AlpineBits®

AlpineBits HotelData 2026-10

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.

©AlpineBits Alliance
This document is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License.
Permissions beyond the scope of the license may be available at www.alpinebits.org.

Table of Contents

1. Introduction **12**

2. Protocol and error handling **13**

2.1. Request and response 13

2.2. HTTP status codes 14

2.2.1. Authentication 15

2.2.2. Server Validation 15

2.2.3. Internal Server Errors 16

2.2.4. Data Validation Errors 16

2.3. AlpineBits® responses and error handling 16

2.4. Implementation tips and best practice 21

3. Handshaking action **22**

3.1. Client request 22

3.2. Server Response 24

3.3. List of AlpineBits® supported `actions` and `capabilities` 25

3.4. Unknown or missing actions 27

3.5. Implementation tips and best practice 27

3.6. Tabular representation of OTA_PingRQ 28

3.7. Tabular representation of OTA_PingRS 28

4. Data exchange actions **30**

On copyright and license of exchanged contents 32

4.1. FreeRooms: room availability notifications 33

4.1.1. Client request 33

CompleteSet 35

Deltas 36

Closing seasons 37

4.1.2. Server response 37

4.1.3. Implementation tips and best practice 37

4.1.4. Tabular representation of OTA_HotelInvCountNotifRQ 38

4.1.5. Tabular representation of OTA_HotelInvCountNotifRS 38

4.2. GuestRequests: quote requests, booking reservations and cancellations 39

Push Support 39

4.2.1. First client request 39

4.2.2. Server response 40

RoomStays 43

Services 48

ResGuests 54

ResGlobalInfo 56

4.2.3. Follow-up client request (acknowledgement) 61

4.2.4. Follow-up server response 62

4.2.5. Implementation tips and best practice 63

4.2.6. Requests Push client request 65

4.2.7. Requests Push server response 65

4.2.8. Guest Requests status update 67

4.2.9. Tabular representation of OTA_ReadRQ 68

4.2.10. Tabular representation of OTA_ResRetrieveRS 68

4.2.11. Tabular representation of OTA_NotifReportRQ 73
4.2.12. Tabular representation of OTA_NotifReportRS 74
4.2.13. Tabular representation of OTA_HotelResNotifRQ 74
4.2.14. Tabular representation of OTA_HotelResNotifRS 79
4.3. Inventory: room category information 80
4.3.1. Inventory/Basic (push) client request 80
4.3.2. Inventory/Basic (push) server response 86
4.3.3. Inventory/Basic (pull) client request 86
4.3.4. Inventory/Basic (pull) server response 86
4.3.5. Inventory/HotelInfo (push) client request 87
4.3.6. Inventory/HotelInfo (push) server response 95
4.3.7. Inventory/HotelInfo (pull) client request 95
4.3.8. Inventory/HotelInfo (pull) server response 96
4.3.9. Implementation tips and best practice 96
4.3.10. Tabular representation of OTA_HotelDescriptiveContentNotifRQ 96
4.3.11. Tabular representation of OTA_HotelDescriptiveContentNotifRS 104
4.3.12. Tabular representation of OTA_HotelDescriptiveInfoRQ 104
4.3.13. Tabular representation of OTA_HotelDescriptiveInfoRS 104
4.4. RatePlans 105
4.4.1. Client request 105
Booking rules 108
Rates 111
Supplements 114
Offers 117
Joining rate plans 121
4.4.2. Computing the cost of a stay 122
4.4.3. Synchronization 126
Messages limits 128
CompleteSet 129
4.4.4. Server response 129
4.4.5. Implementation tips and best practice 129
4.4.6. Tabular representation of OTA_HotelRatePlanNotifRQ 130
4.4.7. Tabular representation of OTA_HotelRatePlanNotifRS 135
4.5. BaseRates 136
4.5.1. Client request 136
4.5.2. Server response 137
4.5.3. Tabular representation of OTA_HotelRatePlanRQ 137
4.5.4. Tabular representation of OTA_HotelRatePlanRS 138
4.6. ActivityData 139
4.6.1. Exchange of hotel activities 139
4.6.2. ActivityData: Synchronization and deletion 142
4.6.3. Server response 143
4.6.4. Tabular representation of OTA_HotelPostEventNotifRQ 143
4.6.5. Tabular representation of OTA_HotelPostEventNotifRS 144
A. AlpineBits® developer resources 146
B. Protocol Version Compatibility 147

B.2026-10 Major overhaul in version 2026-10 147
Protocol changes 147
RatePlans 147
B.2024-10 Major overhaul in version 2024-10 147
RelaxNG 148
HotelCode and HotelName 148
Inventory 148
RatePlans 148
B.2022-10 Major overhaul in version 2022-10 148
GuestRequests 148
FreeRooms 148
B.2020-10 Major overhaul in version 2020-10 148
Currency 148
Copyright and License information on multimedia objects 148
GuestRequests 149
FreeRooms 149
RatePlans 149
Inventory/HotelInfo 149
Activities 149
B.2018-10 Major overhaul in version 2018-10 149
Housekeeping / Handshaking 149
New action GuestRequests (Push) 150
New attribute RoomType 150
Reference to the online presence of the lodging structure 150
Review price calculation 150
Interaction with channel managers 150
Server request for complete data set 150
RatePlans 150
B.2017-10 Major overhaul in version 2017-10 150
AlpineBits® server response outcomes 150
FreeRooms 151
GuestRequests 151
SimplePackages 151
Inventory 151
RatePlans 151
BaseRates 152
B.2015-07b Minor updates in version 2015-07b 152
B.2015-07 Major overhaul in version 2015-07 152
Inventory 152
B.2014-04 Major overhaul in version 2014-04 153
HTTPS layer 153
FreeRooms 153
GuestRequests 153
SimplePackages 153
Inventory and RatePlans 153
B.2013-04 Compatibility between a 2012-05b client and a 2013-04 server 153

Housekeeping 153
FreeRooms 153
GuestRequests 154
SimplePackages 154
B.2013-04 Compatibility between a 2013-04 client and a 2012-05b server 154
Housekeeping 154
FreeRooms 154
GuestRequests 154
SimplePackages 154
C. External links 155
D. Code Lists 156
ABH: AlpineBits Hotel Amenity Code 157
ABR: AlpineBits Room Amenity Type 162

Disclaimer

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

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

About the AlpineBits Alliance

The "AlpineBits Alliance" is a group of SME operating in the tourism sector working together to innovate and open the data exchange in 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
c/o NOI Techpark – Building D3
via Gianni Brida 4 | Gianni-Brida-Straße 4
39100 Bolzano | Bozen - ITALY
VAT Reg No: IT02797280217
https://www.alpinebits.org
info@alpinebits.org

AlpineBits Alliance Members

ADDITIVE GmbH - https://www.additive.eu
aries.creative KG SAS - https://www.ariescreative.com
ASA KG - https://www.asahotel.com
Brandnamic GmbH - https://www.brandnamic.com
Consisto GmbH - https://www.consisto.it
DESTINATION srl - https://www.destinationsrl.it
Factory GmbH - https://www.factory.it
GardenaNet snc - https://www.gardena.net
HGV - https://www.hgv.it
Hotelfabrik - https://www.hotelfabrik.it
Hundertrot GmbH - WESEO GmbH - https://www.weseo.at
IDM Südtirol - Alto Adige - https://www.idm-suedtirol.com
inetcons by Internet Consulting GmbH - https://www.inetcons.it
Internet Service GmbH - https://www.internetservice.it
Just Impact Group GmbH - https://justimpact.group
LTS - Landesverband der Tourismusorganisationen Südtirols - https://www.lts.it
NOI Spa/AG - https://noi.bz.it
PCS GmbH - https://www.pcs-software.it
Peer GmbH - https://www.peer.media
Schneemenschen GmbH - https://www.schneemenschen.de
Südtiroler Bauernbund - Roterhahn - https://www.roterhahn.it
Trentino holidays - https://www.thol.it
XLbit srl - https://www.xlbit.com
Yanovis - https://www.yanovis.com

Authors of this document:
AlpineBits® repository Contributors - https://gitlab.com/alpinebits/hoteldata/standard-specification/-/graphs/master

Document Change Log

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

protocol version	doc. release date	description
2026-10	2026-10-22	Status: • official release Updates and Additions: • All sample files and the snippets embedded in this document were reworked for this release: dates were rebased, the two were realigned where they had drifted apart, and the root **`Version`** attribute now carries the OTA2015A version of the message it appears on (see section 3.5) • New samples cover features that were described but never shown: the follow-up response to an acknowledgment, rate plan joining, special offers, `Overlay` and `Remove` messages, the out-of-order and out-of-market counts, availability for specific rooms, and the alternative forms of **`Commission`** and **`CardNumber`** • The AlpineBits ABH and ABR code lists are now used in the Inventory samples • The examples now describe one single property throughout: every sample addresses **`HotelCode`** 123 "AlpineBits Hotel", and its room categories are named `SINGLE`, `DOUBLE`, `SUITE` and `PITCH` in every action, as section 4.1.1 requires; the renaming example of section 4.3.1 follows suit, showing `EZ` renamed to `SINGLE` and `DZ` to `DOUBLE` • GuestRequests: the push response was described as carrying **zero or more** **`HotelReservation`** elements, a cardinality no valid message ever had: both the AlpineBits® schema and OTA2015A require at least one. The document now says **one or more** • GuestRequests: the original booking ID, assigned by the system where the guest completed the reservation, can be transmitted in a **`HotelReservationID`** element with **`ResID_Type`** 14, next to the ID assigned by the channel manager in **`UniqueID`** • GuestRequests: the **`End`** attribute of **`TimeSpan`** is now explicitly exclusive (it is the departure date), both in **`RoomStay`** and in **`Service`**; **`ExpireDateExclusiveInd`** is documented as optional, defaulting to `false`; seven examples show the valid combinations of **`TimeSpan`**, **`Start`** and **`End`** in a **`Service`** • RatePlans: the algorithm of section 4.4.2 "Computing the cost of a stay" was restructured: the guests are first classified as adults or children according to the first **`OfferRule`** element (new step 0), then **all** the bookability checks are performed on the resulting numbers of adults and children and before any transformation (steps 1 to 3; the former step 4a is now step 3), then the transformations are applied (steps 4 and 5) and finally the cost is computed (step 6)

