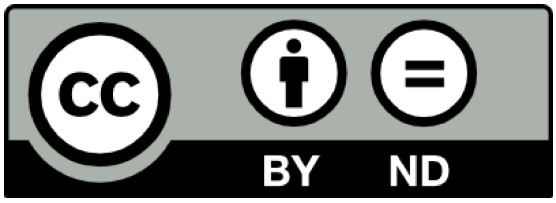
www.alpinebits.org

AlpineBits is an interface specification for exchanging data in the tourism sector, specially tailored for alpine tourism.

The interface is based on XML messages that validate against version 2015A of the OpenTravel Schema by the OpenTravel Alliance.



Disclaimer

This specification is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

If you find errors or you have proposals for enhancements, do not hesitate to contact us using the online discussion group: https://groups.google.com/forum/#!forum/alpinebits.

About the AlpineBits Alliance

The “AlpineBits Alliance” is a group of SME operating in the touristic sector working together to innovate and open the data exchange in the alpine tourism, and therefore to optimize the online presence, sales and marketing efforts of the hotels and other accommodations in the alpine territory and also worldwide.

AlpineBits Alliance
Via Bolzano 63/A
39057 Frangarto / Appiano s.s.d.v. (BZ) - ITALY
VAT Reg No: IT02797280217
www.alpinebits.org
info@alpinebits.org

AlpineBits Alliance Members

Altea Software Srl - www.altea.it
aries.creative KG - www.ariescreative.com
ASA OHG - www.asaon.com
Athesia Druck GmbH - www.sentres.com
Brandnamic GmbH - www.brandnamic.com
Dolomiti.it Srl - www.dolomiti.it
GardenaNet snc - www.gardena.net
HGV - www.hgv.it
Internet Consulting GmbH - www.inetcons.it
LTS - www.lts.it
Marketing Factory GmbH - www.marketingfactory.it
MM-One Group Srl - www.mm-one.com
PCS Hotelsoftware GmbH - www.pcs-phoenix.com
Peer GmbH - www.peer.biz
Rateboard GmbH - www.rateboard.info
Schneemenschen GmbH - www.schneemenschen.de
SiMedia GmbH - www.simedia.com
trick17.media OHG - www.trick17.it
Vioma GmbH - www.vioma.de

Author of this document:
Chris Mair - www.1006.org

Special thanks to IDM Südtirol - Alto Adige - www.idm-suedtirol.com

Document Change Log

Important note: make sure to have the latest version of this document! The latest version is available from http://www.alpinebits.org.

protocol version	doc. release date	description
2015-07b	2016-08-01	Status: ● official release Updates and additions: ● RatePlans: Section 4.5 has been rewritten to clarify details that the previous version just skipped over, including a detailed description of the price calculation algorithm ● RatePlans: new optional capability OTA_HotelRatePlanNotif_accept_RatePlanJoin to allow displaying alternative treatments for the same price list ● Schemas updates and validation: explicitly forbid some values (e.g. base prices of 0 EUR) and limit length of some attributes ● Minor fixes and clarifications
2015-07	2015-10-28	Status: ● official release Updates and additions: ● **OTA compatibility: version 2015A is now used** ● some rewrite of the text to add more clarity ● GuestRequests: a series to modifications and additions to make it more flexible especially for reservations ● GuestRequests: added refusals (warnings) ● GuestRequests: added booking modifications (ResStatus = 'Modify') ● RatePlans: changes to capabilities ● RatePlans: some clarifications and small additions ● RatePlans: partial rewrite and better explanation of Supplements ● RatePlans: added the explicit response messages ● Inventory: changes to capabilities ● Inventory: replaced by OTA_HotelDescriptiveContentNotifRQ ● Inventory: added possibility to send multimedia content ● Inventory: added additional descriptive content that may be sent separately from the basic data
2014-04	2014-12-23	Status: ● official release with minor errata fixed ● section 4.2.3.: the example was not correct about the fact that the presence of a SelectionCriteria Start requires the server to send the list of inquiries again, regardless whether the client has retrieved them before or not (example fixed and misleading sentence removed) ● section 4.1.1: the document did not mention that it is allowed to send a single empty AvailStatusMessage element in a CompleteSet request to reset the room availabilities in a given Hotel - the empty AvailStatusMessage is required for OTA compatibility (this special case is now explicitly mentioned) ● section 4.12: in the table at the end of the section the code for “Invalid hotel” was wrongly given as 61 instead of 361 (typo fixed) ● section 4.5.2: the document did not mention that it is allowed to send a single empty RatePlan element in a CompleteSet request to reset the rate plans in a given Hotel - the empty RatePlan is required for OTA compatibility (this special case is now explicitly mentioned)
2014-04	2014-10-15	Status: ● official release

		Updates and additions: ● this is a major overhaul of AlpineBits - see section B.3. for a list of breaking changes, updates and additions ● New section: **Inventory** - room category information ● New section: **RatePlans**
2013-04	2013-05-24	Status: ● official release Updates: ● Section 2 (HTTPS layer): added information regarding the new **X-AlpineBits-ClientID** and **X-AlpineBits-ClientProtocolVersion** fields in the HTTP header ● Section 3.2 (capabilities): added capability for FreeRoom **deltas** ● Section 4 (Intro): changed the text a bit to make it clearer that AlpineBits **does** indeed support booking requests and not only requests for quotes ● Section 4.1 (FreeRooms): added the possibility to send partial information (**deltas**); added **warning** response; much improved description of the response in general ● Section 4.2 (GuestRequests): slightly improved the description of the response in case of error ● Section 4.3 (Simple Packages): added **limitation** (just one Hotel per request); added **warning** response; much improved description of the response in general; clarified text to explicitly state that it is not allowed to mix package add and delete requests in a single message ● Appendix A (example code): this document should be language neutral so the code that used to be here has been removed with a message to check the official AlpineBits site (with the current release the code is still in the documentation kit, however) ● Appendix B (compatibility matrix): new appendix
2012-05b	2012-10-01	Status: ● official release Updates: ● OTA compatibility: the attribute **`Thu`** is renamed to **`Thur`** and the attribute **`Wed`** is renamed to **`Weds`** ● **SimplePackages:** the element **`Image`** is listed as **mandatory** in the table as it already was in the text and schema files ● **SimplePackages:** The element **`RateDescription`** is listed as non-repeatable in the table as it already was in the text and schema files
2012-05	2012-05-31	Status: ● official release Updates: ● major rewrite of the text ● FreeRooms: action OTA_HotelAvailNotif is no longer mandatory Additions: ● GuestRequests: reservation inquiries ● SimplePackages: package availability notifications
2011-11	2011-11-18	Minor alterations and release under Creative Commons Attribution-NoDerivs 3.0 Unported License
2011-10	2011-10-20	production release with minor alterations
2011-09	2011-09-08	first draft of redesigned version (using POST instead of SOAP)
2010-10	2010-10-20	second draft
2010-08	2010-08-01	first draft

Table of Contents

1. Introduction

This documents describes a standard for exchanging traveling and booking information, called **AlpineBits**.

AlpineBits builds upon established standards:

- client-server communication is done through stateless HTTPS (the client POSTs data to the server and gets a response) with basic access authentication[1] and
- the traveling and booking information are encoded in XML following version 2015A of the OpenTravel Schema [3, 4, 5] (from here on called OTA2015A) by the OpenTravel Alliance [2].

At the current version of the standard, the scope of AlpineBits covers exchanging the following types of information:

- room availability (FreeRooms),
- reservation inquiries (GuestRequests),
- package availability (SimplePackages),
- room category information (Inventory) and
- prices (RatePlans).

AlpineBits relies on its underlying transport protocol to take care of security issues. Hence **the use of HTTPS is mandatory**.

2. The HTTPS request and response structure

An AlpineBits compliant server exposes a **single** HTTPS URL. Clients send POST requests to that URL.

The POST request **must** transmit the access credentials using **basic access authentication**.

The HTTPS header of the POST request **must** contain an X-AlpineBits-ClientProtocolVersion field. The value of this field is the protocol version supported by the client (see the first column of the changelog table). A server that does not receive the field will simply conclude that the client speaks a protocol version preceding the version when this field was introduced (2013-04).

The HTTPS header of the POST request **may** contain an X-AlpineBits-ClientID field. The value of this field is an arbitrary string a server implementer **might** want to use to identify the client software version or installation ID.

The POST request **must** follow the multipart/form-data encoding scheme, as commonly used in the context of HTML forms for file uploads.

The POST request **may** be compressed using the gzip algorithm, in which case the HTTP request header *Content-Encoding* **must** be present and have "gzip" as value. A POST request compressed with gzip **must** be compressed by the client in its entirety (i.e. the whole message must be compressed, not the single parts of the multipart/form-data content). It is a client responsibility to check whether the server supports content compression, this is done by checking the value of the **HTTP response header** *X-AlpineBits-Server-Accept-Encoding* which is set to "gzip" by servers who support this feature. The so called "Housekeeping" actions **must not** be compressed.

The POST requests **must** have **at least** one parameter named **`action`.** Depending on the value of **`action`**, one additional parameter named **`request`** might be required.

Following is a capture of an example POST. In this example, the value of **`action`** is the string `OTA_HotelAvailNotif`, indicating the client wishes to perform a room availability notification. The value of **`request`** is an XML document (not fully shown).

```
POST / HTTP/1.1
Authorization: Basic Y2hyaXM6c2VjcmV0
Host: localhost
Accept: */*
X-AlpineBits-ClientProtocolVersion: 2015-07
X-AlpineBits-ClientID: sample-client.php v. 2015-07 1.0
Content-Length: 1989
Expect: 100-continue
Content-Type: multipart/form-data; boundary=----------------------------9d7042ecb251

------------------------------9d7042ecb251
Content-Disposition: form-data; name="action"

OTA_HotelAvailNotif:FreeRooms
------------------------------9d7042ecb251
Content-Disposition: form-data; name="request"

<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelAvailNotifRQ
        [...]
</OTA_HotelAvailNotifRQ>
------------------------------9d7042ecb251--
```

Note the Authorization field, with the username/password string (`chris:secret`) encoded in base64 (`Y2hyaXM6c2VjcmV0`) as defined by the basic access authentication standard [1].

Also note the X-AlpineBits-ClientProtocolVersion and X-AlpineBits-ClientID fields and the multipart content with the values of parameters **`action`** and **`request`.**

As a result of the POST request, the server answers with a response. Of course, the content of the response depends on the POSTed parameters, in particular the value of `action`. AlpineBits currently identifies two kinds of actions: the so-called housekeeping actions explained in section 3 and the actual data exchange actions explained in section 4.

The expected status code of the response is 200 (Ok), indicating that the server could authenticate the user (with or without being able to actually process any action).

In case of authentication failure (either an invalid or missing username/password or a value of X-AlpineBits-ClientID that is not acceptable to the server) the status code is 401 (Authorization Required) and the content is `ERROR:` followed by a colon (`:`) and an error message indicating the reason for the failure, such as no username/password was provided or the password expired, etc. Regarding the X-AlpineBits-ClientID, a server chooses to require the ID or to ignore the ID. If the server chooses to ignore the ID, it **must** do so silently, i.e. it **must** not return a 401 status because of the presence of the ID.

In case of internal server problem the status code is 500 (Internal Server Error).

A server that has **not** announced support for requests compressed with gzip **may** return the status code 501 (not implemented) in case it receives such requests.

An AlpineBits client **must** be able to handle these status codes. It **should** retry a request that has failed (error code 500 or timeout) and only escalate the failure after 2 retries with an appropriate delay.

2.1. Implementation tips and best practice

- For maximum compatibility across different implementations AlpineBits implementers are asked to handle POST requests using a supporting API in their language of choice. See appendix A for examples.

- All POSTs to an AlpineBits server are sent to a single URL. However, a server implementer might make more than one URL available for independent servers, such as servers with support for different versions:
    - https://server.example.com/alpinebits/2011-11
    - https://server.example.com/alpinebits/2012-05b
    - etc…

- Gzip compression can make request smaller by a factor of ten, therefore it was introduced in AlpineBits even though its usage in the POST requests is rare (far more than in the responses). According to the various RFCs compressing the requests is not forbidden, but there are no implementation recommendations; it was therefore chosen to use an approach that major web servers were already supporting at the time of this writing.

3. Housekeeping actions

These actions allow the client to query the server version and capabilities. An AlpineBits server **must** be able to handle **both**.

It is the **client's responsibility** to ensure the server can handle the actions it intends to perform. Clients are therefore invited to query the server's capabilities before performing the data exchange actions described in section 4.

The following table lists all available housekeeping actions.

usage	**mandatory**	**available since version**	**parameter** `action` (string)	**parameter** `request` (string)	**server response** (string)
a client queries the server version	YES	2011-11	`getVersion`	(not sent)	the server version
a client queries the server capabilities	YES	2011-11	`getCapabilities`	(not sent)	the server capabilities

3.1. Query the server version

A client performs this action to query the server version. The values of `action` is the string `getVersion`. The parameter `request` is not specified.

The response content is the string OK: followed by the protocol version supported by the server (see the first column of the changelog table, e.g. 2011-11 for the first release version of AlpineBits).

3.2. Query the server capabilities

A client performs this action to query the server capabilities. The values of `action` is the string `getCapabilities`. The parameter `request` is not specified. Please note that these requests **must** be sent in plain text i.e. not compressed using gzip, even when the server supports compressed requests.

The response is a string starting with `OK:` followed by a list of comma separated tokens each of which indicates a single capability of the server.

AlpineBits specifies the following capabilities:

- `action_getVersion`
  the server implements the `getVersion` action

- `action_getCapabilities`
  the server implements the `getCapabilities` action

- `action_OTA_HotelAvailNotif`
  the server implements handling room availability notifications (FreeRooms)

- `OTA_HotelAvailNotif_accept_rooms`
  for room availability notifications (FreeRooms), the server accepts booking limits for specific rooms

- `OTA_HotelAvailNotif_accept_categories`
  for room availability notifications (FreeRooms), the server accepts booking limits for categories of rooms

- `OTA_HotelAvailNotif_accept_deltas`
for room availability notifications (FreeRooms), the server accepts partial information (deltas)

- `action_OTA_Read`
the server implements handling quote requests, booking reservations and cancellations (GuestRequests)

- `action_OTA_HotelRatePlanNotif`
the server implements handling package availability notifications (SimplePackages)

- `action_OTA_HotelDescriptiveContentNotif_Inventory`
the server implements handling room category information (Inventory)

- `OTA_HotelDescriptiveContentNotif_Inventory_use_rooms`
for room category information (Inventory), the server needs information about specific rooms

- `OTA_HotelDescriptiveContentNotif_Inventory_occupancy_children`
for room category information (Inventory), the server supports applying children rebates also for children below the standard occupation

- `OTA_HotelDescriptiveContentNotif_Inventory_accept_basic`
for room category information (Inventory), the server accepts the main inventory information and basic descriptions from this client

- `OTA_HotelDescriptiveContentNotif_Inventory_accept_additional`
for room category information (Inventory), the server accepts the additional descriptions from this client

- `action_OTA_HotelRatePlanNotif_RatePlans`
the server implements handling prices (RatePlans)

- `OTA_HotelRatePlanNotif_accept_MinLOS`
for prices (RatePlans), the server accepts MinLOS restrictions in booking rules

- `OTA_HotelRatePlanNotif_accept_MaxLOS`
for prices (RatePlans), the server accepts MaxLOS restrictions in booking rules

- `OTA_HotelRatePlanNotif_accept_ArrivalDOW`
for prices (RatePlans), the server accepts arrival DOW restrictions in booking rules

- `OTA_HotelRatePlanNotif_accept_DepartureDOW`
for prices (RatePlans), the server accepts departure DOW restrictions in booking rules

- `OTA_HotelRatePlanNotif_accept_RatePlan_BookingRule`
for prices (RatePlans), the server accepts “generic” booking rules

- `OTA_HotelRatePlanNotif_accept_RatePlan_RoomType_BookingRule`
for prices (RatePlans), the server accepts “specific” booking rules for the given room types

- `OTA_HotelRatePlanNotif_accept_RatePlan_mixed_BookingRule`
for prices (RatePlans) and **within the same rate plan,** the server accepts both “specific” and “generic” booking rules. Both "generic" and "specific" rules capabilities **must** still be announced by the server.

