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

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
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
Internet Service GmbH - www.internetservice.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
XLbit snc - www.xlbit.com

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 www.alpinebits.org.

protocol version	doc. release date	description
2017-10	2017-11-08	Status: • official release Notable documentation changes: • **The AlpineBits® documentation is now released under the Creative Commons Attribution-ShareAlike 3.0 Unported License** • The attached Trademark policy is now an integral part of the AlpineBits® specifications Updates and additions: • Chapter 2: added version handshake best practices • FreeRooms: added transmission of number of bookable rooms, added purge of stale data • Inventory: heavily updated and refactored • GuestRequests: added Commission element, added encrypted credit card numbers and made card holder name optional, added hints on how to fill ReservationID • RatePlans: added static rates, added supplements that are only available on given days of the week and supplements that depend on room category, improved and extended offers, added SetForwardMinStay and SetForwardMaxStay, extended descriptions and improved their documentation • new action: BaseRates • added general support of HTML in descriptions as an additional format (with some warnings) • minor textual improvements Reorganization of the appendixes: • Appendix A: common section about server response outcomes • Appendix C: compatibility to previous versions described • Appendix D: updated links to OTA2015A standard resources Removals: • removed SimplePackages
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 • schema 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 appendix C.4 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 FreeRooms **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 B: 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 C: 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),
- 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 B for more resources.

- 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 ten, therefore it was introduced in AlpineBits® even though its usage in POST requests isn't very common (as opposed to responses where its use is widespread). According to the various RFCs, compressing the requests is not forbidden, but there are no implementation recommendations. It was decided to use an approach major web servers were already supporting at the time of this writing.

- When negotiating an AlpineBits® version, clients and servers should behave in the following way (this works starting from 2013-04):
  - Client: the client queries the server version, sending a header with the highest version it supports.
  - Server: if the server supports this same version, it answers with this version and the negotiation terminates successfully; otherwise the server answers with the highest version it supports.
  - Client: if the client recognizes the server version, it starts using the server version and the negotiation terminates successfully; otherwise no communication is possible, since the two parties don't share a common version.

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

- `OTA_HotelAvailNotif_accept_BookingThreshold`
  for room availability notifications (FreeRooms), the server accepts the number of rooms that are considered free but not bookable

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

- `action_OTA_HotelDescriptiveContentNotif_Inventory`
  the server implements handling Inventory/Basic (push)

- `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

- `action_OTA_HotelDescriptiveContentNotif_Info`
  the server implements handling Inventory/HotelInfo (push)

- `action_OTA_HotelDescriptiveInfo_Inventory`
  the server implements handling Inventory/Basic (pull)

- `action_OTA_HotelDescriptiveInfo_Info`
  the server implements handling Inventory/HotelInfo (pull)

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

- `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

- `OTA_HotelRatePlanNotif_accept_OfferRule_BookingOffset`
  for prices (RatePlans), the server accepts the OfferRule restrictions MinAdvancedBookingOffset and MaxAdvancedBookingOffset

- `OTA_HotelRatePlanNotif_accept_OfferRule_DOWLOS`
  for prices (RatePlans), the server accepts the OfferRule restrictions ArrivalDaysOfWeek, DepartureDaysOfWeek, SetMinLOS and SetMaxLOS

- `action_OTA_HotelRatePlan_BaseRates`
  the server implements handling BaseRates

- `OTA_HotelRatePlan_BaseRates_deltas`
  the server supports delta information with BaseRates

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.

For data exchange actions, both parameters (`action` and `request`) are mandatory.

The value of the `request` parameter (sent by the client) and the server response are XML documents following OTA2015A and using the XML root elements specified in the following table:

known as (since)	**usage**	**parameter** `action` (string)	**parameter** `request` and **server response** (XML documents)
FreeRooms (since 2011-11)	a client sends room availability notifications to a server	OTA_HotelAvailNotif:FreeRooms	OTA_HotelAvailNotifRQ OTA_HotelAvailNotifRS
GuestRequests (since 2012-05)	a client sends a request to receive requests for a quote or booking requests from the server	OTA_Read:GuestRequests	OTA_ReadRQ OTA_ResRetrieveRS
GuestRequests/ Acknowledgments (since 2014-04)	a client acknowledges the requests it has received	OTA_NotifReport:GuestRequests	OTA_NotifReportRQ OTA_NotifReportRS
SimplePackages (2012-05 until 2015-07b)	this action has been removed in version 2017-10		
Inventory/Basic (Push) (since 2015-07)	a client sends room category information and room lists	OTA_HotelDescriptiveContentNotif:Inventory	OTA_HotelDescriptiveContentNotifRQ OTA_HotelDescriptiveContentNotifRS
Inventory/Basic (Pull) (since 2017-10)	a client requests room category information and room lists	OTA_HotelDescriptiveInfo:Inventory	OTA_HotelDescriptiveInfoRQ OTA_HotelDescriptiveInfoRS
Inventory/HotelInfo (Push) (since 2015-07)	a client sends additional descriptive content regarding room categories	OTA_HotelDescriptiveContentNotif:Info	OTA_HotelDescriptiveContentNotifRQ OTA_HotelDescriptiveContentNotifRS
Inventory/HotelInfo (Pull) (since 2017-10)	a client requests additional descriptive content regarding room categories	OTA_HotelDescriptiveInfo:Info	OTA_HotelDescriptiveInfoRQ OTA_HotelDescriptiveInfoRS
RatePlans (since 2014-04)	a client sends information about rate plans with prices and booking rules	OTA_HotelRatePlanNotif:RatePlans	OTA_HotelRatePlanNotifRQ OTA_HotelRatePlanNotifRS
BaseRates (since 2017-10)	a client requests information about rate plans	OTA_HotelRatePlan:BaseRates	OTA_HotelRatePlanRQ OTA_HotelRatePlanRS

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, two AlpineBits® schema files (one XML schema file and one Relax NG schema file) are provided as an integral part of the specification in the AlpineBits® documentation kit.

The Relax NG file is somewhat stricter than the XML schema file as RelaxNG is intrinsically more powerful in expressing constraints that express how elements and attributes depend on each other.

Both AlpineBits® schema 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` and **Type** = `16` indicates that this message contains the complete information (as entered by the user). When receiving such a request, a server **must** remove all information about any availability it might have on record regarding the given hotel.

The attribute **ID** is needed for compatibility with OTA, the value is ignored by AlpineBits.

A client might want to let a server know its availability data should be purged based on internal business rules. An example might be availability data that is considered stale, because it hasn't been updated by the user for some time. The client then **should** send a request using an **UniqueID** element with attribute **Instance** = `CompleteSet` and **Type** = `35`. A server **must** accept this value (without returning an error or warning) and **could** make use of this hint and keep the availability data it has on record marking it as purged. Otherwise such a message should be considered equivalent to the one with **Type** = `16`.

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"
                    BookingThreshold="0">

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

</AvailStatusMessage>

<AvailStatusMessage BookingLimit="1"
                    BookingLimitMessageType="SetLimit"
                    BookingThreshold="0">

    <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).

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).

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.

The integer value of the attribute `BookingLimit` indicates the number of available rooms. Since in the example we're dealing with a specific room here (`InvCode` is 101S), 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.

A server **may** support handling rooms that are considered **free but not bookable** (see section GuestRequests for the description of bookings) and **must** set the `OTA_HotelAvailNotif_accept_BookingThreshold` capability accordingly.

If the capability is set, a client **must** send the `BookingThreshold` attribute in the `AvailStatusMessage` element. A client that is only sending bookable rooms **must** set `BookingThreshold` to 0.

The number of bookable rooms are hence given by `BookingLimit` - `BookingThreshold`.

Since overbooking is not allowed and the number of bookable rooms cannot exceed the number of free rooms the inequality 0 ≤ `BookingThreshold` ≤ `BookingLimit` holds.

If the server does not have the capability set, a client **must not** send the `BookingThreshold` attribute in the `AvailStatusMessage` element, and all the rooms are implicitly considered bookable.

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.

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. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed.

See Appendix A for details.

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 day **after** the `End` date.
- 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.

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 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 will send a response indicating the outcome of the request. The response is a OTA_ResRetrieveRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed.

See Appendix A for details.

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">
                <Commission Percent="15"/>
                <!-- 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**),
    - **zero or one** Commission elements with **either** a Percent attribute (the commission in percent) or a CommissionPayableAmount sub-element with attributes Amount and CurrencyCode,
    - **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)** with:
    - the **mandatory** attributes CardCode (the card issuer, two uppercase letters, such as "VI" for Visa) and ExpireDate (four digits),
    - the **optional** sub-element CardHolderName,
    - the **mandatory** sub-element CardNumber (see below);
- **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.