protocol version	doc. release date	description
		Breaking changes: • Protocol: Removed the `multipart/form-data` *Content-Type* of the POST request. The action is now sent in the `X-AlpineBits-Action` header field and the XML document is sent as the request body with `Content-Type` set to `application/xml`. During the transition a server may keep serving earlier versions from the same URL, telling them apart by the *Content-Type* of the request • Schema: the schema has been aligned with OTA2015A wherever it accepted more than OTA does, so that every document valid against the AlpineBits® schema is also valid against OTA2015A. Nothing OTA2015A allows was removed: enumerations were restored for **`RatingSymbol`**, **`OffsetDropTime`** and **`CardCode`**; **`CurrencyCode`** and the OTA code list attributes got their OTA pattern back; numeric attributes, string lengths and **`maxOccurs`** got the upper bounds OTA states; the coordinates of **`Position`** and the ignored **`ID`** of **`UniqueID`** got the length limits OTA states; and **`DiscountPattern`** no longer accepts the empty string • Schema: an empty **`Services`**, **`Features`**, **`TextItems`** or **`OfferRules`** element, and an empty **`MultimediaDescriptions`** element inside a **`GuestRoom`**, are no longer valid, because OTA2015A requires at least one child in each of them. For all of them but one the document already prescribed at least one child, and only the schema was wider; the exception is **`Features`**, described in section 4.3 as containing zero or more **`Feature`** elements, which now requires one (a hotel with no accessibility feature to declare omits the **`Features`** element). An empty **`GuestRooms`** element is not affected and keeps its documented meaning of removing all room categories of a hotel • Schema: **`RatePlanCode`** is now limited to 16 characters everywhere it appears (RatePlans, BaseRates and GuestRequests), down from 64. OTA2015A allows only 16 in a reservation, so a longer code could be published but never reported back • Inventory: inside **`Policies`**, the schema now enforces what section 4.3 has always stated, namely that a **`Policy`** element carries **exactly one** policy; the children are declared as a choice, in the order OTA2015A uses. The schema used to accept any combination of them in a single **`Policy`**, in an order OTA2015A rejects. Messages following the documented model are unaffected • Inventory: added the optional attribute **`MaxAdultOccupancy`** to **`GuestRoom`**, with **`StandardOccupancy`** ≤ **`MaxAdultOccupancy`** ≤ **`MaxOccupancy`**, to state how many adults a room category allows; when it is given, the computation of the cost of a stay (section 4.4.2) rejects a stay with more adults • RatePlans: Removed `Full` as allowed value for **`RatePlanNotifType`**, reverting to the use of `New`

protocol version	doc. release date	description
		• RatePlans: in section 4.4.2 *family offers* are now applied **before** children are transformed into adults to reach the minimum number of full-rate payers (steps 4 and 5), where up to 2024-10 the transformation came first. The conditions of a *family offer* are thus evaluated on the children as classified by the rate plan, and a child who stays free is never transformed into an adult. Note that all the bookability checks are still performed beforehand, in particular the minimum number of adults (**`MinOccupancy`** on the **`Occupancy`** element with **`AgeQualifyingCode`** `10` ("adult")). This new order only makes a difference when a *family offer* grants a child a free stay before the subsequent transformation (step 5) could turn that child into an adult to reach the minimum number of full-rate payers. This is a purely theoretical scenario: in practice, a *family offer* is unlikely to ever be meant for such a child. Hotels relying on the previous behaviour have to review their *family offers* • RatePlans: the static rate **must** now be transmitted in every message, with **`RatePlanNotifType`** `New` as well as `Overlay`, where up to 2024-10 it could be transmitted only with `New`. In an `Overlay` message it serves for control purposes only: the server should compare it with the rate plan on record and, on a discrepancy, refuse the rates with an error (see sections 4.4.1 and 4.4.3) • Schema: the empty string is no longer accepted for **`MinMaxMessageType`** inside an **`OfferRule`**, since it is not specified in the OTA standard; the documented values `SetMinLOS` and `SetMaxLOS` are unaffected • RatePlans: the pricing of pets, prescribed since release 2022-10, can finally be expressed in a valid message: `ALPINEBITS-PETS` is now an allowed value for **`CodeContext`** and **`InvType`** in the schema, and a sample was added • RatePlans: the value that marks the pets' booking rules and supplements was renamed from `ALPINEBITSEXTRA-PETS` to `ALPINEBITS-PETS`: OTA declares **`InvType`** as `StringLength1to16` and the old spelling is 20 characters long, so no message carrying it could validate against OTA2015A. The reserved **`InvType`** value `ALPINEBITSEXTRA`, which stands for a shared list of common **`InvCode`** values that does not exist yet, was renamed to `ALPINEBITS` at the same time, so that the two AlpineBits®-specific values of that attribute share one prefix • BaseRates: inside **`RatePlan`**, the schema now enforces what section 4.5.1 has always stated, namely that **`DateRange`**, **`RatePlanCandidates`** and **`HotelRef`** are given in that order, at most one each, with **`HotelRef`** always present. The schema used to accept them repeated, in any order, and with no **`HotelRef`** at all. Messages following the documented model are unaffected • GuestRequests (push): the **`HotelReservations`** element of a `OTA_HotelResNotifRQ` request must carry at least one **`HotelReservation`**, where the document previously allowed zero; OTA2015A rejects the empty element. The **`ReservationsList`** element of a `OTA_ResRetrieveRS` response may still be empty, as before

protocol version	doc. release date	description
		• GuestRequests (push): the **HotelReservations** element of a OTA_HotelResNotifRS response is now **optional**, as it is in OTA2015A. It was mandatory and requires at least one **HotelReservation**, so a server that could process none of the requests in a batch had no valid way to acknowledge nothing; it can now omit the element and answer with the warning outcome of section 2.3, listing every refusal in **Warnings**. The empty element stays invalid. No message valid until now becomes invalid, but clients **must** be prepared for the element to be absent • GuestRequests: a reservation can now carry all the arriving guests: **ResGuests** may contain one **ResGuest** per guest, each identified by the new **ResGuestRPH** attribute, and each **RoomStay** may list the guests that occupy it in the new **ResGuestRPHs** element. **ResGuests** is now described as optional, as the schema always allowed • GuestRequests: the **Profiles** element of **ResGlobalInfo** may now contain up to three **ProfileInfo** elements, with **ProfileType** 1 (customer), 3 (company) or 4 (booking channel), and a **Profile** may contain a **Customer** element, so that a reservation booked by a company can be described (see the sample GuestRequests-OTA_ResRetrieveRS-allguests-and-company.xml) • GuestRequests (acknowledgement): each **HotelReservation** of an OTA_NotifReportRQ request may now carry a **ResGlobalInfo** element with a **BasicPropertyInfo** element giving the **HotelCode** (and optionally the **HotelName**) of the acknowledged reservation, not only its ID • GuestRequests: **PaymentCard** now has the same definition in the **Guarantee** of a **RoomStay** and in **DepositPayments**. In **Guarantee**, **CardHolderName** and the **CardCode** and **ExpireDate** attributes are now optional, **ExpireDate** must be a valid MMYY date where any four digits were accepted, and **CardNumber** also accepts the **Mask**, **Token** and **TokenProviderID** attributes • GuestRequests and Inventory: the **PlainText** of **BankAcctNumber** (in the **DepositPayments** of a reservation and in the **GuaranteePaymentPolicy** of HotelInfo) and of **BankID** (in HotelInfo) now accepts any string of up to 64 characters (e.g. an IBAN), where only up to 19 digits were accepted • GuestRequests: the attributes of **DepositPayments**, which were untyped, now have a type: **GuaranteeTypeCode** accepts only 3, 28 and 46, **GuaranteeID** up to 64 characters, **PaymentTransactionTypeCode** only charge, reserve and the new value refund (see section 4.2.2 for the rules of a refund), **Start** a date-time and **GuaranteeCode** up to 32 characters • GuestRequests: the **AddressInfo** element of a company profile accepts the optional **Language** attribute • GuestRequests: the **Start** and **End** attributes of the **TimeSpan** of a **Service** now accept a date, a time or a date-time (OTA DateOrTimeOrDateTimeType), where only a date-time was accepted

protocol version	doc. release date	description
		• GuestRequests: added two new values for **`ServiceCategoryCode`** in **`Service`**: `PACKAGE` for package bookings and `RESOURCE` for bookable resources with limited availability, such as bicycles or baby monitors. Calculation of **`AmountAfterTax`** was added.
2024-10	2024-10-24	Status: • official release Updates and Additions: • RelaxNG has been removed as mandatory schema for AlpineBits® • GuestRequests: Added support for **`RoomRates`** • GuestRequests: Added element **`Services`** to specify additional services booked with the reservation • GuestRequests: Extension for **`Comment`** to support "additional info" • GuestRequests: Added element **`DepositPayments`** for information about payments made with the booking • GuestRequests: Added element **`Profiles`** for information about the booking channel • Inventory/Basic: Added custom AlpineBits Room Amenity Types (ABR) • Inventory/HotelInfo: Added custom AlpineBits® Hotel Amenity Codes (ABH) • Inventory/HotelInfo - Breaking change: Removed attributes **`CheckInTime`** and **`CheckOutTime`** in element **`PolicyInfo`** • Inventory: Added **`TPA_Extensions`** for exchanging additional codes based on national/EU regulations (i.e. CIN for Italy) • RatePlans: Capability `OTA_HotelRatePlanNotif_accept_full` added • RatePlans: Added `Full` value to attribute **`RatePlanNotifType`** to replace `New` value • Breaking change: Attribute **`HotelCode`** is now mandatory while attribute **`HotelName`** has become optional • Breaking change: Attributes **`HotelCode`** and **`HotelName`** must not be case sensitive anymore
2022-10	2022-10-20	Status: • official release Updates and Additions: • Handshake: added a complete handshake example in the introduction • Handshake: added new capability to request a handshake refresh • Handshake: added specification for unsupported versions • GuestRequests: added new action for reservation status update • GuestRequests: enhanced specification on mandatory acknowledgement in case of GuestRequest support • GuestRequests: added specification for newsletter registration

protocol version	doc. release date	description
		• GuestRequests: added possibility for multiple room stays in different timespans in a single request • GuestRequests: breaking change in the indication of pets in SpecialRequests • GuestRequests: extended SpecialRequests for a more generic usage • FreeRooms: added new capability for complete sets • Inventory/HotelInfo: added specification in usage of 3/4 board • RatePlan: added pricing of pets • RatePlan: exended charge type of supplements • Schema: minor restriction fixes • Various textual clarifications and enhancements in specification, best practices and appendix
2020-10	2020-10-15	Status: • official release Updates and Additions: • Data Exchange: added support of all currency codes from the ISO 4217 • Data Exchange: added exchange of copyright and license information on multimedia objects like images • GuestRequests: added the possibility for special requests regarding pets • FreeRooms: message changed from OTA_HotelAvailNotif to OTA_HotelInvCountNotif • FreeRooms: changes to capabilities • FreeRooms: added the possibility to exchange information regarding closing seasons and rooms out-of-order • Inventory/HotelInfo: specification for hotel information data has been expanded • RatePlans: new attribute CurrencyCode • Activities: added a new message to exchange information about hotel internal activities
2018-10	2018-11-30	Status: • official release Notable documentation changes: • **The AlpineBits® documentation is now edited using a public git repository, see the repository history for detailed changelog** Updates and Additions: • Handshaking: the Housekeeping action has been renamed to Handshaking and the relative chapter has been completely rewritten, also introducing breaking changes. Both server and client can now announce the capabilities they support. • GuestRequests: it is now possible to push the GuestRequests instead of polling them.

protocol version	doc. release date	description
		• GuestRequests/Inventory: a new attribute has been added in order to allow a finer classification of the guest accommodation. • Inventory: added the possibility to exchange references to the online presence of the lodging structure (e.g. social networks, review aggregators, etc.) • RatePlans: the price calculation has been reviewed and simplified • BaseRates: the interaction with the Channel Managers has been clarified and simplified • Servers can now request the client to send a full data set in case of suspected mis-alignment of data
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 appendices: • 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

protocol version	doc. release date	description
2015-07b	2016-08-01	Status: • official release Updates and additions: • RatePlans: Section 4.5 has been rewritten to clarify details that the previous version just skipped over, including a detailed description of the price calculation algorithm • RatePlans: new optional capability OTA_HotelRatePlanNotif_accept_RatePlanJoin to allow displaying alternative treatments for the same price list • 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** • 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)

protocol version	doc. release date	description
		• 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

protocol version	doc. release date	description
		• **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

1. Introduction

This document describes a standard for exchanging traveling and booking information, called **AlpineBits® HotelData**.