- `OTA_HotelRatePlanNotif_accept_Supplements`
for prices (RatePlans), the server accepts supplements

- `OTA_HotelRatePlanNotif_accept_FreeNightsOffers`
for prices (RatePlans), the server accepts free nights offers

- `OTA_HotelRatePlanNotif_accept_FamilyOffers`
  for prices (RatePlans), the server accepts family offers

- `OTA_HotelRatePlanNotif_accept_overlay`
  for prices (RatePlans), the server accepts the rate plan notif type value `Overlay`

- `OTA_HotelRatePlanNotif_accept_RatePlanJoin`
  for prices (RatePlans), the server supports grouping RatePlans with different MealPlanCodes under a single price list

AlpineBits **requires** a server to support at least all mandatory housekeeping actions.

All other capabilities are optional. It is a **client's responsibility** to check for server capabilities before trying to use them. A server implementation is free to ignore information that requires a capability it doesn't declare. A server **must**, however, implement all capabilities it declares.

3.3. Unknown or missing actions

Upon receiving a request with an unknown or missing value for action, the server response is the string: `ERROR:unknown or missing action`.

3.4. Implementation tips and best practice

- Since the getCapabilities request is authenticated it's possible for a server to announce different capabilities to different users.

- OTA requires the root element of an XML document to have a version attribute. As regards AlpineBits, the value of this attribute is irrelevant.

4. Data exchange actions

These actions allow the actual exchange of data between client and server.

The parameter `request` is mandatory. Both, the client request and the server response are XML documents following OTA2015A as specified in the following table.

known as	usage	since version	parameter `action` (string)	parameter `request` (XML document)	server response (XML document)
FreeRooms	a client sends room availability notifications to a server	2011-11	`OTA_HotelAvailNotif:FreeRooms`	OTA_HotelAvailNotifRQ	OTA_HotelAvailNotifRS
GuestRequests	a client sends a request to receive requests for a quote or booking requests from the server	2012-05	`OTA_Read:GuestRequests`	OTA_ReadRQ	OTA_ResRetrieveRS
GuestRequests (ack's)	a client acknowledges the requests it has received	2014-04	`OTA_NotifReport:GuestRequests`	OTA_NotifReportRQ	OTA_NotifReportRS
SimplePackages	a client sends package availability notifications to a server	2012-05	`OTA_HotelRatePlanNotif:SimplePackages`	OTA_HotelRatePlanNotifRQ	OTA_HotelRatePlanNotifRS
Inventory	a client sends room category (inventory) information	2015-07	`OTA_HotelDescriptiveContentNotif:Inventory`	OTA_HotelDescriptiveContentNotifRQ	OTA_HotelDescriptiveContentNotifRS
RatePlans	a client sends information about rate plans with prices and booking rules	2014-04	`OTA_HotelRatePlanNotif:RatePlans`	OTA_HotelRatePlanNotifRQ	OTA_HotelRatePlanNotifRS

AlpineBits **requires** all XML documents to be encoded in **UTF-8**.

The business logic of an AlpineBits server, i.e. how the server processes and stores the information it receives is implementation-specific.

The format of the requests and responses is, however, exactly specified.

First of all the requests and responses **must** validate against the OTA2015A schema.

Since OTA is very flexible regarding mandatory / optional elements, AlpineBits adds extra requirements about exactly which elements and attributes are required in a request.

If these are not present, a server's business logic is bound to fail and will return a response indicating an error even though the request is valid OTA2015A.

OTA2015A, for instance allows OTA_HotelAvailNotifRQ requests that do not indicate the hotel, this might make perfect sense in some context, but an AlpineBits server will return an error if that information is missing from the request.

To aid developers, an AlpineBits XML Schema file and a set of Relax NG files are provided as an integral part of the specification in the AlpineBits documentation kit.

The Relax NG file set is stricter than the XML schema file. First, there is a RelaxNG file for each request and response type, so the nodes can be validated in a more specific way. Second, RelaxNG is intrinsically more powerful in expressing constraints that express how elements and attributes depend on each other.

Both, the XML Schema file and the Relax NG files, are stricter than OTA2015A in the sense that all documents that validate against AlpineBits will also validate against OTA2015A, but not vice versa.

The AlpineBits documentation kit also provides a sample file for each of the request and response documents.

The latest AlpineBits documentation kit for each protocol version is available from the official AlpineBits website.

4.1. FreeRooms: room availability notifications

When the value of the `action` parameter is `OTA_HotelAvailNotif:FreeRooms` the client intends to send room availability notifications to the server.

A server that supports this action **must** support at least one of two capabilities: `OTA_HotelAvailNotif_accept_rooms` or `OTA_HotelAvailNotif_accept_categories`. This way the server indicates whether it can handle the availability of rooms at the level of distinct rooms, at the level of categories of rooms or both.

4.1.1 Client request

The parameter `request` must contains an OTA_HotelAvailNotifRQ document.

Clients and servers typically wish to exchange only delta information about room availabilities in order to keep the total amount of data to be processed in check.

However, for simplicity let us first consider a request where the client transmits the complete availability information as might be the case for a first synchronization.

Consider the outer part of the example document:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelAvailNotifRQ
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelAvailNotifRQ.xsd"
        Version="1.002">

    <UniqueID Type="16" ID="1" Instance="CompleteSet"/>

    <AvailStatusMessages HotelCode="123" HotelName="Frangart Inn">7

        <!-- ... see below ... -->

    </AvailStatusMessages>

</OTA_HotelAvailNotifRQ>
```

samples/FreeRooms-OTA_HotelAvailNotifRQ.xml - outer part

An OTA_HotelAvailNotifRQ may contain just **one** `AvailStatusMessages` (note the plural) element, hence at most **one** hotel can be dealt with in a single request.

The **UniqueID** element with attribute **Instance** = `CompleteSet` indicates that this message contains the complete information. When receiving such a request, a server **must** remove all information about any availability it might have on record regarding the given hotel.

If the **UniqueID** element is missing, the message contains delta information. In that case the server updates only the information that is contained in the message without touching the other information that it has on record.

AlpineBits requires the attributes **HotelCode** **or** **HotelName** to be present (and match information in the server's database). The fictitious hotel in the example is the "Frangart Inn" with code "123". Specifying both — code and name — is redundant, but allowed, as long as both are consistent.

AlpineBits **requires** a match of **HotelCode**, **HotelName** to be **case sensitive**.

Second, consider the inner part of the example that contains **AvailStatusMessage** (note the singular) elements for three different rooms.

Let's start with the availabilities for room 101S.

```
<AvailStatusMessage BookingLimit="1" BookingLimitMessageType="SetLimit">

    <StatusApplicationControl Start="2010-08-01" End="2010-08-10"
                              InvTypeCode="double" InvCode="101S" />

</AvailStatusMessage>

<AvailStatusMessage BookingLimit="1" BookingLimitMessageType="SetLimit">

    <StatusApplicationControl Start="2010-08-21" End="2010-08-30"
                              InvTypeCode="double" InvCode="101S" />

</AvailStatusMessage>
```

samples/FreeRooms-OTA_HotelAvailNotifRQ.xml - inner part

The use of the **InvCode** attribute tells us we're dealing with a **specific** room (101S) that belongs to the room category given by the **InvTypeCode** (double).

Alternatively, using a **InvTypeCode** without a **InvCode** attribute would indicate that the availability refers to a category of rooms, not a specific room.

AlpineBits **requires** a match of **InvCode** or **InvTypeCode** to be **case sensitive**.

An AlpineBits server **must** be able to treat **at least** one case out of the two cases (specific rooms or categories). A client should perform the `getCapabilities` action to find out whether the server treats the room case (token `OTA_HotelAvailNotif_accept_rooms`), the category case (token `OTA_HotelAvailNotif_accept_categories`) or both.

Mixing rooms and categories in a single request is **not** allowed. An AlpineBits server **must** return an error if it receives such a mixed request.

The attribute **Start** and **End** indicate that room 101S is available from 2010-08-01 to 2010-08-10 and from 2010-08-21 to 2010-08-30.

Regarding the first interval, this means the earliest possible check-in is 2010-08-01 afternoon and latest possible check-out is 2010-08-11 morning (maximum stay is 10 nights).

Since there are no further restrictions, check-ins **after** 2010-08-01 and stays of **less** than 10 nights are allowed as well, provided the check-out is not later than 2010-08-11 morning.

Idem for the other block of 10 nights from 2010-08-21 to 2010-08-30 (latest check-out is 2010-08-31 morning).

Since a specific room is indicated here, the only meaningful value of **BookingLimit** is 0 or 1 (the same room can not be available more than once). In the category case, numbers larger than 1 would also be allowed.

BookingLimit numbers are always interpreted to be absolute numbers. Differential updates are not allowed.

Note that AlpineBits does **not allow** **AvailStatusMessage** elements with overlapping periods. This implies that the order of the **AvailStatusMessage** elements doesn't matter. It is a **client's responsibility** to avoid overlapping. An AlpineBits server's business logic **may** identify overlapping and return an error or **may** proceed in an implementation-specific way.

AlpineBits **requires** the **AvailStatusMessage** element to have attributes **BookingLimit** and **BookingLimitMessageType**. It also requires exactly one **StatusApplicationControl** element with attributes **Start**, **End**, **InvTypeCode** and (optional **InvCode**) for each **AvailStatusMessage** element. It **must** return an error if any of these are missing. There is however one exception: to completely reset all room availability information for a given Hotel a client might send a `CompleteSet` request with just one empty **AvailStatusMessage** element without any attributes. The presence of the empty **AvailStatusMessage** element is required for OTA validation.

Please note that previous versions of AlpineBits allowed some booking restrictions to be used in FreeRooms (length of stay and day of arrival). This possibility has been removed with version 2014-04 as these restrictions are better handled by RatePlans.

AlpineBits **recommends** that Implementers that use delta requests **should** send the full set of information periodically.

A server that supports delta requests **must** indicate so via the `OTA_HotelAvailNotif_accept_deltas` capability. As always, it is the **client's responsibility** to check whether the server supports deltas before trying to send them.

4.1.2. Server response

The server will send a response indicating the outcome of the request. The response is a OTA_HotelAvailNotifRS document. Four types of outcomes are possible: success, advisory, warning or error.

Success

The request was accepted and processed successfully. The client does **not** need to take any further action.

In this case, the OTA_HotelAvailNotifRS response contains nothing but a **single**, empty **Success** element:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelAvailNotifRS
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelAvailNotifRS.xsd"
        Version="1.001">

       <Success/>

</OTA_HotelAvailNotifRS>
```

samples/FreeRooms-OTA_HotelAvailNotifRS-success.xml

Advisory

The request was accepted and processed successfully. However, one or more non-fatal problems were detected and added to the server response. The client does **not** need to resend the request, but **must** notify the user or the client implementer regarding the advisory received.

In this case, the OTA_HotelAvailNotifRS response contains an empty **Success** element followed by **one or more Warning** elements with the attribute **Type** set to the fixed value 11, meaning “Advisory” according to the OTA list “Error Warning Type” (EWT).

Each **Warning** element should contain a human readable text as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelAvailNotifRS
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelAvailNotifRS.xsd"
        Version="1.001">

        <Success/>
        <Warnings>
            <Warning Type="11">
                last full data set received more than 48 hours ago
            </Warning>
        </Warnings>

</OTA_HotelAvailNotifRS>
```

samples/FreeRooms-OTA_HotelAvailNotifRS-advisory.xml

Warning

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the warning is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelAvailNotifRS response contains an empty **Success** element followed by **one or more Warning** elements with the attribute **Type** set to any value allowed by the OTA list “Error Warning Type” (EWT) **other than** 11 (“Advisory”).

Each **Warning** element should contain a human readable text as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelAvailNotifRS
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelAvailNotifRS.xsd"
        Version="1.001">

        <Success/>
        <Warnings>
            <Warning Type="3">
                dates are too far in the future for this server to process
            </Warning>
        </Warnings>

</OTA_HotelAvailNotifRS>
```

samples/FreeRooms-OTA_HotelAvailNotifRS-warning.xml

Error

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the error is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelAvailNotifRS response contains **one or more** `Error` elements with the attribute `Type` set to the fixed value `13`, meaning “Application error” according to the OTA list “Error Warning Type” (EWT) and the attribute `Code` set to any value present in the OTA list “Error Codes” (ERR).

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelAvailNotifRS
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelAvailNotifRS.xsd"
        Version="1.001">

    <Errors>
        <Error Type="13" Code="404">
            Invalid start/end date combination
        </Error>
    </Errors>

</OTA_HotelAvailNotifRS>
```

samples/FreeRooms-OTA_HotelAvailNotifRS-error.xml

To aid implementers pick somewhat consistent error codes, here is a non-exhaustive list of codes that might be typically encountered when processing FreeRooms requests, taken from the OTA list “Error Codes” (ERR):

Code	Text
361	Invalid hotel
392	Invalid hotel code
396	Invalid name
375	Hotel not active
135	End date is invalid
136	Start date is invalid
404	Invalid start/end date combination
131	Room/unit type invalid
69	Minimum stay criteria not fulfilled
70	Maximum stay criteria not fulfilled
362	Invalid number of nights

4.1.3. Implementation tips and best practice

- Note that in the 2011-11 version of AlpineBits the support of this action was mandatory for the server. This is no longer the case.
- Note that sending partial information (deltas) was added with AlpineBits 2013-04.
- For non-delta requests, since no time frame is explicitly transmitted by the client, a server is encouraged to delete and insert all the information stored in its backend, rather than updating it.
- Please note that the **`End`** date of an interval identifies the last day and night of the stay. Departure is the morning of the date after the **`End`** date.
- The OTA lists “Error Warning Type” (EWT) and “Error Codes” (ERR) come with the OTA2015A documentation package. The package can be downloaded from the OTA web site [3]. The file `OpenTravel_CodeList_2015_06_03.xlsm` contains all the lists.

4.2. GuestRequests: quote requests, booking reservations and cancellations

The typical use case for GuestRequests is a portal that collects quote **requests**, booking **reservations** or booking **cancellations** from potential customers and stores them until a client (typically the software used by a hotel) retrieves them.

In this case, the client sends a **first** request to obtain the information from the server about any requests, reservation or cancellations with the parameter **`action`** set to the value `OTA_Read:GuestRequests`.

The server then responds with the requested information.

Successively the client sends a **follow-up** request to acknowledge having received the information, with the parameter **`action`** set to the value `OTA_NotifReport:GuestRequests`.

4.2.1. First client request

The parameter **`action`** is set to the value `OTA_Read:GuestRequests`. and the parameter **`request`** must contain a OTA_ReadRQ document.

For the **mandatory** attributes **`HotelCode`** and **`HotelName`** the rules are the same as for room availability notifications (section 4.1.1).

The element **`SelectionCriteria`** with the **`Start`** attribute is **optional**.

When given, the server will send only inquiries generated after the **`Start`** timestamp, regardless whether the client has retrieved them before or not.

When omitted, the server will send all inquiries it has on record and that the client has not yet retrieved.

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_ReadRQ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns="http://www.opentravel.org/OTA/2003/05"
            xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_ReadRQ.xsd"
            Version="1.001">

    <ReadRequests>
       <HotelReadRequest HotelCode="123" HotelName="Frangart Inn">
           <SelectionCriteria Start="2012-03-21T15:00:00+01:00"></SelectionCriteria>
       </HotelReadRequest>
    </ReadRequests>

</OTA_ReadRQ>
```

samples/GuestRequests-OTA_ReadRQ.xml

4.2.2 Server response

The server response is a OTA_ResRetrieveRS document indicating the outcome of the request. Two types of outcome are possible: error or success.

Error

If the request **could not** be accepted or processed successfully the OTA_ResRetrieveRS response contains one or more **`Error`** elements, each with **`Type`** and **`Code`** attributes containing human readable text. The rules for creating error messages are identical to the FreemRooms error case (please see section 4.1.2).