The CardNumber element is used to send a credit card number. The number can be send **either as plain text** (but note the message transport is encrypted using HTTPS) **or encrypted**.

For the plain text case, the CardNumber element has **no** attributes and **must** contain a PlainText sub-element holding the complete credit card number or its last four digits as in the example above.

For the encrypted case, the CardNumber element **must not** have any sub-elements and **must** have attributes EncryptedValue (a string representation of the encrypted card number) and EncryptionMethod (a string identifying the encryption method, e.g. "RSA-PKCSV2.2").

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** sub elements **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.

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.

Here is an example where a client refuses one reservation request and acknowledges three other requests.

```
<?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 send a response indicating the outcome of the request. The response is a OTA_NotifReportRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed.

See Appendix A for details.

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.

Some guidelines on how to fill the fields in the `HotelReservationID` and `Profile` / `CompanyInfo` elements are provided here in order to allow easier grouping of the data coming from different servers.

The attributes of the `HotelReservationID` attributes can be mapped to the attributes used for internet marketing like in the following table:

utm_source	↔	`ResID_Source`
utm_medium	↔	`ResID_SourceContext`
utm_content	↔	not present in AlpineBits
utm_campaign	↔	`ResID_Value`

We can then distinguish different use cases - here are some examples:

1. Request or reservation via homepage

a) Direct (user types domain into browser address bar)
`ResID_Source` (utm_source) = direct
`ResID_SourceContext` (utm_medium) =
`ResID_Value` (utm_campaign) =
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

b) Referral link to hotel site from other website
`ResID_Source` (utm_source) = otherwebsite.com
`ResID_SourceContext` (utm_medium) = referral
`ResID_Value` (utm_campaign) =
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

c) Referral link to hotel site from portal website with campaign infos
`ResID_Source` (utm_source) = portalwebsite.com
`ResID_SourceContext` (utm_medium) = referral (or exact medium type of portal, e.g. "banner", "membership", "toplinks")
`ResID_Value` (utm_campaign) = [campaign-name]
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

d) Search engine organic
`ResID_Source` (utm_source) = google.com
`ResID_SourceContext` (utm_medium) = organic_search
`ResID_Value` (utm_campaign) =
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

e) Search engine paid (e.g. Google Adwords)
`ResID_Source` (utm_source) = google.com
`ResID_SourceContext` (utm_medium) = cpc
`ResID_Value` (utm_campaign) = [campaign-name]
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

f) Social media content link (e.g. Facebook)
`ResID_Source` (utm_source) = facebook.com
`ResID_SourceContext` (utm_medium) = social_media
`ResID_Value` (utm_campaign) = [facebook-post]
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

g) Social media paid (e.g. Facebook Ads)
`ResID_Source` (utm_source) = facebook.com
`ResID_SourceContext` (utm_medium) = cpc
`ResID_Value` (utm_campaign) = [campaign-name]
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

h) Newsletter
`ResID_Source` (utm_source) = newsletter-[company-name]
`ResID_SourceContext` (utm_medium) = email
`ResID_Value` (utm_campaign) = [newsletter-name] (e.g. NL 09/2017)
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

i) QR code in printed magazine
`ResID_Source` (utm_source) = [magazine-name] (e.g. Bergwelten)
`ResID_SourceContext` (utm_medium) = qr_code
`ResID_Value` (utm_campaign) = [magazine-name-and-issue] (e.g. Bergwelten 08/2017)
`Profile` / `CompanyInfo` (travel agent / booking channel) = homepage

<u>2. Request or reservation via landing page</u>
a) Bookings/enquiries on landing pages from Google Adwords
`ResID_Source` (utm_source) = google.com
`ResID_SourceContext` (utm_medium) = cpc
`ResID_Value` (utm_campaign) = [campaign-name]
`Profile` / `CompanyInfo` (travel agent / booking channel) = landingpage

b) Bookings/enquiries on landing pages from portal links
`ResID_Source` (utm_source) = portalwebsite.com
`ResID_SourceContext` (utm_medium) = referral (or exact medium type of portal, e.g. "banner", "membership", "toplinks")
`ResID_Value` (utm_campaign) = [campaign-name]
`Profile` / `CompanyInfo` (travel agent / booking channel) = landingpage

<u>3. Reservation on a booking platform</u>
a) Bookings on booking portal (e.g. Booking.com)
`ResID_Source` (utm_source) =
`ResID_SourceContext` (utm_medium) =
`ResID_Value` (utm_campaign) =
`Profile` / `CompanyInfo` (travel agent / booking channel) = [booking portal name] (e.g. booking.com)

b) Bookings/enquiries on portals (e.g. suedtirolerland.it)
`ResID_Source` (utm_source) = [portal name] (e.g. suedtirolerland.it)
`ResID_SourceContext` (utm_medium) =
`ResID_Value` (utm_campaign) =
`Profile` / `CompanyInfo` (travel agent / booking channel) = [portal operator] (e.g. Peer.travel)

4.3. SimplePackages: package availability notifications (removed)

This feature has been removed with version 2017-10.

SimplePackages can be still used by servers and clients supporting previous versions of the standard (from 2012-05 to 2015-07b).

4.4. Inventory: room category information

The Inventory request allows up to four values for the `action` parameter (depending on the server exposed capabilities):

- `OTA_HotelDescriptiveContentNotif:Inventory` indicates the client wishes to define a room category, send its ***basic*** description and optionally provide a list of specific rooms for each category (see section **Inventory/Basic (push)**)
- `OTA_HotelDescriptiveInfo:Inventory` indicates the client wishes to **retrieve** the information about room category, basic description and the list of specific rooms (see section **Inventory/Basic (pull)**)

- `OTA_HotelDescriptiveContentNotif:Info` indicates the client wishes to **send** additional descriptive content (see section **Inventory/HotelInfo (push)**)
- `OTA_HotelDescriptiveInfo:Info` indicates the client wishes to **retrieve** the additional descriptive content (see section **Inventory/HotelInfo (pull)**)

4.4.1. Inventory/Basic (push) client request

The parameter `request` contains an OTA_HotelDescriptiveContentNotifRQ document.

Each document contains **one** **HotelDescriptiveContent** element. For the 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 **HotelDescriptiveContent**, a **FacilityInfo** element contains a list of zero or more **GuestRoom** elements. There are two distinct kinds of **GuestRoom** elements:

- the **heading** one is used to define a room category and its ***basic*** description (the category is identified by the attribute **Code**),

- the **following** ones list specific rooms for each category (identified by the attribute **Code**).

More than one category can be transmitted. That means a category defining sequence of one-heading-**GuestRoom**-element plus zero or more following-**GuestRoom**-elements can be repeated for each category.

Sending zero **GuestRoom** elements (meaning an empty **GuestRooms** element) will remove all room categories for the given hotel.

Following is the global structure of an example of such a document:

```
<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>

        <!-- the heading GuestRoom element is used to define a category
             and its basic description -->

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

        <!-- the follow-up GuestRoom elements list the specific
             rooms in the category -->

        <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 heading **GuestRoom** element, used to define a room category and its ***basic*** description, 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.
- **ID**: **optional**, used for updating room category **codes** (see remarks at the end of this section)

The heading **GuestRoom** element also contains the following sub-elements:

- **TypeRoom**: **mandatory** element with the attributes:
  - **StandardOccupancy** (mandatory),
  - **RoomClassificationCode** (mandatory) to classify the kind of guest room following the OTA list “Guest Room Info” (GRI) - 42 means just “Room”, 13 means “Apartments”, etc...
  - **Size** (optional)
- **Amenities: optional** element containing a list of **Amenity** elements, each indentified by the **mandatory** attribute **RoomAmenityCode** following the OTA list "Room Amenity Type" (RMA).

- **MultimediaDescription**: one or more elements with an **InfoCode** attribute with certain values (a subset of the OTA list "Information type" (INF)):
  - **exactly one MultimediaDescription** element with attribute **InfoCode** = `25` (Long name) indicating a title **must** be present,
  - **at most one MultimediaDescription** element with attribute **InfoCode** = `1` (Description) **may be** present and
  - **at most one MultimediaDescription** element with attribute **InfoCode** = `23` (Pictures) **may be** present.

Here is an example of such a heading **GuestRoom** element:

```
<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 heading GuestRoom element

The **MultimediaDescription** elements follow these rules:

- a **MultimediaDescription** element with attribute **InfoCode** = `25` or `1` **must** contain only a single **TextItem** element
- a **MultimediaDescription** element with attribute **InfoCode** = `23` contains **only ImageItem** elements (**one or more).**

Here is an example:

```
<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 - MultimediaDescription elements

TextItems contains a **TextItem** element, which **must** contain one or more **Description** elements, each with the **mandatory** attributes **TextFormat** and **Language**.