AlpineBits® HotelData 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
- traveling and booking information is 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),
- request and reservation inquiries (GuestRequests),
- room category information (Inventory),
- prices (RatePlans),
- prices for revenue management (BaseRates),
- hotel activities (Activities).

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

2. Protocol and error handling

2.1. Request and response

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 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 which a server implementer **might** want to use to identify the client software version or installation ID.

The HTTPS header of the POST request **must** contain an X-AlpineBits-Action field. The value of this field is the AlpineBits® action the client wishes to perform, for instance `OTA_Ping:Handshaking`. The complete list of the allowed values is given in section 3 for the handshaking action and in section 4 for the data exchange actions.

The body of the POST request **must** be the XML document required by the requested action and the HTTP request header *Content-Type* **must** be set to `application/xml`.

Compatibility with protocol versions earlier than 2026-10

The *Content-Type* described above is effective with the release of AlpineBits® 2026-10. Earlier versions keep following the rules described by their own specification, which used the `multipart/form-data` scheme.

For backward compatibility during the transition, a server **may** keep serving earlier versions from the same URL. In that case the *Content-Type* of the request determines which versions the request refers to: a request sent as `application/xml` refers to 2026-10 or later, a request sent as `multipart/form-data` refers to 2024-10 or earlier. The *Content-Type* and the value of the `X-AlpineBits-ClientProtocolVersion` field **must** agree. A server that receives a request where they do not agree **must** answer as described in section 2.2.

The two always agree when both parties follow their respective specifications, because a client sends the request in the *Content-Type* of the version it declares. A client that supports AlpineBits® 2026-10 alongside earlier versions therefore uses the *Content-Type* of whichever version it is addressing, and never mixes the two within the same request.

This allowance is meant to ease the migration to 2026-10 and is expected to be dropped once the earlier versions reach their end of life. A server that supports no version earlier than 2026-10 **must** reject any *Content-Type* other than `application/xml`.

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 a value. A POST request compressed with gzip **must** be compressed by the client in its entirety. It is the responsibility of the client 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 which support this feature. The so called "Handshaking" actions **must not** be compressed.

An exemplary handshake on command line using cURL with file `files/samples/Handshake/Handshake-OTA_PingRQ.xml` would look as follows.

```
curl --trace-ascii -
     -H "X-AlpineBits-ClientProtocolVersion: 2026-10"
     -H "X-AlpineBits-Action: OTA_Ping:Handshaking"
```

```
      -H "Content-Type: application/xml"
      --user user:secret
      --data-binary @Handshake-OTA_PingRQ.xml
      https://development.alpinebits.org/server-2026-10/
```

Following is the capture of the example POST from the command above. The value of the `X-AlpineBits-Action` field is the string `OTA_Ping:Handshaking`, indicating the client wishes to perform the handshake. The body is the XML document `files/samples/Handshake/Handshake-OTA_PingRQ.xml` (not fully shown).

```
POST /server-2026-10/ HTTP/1.1
Host: development.alpinebits.org
Authorization: Basic Y2hyaXM6c2VjcmV0
User-Agent: curl/7.64.0
Accept: */*
X-AlpineBits-ClientProtocolVersion: 2026-10
X-AlpineBits-Action: OTA_Ping:Handshaking
Content-Type: application/xml
Content-Length: 6854
Expect: 100-continue

[here is the content of the file Handshake-OTA_PingRQ.xml]
```

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

Also note the `X-AlpineBits-ClientProtocolVersion` and `X-AlpineBits-Action` fields, the `Content-Type` of the request and the XML document sent as its body.

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

```
HTTP/1.1 200 OK
Date: Wed, 21 Oct 2026 15:16:42 GMT
Server: Apache
X-Powered-By: Express
Access-Control-Allow-Origin: *
Content-Type: application/xml; charset=utf-8
Content-Length: 1653
ETag: W/"675-SunhP4r+79pVYE64e5ahe7LFpDs"
Vary: Accept-Encoding
X-AlpineBits-Server-Accept-Encoding: gzip

[here is the response, that is identical to Handshake-OTA_PingRS.xml in the samples]
```

2.2. HTTP status codes

AlpineBits® uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the 2xx range indicates success. Codes in the 4xx range indicate an error that occurred given the information provided (e.g., a required parameter was omitted, validation errors, etc.). Codes in the 5xx range indicates an error with the AlpineBits server and the client **should** retry.

- When authentication fails, the server **must** return a status code 401 along with an error message in the form of `text/plain` prefixed with `ERROR:`.
- If any preconditions fail before data validation, the server **must** return a status code 400 along with

an error message in the form of `text/plain` prefixed with `ERROR:`.

- Other validations **must** be returned in the form of XML within the corresponding OTA response element, with a status code 200.

2.2.1. Authentication

No valid basic auth credentials provided.

`401 Unauthorized`

```
ERROR:invalid or missing username/password
```

2.2.2. Server Validation

Missing Client Protocol Version

No valid HTTP Header `X-AlpineBits-ClientProtocolVersion` provided.

`400 Bad Request`

```
ERROR:no valid client protocol version provided
```

Unsupported Client Protocol Version

`400 Bad Request`

```
ERROR:your current alpinebits version does not match one of the servers supported
versions
```

Missing Client ID

(optional) No valid HTTP Header `X-AlpineBits-ClientID` provided.

`400 Bad Request`

```
ERROR:no valid client id provided
```

Unsupported GZIP compression

(optional) The server does not support requests compressed with gzip. A client can verify this by checking for the presence of the HTTP response header `X-AlpineBits-Server-Accept-Encoding` set to `gzip`.

`400 Bad Request`

```
ERROR:unsupported GZIP compression
```

Unsupported content type

The request body was not sent with the HTTP header `Content-Type` set to `application/xml`, and

the server does not serve any earlier version for which that *Content-Type* would be valid.

`400 Bad Request`

```
ERROR:unsupported content type
```

Content type does not match the client protocol version

The server serves more than one *Content-Type* from the same URL, but the *Content-Type* of the request and the value of `X-AlpineBits-ClientProtocolVersion` do not agree (see section 2.1).

`400 Bad Request`

```
ERROR:content type does not match the client protocol version
```

Unknown or missing action

No valid HTTP Header `X-AlpineBits-Action` provided, or its value is not among the actions supported by the server.

`400 Bad Request`

```
ERROR:unknown or missing action
```

XML validation error

`400 Bad Request`

```
ERROR:XML validation error
```

2.2.3. Internal Server Errors

Any unexpected error may include an optional message, which must be prefixed with `ERROR:`.

`500 Internal Server Error`

2.2.4. Data Validation Errors

If on protocol level the outcome is 200, it means the message could be processed correctly and there have not been any errors on protocol level. However the message can still contain data inconsistencies or business logic errors. These cases are explained in the dedicated Chapter 5.

`200 OK`

2.3. AlpineBits® responses and error handling

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_HotelInvCountNotifRS` 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_HotelInvCountNotifRS` 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 client's request could be completely received, correctly parsed, was deemed syntactically valid and its contents could be processed successfully by the server. All business rules were satisfied. In case of a successful outcome the response contains **one** empty **`Success`** element, **no** **`Warnings`** and **no** **`Errors`**:

The client, in order to safeguard the validity of the transmitted data, should always make sure that the following conditions are true:

- The XML composed by the client does represent the actual data.
- All the checks that are indicated in this document as "client responsibility" were performed by the client.

```
<!-- 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 successful outcome.

Responses with an AlpineBits® advisory outcome

As is the case for the successful 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_HotelInvCountNotifRS
        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_HotelInvCountNotifRS.xsd"
        Version="3.000">

        <Success/>
        <Warnings>
             <Warning Type="11">
```

```
                last full data set received more than 48 hours ago
            </Warning>
        </Warnings>

</OTA_HotelInvCountNotifRS>
```

files/samples/FreeRooms/FreeRooms-OTA_HotelInvCountNotifRS-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 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="3.000">

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

</OTA_HotelRatePlanNotifRS>
```

files/samples/RatePlans/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 as 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 without the mandatory attribute **`HotelCode`**. 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
        </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 automated way, this means alerting someone using appropriate means.

Note that a server that receives a request that is not authenticated, or that has a missing or invalid `X-AlpineBits-Action` field or request body, 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.

A server **could** perform additional checks after an outcome success answer (asynchronous checks). In case of inconsistencies with a previous stored state, a server **may** request a complete set of data as described below.

Responses with a request for complete sets of data

Along with the previous response cases, a server can request the client to send a full data set right after the current communication ends. The requested data **may** refer to different data from that received in the originating AlpineBits® request (For instance the server may request a full transfer of Availability information upon receiving a request for Rateplans).

In this case, the response contains one or more additional elements whose type depend on the outcome of the request:

- One or more **`Warning`** elements in case of a success, advisory or warning outcome.
- One or more **`Error`** elements in case of an error outcome.

These elements must be empty, must have the attribute **`Type`** set to `11` ("Advisory") and **must** have the mandatory attribute **`Status`** with a value chosen among the following, based on the desired complete data set:

- **`Status`** = `ALPINEBITS_SEND_HANDSHAKE` to request the handshake update
- **`Status`** = `ALPINEBITS_SEND_FREEROOMS` to request the full set of free rooms data
- **`Status`** = `ALPINEBITS_SEND_RATEPLANS` to request the full set of rate plans data
- **`Status`** = `ALPINEBITS_SEND_INVENTORY` to request the room configuration

If the server sends more than one of such elements, they must be referred to different data types, meaning that, at the current AlpineBits® version, a server can send up to three different requests: one for each of the above **`Status`** values.

Upon receiving a response with one or more of these elements, the client **should** transmit the corresponding data set as soon as possible in any order (see Tips and best practice).

The server **should** only send requests for kind of data supported by the client (the logic by which the server keeps track of these information is beyond the scope of this specification). The client **must** support receiving requests regarding any data, even if it is not directly supported/handled by it. The client **is free to ignore** request for data it does not handle, but might process such requests as it deems more appropriate, for instance by notifying the user.

Tips and best practice

A server **should not** request a complete data set in response to a complete set just received regarding the same data type. In this case, a client **must** ignore the request.

It is a client's responsibility to keep track of which data was requested by the server in case it receives more than one element requesting a different complete data set.

A server should use this feature with common sense and only when necessary in order to avoid useless heavy-data transfer.

A client should implement a precaution against too frequent complete set requests, for example limiting its number in a given timespan.

Some examples:

The server accepts the message and requests a full synchronization of FreeRooms data:

```
<Success/>
<Warnings>
    <Warning Type="11" Status="ALPINEBITS_SEND_FREEROOMS"></Warning>
</Warnings>
```

The server sends a warning together with a request for a full synchronization of RatePlans data:

```
<Success/>
```

```
<Warnings>
    <Warning Type="3">
        Rate elements referred to same InvTypeCode with overlapping periods
    </Warning>
    <Warning Type="11" Status="ALPINEBITS_SEND_RATEPLANS"></Warning>
</Warnings>
```

The server sends an error together with a request for a full synchronization of Inventory and FreeRooms data:

```
<Errors>
    <Error Type="13" Code="321">
        missing Start attribute in StatusApplicationControl element
    </Error>
    <Error Type="11" Status="ALPINEBITS_SEND_INVENTORY"></Error>
    <Error Type="11" Status="ALPINEBITS_SEND_FREEROOMS"></Error>
</Errors>
```

2.4. Implementation tips and best practice

- 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/2024-10
  - https://server.example.com/alpinebits/2026-10
  - etc…
- Gzip compression can make a request smaller by a factor of 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.
  - 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. Handshaking action

The handshaking action allows client and server to agree on the supported AlpineBits® versions and capabilities. Any AlpineBits® client or server **must** be able to handle this request. In the rest of this chapter the word "actor" will be used when referring to either client or server.

It is the **client's responsibility** to ensure the server can handle the actions it intends to perform. The client **should** update capabilities periodically. The server **may** request an update of the capabilities from the client by sending a response described in section 2.3 under paragraph "Responses with a request for complete sets of data"

Please note that this chapter was completely rewritten in AlpineBits® 2018-10 version, and it introduces breaking changes. This was needed to significantly streamline the handshaking progress.

The following table lists all available handshaking actions.

usage (since)	**mandatory**	`X-AlpineBits-Action` **field (string)**	**request body and server response (XML document)**
client sends its supported actions and capabilities (since 2018-10)	YES	`OTA_Ping:Handshaking`	OTA_PingRQ OTA_PingRS

3.1. Client request

The request body must contain an `OTA_PingRQ` document.

The **`EchoData`** element is a string representing a JSON object whose rules are defined below.

As mandated by OTA, the server, additional to it's response, **must** reply the content of the **`EchoData`** element identical to what the client sent. Therefore to avoid discrepancies and problems when parsing and comparing EchoData due to different indents or whitespaces, it's advisable to transmit the EchoData JSON string as a single line, without additional spaces.

The JSON object **must** adhere to the following rules:

- The root element is a JSON array with name `versions`.
- Each element of the array `versions` is a JSON object.

Each object of the `versions` array is built according to the following rules:

- A member with name `version` and value corresponding to a valid AlpineBits® version string **must** be present (see the first column of the changelog table, e.g. 2018-10 for the first version of AlpineBits® that supports handshaking)
- A JSON array with name `actions` **may** be present.

Each element of the `actions` array is a JSON object built according to the following rules:

- A member with the name `action` and value corresponding to one of the supported `actions` listed in section 3.3.
- A JSON array with name `supports`, listing all the supported `capabilities` for the sibling `action`.

The client must always include all versions he supports in the message and set the header field `X-AlpineBits-ClientProtocolVersion` to the highest version.

If there is no common version after the server did the intersection, no communication is possible. For more details on server responses in the handshake see section 3.2.

```
{"versions":[{"version":"2026-
10","actions":[{"action":"action_OTA_Read"},{"action":"action_OTA_HotelResNotif_GuestRequ
```

ests"},{"action":"action_OTA_HotelResNotif_GuestRequests_StatusUpdate"},{"action":"action_OTA_HotelInvCountNotif","supports":["OTA_HotelInvCountNotif_accept_rooms","OTA_HotelInvCountNotif_accept_categories","OTA_HotelInvCountNotif_accept_deltas","OTA_HotelInvCountNotif_accept_out_of_market","OTA_HotelInvCountNotif_accept_out_of_order","OTA_HotelInvCountNotif_accept_complete_set","OTA_HotelInvCountNotif_accept_closing_seasons"]},{"action":"action_OTA_HotelDescriptiveContentNotif_Inventory","supports":["OTA_HotelDescriptiveContentNotif_Inventory_use_rooms","OTA_HotelDescriptiveContentNotif_Inventory_occupancy_children"]},{"action":"action_OTA_HotelDescriptiveContentNotif_Info"},{"action":"action_OTA_HotelDescriptiveInfo_Inventory"},{"action":"action_OTA_HotelDescriptiveInfo_Info"},{"action":"action_OTA_HotelRatePlanNotif_RatePlans","supports":["OTA_HotelRatePlanNotif_accept_ArrivalDOW","OTA_HotelRatePlanNotif_accept_DepartureDOW","OTA_HotelRatePlanNotif_accept_RatePlan_BookingRule","OTA_HotelRatePlanNotif_accept_RatePlan_RoomType_BookingRule","OTA_HotelRatePlanNotif_accept_RatePlan_mixed_BookingRule","OTA_HotelRatePlanNotif_accept_Supplements","OTA_HotelRatePlanNotif_accept_FreeNightsOffers","OTA_HotelRatePlanNotif_accept_FamilyOffers","OTA_HotelRatePlanNotif_accept_full","OTA_HotelRatePlanNotif_accept_overlay","OTA_HotelRatePlanNotif_accept_RatePlanJoin","OTA_HotelRatePlanNotif_accept_OfferRule_BookingOffset","OTA_HotelRatePlanNotif_accept_OfferRule_DOWLOS"]},{"action":"action_OTA_HotelRatePlan_BaseRates","supports":["OTA_HotelRatePlan_BaseRates_deltas"]},{"action":"action_OTA_HotelPostEventNotif_EventReports"}]},{"version":"2024-10","actions":[{"action":"action_OTA_Read"},{"action":"action_OTA_HotelRatePlan_BaseRates","supports":["OTA_HotelRatePlan_BaseRates_deltas"]},{"action":"action_OTA_HotelPostEventNotif_EventReports"}]}]}

`files/samples/Handshake/Handshake-OTA_PingRQ.xml` - Handshake client request (see human readable JSON below)


```
{
    "versions": [
        {
            "version": "2026-10",
            "actions": [
                {
                    "action": "action_OTA_Read"
                },
                {
                    "action": "action_OTA_HotelResNotif_GuestRequests"
                },
                {
                    "action": "action_OTA_HotelInvCountNotif",
                    "supports": [
                        "OTA_HotelInvCountNotif_accept_rooms",
                        "OTA_HotelInvCountNotif_accept_categories",
                        "OTA_HotelInvCountNotif_accept_deltas",
                        ...
                    ]
                },
                ...
            ]
        },
        {
            "version": "2024-10",
            "actions": [
                {
                    "action": "action_OTA_Read"
                },
                {
                    "action": "action_OTA_HotelRatePlan_BaseRates",
                    "supports": [
                        "OTA_HotelRatePlan_BaseRates_deltas"
                    ]
                },
                {
                    "action": "action_OTA_HotelPostEventNotif_EventReports"
                }
            ]
        }
    ]
```

```
}
```

[small]Human readable JSON extract of the request above (for readability only: transmit it as a single line string as defined above)

For every successfully negotiated (see below) action, the client **must** set the `X-AlpineBits-ClientProtocolVersion` HTTP header according to the sibling `version` when performing the subsequent data exchange with the server.

In order to avoid misinterpretations, the AlpineBits® versions below `2018-10` **may** be part of the JSON object, but there **must not** be any `actions` array specified for them and will be ignored. The handshaking of the legacy versions **must** happen in subsequent requests following the legacy rules.

3.2. Server Response

The response is an `OTA_PingRS` document.

As mandated by OTA, the content of the **`EchoData`** element **must** be identical to what the client sent. To avoid discrepancies and problems when parsing and comparing EchoData due to different indents or whitespaces, it's advisable to transmit the EchoData JSON string as a single line, without additional spaces.

This assures the client that the server got and understood the complete message. In cases the server cannot process the request (e.g. due to syntax errors) he has to answer with an HTTP status 400 Bad request.