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_ResRetrieveRS
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_ResRetrieveRS.xsd"
    Version="7.000">

    <Errors>
        <Error Type="13" Code="392">
            Invalid hotel code
        </Error>
    </Errors>

</OTA_ResRetrieveRS>
```

samples/GuestRequests-OTA_ResRetrieveRS-error.xml

Success

In case of success, the OTA_ResRetrieveRS response will contain a **single**, empty **`Success`** element followed by a **`ReservationsList`** element with **zero or more** **`HotelReservation`** elements containing the requested information (zero elements indicate the server has no information for the client at this point).

Each **`HotelReservation`** **must** have the attributes **`CreateDateTime`** (the timestamp the information was collected by the portal). Furthermore the **`ResStatus`** attribute **must** be set. AlpineBits expects it to be one of the following four:

- `Requested` - this is a **request** for a quote
- `Reserved` - this is a booking **reservation**
- `Modify` - this is a booking **modification**
- `Cancelled` - this is a booking **cancellation**

The following example is a **reservation** (thus, **`ResStatus`** is `Reserved`). The documentation kit also has an example of a quote **request**. A **cancellation** is discussed later.

First, consider the outer part of the OTA_ResRetrieveRS document:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_ResRetrieveRS
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_ResRetrieveRS.xsd"
        Version="7.000">

    <Success/>

    <ReservationsList>

        <HotelReservation CreateDateTime="2012-03-21T15:00:00+01:00"
                          ResStatus="Reserved">

             <!-- Type 14 -> Reservation -->
            <UniqueID Type="14" ID="6b34fe24ac2ff810"/>

            <RoomStays>      <!-- stays, see below -->                    </RoomStays>

            <ResGuests>      <!-- customer data, see below -->            </ResGuests>

            <ResGlobalInfo>  <!-- additional booking data, see below -->  </ResGlobalInfo>

       </HotelReservation>

    </ReservationsList>

</OTA_ResRetrieveRS>
```

samples/GuestRequests-OTA_ResRetrieveRS-reservation.xml - outer part

Each `HotelReservation` contains a **mandatory** `UniqueID` element that the client can use to recognize information it has already processed.

The `UniqueID` element **must** have the `Type` attribute set according the OTA Unique Id Type list (UIT). The value must be consistent with the `ResStatus` attribute of the surrounding `HotelReservation` element:

- For `ResStatus` = `Requested`, the `Type` must be 14 (Reservation)
- For `ResStatus` = `Reserved`, the `Type` must be 14 (Reservation)
- For `ResStatus` = `Modify`, the `Type` must be 14 (Reservation)
- For `ResStatus` = `Cancelled`, the `Type` must be 15 (Cancellation)

The attribute `ID` is a free text field suitable for uniquely identifying the `HotelReservation`.

The actual data is then split into three parts: each `HotelReservation` contains the elements: `RoomStays`, `ResGuests` and `ResGlobalInfo` (all **mandatory**) discussed in the following paragraphs.

First part: `RoomStays`.

The `RoomStays` element contains **one or more** `RoomStay` elements, each indicating a desired stay.

```
<RoomStays>

    <RoomStay>

        <RoomTypes>
            <RoomType RoomTypeCode="bigsuite" RoomClassificationCode="42"/>
        </RoomTypes>

        <RatePlans>
            <RatePlan RatePlanCode="123456-xyz">
                <!-- Code 1 -> All inclusive -->
                <MealsIncluded MealPlanIndicator="true" MealPlanCodes="1"/>
            </RatePlan>
        </RatePlans>

        <!-- 2 adults + 1 child + 1 child = 4 guests -->
        <GuestCounts>
            <!-- 2 adults -->
            <GuestCount Count="2"/>
            <!-- 1 child -->
            <GuestCount Count="1" Age="9"/>
            <!-- 1 child -->
            <GuestCount Count="1" Age="3"/>
        </GuestCounts>

        <TimeSpan Start="2012-01-01" End="2012-01-12"/>

        <Guarantee>
            <GuaranteesAccepted>
                <GuaranteeAccepted>
                    <PaymentCard CardCode="VI"
                                 ExpireDate="1216">
                        <CardHolderName>Otto Mustermann</CardHolderName>
                        <CardNumber>
                            <PlainText>4444333322221111</PlainText>
                        </CardNumber>
                    </PaymentCard>
                </GuaranteeAccepted>
            </GuaranteesAccepted>
        </Guarantee>

        <Total AmountAfterTax="299" CurrencyCode="EUR"/>

    </RoomStay>

</RoomStays>
```

`samples/GuestRequests-OTA_ResRetrieveRS-reservation.xml` - `RoomStay` element

Each `RoomStay` element contains:

- **one** `RoomType` element (**mandatory**): see below for explanation.
- **one** `RatePlan` element with a `RatePlanCode` attribute (**mandatory** for **reservations**, **optional** for quote **requests**) and **one** `MealsIncluded` element (**mandatory** for **reservations**, **optional** for quote **requests**): the **MealsIncluded** element must contain the **MealPlanCodes** attribute (values see below) and must have the **MealPlanIndicator** attribute set to true.
- **one** `GuestCounts` element (**mandatory**) indicating the number of all adults (identified by **one** `GuestCount` element **with no** `Age` attribute given) and the number of all children (identified by **zero or more** `GuestCount` element **with** `Age` attribute given); of course **all** guests must be listed
- **one** `TimeSpan` element (**mandatory**): see below for explanation

- **one** `PaymentCard` element (only for **reservations**, **optional);** the element **must** have attributes **CardCode** (the card issuer, two uppercase letters, such as “VI” for Visa) and **ExpireDate** (four digits) and **must** have the sub-elements **CardHolderName** and **CardNumber/PlainText** (the cardnumber consisting of digits, either the complete number or the last 4 digits are given)
- **one** `Total` element (**mandatory** for **reservations**, **optional** for quote **requests**) containing the cost after taxes the portal has displayed to the customer, both attributes `AmountAfterTax` and `CurrencyCode` are required.

For reservations, the `RoomType` element is **mandatory.** Reservations **must** refer to a specific room category (specified by `RoomTypeCode`). Quote requests **should** refer a specific room category whenever possible (specified by the **optional** attribute `RoomTypeCode`) but **may** refer to a generic GRI (specified by the **optional** attribute `RoomClassificationCode`) or **may** be completely open if the `RoomType` element is present without attributes.

The `RoomTypeCode` **must** be identical to the `InvTypeCode` attribute used for room categories (see section 4.1.1).

The `RoomClassificationCode` follows the OTA list “Guest Room Info” (GRI). It is used to loosely classify the kind of guest room (42 means just “Room”, 13 means “Apartments”, etc.) wished by the guest.

Regarding the **MealPlanCodes** attribute, AlpineBits **does not** use the single Breakfast/Lunch/Dinner booleans, but relies on the **MealPlanCodes** attribute only. The following codes (a subset of the full OTA list) are allowed:

- 1 - all inclusive
- 3 - bed and breakfast
- 10 - full board
- 12 - half board
- 14 - room only

The `TimeSpan` element deserves a more detailed explanation.

For **reservations** (`ResStatus` is `Reserved` or `Modify`), the arrival and departure date **must** be given with the `Start` and `End` attributes of the `TimeSpan` element. **No** other attributes and **no** subelements **must** be present in `TimeSpan`.

For quote **requests** (`ResStatus` is `Requested`) the timespan can be given in the same way (i.e. using the `Start` and `End` attributes) or it **may** be given as a window. In this case the `TimeSpan` element must have the `Duration` attribute (encoded in ISO 8601) and the `StartDateWindow` sub element with attributes `EarliestDate` and `LatestDate` which **must** be greater than `EarliestDate` indicating a range of possible start dates.

`Duration` are given in nights, the form is thus always PxN where x is a number.

If multiple `RoomStay` elements are given, all the `TimeSpan` elements must have exactly the same values.

As a special case, however, **only** for quote **requests** (`ResStatus` is `Requested`), it is possible to add **at most one optional** `RoomStay` element that contains **only** the `TimeSpan` element. In this case, this last `TimeSpan` is allowed to have different values (as a matter of fact, they must be different) and it ought to be interpreted by the client as an alternative period with regard to the preceding `RoomStay` element(s).

Second part: `ResGuests`.

Nested inside the `ResGuests` element is **exactly one** `Customer` element, providing the data of the primary guest.

The `Gender` attribute can be `Male, Female` or `Unknown`. The `BirthDate` attribute follows ISO 8601. The `Language` follows ISO 639-1 (two-letter lowercase language abbreviation). It identifies the language to be used when contacting the customer.

These three attributes are all **optional**. However, it is recommended that at least gender and language be specified (so the customer can be addressed properly).

```
<ResGuests>
    <ResGuest>
        <Profiles>
            <ProfileInfo>
                <Profile>

                    <Customer Gender="Male" BirthDate="1980-01-01" Language="de">

                        <PersonName>
                            <NamePrefix>Herr</NamePrefix>
                            <GivenName>Otto</GivenName>
                            <Surname>Mustermann</Surname>
                            <NameTitle>Dr</NameTitle>
                        </PersonName>

                        <!-- Code 1 -> Voice -->
                        <Telephone PhoneTechType="1" PhoneNumber="+4934567891"/>
                        <!-- Code 3 -> Fax -->
                        <Telephone PhoneTechType="3" PhoneNumber="+4934567892"/>
                        <!-- Code 5 -> Mobile -->
                        <Telephone PhoneTechType="5" PhoneNumber="+4934567893"/>

                        <Email Remark="newsletter:yes">otto.mustermann@example.com</Email>

                        <Address Remark="catalog:yes">

                            <AddressLine>Musterstraße 1</AddressLine>
                            <CityName>Musterstadt</CityName>
                            <PostalCode>1234</PostalCode>
                            <CountryName Code="DE"/>

                        </Address>

                    </Customer>

                </Profile>
            </ProfileInfo>
        </Profiles>
    </ResGuest>
</ResGuests>
```

`samples/GuestRequests-OTA_ResRetrieveRS-reservation.xml` - `Customer` element

The `Customer` element contains:

- **one** `PersonName` element with `NamePrefix`, `GivenName` (**mandatory**), `Surname` (**mandatory**) and `NameTitle`
- **zero or more** `Telephone` elements with the optional `PhoneTechType` attribute: it indicates the phone technology (1 → voice, 3 → fax, 5 → mobile per OTA)
- **one optional** `Email` element
- **one optional** `Address` element with the (all optional) elements `AddressLine`, `CityName`, `PostalCode` and `CountryName` with `Code` attribute; the `Code` attribute follows ISO 3166-1 alpha-2 (two-letter uppercase country codes)

Note that most elements and attributes under the **ResGuests** element are optional in the AlpineBits schema. It is, however, expected that as much contact information as possible is given.

The **Email** element **may** contain the attribute **Remark** having either values `newsletter:yes` or `newsletter:no` indicating whether the customer does or does not wish to receive an email newsletter. A missing **Remark** indicates the information is not known - for new customers this should be treated the same way as if a `newsletter:no` was given (do not send a newsletter), while existing customer records should not be updated.

Analogously, the **Address** element **may** contain the attribute **Remark** having either values `catalog:yes` or `catalog:no` indicating whether the customer does or does not wish to receive print ads by mail. A missing **Remark** indicates the information is not known - for new customers this should be treated the same way as if a `catalog:no` was given (do not send a mail), while existing customer records should not be updated.

Third part: ResGlobalInfo.

```
<ResGlobalInfo>

    <Comments>

        <Comment Name="included services">
            <ListItem ListItem="1" Language="de">Parkplatz</ListItem>
            <ListItem ListItem="2" Language="de">Schwimmbad</ListItem>
            <ListItem ListItem="3" Language="de">Skipass</ListItem>
        </Comment>

        <Comment Name="customer comment">
            <Text>
                Sind Hunde erlaubt?

                Mfg.
                Otto Mustermann.
            </Text>
        </Comment>

    </Comments>

    <CancelPenalties>
        <CancelPenalty>
            <PenaltyDescription>
                <Text>
                Cancellation is handled by hotel.
                Penalty is 50%, if canceled within 3 days before show, 100% otherwise.
                </Text>
            </PenaltyDescription>
        </CancelPenalty>
    </CancelPenalties>

    <HotelReservationIDs>
        <!-- ResID_Type 13 -> Internet Broker -->
        <HotelReservationID ResID_Type="13"
                            ResID_Value="Slogan"
                            ResID_Source="www.example.com"
                            ResID_SourceContext="top banner" />
    </HotelReservationIDs>

    <Profiles>
        <ProfileInfo>
            <!-- ProfileType 4 -> Travel Agent -->
            <Profile ProfileType="4">
                <CompanyInfo>
                    <CompanyName Code="123" CodeContext="ABC">
                        ACME Travel Agency
                    </CompanyName>
                    <AddressInfo>
                        <AddressLine>Musterstraße 1</AddressLine>
```

```
                        <CityName>Flaneid</CityName>
                        <PostalCode>12345</PostalCode>
                        <CountryName Code="IT"/>
                    </AddressInfo>
                    <!-- Code 1 -> Voice -->
                    <TelephoneInfo PhoneTechType="1" PhoneNumber="+391234567890"/>
                    <Email>info@example.com</Email>
                </CompanyInfo>
            </Profile>
        </ProfileInfo>
    </Profiles>

    <!-- this is needed for OTA-2015A compatibility -->
    <BasicPropertyInfo/>


</ResGlobalInfo>
```

samples/GuestRequests-OTA_ResRetrieveRS-reservation.xml - **ResGlobalInfo** element

The **ResGlobalInfo** element contains:

- **one** **Comment** element (**optional**) with attribute **Name** set to `included services` containing the included services given **as free text fields** using **ListItem** elements (see below). In most cases the AlpineBits client software will just display this to a human hotel employee with no further processing
- **one** **Comment** element (**optional**) with attribute **Name** set to `customer comment` containing a single **Text** element freely filled out by the customer and fed through unchecked by the portal
- **only allowed for reservations, one** **PenaltyDescription** element (**optional**) containing a **single** **Text** element with no attributes that clearly states the cancellation policy the portal and the hotel have previously agreed upon and the portal has communicated to the customer - the language or languages of this text is chosen by the portal
- **one or more** **HotelReservationID** elements (**optional**) that can be used to transmit miscellaneous IDs associated with the reservation the trading partners have agreed upon; **ResID_Type** **must** be specified with a value from the OTA "Unique Id Type" list (UIT); the other attributes, **ResID_Value**, **ResID_Source** and **ResID_SourceContext** are all **optional**; historically, AlpineBits has used these fields to handle internet campaign management: in this case the agreement is to use a **ResID_Type** value of "13" (internet broker) and the three attributes **ResID_Value**, **ResID_Source** and **ResID_SourceContext** identify, respectively, the campaign name (Slogan in our example), the campaign source (www.example.com) and the campaign marketing medium (top banner) following the scheme used by Google Analytics
- **one optional** **Profile** element with attribute **ProfileType** set to "4" with information about the booking channel; nested under **Profile**, a **CompanyName** element with attributes **Code** and **CodeContext** **must** be present, while the **AddressInfo**, **TelephoneInfo** and **Email** elements are optional: these contain the same data as the equivalent fields in the customer part (note the element names: **AddressInfo** and **TelephoneInfo** vs **Address** and **Telephone** in the customer part - also different ordering - dictated by OTA)
- one empty **BasicPropertyInfo** element (not used by AlpineBits, required for OTA schema validity)

Each **ListItem** element **must** have **Language** and **ListItem** attributes. At most **one** **ListItem** element is allowed for each combination of **Language** and **ListItem**.

Modifications and Cancellations.

A booking modification (**ResStatus** is `Modify`) is identical to a booking reservation (**ResStatus** is `Reserved`). However, for a booking modifications the client will recognize the **UniqueId** attribute and act accordingly, updating the reservation instead of adding a new one.

