CurrencyCode* associated with that price (which will always be the default currency code for that property). The <RoomRates> tag will also contain a <TimeSpan> tag, which indicates the day the node's availability is valid for. You can simply read the *Start* attribute's value. In this implementation of the message, the *Duration* attribute will always contain the "P1N" value to mean one night.
- There is a <BasicPropertyInfo> tag which shows the ID of the property. This is

repeated for every <RoomStay> tag, as per OTA specifications.

Please note that the sequence in which <RoomStay> tags will appear is per room type, and not per date. This means that if you're requesting availability for a property that has 1 private room and one shared room, and you're requesting availability for a two-day interval, you'll receive first all the data for the private room, day by day, then the data for the shared room, day by day. The sample response above shows this very clearly.

The same is valid if there are several private/shared room types in the property's availability. The system will output the availability for the first room type, day by day, then for the second room type, and so on.

11.5. The OTA_HotelDescriptiveInfoRQ / OTA_HotelDescriptiveInfoRS messages

This message couple is used to read current rooms inserted for a property on our system, regardless of availability. It is particularly useful to sync a PMS to the existing rooms in our system at the first set-up. It is recommended for properties to immediately insert all possible rooms types they might want to allocate to our system. However, if rooms or room types are added later, this message can be used to sync the PMS again to reflect new room types or room quantities. *IMPORTANT*: it is not currently possible to insert rooms or room type via PMS, it must be done using the property's backoffice. This is both for security and technical reasons.

11.5.1. The OTA_HotelAvailRQ message

This is a sample message:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelDescriptiveInfoRQ EchoToken="1" Target="Test"
Version="1.006"
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelDescriptiveInfoRQ.xsd"  PrimaryLangID="en">

<POS>
<Source>
<RequestorID ID="username" MessagePassword="password" Type="10" />
</Source>
</POS>
<HotelDescriptiveInfos>
      <HotelDescriptiveInfo HotelCode="1">
      <FacilityInfo SendGuestRooms="true"/>
      </HotelDescriptiveInfo>
</HotelDescriptiveInfos>
</OTA_HotelDescriptiveInfoRQ>
```

The root element is an <HotelDescriptiveInfos> tag, containing an <HotelDescriptiveInfo> tag. The hotel code must be specified as the *HotelCode* attribute of this tag. will process only the first <HotelDescriptiveInfo> tag and ignore the others, if there are any.

To receive a list of room types, add the <FacilityInfo> tag and set its *SendGuestRooms* attribute to *true*.

Our system will return all rooms inserted in our system regardless of whether they have availability or not.

11.5.2. The OTA_HotelDescriptiveInfoRS message

Here is a sample response message:

```
<?xml version="1.0" encoding="utf-8"?>
<OTA_HotelDescriptiveInfoRS
xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelDescriptiveInfoRS.xsd" TimeStamp="2010-07-06T06:34:14-
04:00" Version="1.006" PrimaryLangID="en" EchoToken="1"
Target="Test">
  <Success/>
  <HotelDescriptiveContents HotelCode="1" HotelCodeContext="HCL"
HotelName="test" HotelCityCode="2049">
    <HotelDescriptiveContent HotelCode="752"
HotelCodeContext="HCL" HotelName="test" HotelCityCode="2049"
CurrencyCode="EUR">
      <FacilityInfo>
        <GuestRooms>
          <GuestRoom Code="1_1" CodeContext="HCL" Quantity="1"
RoomTypeName="private" MaxOccupancy="1">
            <TypeRoom Name="1 bed/s private with bath"
RoomType="1_1" StandardNumBeds="1" RoomClassificationCode="37"/>
          </GuestRoom>
          <GuestRoom Code="1_5" CodeContext="HCL" Quantity="1"
RoomTypeName="private" MaxOccupancy="4">
            <TypeRoom Name="4 bed/s private with bath"
RoomType="1_5" StandardNumBeds="4" RoomClassificationCode="36"/>
          </GuestRoom>
          <GuestRoom Code="1_254" CodeContext="HCL" Quantity="13"
RoomTypeName="private" MaxOccupancy="3">
            <TypeRoom Name="deluxe 3 bed/s private with bath"
RoomType="1_254" StandardNumBeds="3" RoomClassificationCode="36"/>
          </GuestRoom>
          <GuestRoom Code="2_44" CodeContext="HCL" Quantity="2"
RoomTypeName="shared" MaxOccupancy="20" Gender="0">
            <TypeRoom Name="20 bed/s shared mixed with bath"
RoomType="2_44" StandardNumBeds="20" RoomClassificationCode="36"/>
          </GuestRoom>
        </GuestRooms>
      </FacilityInfo>
    </HotelDescriptiveContent>
  </HotelDescriptiveContents>
</OTA_HotelDescriptiveInfoRS>
```

The response will contain one <HotelDescriptiveContents> which will in turn contain one <HotelDescriptiveContent> tag, which will contain a <FacilityInfo> tag. The <FacilityInfo> tag will contain a <GuestRooms> tag, which will contain a <GuestRoom> node per room type. This means that all rooms of a certain type will be in the same node, and their quantity will be indicated by the *Quantity* attribute of the node.

Each <GuestRoom> tag will have the following attributes:

Attribute	***Notes***
Code	Internal code for Hostelsclub Room, see chapter *Hostelsclub Room type codes* below.
CodeContext	This signifies that the Code above is a Hostelsclub internal code, and will always have the value of HCL.
Quantity	This reflects the quantity of rooms inserted in our system.
RoomTypeName	This will have the value of “private” or “shared” in the language of the request, to indicate whether this room is a standard private room or if it is a shared or “dorm” room, which is typical of hostels and other budget accommodation types. Please note that these values are textual, and may change according to our internal translation policies. You should use this value to display the room category to humans, not base your code on it.
MaxOccupancy	This is the maximum number of people that can sleep in this room. A double or twin room will list 2, a triple 3, a 10-bed shared dorm will list 10 and so on. This does not take into account possible extra beds or “sell double room as single room” policies.
Gender	This is an optional attribute that will appear only on shared/dorm rooms. It will be populated with the following values: 0 for a mixed room (where both males and females can book) 1 for a male-only room 2 for a female-only room ***Important: this attribute is not part of the OTA standard. We were forced to add it because the OTA standard does not provide for specification of whether a room is female or male-only. If shared rooms are present in a property, this response message will NOT validate against OTA XSDs.***

Each <GuestRoom> tag will contain a <TypeRoom> tag, which will have the following attributes:

Attribute	***Notes***
RoomType	Same as *Code* above.
Name	This will have a full textual description of the room type (examples: “deluxe 3 bed/s private with bath”, “20 bed/s shared mixed with bath”) in the language of the request. Please note that these values are textual, and may change according to our internal translation policies. You should use this value to display the room category to humans, not base your code on it.
StandardNumBeds	Same as *MaxOccupancy* above.
RoomClassification Code	This is an OTA code, see chapter *OTA Room type codes* below.

12. The PUSH notification system for reservations and cancellations

It is now possible to require our system to send a notification to your system every time a reservation is made or cancelled by the user. Consequently, XML requests of type **OTA_ReadRQ** (which returns a list of reservations) can be made on-demand when the notification arrives, instead of every *n* minutes.

The notification can be configured from the property backoffice by logging in with an XML-enabled account.

The notification performs an HTTP POST request towards your chosen URL, where your system will have to process the message by reading a set of parameters.

The POST is of “multipart/form-data” type and contains the following parameters:

- **propertyID** contains the ID of the property/hostel that has received the reservation/cancellation
- **reservationID** contains the ID of the reservation that has been made/cancelled
- **action** contains the string “booked” for new reservations, or the string **“**canceled” if the reservation has been cancelled.

Your URL must be reachable vie Internet using the HTTP protocol and **must not require authentication**.

Since there is no guarantee that the notification will be performed 100% of the time (your URL might be temporarily unreachable from our sites) it is still advisable to keep performing **OTA_ReadRQ** calls on a regular basis, although less frequently than before (a few times a day, not hourly).

For safety reasons it is advisable to add access restrictions on your URL based on caller IP. Currently Hostelsclub makes notification from the following IP addresses: 209.18.68.71, 209.18.68.73

Here's a simple example of implementation of notification reception using a PHP script running on a web server at the hypotetical URL http://www.example.com/hostelsclub/get_notify.php):

```
<?php

$propertyID = $_POST['propertyID']; //$propertyID now contains the ID of the property which received the new booking

$reservationID = $_POST['reservationID']; //$ reservationID now contains the ID of the new reservation

$action = $_POST['action']; //$action now contains the string “booked” (or “canceled”)

...

?>
```

13. Appendix

This part of the document contain a series of tables with codes for various parts of the system. The longer tables have been put in a separate spreadsheet file for your convenience. The spreadsheet file is distributed with this document. If you haven't received it, please contact our staff to get a copy.

13.1. Error Types

Both <Error> and <Warning> tags have both a *Code* attribute, which refers to the specific error code, and a *Type* attribute, that indicates in what broad category the error falls in. We've had to add a code to use for errors that do not fall into any of the existing categories.