```
<Success/>
<Warnings>
  <Warning Type="11" Status="ALPINEBITS_HANDSHAKE">
    {
      "versions": [{
          "version": "2026-10",
          "actions": [{
                "action": "action_OTA_Read"
              },{
                "action": "action_OTA_HotelResNotif_GuestRequests"
              },{
                "action": "action_OTA_HotelResNotif_GuestRequests_StatusUpdate"
              },{
                "action": "action_OTA_HotelInvCountNotif",
                "supports": [
                  "OTA_HotelInvCountNotif_accept_categories",
                  "OTA_HotelInvCountNotif_accept_deltas",
                  ...]
              },
              ...
            ]
        },
        {
          "version": "2024-10",
          "actions": [{
                "action": "action_OTA_Read"
              },{
                "action": "action_OTA_HotelPostEventNotif_EventReports"
              }]
        }]
    }
  </Warning>
</Warnings>
<EchoData> Note: Return the exact same content as received by the client with PingRQ!
</EchoData>
```

`files/samples/Handshake/Handshake-OTA_PingRS.xml` - Handshake server response (see full file in samples)

The server **must** add to the response a **`Warning`** element with **`Type`** `11` and **`Status`** `ALPINEBITS_HANDSHAKE`. The content of this element **must** be the **intersection** between the client announced `versions`, `actions` and `capabilities` and what the server actually supports.

If a server receives a handshake request, he **must** answer with a list of all compatible versions, out of the list the client sent. The response **must** include only versions the server supports or knows. If there are no compatible versions, the versions list is empty.

The `versions` list **must** describe what the URL the request was sent to actually serves. A server that serves earlier versions from the same URL (see section 2.1) therefore announces them as well, even though their requests use a different *Content-Type*: the client already declared it speaks them, and it reaches them on the same URL by sending the *Content-Type* of the version it addresses. Conversely, a server that dedicates one URL to each version announces, on each of them, only the versions that URL serves.

This lets a single handshake describe a partial migration, where the server has moved only some actions to the newer version, and it lets a server retire an older version gracefully: it can stop announcing it while still accepting it, so that clients migrate as they refresh their capabilities, and stop accepting it once the traffic has ceased.

If the server supports a version, but no intersection of common actions is found, the server **must** return an empty actions list.

For versions 2024-10 or newer the action OTA_Ping is to be considered implicit and **must not** be indicated in the list of supported actions.

3.3. List of AlpineBits® supported `actions` and `capabilities`

- `action_OTA_Ping`
  handshaking action (support for this action is **mandatory**)
- `action_OTA_HotelInvCountNotif`
  the actor implements handling room availability notifications (FreeRooms)
- `OTA_HotelInvCountNotif_accept_rooms`
  for room availability notifications (FreeRooms), the actor handles availabilities for specific rooms
- `OTA_HotelInvCountNotif_accept_categories`
  for room availability notifications (FreeRooms), the actor handles availabilities for room categories
- `OTA_HotelInvCountNotif_accept_complete_set`
  for room availability notifications (FreeRooms), the actor handles CompletSet messages
- `OTA_HotelInvCountNotif_accept_deltas`
  for room availability notifications (FreeRooms), the actor handles partial information (deltas)
- `OTA_HotelInvCountNotif_accept_out_of_order`
  for room availability notifications (FreeRooms), the actor handles the number of rooms that are considered out of order
- `OTA_HotelInvCountNotif_accept_out_of_market`
  for room availability notifications (FreeRooms), the actor handles the number of rooms that are considered free but not bookable
- `OTA_HotelInvCountNotif_accept_closing_seasons`
  for room availability notifications (FreeRooms), the actor handles closing seasons
- `action_OTA_Read`
  the actor implements handling quote requests, booking reservations and cancellations (GuestRequests Pull)

- `action_OTA_HotelResNotif_GuestRequests`
  the actor implements pushing quote requests, booking reservations and cancellations (GuestRequests Push)
- `action_OTA_HotelResNotif_GuestRequests_StatusUpdate`
  the actor implements the status update for quote requests (GuestRequests Push status update)
- `action_OTA_HotelDescriptiveContentNotif_Inventory`
  the actor implements handling Inventory/Basic (push)
- `OTA_HotelDescriptiveContentNotif_Inventory_use_rooms`
  for room category information (Inventory), the actor needs information about specific rooms
- `OTA_HotelDescriptiveContentNotif_Inventory_occupancy_children`
  for room category information (Inventory), the actor supports applying children rebates also for children below the standard occupation
- `action_OTA_HotelDescriptiveContentNotif_Info`
  the actor implements handling Inventory/HotelInfo (push)
- `action_OTA_HotelDescriptiveInfo_Inventory`
  the actor implements handling Inventory/Basic (pull)
- `action_OTA_HotelDescriptiveInfo_Info`
  the actor implements handling Inventory/HotelInfo (pull)
- `action_OTA_HotelRatePlanNotif_RatePlans`
  the actor implements handling prices (RatePlans)
- `OTA_HotelRatePlanNotif_accept_ArrivalDOW`
  for prices (RatePlans), the actor accepts arrival DOW restrictions in booking rules
- `OTA_HotelRatePlanNotif_accept_DepartureDOW`
  for prices (RatePlans), the actor accepts departure DOW restrictions in booking rules
- `OTA_HotelRatePlanNotif_accept_RatePlan_BookingRule`
  for prices (RatePlans), the actor accepts "generic" booking rules
- `OTA_HotelRatePlanNotif_accept_RatePlan_RoomType_BookingRule`
  for prices (RatePlans), the actor 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 actor accepts both "specific" and "generic" booking rules. Both "generic" and "specific" rules capabilities **must** still be announced by the actor.
- `OTA_HotelRatePlanNotif_accept_Supplements`
  for prices (RatePlans), the actor accepts supplements
- `OTA_HotelRatePlanNotif_accept_FreeNightsOffers`
  for prices (RatePlans), the actor accepts free nights offers
- `OTA_HotelRatePlanNotif_accept_FamilyOffers`
  for prices (RatePlans), the actor accepts family offers
- `OTA_HotelRatePlanNotif_accept_full`
  for prices (RatePlans), the actor accepts complete/full/new RatePlans using notif type value `New`
- `OTA_HotelRatePlanNotif_accept_overlay`
  for prices (RatePlans), the actor accepts deltas for a RatePlan using notif type value `Overlay`
- `OTA_HotelRatePlanNotif_accept_RatePlanJoin`
  for prices (RatePlans), the actor supports grouping RatePlans with different MealPlanCodes under a single price list
- `OTA_HotelRatePlanNotif_accept_OfferRule_BookingOffset`
  for prices (RatePlans), the actor accepts the OfferRule restrictions MinAdvancedBookingOffset and MaxAdvancedBookingOffset
- `OTA_HotelRatePlanNotif_accept_OfferRule_DOWLOS`
  for prices (RatePlans), the actor accepts the OfferRule restrictions ArrivalDaysOfWeek, DepartureDaysOfWeek, SetMinLOS and SetMaxLOS

- `action_OTA_HotelRatePlan_BaseRates`
  the actor implements handling BaseRates
- `OTA_HotelRatePlan_BaseRates_deltas`
  the actor supports delta information with BaseRates
- `action_OTA_HotelPostEventNotif_EventReports`
  the actor implements handling hotel internal activities

AlpineBits® **requires** an actor to support at least all mandatory handshaking 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.

3.4. Unknown or missing actions

Upon receiving a request with an unknown or missing value in the `X-AlpineBits-Action` field, the server response is the string: `ERROR:unknown or missing action`.

Please note that the legacy "housekeeping actions" (namely: `getVersion` and `getCapabilities`) have been removed from this version of AlpineBits® and will yield to such a response.

3.5. Implementation tips and best practice

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

  Note that its value is the version of that specific OTA message, not a single value shared by all messages: OTA assigns a version number to every message schema and it differs between messages and even between the request and the response of the same pair (e.g. in OTA2015A `OTA_HotelDescriptiveContentNotifRQ` is at 8.000 while its response is at 3.000). The samples that come with this document carry the OTA2015A version of the message they show, taken from the `version` attribute of the corresponding OTA schema file.

- The handshaking action allows actors that are able to support multiple AlpineBits® versions to negotiate the best combinations of versions and capabilities with a single request. The suggested approach for a client is to list all the AlpineBits® versions and the related `actions` / `capabilities` it supports in a single request.
- Since the server is performing an **intersection** between the received JSON object and its supported versions and capabilities the response might be an empty JSON object (`{}`), this is an indication that there was a failure in parsing the JSON of the request.
- A client that receives an `ERROR` response to a handshaking request **should** consider that the endpoint may serve only versions earlier than 2026-10, whose requests use a different *Content-Type* (see section 2.1), before concluding that its own request was malformed. Since the handshaking request itself uses the *Content-Type* of the version the client declares, an endpoint that does not know that *Content-Type* cannot parse the request and cannot report the version mismatch in a meaningful way. This is the one case the handshaking action cannot negotiate on its own.
- When a client upgrades its AlpineBits® version, it is advised for it to re-configure all connections with its partner servers (provided they support the new version) starting from the handshaking and dealing with each data exchange action as a first synchronization in order to overwrite servers' data. This is to avoid any ambiguity or data-loss due to possible breaking changes between the old and new version. Examples of such breaking changes could be found when upgrading to version 2017-10 (or higher) with the introduction of static rates or when upgrading to version 2020-10 (or higher) with the re-writing of the FreeRooms action introducing the new out-of-order categories. In case a server receives a delta message with an AlpineBits® version which is inconsistent with the previously stored data, a server is free to decide whether to alert the client by sending a warning, to delete the stored data and/or to request a full re-synchronisation from the client (complete set).
- A server may inform the client regarding the deprecation of the current request through the headers `Deprecation` and `Link`, as defined by the IETF (https://datatracker.ietf.org/doc/draft-ietf-httpapi-

deprecation-header). These headers may be added to the response of any type of action that the server deems appropriate to deprecate, and applies to that specific action.

For the complete explanation of the headers, their syntax and use cases, please refer to the source linked above. As example, one endpoint serving an obsolete version of AlpineBits could be returning the following headers:

```
Deprecation: Mon, 19 Oct 2020 23:59:59 GMT
Link: <https://developer.alpinebits.org/deprecation.txt>; rel="deprecation";
type="text/plain"
```

The linked resource may contain information about the deprecation that happened in the specified date (e.g. introducing a new endpoint replacing the one the client is currently using), sharing details about it. Since this information is not useful for users, a client is not required to show it and may choose to collect it and report it to its developers or administration team. In case the scope of the deprecation is broader than the specific action, this information might be added to the linked resource. This information is meant as a helpful way to harmonize server and client versions, and it might be ignored by the client.

3.6. Tabular representation of OTA_PingRQ

Level	Element	Attribute	Type	Cardinality
	OTA_PingRQ		element	1
›	**EchoData**		string(1-∞)	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

3.7. Tabular representation of OTA_PingRS

Level	Element	Attribute	Type	Cardinality
	OTA_PingRS		element	1
Start choice				
›	**Errors**		element	1
› ›	**Error**		element	1 - 99
		Type	string enum (11|13)	1
		Code		0 - 1
		Status	string enum (ALPINEBITS_HANDSHAKE)	0 - 1
End choice				
Start choice				
›	**Success**		element	1
›	**Warnings**		element	0 - 1
› ›	**Warning**		element	1 - ∞
		Type		1

Level	Element	Attribute	Type	Cardinality
		Status	string enum (ALPINEBITS_HANDSHAKE)	0 - 1
›	**EchoData**		string(1-∞)	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4. Data exchange actions

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

For data exchange actions, both the `X-AlpineBits-Action` field and the request body are mandatory.

The request body (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	`X-AlpineBits-Action` field (string)	request body and server response (XML documents)
FreeRooms (since 2011-11)	a client sends room availability notifications to a server	`OTA_HotelInvCountNotif:FreeRooms`	`OTA_HotelInvCountNotifRQ` `OTA_HotelInvCountNotifRS`
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 (Push) (since 2018-10)	a client sends requests for a quote or booking requests to the server	`OTA_HotelResNotif:GuestRequests`	`OTA_HotelResNotifRQ` `OTA_HotelResNotifRS`
GuestRequests Status Update (Push) (since 2022-10)	a client sends status updates for a quote or booking requests to the server	`OTA_HotelResNotif:GuestRequests_StatusUpdate`	`OTA_HotelResNotifRQ` `OTA_HotelResNotifRS`
GuestRequests/Acknowledgments (since 2014-04)	a client acknowledges the requests it has received	`OTA_NotifReport:GuestRequests`	`OTA_NotifReportRQ` `OTA_NotifReportRS`
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 properties	`OTA_HotelDescriptiveContentNotif:Info`	`OTA_HotelDescriptiveContentNotifRQ` `OTA_HotelDescriptiveContentNotifRS`
Inventory/HotelInfo (Pull) (since 2017-10)	a client requests additional descriptive content regarding properties	`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`

known as (since)	usage	`X-AlpineBits-Action` field (string)	request body and server response (XML documents)
Activities (since 2020-10)	a client requests information about hotel activities	`OTA_HotelPostEventNotif:EventReports`	`OTA_HotelPostEventNotifRQ` `OTA_HotelPostEventNotifRS`

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

Caution must be taken when dealing with special and unusual characters (e.g. emojis, kanjis,…). Their use is increasing and where the user is allowed to enter arbitrary text (mostly from mobile devices) they could be inserted and sent through APIs. AlpineBits® and XML format can handle any character without issue but if unexpected, they might lead to unpredictable results, particularly on unprepared applications. Usually, supporting Unicode or UTF-8 encoding prevents most potential serious issues, but some unpleasant situations can still occur; for example:

- Incorrect visualization of certain characters might create confusion in text comprehension or in text formatting.
- Unicode characters bigger than 16 bits might not be supported yet by every system and can also cause alterations in stored data.
- Ambiguity between characters and their composite version (e.g. characters with diacritical marks) might create issues with some algorithms (e.g. searches, comparisons).

Therefore, it is a good practice to parse the data received with AlpineBits® for special characters prior to storing it. In worst cases this will lead only to a validation error rather than more unpleasant situations. More in general, it is advised to ensure the necessary checks in order to prepare its own application to receive special characters.

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_HotelInvCountNotifRQ` requests that do not indicate the hotel, this might make perfect sense in some context, but an AlpineBits® server will return an error if that information is missing from the request.

To aid developers, an AlpineBits® XML schema file is provided as an integral part of the specification in the AlpineBits® documentation kit.

The AlpineBits® schema file is 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.

Even though the XML snippets and sample files express prices in EUR, in AlpineBits® all currency codes from the ISO 4217 are allowed. It is important that clients and servers that exchange data with each other use the same currency code. In the event that a server receives unsupported currencies in a

message, it must reply to the client with a warning outcome. Likewise, a client that receives unsupported currencies should discard the whole message and notify the server in some way.

On copyright and license of exchanged contents

Many AlpineBits messages allow to transmit URLs to multimedia objects (mostly images), and whenever possible suggest the explicit identification of the copyright holder. However, in order to know its rights, the receiving end must also be aware of the licensing terms attached to these multimedia resource. Unfortunately there is no place for this information inside the XML messages, hence some other means of conveying it is suggested below as a best practice.

- The publisher of the multimedia object could add some metadata (e.g. EXIF in case of pictures) that contains the licensing information. To this end, AlpineBits recommends the utilization of a standard like the `IPTC` specification. The receiving end **should not** ignore nor remove this metadata and **should** also include them in a derived work (e.g. thumbnails or resized representations).
- The publisher of the multimedia object could add some metadata to the HTTP response that serves the multimedia content, to this end AlpineBits recommends the usage of `X-AlpineBits-License` and `X-AlpineBits-CopyrightHolder` headers. The receiving end **should not** ignore these headers and **should** consider also any derived work to be under the same constraints.

Please note that these approaches are not mutually exclusive. Furthermore it is recommended to provide as much information as possible within each data exchange.

4.1. FreeRooms: room availability notifications

When the value of the `X-AlpineBits-Action` field is `OTA_HotelInvCountNotif: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_HotelInvCountNotif_accept_rooms` or `OTA_HotelInvCountNotif_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 (a better explanation will follow).

4.1.1. Client request

The request body must contain an `OTA_HotelInvCountNotifRQ` document.

Consider the outer part of the example document:

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

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

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

    <Inventories HotelCode="123" HotelName="AlpineBits Hotel">

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

    </Inventories>

</OTA_HotelInvCountNotifRQ>
```

`files/samples/FreeRooms/FreeRooms-OTA_HotelInvCountNotifRQ.xml` - outer part

An `OTA_HotelInvCountNotifRQ` contains exactly **one `Inventories`** (note the plural) element, hence at most **one** hotel can be dealt with in a single request.

AlpineBits® requires the **mandatory** attribute **`HotelCode`** to be present and match information in the server's database. The attribute **`HotelName`** is **optional**.

Second, consider the inner part of the example that contains **`Inventory`** elements (note the singular) for three different time periods.

```
<Inventory>

    <StatusApplicationControl Start="2027-08-01" End="2027-08-10"
                              InvTypeCode="DOUBLE" />
     <InvCounts>

          <InvCount CountType="2" Count="3" />

     </InvCounts>

</Inventory>

<Inventory>

    <StatusApplicationControl Start="2027-08-11" End="2027-08-20"
                              InvTypeCode="DOUBLE" />
```

```
</Inventory>

<Inventory>

    <StatusApplicationControl Start="2027-08-21" End="2027-08-30"
                              InvTypeCode="DOUBLE" />
     <InvCounts>

          <InvCount CountType="2" Count="1" />

     </InvCounts>

</Inventory>
```

`files/samples/FreeRooms/FreeRooms-OTA_HotelInvCountNotifRQ.xml` - inner part

The absence of the **`InvCode`** attribute tells us we're dealing with a room category (DOUBLE) given by the **`InvTypeCode`**.

Alternatively, using a **`InvCode`** together with a **`InvTypeCode`** attribute would indicate that the availability refers to a **specific room**, not a room category.

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

An AlpineBits® server that implements the FreeRooms action **must** be able to treat **at least** one out of two cases (specific rooms or categories). A client should perform the `OTA_Ping:Handshaking` action to find out whether the server treats the room case (token `OTA_HotelInvCountNotif_accept_rooms`), the category case (token `OTA_HotelInvCountNotif_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.

```
<!-- room 101 is bookable -->
<Inventory>
    <StatusApplicationControl Start="2027-08-01" End="2027-08-10" InvTypeCode="DOUBLE"
InvCode="101" />
    <InvCounts>
        <InvCount CountType="2" Count="1" />
    </InvCounts>
</Inventory>

<!-- room 102 is booked: no InvCounts at all means fully booked -->
<Inventory>
    <StatusApplicationControl Start="2027-08-01" End="2027-08-10" InvTypeCode="DOUBLE"
InvCode="102" />
</Inventory>
```

`files/samples/FreeRooms/FreeRooms-OTA_HotelInvCountNotifRQ-rooms.xml` - availability reported for specific rooms

The **`Inventory`** element for a given room/room category can contain an **`InvCounts`** (plural) element. Based on the supported capabilities, the **`InvCounts`** element contains from 1 to 3 **`InvCount`** (singular) elements, one for each of the allowed **`CountType`** attributes:

- **`CountType`** = 2 indicates rooms that are **bookable** (this **`CountType`** **must** be supported).
- **`CountType`** = 6 indicates rooms that are **out of order** (i.e. rooms that are not booked nor bookable, maybe due to renovations), requires the `OTA_HotelInvCountNotif_accept_out_of_order` capability.

- **`CountType`** = 9 indicates rooms that are **available but not bookable** (see section GuestRequests for the description of bookings), requires the `OTA_HotelInvCountNotif_accept_out_of_market` capability.

If the server does not have the corresponding capability set, a client **must not** send the **`CountType`** = 6 or the **`CountType`** = 9 elements, and all the rooms not specified in the **`CountType`** = 2 element are to be considered as already booked.

```
<Inventory>
    <StatusApplicationControl Start="2027-08-01" End="2027-08-10" InvTypeCode="DOUBLE" />
    <InvCounts>
        <!-- 2 -> bookable -->
        <InvCount CountType="2" Count="5" />
        <!-- 6 -> out of order: neither booked nor bookable -->
        <InvCount CountType="6" Count="2" />
        <!-- 9 -> free but not bookable through this channel -->
        <InvCount CountType="9" Count="1" />
    </InvCounts>
</Inventory>
```

`files/samples/FreeRooms/FreeRooms-OTA_HotelInvCountNotifRQ-out-of-order-and-out-of-market.xml` - the three counts in a single Inventory element

The integer value of the attribute **`Count`** indicates the room count for the given **`CountType`** and must be a non negative number (≥ 0).

If an **`InvCount`** element for a specific **`CountType`** is missing it must be considered as if its **`Count`** attribute is 0. Likewise, if the whole **`InvCounts`** container is missing it is considered as if the whole room/room category is fully booked in the given time period (same as **`CountType`** = 2 with **`Count`** = 0).

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

Since in the example we're dealing with a room category (**`InvCode`** is missing), the value of **`Count`** can be > 1. In the case of a specific room (**`InvCode`** is specified), the only meaningful value would be 1 (the same room can not be "counted" more than once).

Since overbooking is not allowed and the number of counted rooms (bookable or not) cannot exceed the total number of rooms, the inequality 0 ≤ 'sum of **`Count`** values' ≤ 'total number of rooms' holds.

Considering the example, the attributes **`Start`** and **`End`** and the corresponding **`InvCount`** element indicate that room category DOUBLE has 3 rooms available from 2027-08-01 to 2027-08-10 and 1 room available from 2027-08-21 to 2027-08-30; from 2027-08-11 to 2027-08-20 there are no DOUBLE rooms available.

Regarding the first interval, this means the earliest possible check-in is 2027-08-01 afternoon and latest possible check-out is 2027-08-11 morning (maximum stay is 10 nights).

Check-ins **after** 2027-08-01 and stays of **less** than 10 nights are allowed as well, provided the check-out is not later than the morning of 2027-08-11.

Same applies for the third block of 10 nights from 2027-08-21 to 2027-08-30 (latest check-out is 2027-08-31 morning).

During the second time period, from 2027-08-11 to 2027-08-20, no check-in is possible.

Note that AlpineBits® does **not allow** **`Inventory`** elements with overlapping periods regarding the same room or room category. This implies that the order of the **`Inventory`** 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.

CompleteSet

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, AlpineBits® considered several use-cases in which it is important to transmit the complete availability information, as might be the case for a first synchronization or for aligning two systems after a communication issue. For this reason it defines a request that has the purpose of resetting the server information with that contained in the message. Such request is called a `CompleteSet`.