The following rules hold:

- The attribute **TextFormat** is set to `PlainText` or `HTML` and the attribute **Language** is 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 **TextItem** element with `TextFormat` set to `HTML` is intended as rich text alternative of a **TextItem** 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 **TextItem** elements with `TextFormat` set to `HTML` is **not** recommended but left as an option for implementers that absolutely need it.

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 **following GuestRoom** element lists all the rooms belonging to a category, as we've seen in the first code sample, repeated here:

```
<!-- the follow-up GuestRoom elements list the specific
     rooms in the category -->

<GuestRoom Code="DZ">
  <TypeRoom RoomID="101"/>
</GuestRoom>

<GuestRoom Code="DZ">
  <TypeRoom RoomID="102"/>
</GuestRoom>
```

`samples/Inventory-OTA_HotelDescriptiveContentNotifRQ.xml` - part showing follow-up GuestRoom elements

Following GuestRoom elements have a mandatory **Code** attribute that identifies the category the specific room belongs to. **No further attributes** are allowed.

They also have a mandatory sub-element **TypeRoom** with a **mandatory** attribute **RoomID** that identifies the specific room. **No** further elements or attributes are **allowed**, included (but not limited to) the **MultimediaDescription** element.

If a server sets the capability `OTA_HotelDescriptiveContentNotif_Inventory_use_rooms`, a client **must** send the full list of rooms and the server **must** store it.

Otherwise, if a server does not set that capability, a client might or might not send the room list. A server that has no use for the room list is then free to discard it upon receiving them, but **must** do so silently (i.e. without returning errors or warnings).

Some remarks.

Note that AlpineBits® does not support deltas for Inventory/Basic (push) requests. After successfully processing a request, any previously data sent via an Inventory/Basic (push) request (and **only** the data sent via an Inventory/Basic (push) request) must be removed.

Moreover, all Inventory/HotelInfo, FreeRooms or RatePlans data that refer to any now missing room categories must be considered outdated.

Also 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).

It happens that hotels change room category codes (for whatever reason). It is therefore important that these changes are communicated to the server in order to avoid losing any data linked to the renamed categories. The old code of the room category can be transferred via the attribute **ID** of the heading **GuestRoom** elements.

Here is an example:

```
<HotelDescriptiveContent HotelCode="123" HotelName="Frangart Inn">
  <FacilityInfo>
    <GuestRooms>

      <!-- "single", formerly known as "EZ" →

      <GuestRoom Code="single" MaxOccupancy="2" MinOccupancy="1" ID="EZ">
        <TypeRoom StandardOccupancy="2" RoomClassificationCode="42"/>
      </GuestRoom>

      <!-- "double", formerly known as "DZ" →

      <GuestRoom Code="double" MaxOccupancy="2" MinOccupancy="1" ID="DZ">
        <TypeRoom StandardOccupancy="2" RoomClassificationCode="42"/>
      </GuestRoom>

     </GuestRooms>
    </FacilityInfo>
  </HotelDescriptiveContent>
```

The server **will** first look for a category “single”. If that’s found, the server **will** silently ignore the ID attribute and replace the data it had on record for “single”.

If “single” is not found, the server will look for a category “EZ”. If that’s found, the server **must** rename it to “single”. If “EZ” is not found either, the server will just store the new category “single”.

This makes sure that “EZ” is renamed to “single”, without causing problems if “EZ” wasn’t known in the first place or in case the renaming has already happened.

The same procedure is to be applied for “DZ” renamed to “double”.

4.4.2. Inventory/Basic (push) server response

The server will send a response indicating the outcome of the request. The response is a OTA_HotelDescriptiveContentNotifRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed.

See Appendix A for details.

4.4.3. Inventory/Basic (pull) client request

The parameter `request` contains an OTA_HotelDescriptiveInfoRQ document.

Analogous to the Inventory/Basic (push) request, a client can send a pull request to **query** the **basic** data the server has stored about a hotel.
Here is such a client request:

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

    <HotelDescriptiveInfos>
        <HotelDescriptiveInfo HotelCode="123" HotelName="Frangart Inn"/>
    </HotelDescriptiveInfos>

</OTA_HotelDescriptiveInfoRQ>
```

samples/Inventory-OTA_HotelDescriptiveInfoRQ-basicpullRQ.xml

As expected, the document must contain **one** **HotelDescriptiveInfo** element with attributes **HotelCode** and **HotelName** that follow the same rules as for room availability notifications (section 4.1.1). Again, information about only one hotel per message can be queried.

Please note that the **content** of the XML message is **identical both** for Inventory/Basic (pull) and Inventory/HotelInfo (pull), but the `action` is used by the server to distinguish the two requests.

4.4.4. Inventory/Basic (pull) server response

The server will send a response indicating the outcome of the request. The response is a OTA_HotelDescriptiveInfoRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed. See Appendix A for details.

In case of a successful outcome, the **Success** element is followed by a **HotelDescriptiveContent** element with the information about the hotel.
Here is such a server response. The complete file is in the developer kit.

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

  <Success/>
  <HotelDescriptiveContents>
    <HotelDescriptiveContent HotelCode="123" HotelName="Frangart Inn">

    <!-- ... see file in development kit ... -->

    </HotelDescriptiveContents>
  </HotelDescriptiveContents>


</OTA_HotelDescriptiveInfoRS>
```

samples/Inventory-OTA_HotelDescriptiveInfoRS-basicpullRS.xml - outer part

4.4.5. Inventory/HotelInfo (push) client request

The parameter `request` contains an OTA_HotelDescriptiveContentNotifRQ document.

The purpose of this request is twofold:

1. A given client could have information that is technically fundamental (data transmitted via Inventory/Basic (push)), but may lack **additional** content such as inspiring pictures of the rooms. The present Inventory/HotelInfo (push) request introduces the 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.)

2. Moreover many AlpineBits® members have expressed the need to transmit all kind of **additional** master data about hotels, such as (but not limited to):

    - hotel type / category
    - address info
    - contact info
    - short/long descriptions
    - logos, seasonal pictures, gallery pictures
    - hotel amenities
    - check-in / check-out times
    - accepted payment methods / deposit
    - ratings info
    - facilities info

Using the Inventory/HotelInfo (push) request it is therefore possible to send **additional data** that is allowed by OTA under the **HotelDescriptiveContent** element.

Each message **must** contain **one** **HotelDescriptiveContent** element with attributes **HotelCode** and **HotelName** that follow the same rules as for room availability notifications (section 4.1.1). Again, information about only one hotel per message can be transmitted.

AlpineBits® allows the following child-element right below **HotelDescriptiveContent**:

- **HotelInfo**: can hold **any** content allowed by OTA,
- **FacilityInfo**: is a subset of the **FacilityInfo** in **Inventory/Basic (push)** messages:
    - **GuestRoom** elements may **only** have a **Code** attribute and may **only** contain **MultimediaDescription** sub-elements;
    - the **MultimediaDescription** elements **must not** have the **InfoCode** attribute and just contain a single **ImageItems** element with a list of **ImageItem** elements
- **Policies**: can hold **any** content allowed by OTA,
- **AffiliationInfo**: can hold **any** content allowed by OTA,
- **ContactInfos**: can hold **any** content allowed by OTA.

The restriction on **FacilityInfo** is meant to make sure that **Inventory/HotelInfo (push)** message are **only** used to send **additional** descriptive content for room categories. It is supposed to be used **in addition** (**not instead of**) **Inventory/Basic (push)**.

Given the somewhat experimental nature of the **Inventory/HotelInfo (push)** request type, future versions of AlpineBits® **might further restrict or clarify** the acceptable sub-elements of **HotelDescriptiveContent**.

The development kit contains an example in the samples folder that documents **only** the **FacilityInfo** element; a non-normative example of a possible usage of all the other nodes mentioned above is available in the samples-contrib folder and is not fully copied here due to the limited space:

```
<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" >

      <!-- ... see file in development kit ... -->

    </HotelDescriptiveContent>
  </HotelDescriptiveContents>

</OTA_HotelDescriptiveContentNotifRQ>
```

samples/Inventory-OTA_HotelDescriptiveContentNotifRQ-hotelinfo.xml - outer part

Note that AlpineBits® does not support deltas for Inventory/HotelInfo (push) requests. Data sent in a Inventory/HotelInfo (push) request replaces all data sent in a previous Inventory/HotelInfo (push) request (although the server will of course **not** replace any Inventory/Basic (push) data it has on record). If multiple clients are providing Inventory/HotelInfo (push) data, it is up to the server implementation to guarantee the integrity of the data.

If a server receives Inventory/HotelInfo (push) data with additional descriptive content regarding unknown room categories it must react accordingly (i.e. return a warning). However, a message that has additional descriptive regarding only **some** of the categories the server has on record, will **not** trigger a warning response. It is very well possible that some room categories do not have **additional** descriptive content.