Besides quote requests and booking reservations, also **cancellations** can be handled. For cancellations the **ResStatus** is `Cancelled` as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_ResRetrieveRS
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_ResRetrieveRS.xsd"
        Version="7.000">

    <Success/>

    <ReservationsList>

        <HotelReservation CreateDateTime="2012-03-21T15:00:00+01:00"
                          ResStatus="Cancelled">

             <!-- Type 15 -> Cancellation -->
            <UniqueID Type="15" ID="c24e8b15ca469388"/>

            <!-- the following are optional for cancellations: -->
            <!--
            <RoomStays>      ...  </RoomStays>
            <ResGuests>      ...  </ResGuests>
            <ResGlobalInfo> ...  </ResGlobalInfo>
            -->

       </HotelReservation>

    </ReservationsList>

</OTA_ResRetrieveRS>
```

samples/GuestRequests-OTA_ResRetrieveRS-cancellation.xml

Each **HotelReservation** **must** of course have the attributes **CreateDateTime** and **ResStatus** set to `Cancelled`.

Of course, the element **UniqueID** is **mandatory**, again with **mandatory** attribute **Type** 15 and attribute **ID** referring to the reservation that is being cancelled! Other reservation elements **may** be also sent.

4.2.3. Follow-up client request (acknowledgement)

A client, upon receiving a non-empty response to its first request (`OTA_Read:GuestRequests`), should initiate another request, acknowledging **or** refusing the **UniqueID** values it got (referring to quotes, reservations or cancellations).

For this follow-up request the parameter `action` is set to the value `OTA_NotifReport:GuestRequests` and the parameter `request` must contain a OTA_NotifReportRQ document.

Here is an example where a client refuses one reservation request and acknowledges three other requests. For the **refusal**, a standard `Warning` element is used. The value of the attribute `Type` can be set to any value of the OTA list “Error Warning Type” (EWT). The value 3 stands for “Biz rule”. The attribute `Code` can be set to any value present in the OTA list “Error Codes” (ERR). The value 450 stands for “Unable to process”. For the **acknowledgments,** the client uses again the `HotelReservation` element, one for each ID it wishes to acknowledge.

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_NotifReportRQ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_NotifReportRQ"
        Version="1.000">

    <Success/>

    <Warnings>
        <!-- refuse reservation with ID=f054bbd2f5ebab9 -->
        <Warning Type="3" Code="450" RecordID="f054bbd2f5ebab9">
            Unable to process reservation
        </Warning>
    </Warnings>

    <NotifDetails>
        <HotelNotifReport>
            <HotelReservations>

                <HotelReservation>
                    <!-- ACK reservation with ID="6b34fe24ac2ff810" -->
                    <UniqueID Type="14" ID="6b34fe24ac2ff810"/>
                </HotelReservation>

                <HotelReservation>
                    <!-- ACK cancellation with ID="c24e8b15ca469388" -->
                    <UniqueID Type="15" ID="c24e8b15ca469388"/>
                </HotelReservation>

                <HotelReservation>
                    <!-- ACK quote request with ID="1000000000000001" -->
                    <UniqueID Type="14" ID="1000000000000001"/>
                </HotelReservation>

            </HotelReservations>
        </HotelNotifReport>
    </NotifDetails>

</OTA_NotifReportRQ>
```

samples/GuestRequests-OTA_ReadRQ-ack.xml

The following rules apply to acknowledgements:

- For every `UniqueID`, the server **must** remember whether or not the client has acknowledged it yet.
- It is a client's responsibility to send the acknowledgments - if it doesn't, it must be prepared to deal with duplicates the next time it queries the server.

Here is a sample sequence of messages exchanged between a client and a server

time	client request	server response	comment
08:00	action = OTA_Read:GuestRequests; request = OTA_Read with SelectionCriteria Start today 00:00	OTA_ResRetrieveRS with UniqueID=1 (Reservation) and UniqueID=2 (Request)	the server answers with today's two requests (1 and 2)
08:01	action = OTA_NotifReport:GuestRequests; request = OTA_NotifReportRQ with UniqueID=1	OTA_NotifReportRS Success	server knows the client got 1, it will not be sent again
09:00	action = OTA_Read:GuestRequests; request = OTA_Read	OTA_ResRetrieveRS with UniqueIDs 2 and 3 (Request)	the client wishes to read all new requests, 1 is not sent (since it was ack'ed), 2 is sent again and 3 is sent because it's new
10:00	action = OTA_Read:GuestRequests; request = OTA_Read	OTA_ResRetrieveRS with UniqueIDs 2 and 3 (Request)	same server response as at 09:00 because 2 and 3 were not ack'ed
10:01	action = OTA_NotifReport:GuestRequests; request = OTA_NotifReportRQ with UniqueID=2, 3	OTA_NotifReportRS Success	server now knows the client got all three
11:00	action = OTA_Read:GuestRequests; request = OTA_Read with SelectionCriteria Start today 00:00	OTA_ResRetrieveRS with UniqueIDs 1, 2 and 3 (Request)	the SelectionCriteria Start overrides the fact that all three guest requests had already been ack'ed, so the server sends all three again

4.2.4. Follow-up server response

The server will respond with a OTA_NotifReportRS document that contains either an **Error** element following the same rules as for the preceding server response or an empty **Success** element i.e. the server must return an error if it detects any mismatch in the acknowledge message. An example of mismatches is if IDs are ack'ed for the wrong Type.

4.2.5. Implementation tips and best practice

- If a non-standard `MealsIncluded` has to be transmitted, consider using the closest standard `MealsIncluded` combination. This needs prior agreement among the parts, which is not covered by AlpineBits. For example in South-Tyrol some hotels offer "Dreiviertel-Pension" (half board plus afternoon snack, hence a non-standard `MealsIncluded` combination) to their guests. This may be transmitted as half board, since "Dreiviertel-Pension" replaces half board for these hotels.

- The value of the `PhoneNumber` attribute (element `Telephone`) should contain the standard international format (as in +<country code><phone number>) whenever possible.

4.3. SimplePackages: package availability notifications

When the value of the `action` parameter is `OTA_HotelRatePlanNotif:SimplePackages` the client intends to send package availability notifications to the server.

Please note that SimplePackages **do not aim** at fully describing all possible aspects and properties of a complex package, and in particular they **do not aim** at describing all properties in a fully machine processable way. Information given as free text is typically intended to be visualized on a portal, not as an input to a fully automated booking workflow (hence the name *"Simple"*).

The availability of multiple packages may be transmitted within a single request, which **must** be treated in a single transaction by the server. Hence an error response means that **no data** has been accepted by the server.

4.3.1. Client Request (notify package availability)

The parameter `request` **must** contains an OTA_HotelRatePlanNotifRQ document with **one or more** `RatePlan` elements each describing an available package. The **mandatory** `Start` and `End` attributes indicate the start and end dates (ISO 8601) of the package's public visibility. Below is the outer part of the request document:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelRatePlanNotifRQ
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelRatePlanNotifRQ.xsd"
    Version="2.001">

    <RatePlans>

        <RatePlan Start="2012-01-01" End="2012-12-31">

            <Rates>
                <!-- availability, costs, meals, see below -->
            </Rates>

            <Description Name="title">
                <!-- see below -->
            </Description>

            <Description Name="intro">
                <!-- see below -->
            </Description>

            <Description Name="gallery">
                <!-- see below -->
            </Description>

            <Description Name="details">
                <!-- see below -->
            </Description>

            <UniqueID Type="18" ID="3f6fce1f0da2ef57"/>

            <HotelRef HotelCode="123" HotelName="Frangart Inn"/>

       </RatePlan>

    </RatePlans>

</OTA_HotelRatePlanNotifRQ>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRQ.xml - outer part

Looking towards the end of the document first:

- the **mandatory** `UniqueID` element uniquely identifies the package, it **must** be of `Type` 18 (meaning other per OTA code table) and the `ID` **must** be present; if a client sends a notification using an `UniqueID` it has sent before it wishes to update (overwrite) all information stored on the server
- the **mandatory** `HotelRef` element associates a hotel to each package. The attributes `HotelCode` and `HotelName` are described in room availability notifications (section 4.1.1)

While it is possible to send more than one `RatePlan` element, AlpineBits requires that all `RatePlan` elements refer to the same Hotel. Hence at most **one** hotel can be dealt with in a single request. An AlpineBits server **must** return an error if it receives a request referring to more than one Hotel.

The **mandatory** `Rates` element contains **one or more** `Rate` elements with data about dates, costs and meals:

```
<Rates>

<Rate MinGuestApplicable="2" Start="2012-08-01" End="2012-08-31" Sat="true" Duration="P7N">

    <BaseByGuestAmts>
        <BaseByGuestAmt NumberOfGuests="1" AmountAfterTax="499.00" CurrencyCode="EUR"/>
    </BaseByGuestAmts>

    <RateDescription Name="included services">
        <!-- included services in free text form -->
        <ListItem ListItem="1" Language="en">parking lot (included)</ListItem>
        <ListItem ListItem="2" Language="en">swimming pool (included)</ListItem>
        <ListItem ListItem="3" Language="en">full board (+25 EUR)</ListItem>
        <ListItem ListItem="1" Language="it">parcheggio</ListItem>
        <ListItem ListItem="2" Language="it">piscina</ListItem>
        <ListItem ListItem="3" Language="it">pensione completa (+25 EUR)</ListItem>
        <ListItem ListItem="1" Language="de">Parkplatz</ListItem>
        <ListItem ListItem="2" Language="de">Schwimmbad</ListItem>
        <ListItem ListItem="3" Language="de">Vollpension (+25 EUR)</ListItem>
    </RateDescription>

    <MealsIncluded Breakfast="true" Lunch="false" Dinner="true"/>

</Rate>

<Rate MinGuestApplicable="2" Start="2012-12-01" End="2012-12-31" Sat="true" Duration="P7N">

    <BaseByGuestAmts>
        <BaseByGuestAmt NumberOfGuests="1" AmountAfterTax="599.00" CurrencyCode="EUR"/>
    </BaseByGuestAmts>

    <RateDescription Name="included services">
        <!-- included services in free text form -->
        <ListItem ListItem="1" Language="en">parking lot</ListItem>
        <ListItem ListItem="2" Language="en">skipass</ListItem>
        <ListItem ListItem="1" Language="it">parcheggio</ListItem>
        <ListItem ListItem="2" Language="it">skipass</ListItem>
        <ListItem ListItem="1" Language="de">Parkplatz</ListItem>
        <ListItem ListItem="2" Language="de">Skipass</ListItem>
    </RateDescription>

    <MealsIncluded Breakfast="true" Lunch="true" Dinner="true"/>

</Rate>

</Rates>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRQ.xml - `Rate` elements

In the example the first of the two **Rate** elements means that the package is 7 nights with arrival date any Saturday in August 2012 (attributes **Start** and **End** are **mandatory**, the optional **Sat** attribute indicates a DOW (day of week) limitation, indicating the rate is available only for arrival days on a Saturday. The **Duration** attribute is **mandatory** and is encoded in ISO 8601. AlpineBits allows only durations given in nights, the form is thus always `PxN` where `x` is a number.

Total cost is 499 EUR (expressed by **mandatory** attributes **AmountAfterTax** and **CurrencyCode**) per person (**mandatory** attribute **NumberOfGuests**) with a minimum of 2 persons (**mandatory** attribute **MinGuestApplicable**). AlpineBits requires **NumberOfGuests** to be always 1. That means package prices are always per person. Also, package prices are always to be considered starting prices: the 499 EUR in this example should be displayed to the customer as "starting from 499 EUR".

The **mandatory** element **MealsIncluded** has attributes **Breakfast**, **Lunch** and **Dinner** set to true or false - all three **must** be be given, even if their value is false; the **optional** attribute **MealPlanCodes** can be used as described in section 4.2, and is not shown in the example.

The included services are given **as free text fields** in the optional **RateDescription** element using **ListItem** elements. Each **ListItem** element **must** have **Language** and **ListItem** attributes. At most **one** **ListItem** element is allowed for each combination of **Language** and **ListItem**. These are intended for humans, not for automated processing.

The second **Rate** element specifies another period of validity of the same package: the example package can be booked also in December, it has full board then, includes a skipass instead of the swimming pool and costs 599 EUR.

Following are four **Description** elements.

```
<!-- the title of the package (plain text),
     repeated for each language -->

<Description Name="title">
    <Text TextFormat="PlainText" Language="en">Hiking in the Atacama Desert</Text>
    <Text TextFormat="PlainText" Language="it">Escursione nel deserto di Atacama</Text>
    <Text TextFormat="PlainText" Language="de">Bergwandern in der Atacamawüste</Text>
</Description>

<!-- the short introductory text (plain text) and optional URLs,
     repeated for each language  -->

<Description Name="intro">

    <Text TextFormat="PlainText" Language="en">
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
        incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </Text>
    <URL>http://www.alpinebits.org/en/</URL>

    <Text TextFormat="PlainText" Language="it">
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
        incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </Text>
    <URL>http://www.alpinebits.org/it/</URL>

    <Text TextFormat="PlainText" Language="de">
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
        incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </Text>
    <URL>http://www.alpinebits.org/de/</URL>
```

```
</Description>

<!-- the images associated with the package: copyright/caption, image and optional URLs,
       repeated for each language -->

<Description Name="gallery">

    <Text TextFormat="PlainText" Language="en">
        (C) 2012 Example Inc.
    </Text>
    <Image>http://www.example.com/image-en.gif</Image>
    <URL>http://www.example.com/en</URL>

    <Text TextFormat="PlainText" Language="it">
        (C) 2012 Example Inc.
    </Text>
    <Image>http://www.example.com/image-it.gif</Image>
    <URL>http://www.example.com/it</URL>

    <Text TextFormat="PlainText" Language="de">
        (C) 2012 Example Inc.
    </Text>
    <Image>http://www.example.com/image-de.gif</Image>
    <URL>http://www.example.com/de</URL>

</Description>


<!-- the detail description as texts (plain text, HTML) or ListItems,
     repeated for each language -->

<Description Name="details">

    <Text TextFormat="PlainText" Language="en">
       Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
       incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
       exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </Text>
    <Text TextFormat="HTML" Language="en">
        <![CDATA[
        Duis aute <b>irure dolor</b> in reprehenderit involuptate velit esse cillum dolore
        eu <a href="http://www.alpinebits.org/">fugiat nulla pariatur</a>. Excepteur sint
        occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
        id est laborum.
        ]]>
    </Text>
    <ListItem ListItem="1" Language="en">commodo consequat</ListItem>
    <ListItem ListItem="2" Language="en">voluptate velit</ListItem>

    <!-- other languages not shown... -->

</Description>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRQ.xml - **Description** elements

Title.

The first **Description** element has **Name** set to `title` and is **mandatory**. This is the title of the package.

It **must** contain nothing but **Text** elements (at least one). Each **Text** element **must** have **TextFormat** set to `PlainText` and the **Language** attribute **must** be given.

At **most** one **Text** element per language is allowed.

Intro.

The second **Description** element has **Name** set to intro and is mandatory as well. This is the short introductory text of the package with optional URLs.

It must contain one or more **Text** elements. Each **Text** element must have **TextFormat** set to PlainText and the **Language** attribute must be given. Each **Text** element can be followed by zero or more **URL** elements.

The **URL** elements are implicitly associated with the text (and the language of the text) that precedes them.

At **most** one **Text** element per language is allowed.

Gallery.

The third **Description** element with **Name** set to gallery is **optional**. It contains **one or more** images associated with the package.

For each image there is a **Text** element with the **TextFormat** set to PlainText. It describes the caption/copyright for the image. The **Language** attribute is mandatory.

Following the **Text** element is the **mandatory** **Image** element containing the image location (HTTP URL). The **Image** element can be followed by zero or more **URL** elements.

The **Image** and **URL** elements are implicitly associated with the caption/copyright text and the language that precedes them. In the present example there are 3 images.

Of course, there can be more than one **Image** per language.

Details.

The fourth and last **Description** element with **Name** set to details, is **optional**. It can contain **zero or more** **Text** elements and **zero or more** **ListItem** elements. It **must** contain **at least one** element however.

Each **Text** element **must** have **TextFormat** set to PlainText or HTML, and the **Language** attribute **must** be given. At most **one** **Text** element is allowed for each combination of **Language** and **TextFormat**. The presence of a **Text** element with **TextFormat** set to HTML is intended as a rich text alternative of a **Text** element with **TextFormat** set to PlainText of the same **Language** and makes the latter **mandatory**.

Please note that an AlpineBits server is explicitly allowed to **filter, shorten or even skip the HTML content**, therefore the usage of **Text** elements with **TextFormat** set to HTML is **not** recommended but left as an option for implementers that absolutely need it.