If a server provides the capability `OTA_HotelInvCountNotif_accept_complete_set`, then a client **may** send a `CompleteSet` message as explained below.

In the example, 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®.

In a `CompleteSet` message **all** the rooms/room categories managed by the client must be specified for any time period for which the client has data on records. This means that also in cases where a room/room category is fully booked (with no availability), it must have a corresponding **`Inventory`** element with its own **`StatusApplicationControl`** sub element and a coherent set of **`InvCount`** elements contained in it (or no **`InvCounts`** container at all, as stated above). This states unambiguously that the room/room category has no availability for the specified period.

A client **must not** include in a `CompleteSet` message a period for which it has no data on record (for instance a client that is forwarding data received from a source, **must not** forward any information besides what was originally received from the source).

Normally the FreeRooms message requires exactly one **`StatusApplicationControl`** element with attributes **`Start`**, **`End`**, **`InvTypeCode`** (and optional **`InvCode`**) for each **`Inventory`** element. The server **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 **`Inventory`** element without any attributes. The presence of the empty **`Inventory`** element is required for OTA validation.

In addition, 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 **should** then 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 flagging it as purged. Otherwise such a message should be considered equivalent to the one with **`Type`** = `16`.

Deltas

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.

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

AlpineBits® **recommends** that implementers that use delta requests **should** send the full set of information periodically if both client and server support it.

If the entire period is explicitly transmitted and processed in a delta message, this leads to the complete overwriting of the previous situation.

If a server does not support `CompleteSet` messages, it might lead to cases where the hotel would have

to intervene and delete obsolete data from the server manually in its backend.

A server **should** provide at least one of `OTA_HotelInvCountNotif_accept_deltas` and `OTA_HotelInvCountNotif_accept_complete_set` capabilities in order to accept FreeRooms messages.

Closing seasons

A special **`Inventory`** element can be used to send information about closing seasons, i.e. time periods in which the hotel is completely closed. It is in fact important to distinguish the cases in which a hotel is fully booked (where one could still hope for a last minute cancellation) from those in which a hotel is closed (where it could not even answer the phone).

Closing seasons elements require both parties to expose the `OTA_HotelInvCountNotif_accept_closing_seasons` capability during handshaking and can **only** be sent as first **`Inventory`** elements in a `CompleteSet` message. The content of the closing seasons **`Inventory`** element is defined as follows:

- **One `StatusApplicationControl`** sub element indicating a time period with the **mandatory `Start`** and **`End`** attributes and with the mandatory attribute **`AllInvCode`** set to `true`.
- **No `InvCounts`** sub elements are allowed.

Multiple **`Inventory`** elements can be specified this way to indicate multiple closing periods.

Each of the closing time periods **must not** overlap with other closed periods nor with any time period that states a room/room category is available (it is therefore possible for a closed period to overlap with a period where all the rooms/room categories are **unavailable**). A server **must** return an error if it detects such inconsistencies.

Delta messages are still to be considered as the most up to date, so in case a delta message would set a room/room category as available in a period previously set as closed, it must be considered as if the hotel revoked the closed period for the overlapping days.

As a best practice, the client should avoid such overwriting or at least send a complete set afterwards in order to explicitly set the correct closed periods. If the server deems it appropriate to force a full synchronization, it might request a `CompleteSet` in one of its next responses (see section 2.3).

4.1.2. Server response

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

See section 2.3 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.
- Note that this action has been completely re-written in the 2020-10 version of AlpineBits®.
- It is important that applications that forward data received from other sources (e.g. channel managers and alike) do not forward any information not present in the original data. For example, if such an application manages on its system a wider time frame than the one received from the source, it should not forward any information beyond the most future date it has received. This is necessary to keep the information clean through the forwarding without complementary (and arbitrary) data to be added by third party applications.
- For `CompleteSet` requests, since no time frame is explicitly transmitted by the client, a server is encouraged to delete and insert all 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.

4.1.4. Tabular representation of OTA_HotelInvCountNotifRQ

Level	Element	Attribute	Type	Cardinality
	OTA_HotelInvCountNotifRQ		element	1
›	**UniqueID**		element	0 - 1
		Type	string enum (16|35)	1
		ID	string(1-32)	1
		Instance	string enum (CompleteSet)	1
›	**Inventories**		element	1
› ›	**Inventory**		element	1 - ∞
› › ›	**StatusApplicationControl**		element	0 - 1
		Start	date (\S+)	1
		End	date (\S+)	1
		InvTypeCode	string(1-8)	0 - 1
		InvCode	string(1-16)	0 - 1
		AllInvCode	boolean (\S+)	0 - 1
› › ›	**InvCounts**		element	0 - 1
› › › ›	**InvCount**		element	1 - 3
		CountType	string enum (2|6|9)	1
		Count	integer ([0-9]+)	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.1.5. Tabular representation of OTA_HotelInvCountNotifRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelInvCountNotifRS		element	1
Start choice				
›	**Success**		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

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, reservations or cancellations with the `X-AlpineBits-Action` field 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 `X-AlpineBits-Action` field set to the value `OTA_NotifReport:GuestRequests`.

Please note that a server that has announced support for `OTA_Read:GuestRequests` MUST also support the action `OTA_NotifReport:GuestRequests` so that the client can execute follow-up acknowledgement requests. See section 4.2.3 and section 4.2.4 for more details.

Push Support

Starting with AlpineBits version 2018-10 it is possible to push the GuestRequest instead of polling them. Please note that this functionality relies on a few arrangements between the two parties that are not negotiated through AlpineBits, namely the endpoint (URL) where the client can receive the pushed GuestRequests, its credentials, etc. This behavior is described in section 4.2.6 and 4.2.7, but the XML structure of the **`HotelReservation`** is the same as in the rest of this chapter.

4.2.1. First client request

The `X-AlpineBits-Action` field is set to the value `OTA_Read:GuestRequests` and the request body 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
<?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="5.001">

    <ReadRequests>
        <HotelReadRequest HotelCode="123" HotelName="AlpineBits Hotel">
           <SelectionCriteria Start="2027-03-21T15:00:00+01:00"></SelectionCriteria>
       </HotelReadRequest>
    </ReadRequests>

</OTA_ReadRQ>
```

`files/samples/GuestRequests/GuestRequests-OTA_ReadRQ-selection-criteria.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 section 2.3 for details.

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

<OTA_NotifReportRS 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_NotifReportRS.xsd"
                   Version="3.000">

    <Success/>

</OTA_NotifReportRS>
```

`files/samples/GuestRequests/GuestRequests-Acknowledgments-OTA_NotifReportRS-success.xml` - the follow-up response, here reporting success

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

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

- `Requested` - this is a **request** for a quote, newsletter or catalog.
- `Reserved` - this is a booking **reservation**.
- `Modify` - this is a booking **modification**.
- `Cancelled` - this is a booking **cancellation**.

For a newsletter- or catalog **request** the optional attribute **`RoomStayReservation`** must be explicitly set to `false`.

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="2026-11-21T12:00:00+01:00"
                          ResStatus="Reserved">
```

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

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

            <Services>      <!--- optional services, see below-->        </Services>

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

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

        </HotelReservation>

    </ReservationsList>

</OTA_ResRetrieveRS>
```

`files/samples/GuestRequests/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 four parts: each **`HotelReservation`** contains the elements: **`RoomStays`**, **`Services`**, **`ResGuests`** and **`ResGlobalInfo`** discussed in the following paragraphs.

Modifications and Cancellations.

A booking modification (**`ResStatus`** is `Modify`) is identical to a booking reservation (**`ResStatus`** is `Reserved`). However, for a booking modification 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="2027-03-21T15:00:00+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>
```

files/samples/GuestRequests/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.

RoomStays

RoomStays is **mandatory** for **reservations**, **optional** for quote **requests**, **disallowed** for newsletter- or catalog **requests**.

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

```
<RoomStays>

    <RoomStay>

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

        <RatePlans>
            <RatePlan RatePlanCode="123456-xyz">
                <Commission Percent="15"/>
                <!-- Code 1 -> All inclusive -->
                <MealsIncluded MealPlanIndicator="true" MealPlanCodes="1"/>
            </RatePlan>
        </RatePlans>

        <RoomRates>
            <RoomRate> ... </RoomRate>
        </RoomRates>

        <!-- 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="2027-01-01" End="2027-01-12"/>

        <Guarantee>
            <GuaranteesAccepted>
                <GuaranteeAccepted>
                    <PaymentCard CardCode="VI"
                                 ExpireDate="1230">
                        <CardHolderName>Otto Mustermann</CardHolderName>
                        <CardNumber>
                            <PlainText>4444333322221111</PlainText>
                        </CardNumber>
                    </PaymentCard>
                </GuaranteeAccepted>
            </GuaranteesAccepted>
        </Guarantee>

        <Total AmountAfterTax="1125" CurrencyCode="EUR"/>

        <ServiceRPHs>
            <ServiceRPH RPH="1" />
            <ServiceRPH RPH="2" />
        </ServiceRPHs>

    </RoomStay>

</RoomStays>
```

`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation.xml` - **`RoomStay`** element common example
`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-allguests-and-company.xml` - **`RoomStay`** element with all guests and company

Each **`RoomStay`** element contains:

- **One `RoomType`** element (**mandatory** for **reservations**, **optional** for quote **requests**): see below for an 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`**; the **`Amount`** must follow the currency decimal format.

```
<RatePlans>
    <RatePlan RatePlanCode="STD-HB">
        <Commission>
            <CommissionPayableAmount Amount="82.50" CurrencyCode="EUR"/>
        </Commission>
        <MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
    </RatePlan>
</RatePlans>
```

`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-alternatives.xml` - a commission expressed as an amount instead of a percentage

- **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 `RoomRate`** element with:
    - A **`RatePlanCode`** attribute (**mandatory**).
    - A **`RoomTypeCode`** attribute (**mandatory**).
    - One or more **`Rate`** elements (**mandatory**), where each element contains:
      - An **`EffectiveDate`** (**mandatory**) attribute: Indicating the starting date.
      - An **`ExpireDate`** attribute: Indicating the ending date.
      - An **optional `ExpireDateExclusiveInd`** attribute (defaults to `false`): When `false`, it indicates that the **`ExpireDate`** is the last day of the applicable period (e.g. when expire date is Oct 15 the last date of the period is Oct 15). When `true`, it indicates that the **`ExpireDate`** is the first day after the applicable period (e.g. when expire date is Oct 15 the last date of the period is Oct 14).
      - An **`RateTimeUnit`** attribute: Indicating the time unit for the rate.
      - An **`UnitMultiplier`** attribute: Indicating the number of rate time units such as "3 Days".
      - A **`Base`** element (**mandatory**) : The base amount charged for the accommodation or service per unit of time (eg: Nightly, Weekly, etc). Note that any additional charges should be itemized in the other elements.
  - **One `GuestCounts`** element (**mandatory**) indicating the number of all adults (identified by **zero or 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 and their total (adults + children) **must not** be zero.

- **One `TimeSpan` element (mandatory)**: see below for an explanation.
- **One `PaymentCard` element, either in `DepositPayments` or `Guarantee` (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 `AmountAfterTax` must follow the currency decimal format.
- **One `ResGuestRPHs` element which holds a list of unsigned integers that reference the `ResGuestRPH` attribute in the `ResGuest` object. It is used to specify which guests occupy this room.

The `CardNumber` element is used to send a credit card number. The number can be sent **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").