4.4.6. Inventory/HotelInfo (push) server response

The server will send a response indicating the outcome of the request. The response is a OTA_HotelDescriptiveContentNotifRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed.

See Appendix A for details.

4.4.7. Inventory/HotelInfo (pull) client request

The parameter `request` contains an OTA_HotelDescriptiveInfoRQ document.

Analogous to the Inventory/HotelInfo (push) request, a client can send a pull request to **query** the **additional** data the server has stored about a hotel.

Here is such a client request:

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

    <HotelDescriptiveInfos>
        <HotelDescriptiveInfo HotelCode="123" HotelName="Frangart Inn"/>
    </HotelDescriptiveInfos>

</OTA_HotelDescriptiveInfoRQ>
```

samples/Inventory-OTA_HotelDescriptiveInfoRQ-hotelinfo.xml

As expected, the document must contain **one HotelDescriptiveInfo** element with attributes **HotelCode** and **HotelName** that follow the same rules as for room availability notifications (section 4.1.1). Again, information about only one hotel per message can be queried.

Please note that the **content** of the XML message is **identical both** for Inventory/Basic (pull) and Inventory/HotelInfo (pull), but the `action` is different and used to distinguish the two requests.

4.4.8. Inventory/HotelInfo (pull) server response

The server will send a response indicating the outcome of the request. The response is a OTA_HotelDescriptiveInfoRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed. See Appendix A for details.

In case of a successful outcome, the **Success** element is followed by a **HotelDescriptiveContent** element with the information about the hotel.

The development kit contains an example in the samples folder that documents **only** the **FacilityInfo** element; a non-normative example of a possible usage of all the other nodes mentioned above is available in the samples-contrib folder and is not fully copied here due to the limited space:

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

  <Success/>
  <HotelDescriptiveContents>
    <HotelDescriptiveContent HotelCode="123" HotelName="Frangart Inn">

    <!-- ... see file in development kit ... -->

    </HotelDescriptiveContents>
  </HotelDescriptiveContents>


</OTA_HotelDescriptiveInfoRS>
```

samples/Inventory-OTA_HotelDescriptiveInfoRS-hotelinfo.xml - outer part

4.4.9. Implementation tips and best practice

Please note that section 4.4 has been changed multiple times since its creation. See appendix C for compatibility information.

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 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.

The two optional attributes **RatePlanID** and **RatePlanQualifier** **may** be used by the client to identify a "master" rateplan and its alternative versions if the server supports them (see section joining rate plans).

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,
- **one** **Offer** element with **one** **OfferRule** element: it defines the cut-off age above which guests are considered adults, the allowed age range for children (or whether children are allowed at all), and optionally introduces more restrictions to the applicability of the rate plan,
- **zero**, **one** or **two** additional **Offer** elements: indicates potential discounts such as free nights or kids that go free,
- **zero to five** **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="New" CurrencyCode="EUR" RatePlanCode="Rate1-4-HB">

            <BookingRules>
                <BookingRule Start="2014-03-03" End="2014-04-17">
                    ...
                </BookingRule>
            </BookingRules>

            <Rates>
                <!-- "static" and "date dependent" Rate element described below -->
                <Rate> ... </Rate>
            </Rates>

            <Supplements>
                <!-- "static" and "date dependent" Supplement element described below -->
                <Supplement> ... </Supplement>
            </Supplements>

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

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

        </RatePlan>

    </RatePlans>

</OTA_HotelRatePlanNotifRQ>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - outer part

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

Each **Description** element **must** have a **Name** attribute. Within a rate plan, the **Name** attribute values must be distinct. Allowed values are:

- `title`
- `intro`
- `description`
- `gallery`
- `codelist`

For the **optional** **Description** elements with names `title`, `intro` or `description`, the following rules hold:

- 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.

Here is an example:

```
<Description Name="title">
    <Text TextFormat="PlainText" Language="en">Wellness Offer</Text>
    <Text TextFormat="PlainText" Language="de">Wellness Angebot</Text>
    <Text TextFormat="PlainText" Language="it">Offerta Wellness</Text>
</Description>
<Description Name="intro">
    <Text TextFormat="PlainText" Language="en">Lorem ipsum EN</Text>
    <Text TextFormat="PlainText" Language="de">Lorem ipsum DE</Text>
    <Text TextFormat="PlainText" Language="it">Lorem ipsum IT</Text>
</Description>
<Description Name="description">
    <Text TextFormat="PlainText" Language="en">Lorem ipsum sit amet EN</Text>
    <Text TextFormat="PlainText" Language="de">Lorem ipsum sit amet DE</Text>
    <Text TextFormat="PlainText" Language="it">Lorem ipsum sit amet IT</Text>
    <Text TextFormat="HTML"      Language="en">Lorem ipsum &lt;br&gt; sit amet EN</Text>
    <Text TextFormat="HTML"      Language="de">Lorem ipsum &lt;br&gt; sit amet DE</Text>
    <Text TextFormat="HTML"      Language="it">Lorem ipsum &lt;br&gt; sit amet IT</Text>
</Description>
```

The **optional** **Description** element with name `gallery` is used to store a sequence of images. Each sequence describes one (independent) image and is made of:

- **one** **Image** element containing the image URL,
- **zero or more** **Text** elements with attributes **TextFormat** (set to `PlainText`) and **Language** (at most one **Text** element per language) that describe the image above
- **zero or one** **Text** elements with just the attribute **TextFormat** (set to `PlainText`) containing copyright information for the image above,
- **zero or one** **URL** elements containing a link to the attribution of image above.

Here is an example:

```
<Description Name="gallery">

    <Image>http://www.example.com/my-beautiful-room.jpg</Image>
    <Text TextFormat="PlainText" Language="en">Description EN</Text>
    <Text TextFormat="PlainText" Language="it">Description IT</Text>
    <Text TextFormat="PlainText" Language="de">Description DE</Text>
    <Text TextFormat="PlainText">(C) 2012 Example Inc.</Text>
    <URL>http://www.example.com/attribution/</URL>

    <Image>http://www.example.com/my-even-more-beautiful-room.jpg</Image>
    <Text TextFormat="PlainText" Language="en">Description EN</Text>
    <Text TextFormat="PlainText" Language="it">Description IT</Text>
    <Text TextFormat="PlainText" Language="de">Description DE</Text>
    <Text TextFormat="PlainText">(C) 2012 Example Inc.</Text>
    <URL>http://www.example.com/attribution/</URL>

    <!-- ... -->

</Description>
```

The **optional Description** element with name `codelist` is used to exchange “theme” information about the rate plan that is not meant to be read by humans.

Such a **Description** element contains nothing but **one or more ListItem** elements with no attributes, each containing a code (the quoted part) from the following list:

- "ALPINEBITS:1001" meaning Bicycle touring
- "ALPINEBITS:1002" meaning Car-free Holiday
- "ALPINEBITS:1003" meaning Cars & Motorcycle
- "ALPINEBITS:1004" meaning Christmas Markets
- "ALPINEBITS:1005" meaning Culture
- "ALPINEBITS:1006" meaning Ecologic Holiday
- "ALPINEBITS:1007" meaning Events
- "ALPINEBITS:1008" meaning Family
- "ALPINEBITS:1009" meaning Food
- "ALPINEBITS:1010" meaning Golf
- "ALPINEBITS:1011" meaning Hiking
- "ALPINEBITS:1012" meaning Horseback Riding
- "ALPINEBITS:1013" meaning Luxury Holiday
- "ALPINEBITS:1014" meaning Mountain Bike
- "ALPINEBITS:1015" meaning Other Summer Activities
- "ALPINEBITS:1016" meaning Other Winter Activities
- "ALPINEBITS:1017" meaning Pets-friendly Holiday
- "ALPINEBITS:1018" meaning Road Bike
- "ALPINEBITS:1019" meaning Romantic Holiday
- "ALPINEBITS:1020" meaning Sci & Snowboard
- "ALPINEBITS:1021" meaning Spa & Health
- "ALPINEBITS:1022" meaning Wine
- "ALPINEBITS:1023" meaning eBike

The "ALPINEBITS" and "OTA" namespaces are reserved, but partners are free to define custom namespaces. A server can safely ignore codes it doesn’t recognize and must not return a warning or error if it encounters one.

Here is an example:

```
<Description Name="codelist">
    <ListItem>ALPINEBITS:1001</ListItem>
    <ListItem>ALPINEBITS:1006</ListItem>
</Description>
```

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.

LengthOfStay elements indicate a minimum length of stay (by setting the attribute **MinMaxMessageType** to `SetMinLOS` or `SetForwardMinStay`) or a maximum length to stay (by setting the attribute **MinMaxMessageType** to `SetMaxLOS` or `SetForwardMaxStay`). Each value of **MinMaxMessageType** must not appear more than once in the same **LengthsOfStay** container element.