The **ListItem** elements are intended to provide a structured description of the package that will be shown to the end user. The appearance of the structured description depends on the server implementation, it is not guaranteed to follow the textual description. Each **ListItem** element **must** have **Language** and **ListItem** attributes. At most **one** **ListItem** element is allowed for each combination of **Language** and **ListItem**.

4.3.2. Client request (notify that a package is no longer available)

A client that wishes to notify the server that a package is no longer available will send **one or more** `RatePlan` elements that only contain a `UniqueID` element and are otherwise empty, such as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelRatePlanNotifRQ
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelRatePlanNotifRQ.xsd"
    Version="2.001">

    <RatePlans>

        <RatePlan>

            <UniqueID Type="18" ID="3f6fce1f0da2ef57"/>

       </RatePlan>

    </RatePlans>

</OTA_HotelRatePlanNotifRQ>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRQ-remove.xml

A server must return an error response if a client tries to notify that a package not in the server records (`ID`) is no longer available.

All given `RatePlan` elements must refer to the same hotel: at most **one** hotel can be dealt with in a single request. An AlpineBits server **must** return an error if it receives a request referring to more than one Hotel.

Please note that it is **not** allowed to mix notifications that packages are available with notifications that packages are no longer available in the same request.

4.3.3. Server response

The server will send a response indicating the outcome of the request. The response is an OTA_HotelRatePlanNotifRS document. Four types of outcome are possible: success, advisory, warning or error.

Success

The request was accepted and processed successfully. The client does **not** need to take any further action.

In this case the OTA_HotelRatePlanNotifRS response contains nothing but a **single**, empty **`Success`** element:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Success/>

</OTA_HotelRatePlanNotifRS>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRS-success.xml

Advisory

The request was accepted and processed successfully. However, one or more non-fatal problems were detected and added to the server response. The client does **not** need to resend the request, but **must** notify the user or the client implementer regarding the advisory received.

In this case, the OTA_HotelRatePlanNotifRS response contains an empty **`Success`** element followed by **one or more** **`Warning`** elements with the attribute **`Type`** set to the fixed value of `11`, meaning “Advisory” according to the OTA list “Error Warning Type” (EWT).

If a **`Warning`** element is not regarding the request as a whole but specific to a **`RatePlan`** element in the request, the **`Warning`** **must** also have a **`RecordID`** attribute.

The value of **`RecordID`** must be the value of the **`RatePlan`**/**`UniqueID`**/**`ID`** the **`Warning`** is referred to.

Each **`Warning`** element should contain human readable text as in the following example:

```
<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Success/>
    <Warnings>
        <Warning Type="11" RecordID="3f6fce1f0da2ef57">
            end date is less than 3 days from now
        </Warning>
```

```
    </Warnings>

</OTA_HotelRatePlanNotifRS>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRS-advisory.xml

Warning

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the warning is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelRatePlanNotifRS response contains an empty **`Success`** element followed by **one or more** **`Warning`** elements with the attribute **`Type`** set to any value allowed by the OTA list “Error Warning Type” (EWT) **other than** `11` (“Advisory”).

If a **`Warning`** element is not regarding the request as a whole but specific to a **`RatePlan`** element in the request, the **`Warning`** **must** also have a **`RecordID`** attribute.

The value of **`RecordID`** must be the value of the **`RatePlan`**/**`UniqueID`**/**`ID`** the **`Warning`** is referred to.

Each **`Warning`** element should contain a human readable text as in the following example:

```
<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Success/>
    <Warnings>
        <Warning Type="3" RecordID="3f6fce1f0da2ef57">
            dates are too far in the future for this server to process
        </Warning>
    </Warnings>

</OTA_HotelRatePlanNotifRS>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRS-warning.xml

Error

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the error is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelAvailNotifRS response contains **one or more** **`Error`** elements with the attribute **`Type`** set to the fixed value of `13`, meaning “Application error” according to the OTA list “Error Warning Type” (EWT) and the attribute **`Code`** set to any value present in the OTA list “Error Codes” (ERR).

```
<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Errors>
        <Error Type="13" Code="404">
            Invalid start/end date combination
        </Error>
    </Errors>

</OTA_HotelRatePlanNotifRS>
```

samples/SimplePackages-OTA_HotelRatePlanNotifRS-error.xml

4.3.4. Implementation tips and best practice

- If a non-standard **`MealsIncluded`** has to be transmitted, consider using the closest standard **`MealsIncluded`** combination. This needs prior agreement among the parts, which is not covered by AlpineBits. For example in South-Tyrol some hotels offer "Dreiviertel-Pension" (half board plus afternoon snack, hence a non-standard **`MealsIncluded`** combination) to their guests. This may be transmitted as half board, since "Dreiviertel-Pension" replaces half board for these hotels.
- The OTA lists “Error Warning Type” (EWT) and “Error Codes” (ERR) come with the OTA2015A documentation package. The package can be downloaded from the OTA web site [3]. The file `OpenTravel_CodeList_2015_06_03.xlsm` contains all the lists.

4.4. Inventory: room category information

When the value of the `action` parameter is `OTA_HotelDescriptiveContentNotif:Inventory` the client informs the server about the available room categories and optionally rooms.

Please note that for this purpose the message used in this version is different than the one used previously! Almost every information sent previously can nevertheless be mapped to the new message, the mapping is outlined in appendix B.2.

4.4.1. Client request

The parameter `request` contains an OTA_HotelDescriptiveContentNotifRQ document.

Each document contains **one HotelDescriptiveContent** element. For the **mandatory** attributes **HotelCode** and **HotelName** the rules are the same as for room availability notifications (section 4.1.1). Note that information about only one hotel per message can be transmitted.

Nested inside the **HotelDescriptiveContent** an enclosing **FacilityInfo** element contains a list of **GuestRoom** elements. These may be of three distinct kinds:

- to define room categories and provide ***basic*** description of them (the category is identified by the attribute **Code**)
- to provide ***additional*** descriptive content related to room categories (identified by the attribute **Code**)
- to list specific rooms for each category (identified by the attribute **Code**)

Here is the global structure of such a document, here the first **GuestRoom** element (collapsed) is the category definition, the following elements are the specific rooms. The differences are explained in detail below:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelDescriptiveContentNotifRQ
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opentravel.org/OTA/2003/05"
  xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
                      OTA_HotelDescriptiveContentNotifRQ.xsd"
  Version="8.000">

 <HotelDescriptiveContents>

    <HotelDescriptiveContent HotelCode="123" HotelName="Frangart Inn">

      <FacilityInfo>

       <GuestRooms>

          <!-- This element defines a category and contains its basic description -->

          <GuestRoom Code="DZ" MaxOccupancy="2" MinOccupancy="1" MaxChildOccupancy="1">
            <!-- -->
          </GuestRoom>

         <GuestRoom Code="DZ">

            <TypeRoom RoomID="101"/>

          </GuestRoom>
```

```
          <GuestRoom Code="DZ">
            <TypeRoom RoomID="102"/>
          </GuestRoom>
        </GuestRooms>
      </FacilityInfo>
    </HotelDescriptiveContent>
  </HotelDescriptiveContents>
</OTA_HotelDescriptiveContentNotifRQ>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRQ.xml - outer part

The **GuestRoom** element, when used to define a room category and its ***basic*** description (as in the first element of the example above), contains the following attributes:

- **Code: mandatory**, identifies the category
- **MinOccupancy**: **mandatory**, sets the minimum number of guests allowed for this room category
- **MaxOccupancy**: **mandatory**, sets the maximum number of guests allowed for this room category
- **MaxChildOccupancy**: **optional**, must be 0 <= **MaxChildOccupancy** <= **MaxOccupancy**. This attribute can be used to influence the computation of the total cost of the stay in the presence of children (see section 4.5 under “Computing the cost of a stay” for details). Note that this attribute **cannot** be used to disallow children in a stay. This attribute **may only** be used if the capability `OTA_HotelDescriptiveContentNotif_Inventory_occupancy_children` is announced by the server.

In the same context of defining room categories and their ***basic*** description, inside **GuestRoom** the following elements are defined:

- **TypeRoom**: **mandatory** element with the **mandatory** attribute **StandardOccupancy**, indicating the type of Room and the **mandatory** attribute **RoomClassificationCode**, used to classify the kind of guest room following the OTA list “Guest Room Info” (GRI) (42 means just “Room”, 13 means “Apartments”, etc...). No further attributes are allowed for this element in this context.
- **Amenities: optional** element containing a list of **Amenity**, each indentified by the **mandatory** attribute **RoomAmenityCode** following the OTA list "Room Amenity Type" (RMA)
- **MultimediaDescription**: **exactly one MultimediaDescription** element with attribute **InfoCode** = `25` (Long name) **must** be present, **at most one MultimediaDescription** element with attribute **InfoCode** = `1` (Description) **may be optionally** present and **at most one MultimediaDescription** element with attribute **InfoCode** = `23` (Pictures) **may be optionally** present. This element is explained in more depth in the following section.

Here is an example of the **GuestRoom** element used to define a room category and its ***basic*** descriptive data:

```
<GuestRoom Code="DZ" MaxOccupancy="2" MinOccupancy="1" MaxChildOccupancy="1">

  <!-- RoomClassificationCode = "42" means Room, 13 Apartment, see OTA table GRI -->
  <TypeRoom StandardOccupancy="2" RoomClassificationCode="42"/>

  <Amenities>
    <!-- 26 means Crib, see OTA table RMA -->
    <Amenity RoomAmenityCode="26"/>
  </Amenities>

  <MultimediaDescriptions>

    <MultimediaDescription InfoCode="25">
      <!-- ... -->
    </MultimediaDescription>

    <MultimediaDescription InfoCode="1">
      <!-- ... -->
    </MultimediaDescription>

    <MultimediaDescription InfoCode="23">
      <!-- ... -->
    </MultimediaDescription>

  </MultimediaDescriptions>

</GuestRoom>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRQ.xml - a GuestRoom used to define and describe a room category

The **GuestRoom** element can also provide ***additional*** descriptive content related to room categories in a separate message, in this case its only **mandatory** attribute is **Code** to identify the room category and the only **mandatory** elements are **MultimediaDescription**, all without attribute **InfoCode**. If ***additional*** descriptive content is sent in the same message as the ***basic*** information, then the related **MultimediaDescription** elements may just be appended to the **MultimediaDescriptions** container.

A server announces if it accepts ***basic*** or ***additional*** descriptions using the corresponding capabilities `OTA_HotelDescriptiveContentNotif_Inventory_accept_basic` and `OTA_HotelDescriptiveContentNotif_Inventory_accept_additional`. A client receiving both capabilities from the server **must** send all the information it has on record within the same message. In this case the **MultimediaDescription** elements without attribute **InfoCode** must be transmitted beside the ***basic*** descriptions.

The **GuestRoom** element is also used to list all the rooms belonging to a category (as seen in the first example), in this case the **only** allowed and **mandatory** attribute is **Code** that identifies the category for the specific room. **No further attributes** are allowed. Similar restrictions also apply to the sub element inside **GuestRoom**: the **mandatory** element **TypeRoom** with the **only** and **mandatory** attribute **RoomID** that identifies the specific room is allowed, further elements or attributes are **not allowed**, included but not limited to the **MultimediaDescription** element.

A server **must** accept a list of rooms, that **should** be sent by the client - if available - regardless of the capability `OTA_HotelDescriptiveContentNotif_Inventory_use_rooms`. If the capability is set by the server then the full list of rooms **must** be sent. A server that has no use for the Room list data is

free to discard these upon receiving them, but **must** do so silently (i.e. without returning Errors or Warnings).

Basic and additional descriptive content

Two levels of descriptive content were already introduced: ***basic*** and ***additional***. Both levels are transmitted using **MultimediaDescription** elements. Descriptions of the ***basic*** level **must** have the **InfoCode** attribute set, whereas the descriptions of the ***additional*** level **must not** have the InfoCode attribute. The general rules for this element are the following:

- the **optional InfoCode attribute**, set only for ***basic*** descriptions, **must** have one of the following values: `1` for (textual) Description, `25` for (textual) Long Name (used as title in AlpineBits), `23` for Pictures. This is a subset of OTA list "Information type" (INF).
- a **MultimediaDescription** element with attribute **InfoCode** = `25` or `1` **must** contain only a single **TextItem** element
- a **MultimediaDescription** element with attribute **InfoCode** = `23` **must** contain only a single **ImageItem** element
- a **MultimediaDescription** element for ***additional*** descriptive content, hence without attribute **InfoCode** **must** contain (if present) **only** a single **ImageItems** element which **must** contain a list of one or more **ImageItem** elements, no further elements are allowed.

An example of the element **MultimediaDescription** used to transmit the ***basic*** descriptive content follows:

```
<MultimediaDescriptions>

  <MultimediaDescription InfoCode="25">

    <TextItems>

      <TextItem>

        <Description TextFormat="PlainText" Language="en">Double room</Description>

        <Description TextFormat="PlainText" Language="de">Doppelzimmer</Description>

        <Description TextFormat="PlainText" Language="it">Camera doppia</Description>

      </TextItem>

    </TextItems>

  </MultimediaDescription>

  <MultimediaDescription InfoCode="1">

    <TextItems>

      <TextItem>

        <Description TextFormat="PlainText" Language="en">Description of the double
room.</Description>

        <Description TextFormat="PlainText" Language="de">Doppelzimmer
Beschreibung.</Description>

        <Description TextFormat="PlainText" Language="it">Descrizione della camera
doppia.</Description>

      </TextItem>
```

```
    </TextItems>

  </MultimediaDescription>

  <MultimediaDescription InfoCode="23">

    <ImageItems>

      <!-- 6 means Guest room, see OTA table PIC -->
      <ImageItem Category="6">

        <ImageFormat CopyrightNotice="Copyright notice 2015">

          <URL>http://www.example.com/image.jpg</URL>

        </ImageFormat>

        <Description TextFormat="PlainText" Language="en">Picture of the room</Description>

        <Description TextFormat="PlainText" Language="de">Zimmerbild</Description>

        <Description TextFormat="PlainText" Language="it">Immagine della
stanza</Description>

      </ImageItem>

    </ImageItems>

  </MultimediaDescription>

<MultimediaDescriptions>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRQ.xml - a List of MultimediaDescription

TextItems contains a list of **TextItem** elements, which **must** contain one or more **Description** elements with the **mandatory** attributes **TextFormat** (**must** be `PlainText`) and the **mandatory** attribute **Language**. The **Language** attribute must follow ISO 639-1 (two- letter lowercase language abbreviation). There can be at most one **Description** for any given value of the **Language** attribute.

ImageItems contains a list of one or more **ImageItem** elements with **mandatory Category** attribute following the OTA list "Picture Category Code" (PIC). Typical values are `6` (Guest Room) and `17` (Map) but the whole table is allowed. The **ImageItem** element contains a **single**, **mandatory ImageFormat** element with the **optional** attribute **CopyrightNotice**. Inside, a **single mandatory** element **URL**, holds the URL where the picture can be retrieved as its value. **Zero or more Description** elements follow, under the same rules outlined above for descriptions contained in the element **TextItem**. Images **should** be processed by the server in the order they are submitted (i.e. the order used to show the pictures the to end users should be consistent with the one sent by the client).

The distinction between ***basic*** and ***additional*** descriptions has been introduced because a single client could have information that are fundamental on the technical side (everything outlined in the first scenario above, e.g. minimum and maximum occupancy) but may lack inspiring pictures of the rooms. AlpineBits therefore introduces a possibility for the server to accept ***additional*** descriptive content from one or multiple clients, while still having a single reputable source for the ***basic*** information (technically speaking a server may accept basic data from multiple clients, but it is highly discouraged to do so to prevent inconsistencies).

Note that AlpineBits does not support deltas for Inventory. After successfully processing an Inventory request of type ***basic***, a server should consider deleted all previous ***basic*** inventory data sent via

AlpineBits. Further, all ***additional*** Inventory data, FreeRooms and RatePlans that refer to any now missing inventory data should be considered outdated. Any ***additional*** Inventory data referring to inventory data that is still valid **must not** be updated.

An Inventory request of type ***additional*** replaces all previous ***additional*** Inventory data, the server **must not** update the ***basic*** inventory data. If multiple clients are providing ***additional*** Inventory data it is up to the server implementation to guarantee the integrity of the data.