```
<Warning Code="193" Type="11"> This reservation cannot be
cancelled</Warning>
```

The OTA allows for the following error types (table EWT - Error Warning Type) :

Error Type	***Short OTA description***	***Full OTA Description***	***Notes***
1	Unknown	Indicates an unknown error.	Also used to retrieve “Too Many Connections” error (see *note(1)* below)
2	No implementation	Indicates that the target business system has no implementation for the intended request.	
3	Biz rule	Indicates that the XML message has passed a low-level validation check, but that the business rules for the request message were not met.	
4	Authentication	Indicates the message lacks adequate security credentials	
5	Authentication timeout	Indicates that the security credentials in the message have expired	
6	Authorization	Indicates the message lacks adequate security credentials	
7	Protocol violation	Indicates that a request was sent within a message exchange that does not align to the message	
8	Transaction model	Indicates that the target business system does not support the intended transaction-oriented operation	
9	Authentical model	Indicates the type of authentication requested is not recognized	

Error Type	***Short OTA description***	***Full OTA Description***	***Notes***
10	Required field missing	Indicates that an element or attribute that is required in by the schema (or required by agreement between trading partners) is missing from the message	
11	-	-	Other errors/warnings (especially warnings), which do not fall into any of the above categories.

note(1) - “Too Many Connections” error: this error is returned when the system has reached the maximum simultaneous requests. This error should be raised even if there is already a running request to the same target structure of the request that receive the error. We recommend to try to send the request again later.

13.2. Error Codes

The list of error codes is provided in a separate Spreadsheet file, for ease of export and usage. If you have not received it, please contact us.

It must be noted that we had to introduce an error code (999) for situations that were not covered by the OTA codes. They are all cases of business rules/structural xml issues.

13.3. Currency codes

The list of currency codes is provided in a separate Spreadsheet file, for ease of export and usage. If you have not received it, please contact us.

13.4. Hotel Category codes

13.4.1. Property Class Type

These are the categories we use from the Property Class Type (PCT) OTA codes. They are used for *PropertyClassCode* attributes.

Code	***Description***
3	Apartment
4	Bed and breakfast
6	Campground
16	Guest house limited service
19	Hostel
20	Hotel

13.4.2. Segment Category Code

These are the Segment Category Code (SEG) OTA codes used for *SegmentCategoryCode* attributes: these are not all the OTA attributes we support during request (we try to support all those which apply to Hotels), but they are the ones we return (and ones you can certainly know we will understand in your requests).

Code	***Description***
2	Budget
17	Midscale
14	Upscale
8	Luxury

When you make a request for properties falling in a certain segment, we will return properties which according to the data we have will match your criteria.

13.5. OTA Room type codes

These are used in some cases in output descriptions – it is a selection of codes from OTA GRI - Guest Room Info

Code	***Description***
10	double bedrooms
19	twin bedrooms
37	Single-bedded accommodations
36	Family/oversized accommodations

13.6. Hostelsclub Room type codes

Due to the fact that our system supports shared rooms, which have a far more variable nature than standard private rooms, we have a big number of room types in our system. More keep being added automatically as owners insert new room types (for ex. 27 bed female shared room with shared toilet).

Please note that all our room type codes look like this: 1_2, 2_12, etc. The first number, which is always either a 1 or a 2, indicates whether the room is a private room (1) or a shared room or dormitory (2). The part after the underscore is the actual code used to identify the other characteristics of the room. The first part, which identifies whether the room is private or shared, is added for your convenience so you know that a room is of one type or the other without having to look up the code in the code table.

The codes are provided in a separate Spreadsheet file, for ease of export and usage. If you have not received it, please contact us.

In the descriptions, all rooms "with WC" contain a private WC in the room. Others have shared toilet facilities. A "shared" room is a room where people can book one or more beds, and share the room with strangers (typical of hostels). Shared rooms can be mixed (with people of all sexes allowed), only for female and only for male occupants.

13.7. Languages

These are the languages supported by our system at the time of writing. More keep being added – in general, if a language is supported in our public site, it is always supported in the XML. We try to use two-letter codes which are ISO – compatible, but when this is not possible our XML will also recognize ISO codes. Please see the section above about language recognition for details about Chinese codes (zh-tw and zh-cn).

The codes will be the same as those used in the URL of our public site. So if a new language is added and you would like to know its code, check our site. You can infer the code from our URLs. For example, the home page of our site in French is http://www.hostelsclub.com/index-fr.html, where "fr" stands for French.

The codes are provided in a separate Spreadsheet file, for ease of export and usage. If you have not received it, please contact us.

13.8. Credit Card Codes

Some credit card codes are included in the OTA specifications, but we had to add some. If a credit card type is not in this list, it means it is not accepted by our system at the moment.

Code	***Card Name***
AX	Amex
DN	Diners
JC	JCB
MC	Mastercard
VI	Visa
VE	Visa Electron
AU	Aura

13.9. City and Country Codes

City and country codes can be found downloading XML files on our site.

The address of the files is

https://www.hostelspoint.com/xml_aff/cities_LANGUAGE_CODE_HERE.xml

For example, the English version of the file has the following address:

https://www.hostelspoint.com/xml_aff/cities_en.xml

These files are updated in real time.

Please note that we have some countries which are "duplicated" in our system.

The United Kingdom is also present as Great Britain (minus Northern Ireland), and in its component countries of Northern Ireland, Wales, England and Scotland. Use the ISO code GB for search in the whole of United Kingdom.

Holland is a duplicate of the Netherlands. This has been duplicated because in many European languages people will refer to the Netherlands using the (less correct) term "Holland".