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**): `SetMinLOS`, `SetMaxLOS`, arrival DOW, master status `Open`,
- criteria of the booking rule (if any) that **applies to the departure day** (**Start** ≤ departure day ≤ **End**): departure DOW,
- criteria of the booking rule (if any) that **applies and need to be checked** for each day of the stay (excluding the departure day): the day **must not** be denied by a master status `Close` rule. The

whole length of the stay **must** be consistent with `SetForwardMinStay` and `SetForwardMaxStay` attributes.

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.

In example A, length of stay (LOS) restrictions are 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="2016-03-03" 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 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.

```
<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)

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

AlpineBits® classifies Rate elements into two kinds: static **Rate** elements and date depending **Rate** elements.

Static rates.

Static **Rate** elements are used to avoid sending the same information repeatedly for each rate of the same rate plan. They:

- contain static information that is valid for the whole Rateplan and is not date depending,
- can **only** be transmitted in messages with **RatePlanNotifType** set to `New`,
- can only be used at position 1 in a list of **Rate** elements.

Here is an example of such a static rate:

```
<!-- first in list: the static rate - values apply to every rate in the rate plan -->

<Rate RateTimeUnit="Day" UnitMultiplier="1">
  <BaseByGuestAmts>
    <BaseByGuestAmt Type="7"/>
  </BaseByGuestAmts>
  <MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
</Rate>
```

`samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml` - static rate

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 **both** of the **optional** attributes **RateTimeUnit** (`Day` is the only allowed value) and **UnitMultiplier** (number of nights) to the static **Rate** element. This will implicitly set the time unit for all the rates in the rate plan.

The **mandatory** **BaseByGuestAmt** element with the **only** and **mandatory** **Type** attribute determines if the amounts given in the rates of the containing rate plan are to be considered per person (**Type** `7`) or per room (**Type** `25`).

Finally, a static rate also sets the board type for the rate plan. This is done with the **mandatory** **MealsIncluded** element. The **mandatory MealPlanIndicator** attribute **must** be true and the **mandatory** **MealPlanCodes** attribute assumes one of the following values (a subset of the full OTA list):

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

Note that AlpineBits® **does not** use the single Breakfast/Lunch/Dinner booleans.

Date depending rates.

These date depending rates (just “called” rates from here on) 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**.

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

Here is an example of such a rate (the prices are considered "per person" because of the static rate described above):

```
<!-- following: "normal" rates ... -->

<Rate InvTypeCode="double" Start="2014-03-03" End="2014-03-08">
<BaseByGuestAmts>
  <BaseByGuestAmt NumberOfGuests="1" AgeQualifyingCode="10" AmountAfterTax="106"/>
  <BaseByGuestAmt NumberOfGuests="2" AgeQualifyingCode="10" AmountAfterTax="96"/>
</BaseByGuestAmts>
<AdditionalGuestAmounts>
  <AdditionalGuestAmount AgeQualifyingCode="10"                          Amount="76.8"/>
  <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"/>
</AdditionalGuestAmounts>
</Rate>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - rate

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

- **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 **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:

1. If **AdditionalGuestAmount** are defined at all, **exactly one AdditionalGuestAmount** element with a **AgeQualifyingCode** set to `10` (“adult”) **must** be present.
2. **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**.
3. 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.

All the **Rate** elements in a **Rateplan must** be consistent with the "brackets" defined in the first **OfferRule** element, it is a client responsibility to ensure this.

A server **must** return a warning outcome or error outcome if this is not the case.

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.

Supplements, analogously to Rates are split into static and date depending **Supplement** elements.

The **static** supplements contain the information used to identify and describe the supplement to guests. For each distinct **InvCode** that is specified, there **must** be exactly one static **Supplement** element and there **may** be several date depending **Supplement** elements.

The following attributes and sub-elements are used in **static** supplements:

- The **mandatory** attribute **AddToBasicRateIndicator must** be set to `true` (to indicate the supplement amount must be added to the amount coming from the rate).
- The **mandatory** attribute **MandatoryIndicator** (a boolean value) indicates that the customer must book the supplement (`true`) or can choose to book the supplement (`false`).
- The **mandatory** attribute **ChargeTypeCode must** have one of the following values:
    - `1` - daily,
    - `12` - per stay,
    - `18` - per room per stay,
    - `19` - per room per night,
    - `20` - per person per stay,
    - `21` - per person per night,
    - `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.
- An **optional PrerequisiteInventory** sub-element with the **mandatory InvType** attribute set to `ALPINEBITSDOW` can be used to make supplements available only on certain days of the week. The **mandatory InvCode** attribute can be used to identify those days of the week: it is a string made of seven binary digits. Each position in the string refers to a day of the week, starting with Monday. A 1 at a given position means that the supplement is available on the corresponding day, a 0 means it’s not. A value of 1100000 for instance, would indicate a supplement available only on Monday and Tuesday.

- **Zero to five Description** elements - the format of the descriptions follow what has been explained for rate plan level descriptions (with the same name values).

All the **Supplement** attributes mentioned are **mandatory**.

No other attributes or sub-elements are allowed for static data supplements (in particular attributes **Start** and **End** are not allowed).

The following attributes and sub-elements define the price of the supplement for specific periods of time. They are thus used in **date depending** supplements:

- 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.
- An optional **PrerequisiteInventory** sub-element with the **mandatory InvType** attribute set to `ROOMTYPE` can be used to make supplements available only for certain room categories or price them differently for different room categories. The **mandatory InvCode** attribute can be used to identify the room category the supplement applies to.

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

Multiple date depending **Supplement** elements referring to the same **InvCode** and **PrerequisiteInventory** 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 for more details.

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)):

1. calculate the number of days where the supplement applies (the supplement must not be applied to the departure day, hence the result is **3**),
2. sum the applicable price for each day (80 + 80 + 85 = 245 €),
3. 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 €).

Note that the rate plan is available also in dates for which a Supplement declared as mandatory is not specified. Of course in these dates the supplement will not be available to guests.

Similarly when a supplement is not applicable to a potential stay due to an unmatched **PrerequisiteInventory**, the rate plan - without the supplement - is still available.

Offers

Offers are considered ***static data*** and can **only** be transmitted in messages with **RatePlanNotifType** set to `New`.

The **mandatory first Offer** element **must** contain **one OfferRule** element, **no Discount** element and **no Guest** element.

The **first OfferRule** element contains:

- zero or one **LengthOfStay** elements with **MinMaxMessageType** set to `SetMinLOS`
- zero or one **LengthOfStay** elements with **MinMaxMessageType** set to `SetMaxLOS`
- zero or one **ArrivalDaysOfWeek** elements
- zero or one **DepartureDaysOfWeek** elements

The use of these elements inside an **OfferRule** is analogous to their use inside a **BookingRule** as explained above.

The **first OfferRule** element further contains:

- **one Occupancy** element with the **AgeQualifying** attribute set to `10` ("adult") and an **optional MinAge** attribute (integer value > 0)
- **zero or one Occupancy** element with the **AgeQualifying** code set to `8` ("child") and **optional** attributes **MinAge** and **MaxAge** (integer value > 0)

Rules regarding these **Occupancy** elements:

➔ If the only **Occupancy** element present is the one with **AgeQualifying** attribute `10` ("adult") and no **MinAge** attribute, all guests are to be considered "adults"; if the **MinAge** attribute is present, **MinAge** must be ≤ `18` – all guests ≥ **MinAge** are considered "adults" and all guests < **MinAge** are considered "children".

➔ If and only if the **MinAge** attribute is specified for the "adult" guests, then in order to allow the presence of children, the **Occupancy** element with **AgeQualifying** attribute `8` ("child") **must** be present. If that element has a **MinAge** and/or a **MaxAge** attribute, the age of "children" is restricted to the interval **MinAge** ≤ age < **MaxAge**.

➔ Any of the **Occupancy** elements may contain also the attributes **MinOccupancy** with integer values between 0 and 99 and **MaxOccupancy** with integer values between 1 and 99 having the purpose to restrict the total number of adults or children allowed in a stay with this rate plan.

The **first OfferRule** element **might** also have **any** of the following arguments with values that are durations given in days encoded in ISO 8601 (thus always in the form PxD where x is the integer number of days):

- **MinAdvancedBookingOffset** - the rate plan is only bookable before the given number of days before the arrival date,
- **MaxAdvancedBookingOffset** - the rate plan is only bookable if booked after the given number of days before the arrival date.

The **first OfferRule** element contains all the static booking rules of the rate plan. **All** the restrictions that are defined in this element **must** be fulfilled by a stay in order to make the rate plan bookable.