A server, when receiving a message that contains only ***additional*** descriptive content **must** check if the room categories are already known and **must** give proper feedback (i.e. return a warning) if they aren't (this indicates that the ***basic*** info either on the server or on the client are outdated). Please note that **missing** room categories **must not** be considered an error, as it is well possible that some room categories do not have ***additional*** descriptive data.

Please note that categories in inventories should refer to physical inventories. They **must not** be used as logical inventories. For example it is not allowed to introduce an inventory with code `suite-x` and one with code `suite-x-special-offer` for the purpose of modeling two different products from the same inventory (RatePlans will take care of that).

4.4.2. Server response

The server will send a response indicating the outcome of the request. The response is an OTA_HotelDescriptiveContentNotifRS document. Four types of outcome are possible: success, advisory, warning or error.

Success

The request was accepted and processed successfully. The client does **not** need to take any further action.

In this case the OTA_HotelDescriptiveContentNotifRS response contains nothing but a **single**, empty **`Success`** element:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelDescriptiveContentNotifRS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelInvNotifRS"
    Version="3.000">

    <Success/>

</OTA_HotelDescriptiveContentNotifRS>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRS-success.xml

Advisory

The request was accepted and processed successfully. However, one or more non-fatal problems were detected and added to the server response. The client does **not** need to resend the request, but **must** notify the user or the client implementer regarding the advisory received.

In this case, the OTA_HotelDescriptiveContentNotifRS response contains an empty **`Success`** element followed by **one or more** **`Warning`** elements with the attribute **`Type`** set to the fixed value of `11`, meaning “Advisory” according to the OTA list “Error Warning Type” (EWT).

Each **Warning** element should contain a human readable text as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelDescriptiveContentNotifRS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelDescriptiveContentNotifRS"
    Version="3.000">

    <Success/>
    <Warnings>
        <Warning Type="11">
            description text contains lorem ipsum
        </Warning>
    </Warnings>

</OTA_HotelDescriptiveContentNotifRS>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRS-advisory.xml

Warning

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the warning is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelDescriptiveContentNotifRS response contains an empty **Success** element followed by **one or more** **Warning** elements with the attribute **Type** set to any value allowed by the OTA list “Error Warning Type” (EWT) **other than** `11` (“Advisory”).

Each **Warning** element should contain a human readable text as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelDescriptiveContentNotifRS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelDescriptiveContentNotifRS"
    Version="3.000">


    <Success/>
    <Warnings>
        <Warning Type="3">
            too many products
        </Warning>
    </Warnings>

</OTA_HotelDescriptiveContentNotifRS>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRS-warning.xml

Error

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the error is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelDescriptiveContentNotifRS response contains **one or more** `Error` elements with the attribute `Type` set to the fixed value of `13`, meaning “Application error” according to the OTA list “Error Warning Type” (EWT) and the attribute `Code` set to any value present in the OTA list “Error Codes” (ERR).

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelDescriptiveContentNotifRS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opentravel.org/OTA/2003/05"
    xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_HotelDescriptiveContentNotifRS"
    Version="3.000">


    <Errors>
        <Error Type="13" Code="404">
            inconsistent values for occupancy
        </Error>
    </Errors>

</OTA_HotelDescriptiveContentNotifRS>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRS-error.xml

4.4.3. Implementation tips and best practice

Please note that for this purpose the message used in this version is different than the one used previously! Almost every information sent previously can nevertheless be mapped to the new message, the mapping is outlined in appendix B.1.

4.5. RatePlans

If the **action** parameter is `OTA_HotelRatePlanNotif:RatePlans` the client sends information about rates and related rules.

4.5.1. Client request

The parameter **request** contains an OTA_HotelRatePlanNotifRQ document.

Each document contains **one RatePlans** element. For the **mandatory** attributes **HotelCode** and **HotelName** the rules are the same as for room availability notifications (section 4.1). Note that requests are limited to one hotel per message.

Nested inside **RatePlans** are **RatePlan** elements, one for each rate plan. The **RatePlan** element has the following **mandatory** attributes (but see the next section for exceptions):

- ➔ **RatePlanNotifType** is either `New`, `Overlay` or `Remove` (see section "Synchronization" below),
- ➔ **CurrencyCode** is `EUR`,
- ➔ **RatePlanCode** is the rate plan ID.

The optional **RatePlan** attributes **RatePlanType** and **RatePlanCategory** are used to transmit *special offers* and are defined as follows: A *special offer* or *package* presented by the hotel **must** set **RatePlanType** to 12 (means "Promotional") and **must not** set the **RatePlanCategory** attribute. An offer or package campaigned by a third party (such as a consortium or a tourist organization) in which the hotel participates **must** set **RatePlanType** to 12 **and also** the **RatePlanCategory** attribute with a value defined by the third party.

Two more optional attributes are **RatePlanID** and **RatePlanQualifier**. They can **only be sent** if the server supports the `OTA_HotelRatePlanNotif_accept_RatePlanJoin` capability. These two attributes are used to identify a "master" rateplan and its alternative versions (e.g. different MealPlanCodes for the same offer by the lodging structure). All these alternative versions share the same **RatePlanID**: **exactly one** RatePlan for each **RatePlanID** value **must** have the **RatePlanQualifier** set to `true`, every other RatePlan that shares the same **RatePlanID must** have the **RatePlanQualifier** set to `false`. The RatePlan with **RatePlanQualifier** set to `true` **must** be used by the server for every information outside **BookingRule** and **Rate** elements (e.g. Offers, Supplements, descriptive contents are taken from this RatePlan).

Each **RatePlan** element contains, in order:

- zero or more **BookingRule** elements: used to restrict the applicability of the rate plan to a given stay - zero means no restrictions,
- zero or more **Rate** elements: indicate the cost of stay,
- zero or more **Supplement** elements: to specify supplements such as final cleaning fees or similar extras,
- zero, one or two **Offer** elements: indicates offers such as free nights or kids go free,
- zero, one or two **Description** elements.

Here is the global structure of the document:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelRatePlanNotifRQ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.opentravel.org/OTA/2003/05"
        xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05 OTA_HotelRatePlanNotifRQ"
        Version="1.000">

    <RatePlans HotelCode="123" HotelName="Frangart Inn">

        <RatePlan RatePlanNotifType="Overlay" CurrencyCode="EUR" RatePlanCode="Rate1-4-HB">

            <BookingRules>
                <BookingRule Start="2014-04-15" End="2014-11-02">
                    ...
                </BookingRule>
            </BookingRules>

            <Rates>
                <Rate InvTypeCode="double" Start="2014-04-15" End="2014-05-20">
                    ...
                </Rate>
            </Rates>

            <Supplements>
                <Supplement> ... </Supplement>
            </Supplements>

            <Offers>
                <Offer> ... </Offer>
            </Offers>

            <Description Name="title">
                <!-- ... -->
            </Description>

        </RatePlan>

    </RatePlans>

</OTA_HotelRatePlanNotifRQ>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - outer part

The **optional** **Description** element, if present, **must** have the attribute **Name** set to either `title` or `intro`. When two **Description** elements are used, they **must not** have the same value for the attribute **Name**. Each **Description** element contains one or more **Text** elements with the attribute **TextFormat** set to `PlainText` or `HTML` and the attribute **Language** set to a two-letter lowercase language abbreviation according to ISO 639-1. At most **one** **Text** element is allowed for each combination of **Language** and **TextFormat**.

The presence of a **Text** element with **TextFormat** set to `HTML` is intended as rich text alternative of a **Text** element with **TextFormat** set to `PlainText` of the same **Language** and makes the latter **mandatory**.

Please note that an AlpineBits server is explicitly allowed to **filter, shorten or even skip the HTML content**, therefore the usage of **Text** elements with **TextFormat** set to `HTML` is **not** recommended but left as an option for implementers that absolutely need it.

The elements **BookingRule**, **Rate**, **Supplement** and **Offer** are explained in the following sections.

Booking rules

BookingRule elements **can** be linked to room categories (see section 4.4) via their **Code** attribute. If the **Code** attribute is given, also the **CodeContext** attribute **must** be set and its value **must** be `ROOMTYPE` (OTA lacks a **InvTypeCode** attribute in this context). A **BookingRule** without a **Code** attribute applies to all room categories.

A **BookingRule** element **must** have attributes **Start** and **End** (both must be valid dates in the form YYYY-MM-DD) and satisfy the condition **Start** ≤ **End**. Unless otherwise specified, **Start** and **End** **must** be considered inclusive.

Within the same rate plan, **BookingRule** elements **must not** overlap (concerning their **Start** and **End** attributes) if they belong to the same class. Classes are:

- **BookingRule** elements with no **Code** attribute,
- **BookingRule** elements with the same value for the **Code** attribute.

The server **must** consider overlaps as an **error**.

BookingRule elements are used to define a number of restriction criteria:

- the minimum or maximum length of stay (LOS) using the **LengthOfStay** element,
- the arrival day of week (arrival DOW) using the **ArrivalDaysOfWeek** element,
- the departure day of week (departure DOW) using the **DepartureDaysOfWeek** element,
- a master restriction status (values `Open`/`Close`) using the **RestrictionStatus** element.

Any missing criteria is to be interpreted as unrestricted.

When matching a **BookingRule**, a server **must** consider the following rules:

- criteria of the booking rule (if any) that applies to the arrival day (**Start** ≤ arrival day ≤ **End**): min LOS, max LOS, arrival DOW, master status `Open`,
- criteria of the booking rule (if any) that applies to the departure day (**Start** ≤ departure day ≤ **End**): departure DOW,
- each day of the stay (excluding the departure day) **must not** be denied by a master status `Close` rule.

A stay must be allowed by all applicable booking rules. In particular, there might be a **BookingRule** element **with** a Code attribute and a **BookingRule** element **without** a Code attribute, both applicable to a given stay. In such a case, both rules must allow the stay.

Three booking rule examples are shown here.

```
<BookingRule Start="2016-03-03" End="2016-04-17">
    <LengthsOfStay>
        <LengthOfStay Time="5" TimeUnit="Day" MinMaxMessageType="SetMinLOS"/>
        <LengthOfStay Time="7" TimeUnit="Day" MinMaxMessageType="SetMaxLOS"/>
    </LengthsOfStay>
</BookingRule>
```

Booking rule (example A)

In example A, length of stay (LOS) restrictions are given. **At most one** **LengthOfStay** element with a **MinMaxMessageType** value of `SetMinLOS` and **one** with a **MinMaxMessageType** value of `SetMaxLOS` can be given. The **Time** attribute must be an integer > 0 and the **TimeUnit** must be **Day**. This rule would restrict any stay having 2016-03-03 ≤ arrival day ≤ 2016-04-17 to a duration between 5 and 7 nights.

```
<BookingRule Start="2016-01-01" End="2017-12-31" Code="double" CodeContext="ROOMTYPE">
    <DOW_Restrictions>
        <ArrivalDaysOfWeek   Mon="0" Tue="0" Weds="0" Thur="1" Fri="0" Sat="1" Sun="0"/>
        <DepartureDaysOfWeek Mon="0" Tue="0" Weds="0" Thur="1" Fri="0" Sat="1" Sun="0"/>
    </DOW_Restrictions>
</BookingRule>
```

Booking rule (example B)

In example B, arrival day of week (arrival DOW) and departure day of week (departure DOW) restrictions are given. At most one **ArrivalDayOfWeek** element and at most one **DepartureDayOfWeek** element must be given. The DOW attributes that are `0` or `false` indicate restricted DOWs. A missing DOW attribute or a value of `1` or `true` indicate there is no restriction. This example would restrict any stay requesting a “double” room (note the **Code** attribute) to arrive and depart on a Thursday or Saturday.

Alternatively, example B could also be written as:

```
<BookingRule Start="2016-01-01" End="2017-12-31" Code="double" CodeContext="ROOMTYPE">
    <DOW_Restrictions>
        <ArrivalDaysOfWeek   Mon="0" Tue="0" Weds="0"          Fri="0"         Sun="0"/>
        <DepartureDaysOfWeek Mon="0" Tue="0" Weds="0"          Fri="0"         Sun="0"/>
    </DOW_Restrictions>
</BookingRule>
```

Booking rule (example B - alternative)

The following example (C) forbids any stay in the suite in August (departure on the 1st of August is possible).

```
<BookingRule Start="2016-08-01" End="2016-08-31" Code="suite" CodeContext="ROOMTYPE">
    <RestrictionStatus Restriction="Master" Status="Close"/>
</BookingRule>
```

Booking rule (example C)

Both attributes, **Restriction** and **Status** are **mandatory**. The value `Close` is necessary for the restriction to occur. An `Open` value would be equivalent to no restriction.

Rates

Rate elements must have an **InvTypeCode** attribute that links them to room categories (see section 4.4).

A **Rate** element **must** have attributes **Start** and **End** (both must be valid dates in the form YYYY-MM-DD) and satisfy the condition **Start** ≤ **End**.

A stay matches the **Start** and **End** attributes if the the arrival day is ≥ **Start** and the departure day ≤ **End** + 1. When the server computes the total cost of the stay it **must** find a matching rate for each night of the stay. Otherwise it **cannot** compute the total cost, and the stay is not possible.

Within the same rate plan, two **Rate** elements **must not** overlap (concerning their **Start** and **End** attributes) if they have the same **InvTypeCode** attribute.

The server **must** consider overlaps as an **error**.

By default, all rates are per night. It is, however, possible to specify rates per an arbitrary amount of nights. This is done by adding the attributes **RateTimeUnit** (`Day` is the only allowed value) and **UnitMultiplier** (number of nights) to the **Rate** element. All rates in the same rate plan must have the same value for **UnitMultiplier**.

Rate elements specify costs (all amounts are taken to be in EUR and after taxes). **Rate** sub-elements are: **BaseByGuestAmt**, **AdditionalGuestAmount** and **MealsIncluded**.

Here is an example of a rate:

```
<Rate InvTypeCode="double" Start="2014-03-03" End="2014-03-08">
<BaseByGuestAmts>
  <BaseByGuestAmt Type="7" NumberOfGuests="1" AgeQualifyingCode="10" AmountAfterTax="106"/>
  <BaseByGuestAmt Type="7" NumberOfGuests="2" AgeQualifyingCode="10" AmountAfterTax="192"/>
</BaseByGuestAmts>
<AdditionalGuestAmounts>
  <AdditionalGuestAmount AgeQualifyingCode="8"              MaxAge="3"  Amount="0"   />
  <AdditionalGuestAmount AgeQualifyingCode="8"  MinAge="3"  MaxAge="6"  Amount="38.4"/>
  <AdditionalGuestAmount AgeQualifyingCode="8"  MinAge="6"  MaxAge="10" Amount="48"  />
  <AdditionalGuestAmount AgeQualifyingCode="8"  MinAge="10" MaxAge="16" Amount="67.2"/>
  <AdditionalGuestAmount AgeQualifyingCode="10"                         Amount="76.8"/>
</AdditionalGuestAmounts>
<MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
</Rate>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - rate

The **BaseByGuestAmt** elements (**at least one** must be present) have the following attributes:

- **Type**(**mandatory**) with values `7` (“per person”) or `25` (“per room”),
- **NumberOfGuests** (**mandatory**) is an integer value > 0,
- **AmountAfterTax** (**mandatory**) is a decimal value > 0.0 (0.0 **is not** allowed),
- **AgeQualifyingCode** (**mandatory**) is set to `10` (“adult”).

For a given **RatePlan**, the value of all **Type** attributes **must be the same** for all **BaseByGuestAmt** elements in every **Rate** in the rate plan.

For a given **Rate**, all **BaseByGuestAmt** elements **must have distinct NumberOfGuests** values.

One or more **BaseByGuestAmt** elements are needed to cover all possible guest occupancies compatible with the room category occupancy limits. In particular, one **BaseByGuestAmt** element with attributes **NumberOfGuests** equal to the standard occupancy **must** be present. Additional **BaseByGuestAmt** elements with values between the minimum and the standard occupancy **may** be present. See the section “Computing the cost of a stay” below for details.

The **AdditionalGuestAmount** elements (zero or more can be present) are used to transmit the prices for guests that are in a room beyond its standard occupancy. Specific prices for children may also be sent with these elements. They have the following attributes:

- **AgeQualifyingCode** (**mandatory**) is set to `8` (“child) or `10` (“adult”),
- **MinAge** and **MaxAge** are both integer values > 0 (a value of 0 is forbidden by OTA, even for **MinAge**); when both are given together, the inequation **MaxAge** > **MinAge** must hold,
- **Amount** (**mandatory**) is a decimal value ≥ 0.0 (0.0 **is** allowed)

AdditionalGuestAmount elements must also comply with these rules that help resolve ambiguities when computed the total cost of a stay:

- If **AdditionalGuestAmount** are defined at all, **exactly one AdditionalGuestAmount** element with a **AgeQualifyingCode** set to `10` (“adult”) **must** be present.
- **AdditionalGuestAmount** elements having **AgeQualifyingCode** set to `8` (“child”) **must have at least one** of the attributes **MinAge** or **MaxAge**. Contrary, those with **AgeQualifyingCode** set to `10` (“adult”) **must have neither**.
- The attributes **MinAge** and **MaxAge** are used to identify age brackets. An age matches the bracket if and only if the following two conditions hold:
- **MinAge** is not given or **MinAge** ≤ age,
- **MaxAge** is not given or **MaxAge** > age.
  For each rate the set of age brackets **must** be defined in such a way that every possible age matches zero or one age brackets.

  If the set of age brackets “has holes”, i.e. given ages a < b < c, a and c are matched by some brackets, but b is not, a server **may** respond with an **advisory** warning. Please note that the message is valid but it is strongly discouraged as this may change in an upcoming release of AlpineBits.

AlpineBits **requires** all rates in a rate plan to refer to the **same board type**. Ideally one would define the board type at the level of the rate plan, but OTA doesn't support that. So all rates **must** contain a **MealsIncluded** element with the same value for the **MealPlanCodes** attribute.

AlpineBits **does not** use the single Breakfast/Lunch/Dinner booleans, but relies on the **MealPlanCodes** attribute only. The following codes (a subset of the full OTA list) are allowed:

- 1 - all inclusive,
- 3 - bed and breakfast,
- 10 - full board,
- 12 - half board,
- 14 - room only.

The **MealsIncluded** element **must** have the **MealPlanIndicator** attribute set to true.

Supplements

Supplements are supported through **Supplement** elements.

Each **Supplement** element has the following **mandatory** attributes:

- **InvType** is set to `EXTRA`,
- **InvCode** can be set freely (1 - 16 characters according to OTA) and **is used as a key to identify a supplement**.

The **InvType** value `ALPINEBITSEXTRA` is not currently used, but is reserved for a future shared list of common **InvCode** values.

The following attributes and sub-elements describe the supplement, in the rest of the document they'll be referred to as ***static data***:

- The attribute **AddToBasicRateIndicator** **must** be set to `true` (to indicate the supplement amount must be added to the amount coming from the rate).

1. The attribute **MandatoryIndicator** (a boolean value) indicates that the customer must book the supplement (`true`) or can choose to book the supplement (`false`); a missing **MandatoryIndicator** attribute has the same meaning as one set to `false`.

2. The attribute **ChargeTypeCode** **must** have one of the following values:
    a. `1` - daily,
    b. `12` - per stay,
    c. `18` - per room per stay,
    d. `19` - per room per night,
    e. `20` - per person per stay,
    f. `21` - per person per night,
    g. `24` - item.

    Note that when **ChargeTypeCode** is 24, the total cost of stay should be computed by asking the user for the number of items.

1. Up to two **Description** elements (following the same schema as the rate plan **Description** elements).

All the attributes mentioned are **mandatory** for supplements that contain static data, the **Description** element with **Name** set as `title` is mandatory as well, the Description element with Name set as `intro` is optional. Further, supplements that contain static data **must not** contain the attributes **Start** and **End**.

The following attributes and sub-elements define the price of the supplement for specific periods of time, in the rest of the document they will be referred to as ***date depending data***:

- the attribute **Amount** indicates the cost of the supplement; amounts are taken to be in EUR and after taxes; a value of 0 indicates the supplement is free of charge. If the attribute is missing, the Supplement is not available,
- the attributes **Start** and **End** (with the usual meaning) define the period where the **Amount** surcharge is applied.

The attributes **Start** and **End** are **mandatory** for supplements that contain date depending data, the attribute **Amount** is **optional**. No further Element or Attribute is allowed.

Multiple date dependent **Supplement** elements referring to the same **InvCode** might be used to specify different prices for different date ranges. **Overlaps are not allowed**: it is a client responsibility to check that different Amount are never set for the same date and the same **Supplement**; a server **must** return an error if it detects such an inconsistency in the data.

When defining supplements, static and date depending data **must** be transmitted in **separate** **Supplement** elements.

Here is a complete example of a supplement:

```
<Supplements>

<Supplement InvType="EXTRA"
            InvCode="0x539"
            AddToBasicRateIndicator="true"
            MandatoryIndicator="true"
            ChargeTypeCode="18">
    <Description Name="title">
        <Text TextFormat="PlainText" Language="de">Endreinigung</Text>
        <Text TextFormat="PlainText"  Language="it">Pulizia finale</Text>
    </Description>
    <Description Name="intro">
        <Text TextFormat="PlainText" Language="de">
            Die Endreinigung lorem ipsum dolor sit amet.
        </Text>
        <Text  TextFormat="PlainText" Language="it">
            La pulizia finale lorem ipsum dolor sit amet.
```

```
        </Text>
    </Description>
</Supplement>

<Supplement InvType="EXTRA"
            InvCode="0x539"
            Amount="20"
            Start="2014-10-01"
            End="2014-10-11">
</Supplement>

</Supplements>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - supplement

Static data may **only** be transmitted in messages with **RatePlanNotifType** set to `New`. Moreover, all the static data **must** be defined within a single **Supplement** element.

Date depending data may be also transmitted in messages with **RatePlanNotifType** set to `Overlay`. See the section "Synchronization" below.

Supplements contribute to the total cost of a stay. The general rule is that this contribution **must** always be added to the amount calculated from the applied rates on a day by day basis. The departure day supplements **must not** be applied, as the guest is leaving.

In case of **ChargeTypeCode** with value `12`, `18` and `20` (see above, all supplements that are *per stay*) the cost of a supplement may vary in the period of the stay. In this case, the total cost of the supplement **must** be calculated using the following algorithm (assuming a 3-night stay with a cost of € 80 for the first two days and € 85 for last two (including the departure day)):

- calculate the number of days where the supplement applies (the supplement must not be applied to the departure day, hence the result is **3**),
- sum the applicable price for each day (80 + 80 + 85 = 245 €),
- divide the result for the number of days obtained at step 1 and round the result at the second decimal place (245 / 3 = 81,67 €).

Offers

Offers are supported through the **Offer** element. AlpineBits only supports offers having a **Discount** element with the **Percent** attribute set to `100`. There are two use cases:

- *free nights offers*, such as "7+1" formulas and the like,
- *family offers*, such as "first kid goes free".

Rate plans may only have at most two **Offer** elements - at most one of each kind. Rate plans are applicable to a stay **only if all offers** that are defined in the rate plan are **applicable**.

A *free nights offer* has a **Discount** element with the following attributes, all **mandatory**:

- **Percent** - value is always `100`.
- **NightsRequired** - how many nights at least must be booked for the discount to apply.
- **NightsDiscounted** - how many nights are discounted (if the stay is n times the required nights, the discounted nights also are n times as many).
- **DiscountPattern** - the pattern is required to be in the form (nights required - nights discounted) times the `0` followed by (nights discounted) times the `1`. No other pattern is allowed.

Note that there is some redundancy in the last three attributes. This is done on purpose for clarity.

Free night offers apply to **every** amount referring to the discounted night (rates as well as per-day or per-night supplements, including mandatory ones).

Free night offers **may be only used** in conjunction with rates that have a **UnitMultiplier** of `1`.

Here is an example of a *free nights offer*:

```
<Offer>
<Discount Percent="100" NightsRequired="7" NightsDiscounted="1" DiscountPattern="0000001"/>
</Offer>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - free nights offer

A *family offer* has a **Discount** element with just the **Percent** attribute set to `100` followed by **at most** one **Guest** element defining who goes free. **Guest** attributes (**all mandatory**) are:

- **AgeQualifyingCode** is set to 8.
- **MaxAge** is an integer value > 0: the discount only applies to guests having age < **MaxAge**.
- **MinCount** is an integer value ≥ 0: it identifies the minimum number of guests having age < **MaxAge** that are required for the offer to be applicable.
- **FirstQualifyingPosition** - always set to `1`,
- **LastQualifyingPosition** - number of persons the discount applies to,

If the attribute **MinCount** is set to 0, the offer (and therefore the rate plan) is applicable also if no guest matches the **MaxAge** criteria, however no guest will receive the discount.

Family offers apply to **every** amount referring to the discounted guest (rates as well as per-person supplements, including mandatory ones).

In case the number of age-matching guests exceeds the number of discounted guests, AlpineBits requires to discount the guests starting from the youngest.

Here is an example of a *family offer*. If at least one child of age < 5 years is present, she will enjoy a free stay. If no child of age < 5 years is present, the offer (and therefore the rate plan) will not be applicable.

```
<Offer>
    <Discount Percent="100"/>
    <Guests>
        <Guest AgeQualifyingCode="8" MaxAge="5" MinCount="1"
               FirstQualifyingPosition="1" LastQualifyingPosition="1" />
    </Guests>
</Offer>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - *family offer*

Here is another example: up to two children, each of age < 6 years will stay free. If no child of age < 6 years is present, the offer (and therefore the rate plan) is still applicable, but nobody will get the discount.

```
<Offer>
    <Discount Percent="100"/>
    <Guests>
        <Guest AgeQualifyingCode="8" MaxAge="6" MinCount="0"
               FirstQualifyingPosition="1" LastQualifyingPosition="2" />
    </Guests>
</Offer>
```

family offer

4.5.2. Computing the cost of a stay

The information contained in a rate plan message (together with information about the stay and the inventory) can be used to compute the total cost of a given stay, provided the stay is possible at all.

The computation is somewhat complex due to the large number of rules involved. The rationale is that the algorithm should be as unambiguous and as top-down as possible. It is never necessary to perform permutations or recursion to find a “optimized solution”. Elements with Start and End attributes, for example, must not overlap. The same is true for age brackets, as we’ve seen. The application of children rebates is very carefully designed to give a unique result and the same holds for offers, etc.

The required steps to perform such a computation are outlined in this section. Also see the section “Implementation tips and best practice” below for a link to a reference implementation.

The following information is needed about the stay:

- the number of adult guests: **n** ≥ 0,
- the ages (in years) of all non-adult guests (that are potentially eligible for children rebates): an array of integers **c**,
- the requested room category (i.e. **InvTypeCode** / **Code**): **code**,
- the arrival date **arr** and the departure date **dep** where **dep** > **arr**.

The total number of guests (**n** + the length of the array **c**) must be > 0.

Note that currently there is no possibility to specify a cut-off age for children rebates in AlpineBits. This is planned for a future release, though.

The following information is needed about the requested room category from inventory (see section 4.4):

- the value of **MinOccupancy** **min** > 0,
- the value of **StandardOccupancy** **std** ≥ **min**,
- the value of **MaxOccupancy** **max** ≥ **std**,
- (optional) the value of **MaxChildOccupancy** **mco**, such that 0 ≤ **mco** ≤ **max**.

From these values the minimum number of guests that ought to pay the full rate (**minfull**) can be computed:

- if **MaxChildOccupancy** is not given, **minfull** = **std**,
- otherwise, **minfull** = maximum(**min**, minimum(**max** - **mco**, **std**)).

Then, to verify that a stay is allowed and to compute its total cost, the following steps need to be performed.

Step 1 (occupancy check)

> Verify that **min** ≤ total number of guests (**n** + the length of the array **c**) ≤ **max**. Unless this inequation holds, the stay in the selected room category is **not** possible and **no** cost can be computed.

Step 2 (transformation)

> While **n** < **minfull** and the length of **c** is > 0, keep removing the greatest element from the array **c** and incrementing **n** by 1. In simple words: transform kids to adults as long as there are any left in an attempt to reach the minimum number of guests that pay the full rate.

Step 3 (family offers)

If there are any matching *family offers*, apply them. When a *family offer* is applied, the corresponding elements from the array **c** are removed. The number of removed elements is referred to as **numfree** below.

Pay attention to the fact that all offers **must** be applicable for a stay to be possible. Hence a rate plan with a family offer that **cannot** be applied (because guests with the required ages are **not** present and MinCount is **not** set to 0) **cannot** be used to satisfy the stay.

Step 4a (restrictions check)

Verify that no **BookingRule** elements impose restrictions that forbid the stay. The restrictions to consider have been detailed earlier in this section (see the section "booking rules"): the minimum/maximum length of stay (LOS), the arrival/departure day of week (DOW), the master restriction status.

Step 4b (compute cost)

Loop over the dates of the period of stay, finding the rate with matching **Start** and **End** values. Thanks to the fact that rates must not overlap, there is no ambiguity there.

It might be necessary, and it is explicitly allowed

- to "stitch" rates together to cover longer stays, accumulating the amount due and/or
- to "split" rates having a **UnitMultiplier** > 1, dividing the amount due by the fraction of nights used.

As said earlier, a stay is only possible if every night of the stay can be covered by a rate.

The detailed implementation for this particular loop-over-and-match-rate step will likely depend on the data model used. However it is important that for each rate, the following sub-steps are performed exactly as described here in order to pick the **correct amount** from the **BaseByGuestAmt** and **AdditionalGuestAmount** elements.

So, for each date and matching rate, consider these contributions to the total cost:

- If there are elements in **c** that have no matching **AdditionalGuestAmount** elements with **AgeQualifyingCode** set to `8` ("child"), i.e. no age brackets match, **only for the evaluation of this specific rate**, those elements are removed from **c** to yield a **c'**. The value of **n** is then incremented accordingly to yield a **n'**. This is the *local* transformation step.
- Out of the **n'** guests, up to and including **std**, **each** pay an amount given by the one **BaseByGuestAmt** element having:
    - a **NumberOfGuests** value of minimum(**n'** + length of **c'** + **numfree** , **std**) if the **Type** value is `7` ("per person") or
    - a **NumberOfGuests** value of minimum(**n'**, **std**) if the **Type** is `25` ("per room").
      If the correct **BaseByGuestAmt** element is not available, the stay as a whole is not possible (the rate plan is incomplete and cannot be applied).

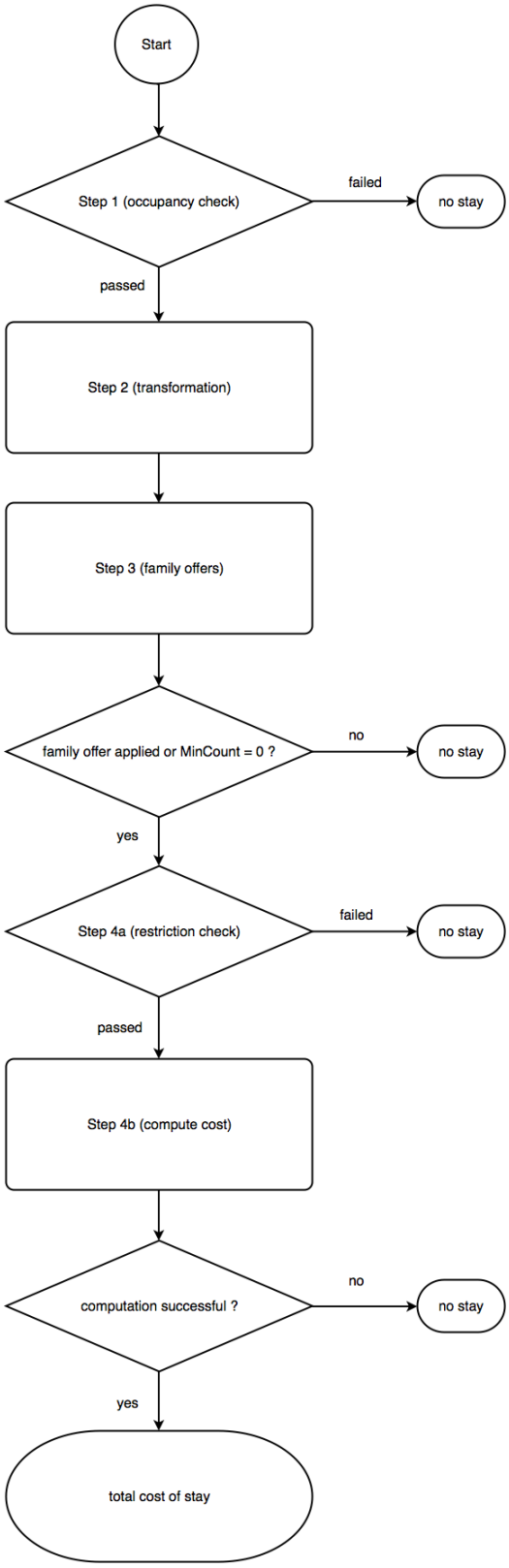
- The remaining guests among the **n'** - if any - **each** pay an amount given by the **AdditionalGuestAmount** elements with **AgeQualifyingCode** set to `10` ("adult").
  If the correct **AdditionalGuestAmount** element is not available, the stay as a whole is not possible (the rate plan is incomplete and cannot be applied).