Here is a quite minimal first **OfferRule** element without any restrictions: guests are considered adults if they are of age 16 or older and children (of any age) are allowed:

```
<Offer>
    <OfferRules>
        <OfferRule>
            <Occupancy AgeQualifyingCode="10" MinAge="16"/>
            <Occupancy AgeQualifyingCode="8"/>
        </OfferRule>
    </OfferRules>
</Offer>
```

samples/RatePlans-OTA_HotelRatePlanNotifRQ.xml - offer

The first offer element is followed by **zero**, **one** or **two** additional **Offer** elements describing discounts.

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 contain **at most one** of each kind. Note that discounts (if given at all) do not necessarily need to apply to a stay in order to make the rate plan **bookable**.

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

- **Percent** (**mandatory**) - value is always `100`.
- **NightsRequired** (**mandatory**) - how many nights at least must be booked for the discount to apply.
- **NightsDiscounted** (**mandatory**) - how many nights are discounted.
- **DiscountPattern** (**optional**) - 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.

If the **DiscountPattern** is **present**, the discount pattern can be applied repeatedly from the beginning of the stay. If, for instance, **NightsRequired** is 7 and the stay is 14 nights, the pattern applies exactly twice, thus two times the **NightsDiscounted** nights are free (corresponding to the nights having a `1` in the pattern).

If the **DiscountPattern** is **absent**, the discount is not repeatable and the free nights are simply the last **NightsDiscounted** nights of the stay.

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`.

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.

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.

Following are a few complete examples of the **Offers** element.

Example A (early booking offer): must be booked at least 30 days in advance, guests ≥ 16 are considered adults, guest < 10 are not allowed.

```
<Offers>
    <Offer>
        <OfferRules>
            <OfferRule MinAdvancedBookingOffset="P30D">
                <Occupancy AgeQualifyingCode="10" MinAge="16" />
                <Occupancy AgeQualifyingCode="8" MinAge="10" MaxAge="16" />
            </OfferRule>
        </OfferRules>
    </Offer>
</Offers>
```

Offers - example A

Example B (last minute offer): cannot be booked more than 7 days in advance, guests ≥ 16 are considered adults, children of any age allowed.

```
<Offers>
    <Offer>
        <OfferRules>
            <OfferRule MaxAdvancedBookingOffset="P7D">
                <Occupancy AgeQualifyingCode="10" MinAge="16" />
                <Occupancy AgeQualifyingCode="8" />
            </OfferRule>
        </OfferRules>
    </Offer>
</Offers>
```

Offers - example B

Example C (four nights for the price of three): only applicable to a stay of exactly 4 nights with arrival on Sunday and departure on Thursday, last night is free, guests ≥ 16 are considered adults, children of any age allowed.

```
<Offers>
  <Offer>
    <OfferRules>
      <OfferRule>
        <LengthsOfStay>
          <LengthOfStay Time="4" TimeUnit="Day" MinMaxMessageType="SetMinLOS" />
          <LengthOfStay Time="4" TimeUnit="Day" MinMaxMessageType="SetMaxLOS" />
        </LengthsOfStay>
        <DOW_Restrictions>
          <ArrivalDaysOfWeek   Sun="1" Mon="0" Tue="0" Weds="0" Thur="0" Fri="0" Sat="0" />
          <DepartureDaysOfWeek Sun="0" Mon="0" Tue="0" Weds="0" Thur="1" Fri="0" Sat="0" />
        </DOW_Restrictions>
        <Occupancy AgeQualifyingCode="10" MinAge="16" />
        <Occupancy AgeQualifyingCode="8" />
      </OfferRule>
    </OfferRules>
  </Offer>
  <Offer>
    <Discount Percent="100" NightsRequired="4" NightsDiscounted="1" />
  </Offer>
</Offers>
```

Offers - example C

Example D (another free nights example): the last night is free if the stay is ≥ 4 nights. Since there is no **DiscountPattern**, you still get just one (the last) night free, even if the stay is ≥ 8 nights. Guests ≥ 16 are considered adults, children of any age allowed. Note the rate plan is still bookable for stays of less than 4 nights (but then there’s no discount).

```
<Offers>
    <Offer>
        <OfferRules>
            <OfferRule>
                <Occupancy AgeQualifyingCode="10" MinAge="16" />
                <Occupancy AgeQualifyingCode="8" />
            </OfferRule>
        </OfferRules>
    </Offer>
    <Offer>
        <Discount Percent="100" NightsRequired="4" NightsDiscounted="1" />
    </Offer>

</Offers>
```

Offers - example D

Example E (one child goes free): if there are at least two children < 5, one of them (the younger one) goes free. Guests ≥ 16 are considered adults, children of any age allowed. Note the rate plan is still bookable if the *family offer* isn't applicable.

```
<Offers>
    <Offer>
        <OfferRules>
            <OfferRule>
                <Occupancy AgeQualifyingCode="10" MinAge="16" />
                <Occupancy AgeQualifyingCode="8" />
            </OfferRule>
        </OfferRules>
    </Offer>
    <Offer>
        <Discount Percent="100" />
        <Guests>
            <Guest AgeQualifyingCode="8" MaxAge="5" MinCount="2"
                   FirstQualifyingPosition="1"
                   LastQualifyingPosition="1" />
        </Guests>
    </Offer>
</Offers>
```

Offers - example E

Example F has the same family discount as example E, but here there is also a **restriction**, stating the rate plan is only bookable if there are at least two children < 5.

```
<Offers>
    <Offer>
        <OfferRules>
            <OfferRule>
                <Occupancy AgeQualifyingCode="10" MinAge="16" />
                <Occupancy AgeQualifyingCode="8" MaxAge="5" MinOccupancy="2"/>
            </OfferRule>
        </OfferRules>
    </Offer>
    <Offer>
        <Discount Percent="100" />
        <Guests>
            <Guest AgeQualifyingCode="8" MaxAge="5" MinCount="2"
                   FirstQualifyingPosition="1" LastQualifyingPosition="1" />
        </Guests>
    </Offer>

</Offers>
```

Offers - example F

Example G combines some restriction with a *family offer* and a *free nights offer*! N nights are free according to the pattern, if the stay is at least N times 4 nights. Additionally one child < 5 goes free. The stay **must** be > 4 nights due to LOS restriction, but the rate plan is bookable also without the presence of a child < 5 (the *family offer* just doesn't apply).

```
<Offers>
    <Offer>
        <OfferRules>
            <OfferRule>
                <LengthsOfStay>
                    <LengthOfStay Time="4" TimeUnit="Day" MinMaxMessageType="SetMinLOS" />
                </LengthsOfStay>
                <Occupancy AgeQualifyingCode="10" MinAge="16" />
                <Occupancy AgeQualifyingCode="8" />
                </OfferRule>
            </OfferRules>
        </Offer>
    <Offer>
        <Discount Percent="100" NightsRequired="4"
                  NightsDiscounted="1" DiscountPattern="0001" />
    </Offer>
    <Offer>
        <Discount Percent="100" />
        <Guests>
            <Guest AgeQualifyingCode="8" MaxAge="5" MinCount="1"
                   FirstQualifyingPosition="1" LastQualifyingPosition="1" />
        </Guests>
    </Offer>
</Offers>
```

Offers - example G

Joining rate plans

The use case of having alternative prices for alternative meal plans is fairly common. To this end, AlpineBits® provides the functionality of joining rate plans. In this case, the two optional attributes **RatePlanID** and **RatePlanQualifier** **may** be used by the client to identify a "master" rateplan and its alternative versions. They can **only be sent** if the server supports the `OTA_HotelRatePlanNotif_accept_RatePlanJoin` capability. All these rate plans share the same **RatePlanID**: **exactly one** rate plan (the "master") for each **RatePlanID** value **must** have the **RatePlanQualifier** set to `true`, every other rate plan (the alternative versions) that shares the same **RatePlanID** **must** have the **RatePlanQualifier** set to `false`.

When the server joins rate plans, it **must** use these components from the "master" rate plan:

- descriptive contents
- static information about rates (except the element **`MealsIncluded`**)
- offers
- static information about supplements

and **must** take these components from the alternative versions:

- booking rules
- date dependent information about rates
- date dependent information about supplements.

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 an “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 guest that are considered children (the cut-off age above which guests are considered adults is defined in the first **OfferRule** element): 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.

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 (total 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 1b (offer rule check)

Verify that **arr**, **n** and **c** are consistent with the first **OfferRule** element explained in the Offers section. It is not allowed to promote children to "adults" (that is remove elements from c and increment n) to force a match.

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.

A rate plan can be booked for a given a stay, even if it contains *family offers* that do not match the stay (of course the discount is not applied in that case).

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.

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, for each rate, the following sub-steps are to be performed to pick the **correct amount** from the **BaseByGuestAmt** and **AdditionalGuestAmount** elements:

- Each child (e.g. each element of the array **c**) is matched to the corresponding **AdditionalGuestAmount** element with **AgeQualifyingCode** set to `8` ("child"). At this point the fact that a match can be made for each child is guaranteed by the rules related to the first **OfferRule** element and by the check in step 1b.
- 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).

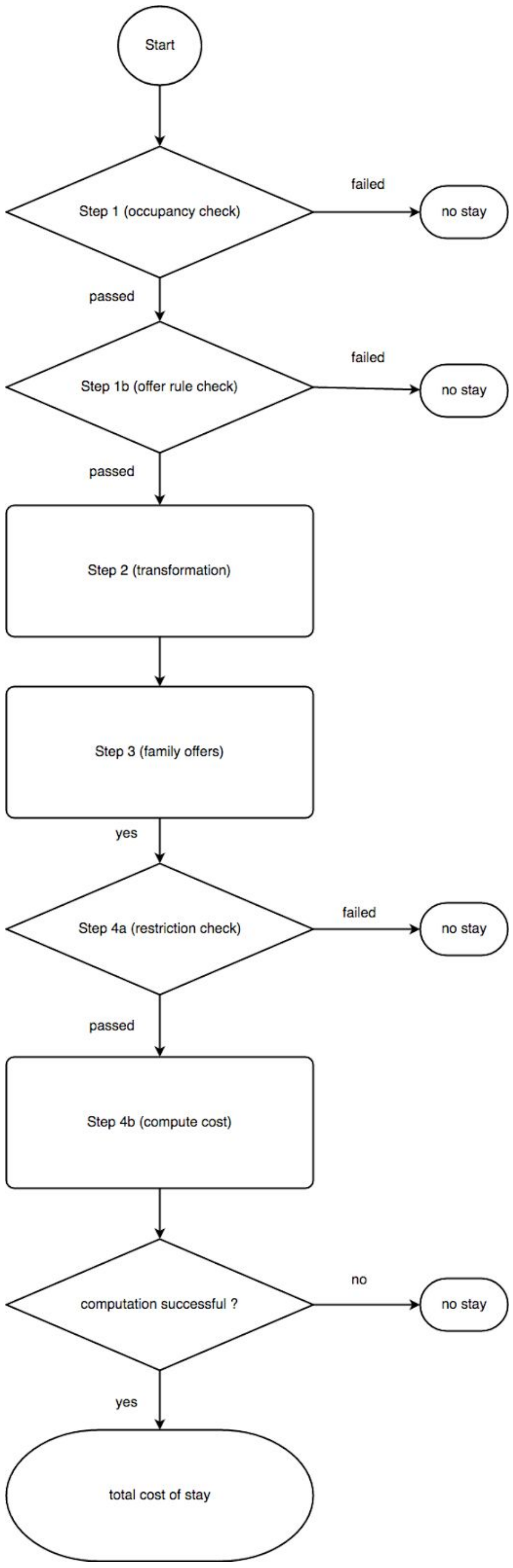
- The remaining guests among the **n** - if any - **each** pay an amount given by the **AdditionalGuestAmount** elements with **AgeQualifyingCode** set to `10` (“adult”).

  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).

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 flowchart of the whole 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. Static rates **must** be sent within this message too. Offers are always considered static and hence **must** also be send 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 or rates, only date depending data may be sent within this message, thus supplements or rates cannot be deleted with an Overlay message. Supplements might be set as not available, though. 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 **BookingRule** elements will delete them. However, one cannot delete sub-elements (such as only the **LengthOfStay** restrictions).

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 a OTA_HotelRatePlanNotifRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed.

See Appendix A for details.

4.5.5. Implementation tips and best practice

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 element **RoomCompanions**
- 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.

4.6. BaseRates

If the `action` parameter is `OTA_HotelRatePlan:BaseRates`, the client sends a pull request to **query** rate plans from the server in order to import data back into a PMS or portal.

4.6.1. Client request

The parameter `request` contains an OTA_HotelRatePlanRQ document.

Nested inside one **RatePlan** element the following sub-elements are given in order:

- zero or one **DateRange** element with **Start** and **End** attributes,
- zero or more **RatePlanCandidate** elements with attributes **RatePlanCode** and **RatePlanID**,
- **one HotelRef** element with attributes **HotelCode** and **HotelName** - the rules are the same as for room availability notifications (section 4.1.1).

Here is an example:

```
<OTA_HotelRatePlanRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="3.000">
  <RatePlans>
    <RatePlan>

      <DateRange Start="2016-12-25" End="2017-01-03" />

      <RatePlanCandidates>
        <RatePlanCandidate RatePlanCode="DZ" RatePlanID="DailyRate"/>
        <RatePlanCandidate RatePlanCode="SPECIAL"/>
      </RatePlanCandidates>

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

    </RatePlan>
  </RatePlans>
</OTA_HotelRatePlanRQ>
```

samples/BaseRates-OTA_HotelRatePlanRQ.xml

Regarding **DateRange** and **RatePlanCandidate** six cases need to be distinguished:

1 - **DateRange** omitted, **RatePlanCandidate** present:
The server will answer with the matching rate plans.

2 - **DateRange** omitted, **RatePlanCandidate** omitted:
This is not possible and will trigger a warning response.

3 - **DateRange** empty, **RatePlanCandidate** present:
If the server supports deltas, it **must** return the changes to the requested rate plans since the last request by the same client. If the server does not support deltas, this is not possible and will trigger a warning response.

4 - **DateRange** empty, **RatePlanCandidate** omitted:
This is not possible and will trigger a warning response.

5 - **DateRange** set, **RatePlanCandidate** present:

Server responds with all rates matching the given date range and rate plan.

6 - **DateRange** set, **RatePlanCandidate** omitted:

The server response contains descriptions of the rate plans having rates in the given date range. Note that the Rate elements are omitted, the response contains just the **RatePlan** elements with the **Description** elements.

4.6.2. Server response

The server will send a response indicating the outcome of the request. The response is a OTA_HotelRatePlanRS document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed. See Appendix A for details.

In case of a successful outcome, the **Success** element is followed by a **RatePlans** element with the information about the hotel.

Here is an example:

```
<OTA_HotelRatePlanRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="3.000">
    <Success/>
    <RatePlans HotelCode="123" HotelName="Frangart Inn">
        <RatePlan RatePlanCode="Base" CurrencyCode="EUR">
            <Rates>
                <Rate RateTimeUnit="Day" UnitMultiplier="1">
                    <BaseByGuestAmts>
                        <BaseByGuestAmt Type="7"/>
                    </BaseByGuestAmts>
                    <MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
                </Rate>
                <Rate Start="2016-12-29" End="2016-12-31" InvTypeCode="EZ">
                    <BaseByGuestAmts>
                        <BaseByGuestAmt NumberOfGuests="1"
                                        AmountAfterTax="130.00"/>
                    </BaseByGuestAmts>
                </Rate>
                <Rate Start="2016-12-29" End="2016-12-31" InvTypeCode="DZ">
                    <BaseByGuestAmts>
                        <BaseByGuestAmt NumberOfGuests="2"
                                        AmountAfterTax="180.00"/>
                    </BaseByGuestAmts>
                </Rate>
            </Rates>
            <Description Name="title">
                <Text TextFormat="PlainText" Language="en">Lorem ipsum.</Text>
                <Text TextFormat="PlainText" Language="it">Lorem ipsum.</Text>
                <!-- more languages ... -->
            </Description>

        </RatePlan>
    </RatePlans>
</OTA_HotelRatePlanRS>
```

samples/BaseRates-OTA_HotelRatePlanRS.xml

A. AlpineBits® server response outcomes

For each data exchange action the server will send a specific response.

The response document type obviously depends on the action. For instance, a FreeRooms response will be a OTA_HotelAvailNotifRS document, whereas a response to a GuestRequests message will be a OTA_ResRetrieveRS document, as listed in section 4 (see the table at the beginning of the section for an overview). The information in the response varies accordingly. E.g. a successful OTA_ResRetrieveRS response contains a `ReservationsList` element, whereas an OTA_HotelAvailNotifRS obviously cannot contain such an element.

What all responses have in common, however, is that they indicate the success or failure of the data exchange action. AlpineBits® distinguishes **four** outcomes that are modeled using the **three** elements provided by OTA in this context: `Success`, `Warnings` and `Errors`. The four outcomes are: **success**, **advisory**, **warning** and **error** and are explained in the following sections.

Responses with an AlpineBits® success outcome

The request could be correctly parsed, was deemed syntactically valid and could be processed successfully in its entirety. All business rules were satisfied. In this case, the response contains **one** empty `Success` element, **no** `Warnings` and **no** `Errors`:

```
<!-- response document -->

    <Success/>

    <!-- according to the document type, more elements might follow -->
```

The client does not need to take any further action, upon receiving a response with a success outcome.