- Finally, each guest from the array **c'** pays an amount given by the **AdditionalGuestAmount** elements with **AgeQualifyingCode** set to `8` ("child") and the matching age bracket (that we know is present after the local transformation step).

At this point, unless a *free nights* offer applies to the date and rate just considered (note that *free nights* offers are compatible only with rates having a **UnitMultiplier** of `1`), sum the contribution to the total cost.

Next, consider the supplements (mandatory and optional ones). Again, the exact implementation of the supplement matching algorithm will depend very much on the data model used. Note in any case, that *free nights* offers also apply to supplements, so the contribution to the cost from supplements that are per day is affected too.

Here is a flow chart of the process:



4.5.3. Synchronization

Clients and servers often wish to exchange only delta information about rates in order to keep the total amount of data to be processed in check.

AlpineBits uses the **RatePlanNotifType** attribute in each **RatePlan** element to define exactly how deltas have to be interpreted. In order to transmit new rate plans or to replace them **RatePlanNotifType** = `New` **must** be used. To transmit changes (deltas) a **RatePlanNotifType** value of `Overlay` must be used.

Note, however, that

- **RatePlanNotifType** = `New`
  At least one **Description** element **must** be present in the rate plan.The server adds the rate plan as a whole. If a rate plan with the same **RatePlanCode** already exists, it is replaced. In case of supplements all static data must be sent within this message. Offers must be set within this message. The attributes **RatePlanID** and **RatePlanQualifier** - if supported by the server - **might only** be sent within this message.

- **RatePlanNotifType** = `Overlay`
  The server updates the rate plan (identified by **RatePlanCode**) using the received data. Elements that are not transmitted are not touched, elements that are transmitted are completely replaced (including all subelements). Since empty elements replace existing elements, sending empty elements can be a means to delete them (see clarification below). In case of supplements only date depending data may be sent within this message, thus supplements cannot be deleted with an Overlay message, but they may be set as not available. Offers cannot be changed with an Overlay message. In order to update offers, the whole RatePlan has to be sent again (using `New`). If the server has no rate plan with the given **RatePlanCode**, it may ignore the client request but must return a warning if it does.

- **RatePlanNotifType** = `Remove`
  The rate plan **must** be empty (no child elements). The server deletes the rate plan (identified by **RatePlanCode**). If the server has no rate plan with the given **RatePlanCode**, it may ignore the client request but must return a warning in this case.

So, when updating a rate plan, sending empty elements will delete them. Note, however, this only works for the elements with “key” attributes **Start**, **End** and **Code**/**InvTypeCode**: i.e. **BookingRule** and **Rate**. One cannot delete sub-elements (such as only the **LengthOfStay** restriction, or only the **MealsIncluded** element, for example). In particular, if some value needs to be updated in a **Rate** that AlpineBits requires to be the same over the whole rate plan, such as **UnitMultiplier**, **Type** or **MealsIncluded** the whole rate plan **must** be sent again, since only updating one rate would break the assumption that these values must be the same for every rate.

That being said, there is the special case of rate plan messages that contain a **UniqueID** element with attribute **Instance** set to `CompleteSet`.

In this case a client indicates it wishes to initiate sending the complete list of its rate plans. The server **must** then consider expired all rate plans it has on record **that are not contained in the current message** (hint: delete them). In that case, the **RatePlan** element will **not** have any **RatePlanNotifType** and **no** child elements must be present (the **RatePlanCode** must of course be present).

Also regarding the `CompleteSet` case, if the client wishes to reset all rate plans for a given hotel it can send a single empty **RatePlan** element (sending no **RatePlan** element at all would violate OTA validation).

Regarding these synchronization mechanisms, a server **must** support everything except **RatePlanNotifType** = `Overlay`. A server **must** set the corresponding capability if it does.

In order to limit the amount of transferred data and processing time, the following rules and recommendations **must** be taken into account:

- **RatePlanNotifType** = `New`
  Complete RatePlans **must** be transmitted one at a time.
- **RatePlanNotifType** = `Overlay`
  Updates to more than one RatePlan **may** be bundled into a single request. However, care should be taken to keep the data size within reasonable bounds. In case of doubt, the updates should be transmitted for one RatePlan at a time.

4.5.4. Server response

The server will send a response indicating the outcome of the request. The response is an OTA_HotelRatePlanNotifRS document. Four types of outcome are possible: success, advisory, warning or error.

Success

The request was accepted and processed successfully. The client does **not** need to take any further action.

In this case the OTA_HotelRatePlanNotifRS response contains nothing but a **single**, empty **Success** element:

```
<?xml version="1.0" encoding="UTF-8"?>

<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Success/>

</OTA_HotelRatePlanNotifRS>
```

samples/RatePlans-OTA_HotelRatePlanNotifRS-success.xml

Advisory

The request was accepted and processed successfully. However, one or more non-fatal problems were detected and added to the server response. The client does **not** need to resend the request, but **must** notify the user or the client implementer regarding the advisory received.

In this case, the OTA_HotelRatePlanNotifRS response contains an empty **Success** element followed by **one or more Warning** elements with the attribute **Type** set to the fixed value `11`, meaning “Advisory” according to the OTA list “Error Warning Type” (EWT).

Each **Warning** element should contain a human readable text as in the following example:

```
<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Success/>
    <Warnings>
        <Warning Type="11">
            end date is less than 3 days from now
        </Warning>
    </Warnings>

</OTA_HotelRatePlanNotifRS>
```

samples/RatePlans-OTA_HotelRatePlanNotifRS-advisory.xml

Warning

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the warning is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelRatePlanNotifRS response contains an empty **Success** element followed by **one or more Warning** elements with the attribute **Type** set to any value allowed by the OTA list “Error Warning Type” (EWT) **other than** `11` (“Advisory”).

Each **`Warning`** element should contain a human readable text as in the following example:

```
<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Success/>
    <Warnings>
        <Warning Type="3">
            dates are too far in the future for this server to process
        </Warning>
    </Warnings>

</OTA_HotelRatePlanNotifRS>
```

samples/RatePlans-OTA_HotelRatePlanNotifRS-warning.xml

Error

The request **could not** be accepted or processed successfully.

The client **must** take action: it **may** try to resend the message if it has reason to assume the error is transient and occurred for the first time, otherwise it **must** escalate the problem to the user or client implementer.

In this case, the OTA_HotelAvailNotifRS response contains **one or more Error** elements with the attribute **Type** set to the fixed value of `13`, meaning “Application error” according to the OTA list “Error Warning Type” (EWT) and the attribute **Code** set to any value present in the OTA list “Error Codes” (ERR).

```
<OTA_HotelRatePlanNotifRS
    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_HotelRatePlanNotifRS.xsd"
    Version="3.14">

    <Errors>
        <Error Type="13" Code="404">
            Invalid start/end date combination
        </Error>
    </Errors>

</OTA_HotelRatePlanNotifRS>
```

samples/RatePlans-OTA_HotelRatePlanNotifRS-error.xml

4.5.4. Implementation tips and best practice

Synchronization tip.

A server, before declaring support for the capability `OTA_HotelRatePlanNotif_accept_RatePlan_mixed_BookingRule` **must** ensure that an update to a generic booking rule has no impact on existing specific rules.

About Supplements.

AlpineBits supplements allow for the following use cases:

- exchange of included, mandatory or optional supplements
- exchange of multi-language descriptions of supplements with title and short text (“intro”)
- exchange of date depending prices
- examples: cleaning fees, parking, New Year's Eve dinner

AlpineBits supplements don’t currently allow for the following use cases (these will be addressed in a future release of AlpineBits, therefore it's possible that substantial modifications will be made):

- supplements available
- exchange of images
- categorization of supplements

Other things to note about supplements:

- AlpineBits **does not** allow the child elements **RoomCompanions** or **PrerequisiteInventory**
- a rate plan **must** *describe* any local taxes and fees in its description (as opposed to giving them as a supplement). The rationale behind this is that a portal does not have enough information to decide whether these local taxes and fees apply to a given booking request or not

Reference Implementation.

A reference implementation for the computation of the cost of a stay is available at https://development.alpinebits.org/#/rtapp. This implementation can be used manually (by using the website) or automatically (by sending data to a web service). The implementation can also be run from the command line, source code is available.

Please let the AlpineBits Alliance know if you find a discrepancy between this document and the reference implementation.

If there is a discrepancy the following rules can be used to solve it:

- if the document is clear, the document prevails,
- if the document is ambiguous, the reference implementation prevails.

A. AlpineBits developer resources

The AlpineBits development home page is at http://www.alpinebits.org/developers/. There are resources linked from that page that help test one's implementation.

Public repositories with schema files and example code snippets are available online at https://github.com/alpinebits/. Contributions are welcome (any programming language).

B. Protocol Version Compatibility

B.1. Minor updates in version 2015-07b

Version 2015-07b is a maintenance release. The section 4.5 about rate plans has been mostly rewritten with more precise and strict information about how to handle corner cases, especially regarding rebates.

While most of this does not lead to breaking changes *per se*, it is likely that 2015-07 servers that were implemented before the release of 2015-07b would compute costs for stays differently, for lack of a sufficiently strict description of details.

One deliberate breaking change of note is that 2015-07b requests the value of the **AmountAfterTax** attribute in **BaseByGuestAmt** elements to be **> 0**, while 2015-07 used to allow a value **≥ 0**.

B.2. Major overhaul in version 2015-07

Inventory

In version 2015-07 the Inventory message was changed from `OTA_HotelInvNotif` to `OTA_HotelDescriptiveContentNotif`. The new message offers the same options as the one used previously beside sending the name of the specific rooms, but allows for much richer descriptions, including pictures. A high-level mapping between the old Inventory and the new one is as follows:

`OTA_HotelInvNotif`	`OTA_HotelDescriptiveContentNotif`
SellableProduct	**GuestRoom**
SellableProduct InvTypeCode	**GuestRoom Code**
SellableProduct InvCode	**TypeRoom RoomID**
Quantities MaximumAdditionalGuests	*Not needed anymore, see StandardOccupancy*
Occupancy MinOccupancy	**GuestRoom MinOccupancy**
Occupancy MaxOccupancy	**GuestRoom MaxOccupancy**
Occupancy AgeQualifyingCode`="8"`	**GuestRoom MaxChildOccupancy**
Not previously possible	**TypeRoom StandardOccupancy**
Room RoomClassificationCode	**TypeRoom RoomClassificationCode**
Amenity AmenityCode	**Amenity RoomAmenityCode**
Text	**TextItem > Description**
Not previously possible	**ImageItem > URL**
Text (*for specific Room*)	*Not possible anymore*

B.3. Major overhaul in version 2014-04

Version 2014-04 was a major overhaul. In most cases, a pre-2014-04 client will not be compatible with a 2014-04 server and viceversa. Here is a list of major changes in 2014-04.

HTTPS layer

The possibility of compression with gzip has been added.

FreeRooms

The possibility to send booking restrictions in FreeRooms has been removed as have the corresponding capabilities. These are better handled by the new RatePlans.

The value of the action parameter has been changed from `FreeRooms` to `OTA_HotelAvailNotif:FreeRooms` for uniformity with the other action values that all follow the `rootElement:actionValue` format.

The possible responses (OTA_HotelAvailNotifRS document) have been re-categorized into four classes: success, advisory (new), warning and error. For error responses the attributes have changed, fixing a bad OTA interpretation.

Finally, the way deltas and complete transmissions are distinguished has changed.
All in all FreeRooms are not compatible with any previous version.

GuestRequests

GuestRequests have been heavily refactored. Previous AlpineBits versions had two type of requests: quotes and booking requests, the current version has three: booking reservations, quote requests and booking cancellations. Also, the client can (and must) now send acknowledgements.

SimplePackages

The possible responses (OTA_HotelRatePlanNotifRS document) have been re-categorized into four classes: success, advisory (new), warning and error. For error responses the attributes have changed, fixing a bad OTA interpretation.

Inventory and RatePlans

These are new message types introduced with version 2014-04.

B.4. Compatibility between a 2012-05b client and a 2013-04 server

Housekeeping

The client will not send the X-AlpineBits-ClientID field in the HTTP header, since it is not aware of this feature. This will cause authentication problems with those 2013-04 servers that require an ID.

The client will not send the X-AlpineBits-ClientProtocolVersion field in the HTTP header, since it is not aware of this feature. This is no problem: a server that is interested in this, will simply recognize the client as preceding protocol version 2013-04.

If the client checks the server version it will see 2013-04 - a version it doesn't recognize. Likewise, if the client checks the capabilities it might see the `OTA_HotelAvailNotif_accept_deltas` capability. Client implementers interested to have their 2012-05b client talk to a 2013-04 server should verify this is not a problem for their client software.

FreeRooms

There is no compatibility problem in the request part: the client will not send partial information (deltas), since it is simply not aware of the existence of the feature.
Please be aware that the lack of this feature (obviously) causes more data to be sent to the server, something not all companies that run servers will be happy with.
The server response might contain a **`Warning`** element the client cannot process. If the client - as it should - carefully parses the response, it will treat this as an error situation and act accordingly. So basically the client is expected to treat the warnings as error, which might be an issue and this should be tested

GuestRequests

No compatibility problems are expected.

SimplePackages

2013-04 introduced the limitation that all packages sent within a single request must refer to the same hotel. An older client not aware of this limit might incur an error returned from the server error.

Similar to the FreeRooms case, the server response might contain a **`Warning`** element the client cannot process. If the client - as it should - carefully parses the response, it will treat this as an error situation and act accordingly. So basically the client is expected to treat the warnings as an error, which might be an issue and should be tested for.

B.5. Compatibility between a 2013-04 client and a 2012-05b server

Housekeeping

The client sends the X-AlpineBits-ClientProtocolVersion field and may send the X-AlpineBits-ClientID field in the HTTP header, but the server will just ignore it - being unaware of the feature.
If the client checks the server version and/or checks the capabilities - as it should - it will note the missing features and not use them.
Hence, technically there is no problem with this combination.

FreeRooms

The client will not send partial information (deltas), since the server does not export the `OTA_HotelAvailNotif_accept_deltas` capability.
The server will never send a response with a **`Warning`** element. This is not a problem for the client.

GuestRequests

In 2013-04 the form of the `ID` attribute is not any more restricted. It has become a free text field. An older server might insist on the old form and throw an error.

SimplePackages

The server will never send a response with a `Warning` element. This is not a problem for the client.

C. Links

[0] **Creative Commons BY ND license:**
http://creativecommons.org/licenses/by-nd/3.0/

[1] **HTTP basic access authentication:**
http://en.wikipedia.org/wiki/Basic_access_authentication

[2] **OpenTravel Alliance:**
http://www.opentravel.org/

[3] **OTA2015A documentation package download:**
http://www.opentravel.org/Specifications/ReleaseNotes.aspx?spec=2015A

[4] **OTA2015A XML schema files online:**
http://www.opentravel.org/Specifications/SchemaIndex.aspx?FolderName=2015A

[5] **browsable interface to the above schema files:**
http://adriatic.pilotfish-net.com/ota-modelviewer/