Responses with an AlpineBits® advisory outcome

As is the case for the success outcome, the request could be correctly parsed, was deemed syntactically valid and could be processed successfully in its entirety. All business rules were satisfied.

However, one or more non-fatal problems were detected and the server wishes to let the client know about them.

In this case, the response contains **one** empty `Success` element followed by **one or more** `Warning` elements with the attribute `Type` set to `11`, meaning “Advisory” according to the “Error Warning Type” (EWT) list in the OTA code list [5].

Each `Warning` element should contain a human readable text.

Here is an example of FreeRooms response with an **advisory** outcome. Let’s imagine a server that handles FreeRooms with delta messages but wishes to receive at least one full data set each 48 hours and a client that hasn’t send one in the last 48 hours. The server might then proceed to accept another delta message, but advise the client with the following response:

```
<?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

A server might or might not implement responses with advisory outcomes.

However, a client **must** recognize advisory outcomes and visualize or log the human readable text, so that the non-fatal problem can be analyzed and corrected at a later stage. Of course, there is no need to resend the message as the server has already processed it successfully.

Responses with an AlpineBits® warning outcome

The request could be correctly parsed, was deemed syntactically valid, but could **not** be processed successfully in its entirety, because some business rules were violated.

Some examples for messages that cause a business rule violation are:

- a message with an unknown HotelCode
- a RatePlans message having overlapping **Rate** elements with the same **InvTypeCode** attribute
- a GuestRequests acknowledgement message with an unknown ID

In this case, the response contains **one** empty **`Success`** element followed by **one or more** **`Warning`** elements with the attribute **`Type`** set to any value allowed by the “Error Warning Type” (EWT) list in the OTA code list [5] **other than** `11` (“Advisory”).

Each **`Warning`** element should contain a human readable text.

Here is an example of RatePlans response with an **warning** outcome. Let's imagine a client that sent rate information concerning dates in the year 2107 because there was a data entry error. While the request was formally correct the server **could not** process the request (and thus **could not** store the information about the rates) because it does not deal with dates in the distant future. So the server will proceed to refuse the request with the following warning response:

```
<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

The value 3 for attribute **Type** stands for "Biz rule" according to the EWT.

Upon receiving an AlpineBits® warning outcome, a client **must** consider the request to be failed **in its entirety** and must act accordingly. Since the request violated a business rule, there is no use to just try resending it. The client must rather escalate the failure. When run interactively, this means alerting the user. When run in an automatized way, this means alerting someone using appropriate means.

It is important to understand that despite the meaning of the word "warning" in other contexts, an AlpineBits® warning outcome indicates a failed request (due to violation of business rules). **It cannot be safely ignored**.

Note that AlpineBits® uses the **Warnings** element also for acknowledgements in the GuestRequests section (see 4.2.3). Those messages are not considered responses (as indicated by the RQ in OTA_NotifReportRQ) and are thus unrelated to the discussion in the present section.

Responses with an AlpineBits® error outcome

The request caused one or more of the following problems:

- it could **not** be correctly parsed
- it was deemed syntactically invalid
- some error occurred while processing the request

and therefore the request could not be be processed successfully in its entirety.

In this case, the response contains **one or more** **Error** elements with the attribute **Type** set to 13, meaning "Application error" according to the "Error Warning Type" (EWT) list and the attribute **Code** set to any value allowed by the "Error Codes" list in the OTA code list [5].

Each **Error** element should contain a human readable text.

As an example, let’s imagine a client sends a message that requires one of the attributes **`HotelCode`** or **`HotelName`** to be present, but doesn’t include any of the two. This makes the request invalid and the server **must** answer with a response indicating the error outcome. Here is an example:

```
<!-- response document -->

    <Errors>
        <Error Type="13" Code="321">
            missing HotelCode or HotelName
        </Error>
    </Errors>
```

Here, the **`Code`** `321` stands for “Required field missing” according to the “Error Codes” list.

Upon receiving an AlpineBits® error outcome, a client **must** consider the request to be failed **in its entirety** and must act accordingly.

If a client software has reason to assume the problem is temporary and occurred for the first time, it **might** try to resend the request at a later moment (and bail out after a small number of retries with no success).

The client **must** then escalate the failure. When run interactively, this means alerting the user. When run in an automatized way, this means alerting someone using appropriate means.

Note that a server that receives a request that is not authenticated, has missing or invalid POST parameters, will just respond with an `ERROR` string as explained in sections 2 and 3. Since at that point no action can be identified the request type and hence the response type is unknown and no exchange of XML documents takes place.

B. 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).

C. Protocol Version Compatibility

C.1. Major overhaul in version 2017-10

AlpineBits® server response outcomes

The AlpineBits® response outcomes (**success**, **advisory**, **warning**, **error**) are now uniformly and more thoroughly defined in the new appendix A.

In particular the distinction between warning and error outcomes is more in line with the OTA documentation [3] (see the section “OpenTravel Message Success, Warnings & Errors Elements”).

Implementations of 2015-07b will likely signal as errors things that in 2017-10 would be considered warnings. This should not lead to breakage, however, since both (warnings and errors) unambiguously indicate failed requests and this has been spelled out with greater clarity in the 2017-10 document.

In any case implementors are invited to have a close look at the new appendix A.

FreeRooms

Two new features have been added to FreeRooms: transmission of the number of bookable rooms and purge requests.

GuestRequests

Two new features have been added to GuestRequests: commissions and encrypted credit card numbers. The credit card holder name was made optional. Some info on how to fill out the ReservationID was added to the best practice section.

SimplePackages

The feature has been removed in version 2017-10.

Inventory

Inventory messages were introduced in version 2014-04 and overhauled in version 2015-07.

In version 2017-10 the section was again changed significantly. They former Inventory request action has been heavily refactored and split into four different actions:

1. **Inventory/Basic: (push)** action `OTA_HotelDescriptiveContentNotif:Inventory` (same string was also used in previous version) is now only used to send room category information and room lists (unlike previous versions, where it had multiple uses),

2. **Inventory/Basic (pull)**: action `OTA_HotelDescriptiveInfo:Inventory` (new) is used to request basic information.

3. **Inventory/HotelInfo (push)**: action `OTA_HotelDescriptiveContentNotif:Info` (new) is used to sends additional descriptive content and

4. **Inventory/HotelInfo (pull)**: action `OTA_HotelDescriptiveInfo:Info` (new) is used to request additional descriptive content.

Of course the corresponding action capabilities have been defined.

Due to the split, these capabilities are no longer necessary and were **removed**:

- `OTA_HotelDescriptiveContentNotif_Inventory_accept_basic`
- `OTA_HotelDescriptiveContentNotif_Inventory_accept_additional`

RatePlans

Concerning **booking rules**, previously the attribute **MinMaxMessageType** of **LengthOfStay** could assume the **two** values `SetMinLOS` and `SetMaxLOS` and there were two corresponding capabilities (`OTA_HotelRatePlanNotif_accept_MinLOS` and `OTA_HotelRatePlanNotif_accept_MaxLOS`). With 2017-10 the list of values have been expanded to four by adding `SetForwardMinStay` and `SetForwardMaxStay`. As all four values **must** be supported the two capabilities have been removed.

It is now possible to define **supplements** that are only available on given days of the week and supplements that depend on room category.

Offers have been improved and extended: besides the discounts, the **Offer** element is now also used to transmit the age above which a guest is considered an adult and a number of restrictions (for which 2 new capabilities (starting with `OTA_HotelRatePlanNotif_accept_OfferRule_`) have been introduced). Free night discounts have also been slightly updated.

Changes have been made to the algorithm describing the computation of the cost of a stay (see section 4.5.2): there is now a new step 1b, step 3 has been changed (a rate plan is now applicable even if it has a family offer that doesn't match the stat) and step 4b has been simplified (as we have complete and gapless age brackets for all possible child ages now).

Descriptions have been extended and documented more thoroughly.

BaseRates

This action has been introduced with 2017-10.

C.2. 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**.

C.3. 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*

C.4. 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.

C.5. 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.

C.6. 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.

D. External links

[0] Creative Commons BY SA license:
https://creativecommons.org/licenses/by-sa/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:
http://opentravelmodel.net/pubs/specifications/OnlinePublicationDetails.html?spec=2015A&specType=1_0&group=19701

[4] OTA2015A XML schema files:
http://opentravelmodel.net/pubs/specifications/OnlinePublicationDetails.html?spec=2015A&specType=OTA_1_0

[5] OTA2015A code list:
http://opentravelmodel.net/pubs/specifications/OnlinePublicationDetails.html?spec=2015A&specType=1_0&group=19708

[6] browsable interface to the OTA XML schema files (choose model "OTA2015A"):
http://adriatic.pilotfish-net.com/ota-modelviewer/