```
<PaymentCard CardCode="VI" ExpireDate="1230">
    <CardHolderName>Otto Mustermann</CardHolderName>
    <CardNumber EncryptedValue="Vk15rP0iQ2xhdWRlU2FtcGxlRW5jcnlwdGVkQ2FyZE51bWJlcg==="
                EncryptionMethod="RSA-PKCSV2.2"/>
</PaymentCard>
```

`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-alternatives.xml` - a credit card number sent encrypted

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", 5 means "Pitches", etc.) wished by the guest.

The optional `RoomType` attribute **may** be specified, see the definition of the `RoomType` attribute used for the Inventory (see section 4.3.1) for the list of values defined by AlpineBits®. If the `RoomType` attribute is specified, then also `RoomClassificationCode` **must** be present.

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

TimeSpan

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**. The **End** attribute **must** be interpreted as exclusive: it is the departure date, not the last night of the stay. For example, a 4-night stay arriving on 23 May 2026 is encoded with **Start** = "2026-05-23" and **End** = "2026-05-27", i.e. the 23rd, 24th, 25th and 26th of May are the nights.

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.

A reservation can contain multiple **RoomStay** elements. Thus, different rooms with different RatePlans can be managed in one reservation. In addition, it is possible to book a stay of, for example, 10 days with a weekly flat rate and a standard offer for the remaining days.

The optional **RoomStayGroupID** attribute comes into play when alternative room stays or periods need to be transmitted. Note that this is valid for quote **requests** only (**ResStatus** is Requested) and they cannot be used together. Each room type with the same **RoomStayGroupID** indicates a group of room types which must be requested together, meaning that **RoomStayGroupID** set to "1" is the first alternative, **RoomStayGroupID** set to "2" the second alternative and so on. One or more alternatives can be specified by using a different identifier.

In general, a separate **RoomStay** element should be used for each room and rate plan.

RoomRates

RoomRates allow to track the exact costs of a room per day.

```
<RoomRates>
    <RoomRate RatePlanCode="123456-xyz" RoomTypeCode="SUITE">
        <Rates>
            <Rate EffectiveDate="2027-01-01" ExpireDate="2027-01-07"
ExpireDateExclusiveInd="false" RateTimeUnit="Day" UnitMultiplier="1">
                <Base AmountAfterTax="100" CurrencyCode="EUR"/>
            </Rate>
            <Rate EffectiveDate="2027-01-08" RateTimeUnit="Day">
                <Base AmountAfterTax="115" CurrencyCode="EUR"/>
            </Rate>
            <Rate EffectiveDate="2027-01-09" RateTimeUnit="Day">
                <Base AmountAfterTax="110" CurrencyCode="EUR"/>
            </Rate>
            <Rate EffectiveDate="2027-01-10" ExpireDate="2027-01-11"
ExpireDateExclusiveInd="false" RateTimeUnit="Day">
                <Base AmountAfterTax="100" CurrencyCode="EUR"/>
            </Rate>
        </Rates>
    </RoomRate>
</RoomRates>
```

`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation.xml` - **`RoomRates`** element

This covers two use cases:

1. In eras of revenue management it becomes more and more frequent that each day has its own unique rate. In order to be able to provide the hotel with data on how much revenue was made on each single day, this information needs to be as detailed as possible.
2. Some portals do grant discounts on the prices they receive from PMS. In order to be able to get the information which price has been offered to the guest back to the PMS, the reservation message has to contain the daily room rates (the base price of a single room on a daily basis).

NB: In a PMS (AlpineBits Client) such gross prices for the rooms often have to be split internally, either for internal cost accounting or for tax regulations (different tax rates). Therefore both the sold offer (rate plan) and the calculated room rate must be known in the PMS.

The amounts specified via the Rate/Base element should only define the room rates. Any other surcharges do not need to be specified in detail. Therefore, the total amount specified under ResGlobalInfo/Total may exceed the sum of the room rates.

Regarding **`RoomRates`** there are the following rules to be followed:

- Since there are separate **`RoomStay`** elements per room and rate plan, only one **`RoomRate`** element is required.
- The **`Rate`** elements defines the unit prices of a certain period per room and time unit.
- **`ExpireDate`** is **optional** and is usually specified inclusive. If this attribute is omitted, the price is valid only for **`EffectiveDate`**.
- The **optional** **`ExpireDateExclusiveInd`** attribute (defaults to `false`) is an indicator that **must** be used only in combination with **`ExpireDate`** and determines whether **`ExpireDate`** is exclusive (`true`) or inclusive (`false`).
- The amounts specified in the **`Base`** element should only define the room rates. Any other surcharges do not need to be specified in detail. Therefore, the total amount specified under ResGlobalInfo/Total may exceed the sum of the room rates.

Services

The **optional** `Services` element contains **one or more** `Service` elements. The `Service` element can be used to specify additional services booked with the reservation and also allows to give a more precise indication of the price composition.

```
<Services>
    <!-- Garage 10 Euro / day -->
    <Service Inclusive="false" Quantity="11" ServiceRPH="1" ServicePricingType="Per use"
             ServiceCategoryCode="OTHER" ServiceInventoryCode="garage1" ID="ab123"
Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="110" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Garage</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <!-- Ski depot, charged for every night of the stay -->
    <Service Inclusive="false" Quantity="7" ServiceRPH="5" ServicePricingType="Per night"
             ServiceCategoryCode="OTHER" ServiceInventoryCode="skidepot" ID="cd456"
Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="35" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Ski depot</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <!-- Spa entrance, charged once for every guest -->
    <Service Inclusive="false" Quantity="4" ServiceRPH="6" ServicePricingType="Per
person"
             ServiceCategoryCode="SPA" ServiceInventoryCode="spaentry" ID="ef789"
Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="60" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Spa entrance</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <!-- Supplement from a rate plan -->
    <Service Inclusive="true" Quantity="1" ServiceRPH="2" ServicePricingType="Per stay"
             ServiceCategoryCode="SUPPLEMENT" ServiceInventoryCode="0x539" ID="456"
Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="50" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Final cleaning</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <!--  Couple Massage -->
    <Service Inclusive="false" Quantity="1" ServiceRPH="3" ServicePricingType="Per use"
             ServiceCategoryCode="ACTIVITY"
             ServiceInventoryCode="M1" ID="555" Type="16">
        <ServiceDetails>
            <!-- optional guest counts -->
            <GuestCounts>
                <!-- 2 adults -->
                <GuestCount Count="2"/>
            </GuestCounts>
            <!-- optional time span -->
            <TimeSpan Start="2027-01-03T15:00:00+02:00" End="2027-01-03T17:30:00+02:00"/>
            <!-- optional guest comment -->
            <Comments>
                <Comment>
                    <Text TextFormat="PlainText">A guest comment</Text>
                </Comment>
```

```
            </Comments>
            <Total AmountAfterTax="50" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">couple massage</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <!-- Board price (2 guests * 7 nights) -->
    <Service Inclusive="true" Quantity="14" ServiceRPH="4" ServicePricingType="Per person
per night"
             ServiceCategoryCode="BOARD" ServiceInventoryCode="HB" ID="HB" Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="140" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Halfboard</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation.xml` - **`Services`** element

Each **`Service`** element contains:

- **One `ServiceDetails`** element with:
  - **One optional `GuestCounts`** element containing one or more **`GuestCount`** elements with the attribute **`Count`** and the optional attribute **`Age`**. The purpose of this element is to identify the number of guests and their age.
  - **One optional `TimeSpan`** element with the mandatory attribute **`Start`** and the optional attribute **`End`**. It specifies when the service is consumed. Both attributes are of OTA type `DateOrTimeOrDateTimeType` and, within the same **`TimeSpan`**, **must** use the same form (a date, a time, or a date-time). The **`End`** attribute **must** be interpreted as exclusive. When **`End`** is omitted, the service refers to the single day or time given by **`Start`**; recurring services **may** therefore be encoded either as a single **`Service`** with **`Start`** and **`End`**, or as one **`Service`** per occurrence with only **`Start`**. When the values carry a time component they **may** include a timezone offset; if the offset is omitted, the local timezone of the accommodation applies. See the dedicated examples at the end of this section.
  - **One optional `Comments`** element containing exactly one **`Comment`** element. The **`Comment`** element contains one **`Text`** element having the attribute **`TextFormat`** set to `PlainText`. This element should be used to add a comment that has been inserted by the customer who has booked the service.
  - **One mandatory `Total`** element with the mandatory attributes **`AmountAfterTax`** and **`CurrencyCode`**; the **`AmountAfterTax`** must follow the currency decimal format. The indicated amount must not be multiplied by the quantity of the service, but already represents the total cost of the service including all taxes.
  - **One optional `ServiceDescription`** element containing one **`Text`** element with the attribute **`TextFormat`** set to `PlainText`. This is used to specify the description of the service.
- The **mandatory** attribute **`ServiceCategoryCode`** is an enumerated type that identifies the category of the service. Allowed values are: `BOARD`, `SUPPLEMENT`, `SPA`, `FOOD`, `BEVERAGE`, `ACTIVITY`, `TOURISTTAX`, `TRANSFER`, `PACKAGE`, `RESOURCE`, `OTHER`
- The **mandatory** attribute **`ServiceInventoryCode`** for identification of the service in conjunction with the ServiceCategoryCode.
- The **mandatory** attribute **`ID`** attribute should be used to specify the unique identifier of the originating system. If the originating system does not manage such an ID, it should be set to the concatenated value of ServiceCategoryCode and ServiceInventoryCode to uniquely identify the service.

- The **mandatory** attribute **`ServicePricingType`** is an enumerated type that indicates how a service was priced. Allowed values are: `Per night`, `Per person`, `Per person per night`, `Per stay`, `Per use`.
- The **mandatory** attribute **`Inclusive`** indicates whether the price for this service is already included in the room rate.
- The **mandatory** attribute **`Quantity`** is used to indicate the quantity of the booked services. Also serves as the number of persons/nights when pricing class is per person or per person per night.
- The **optional** attribute **`ServiceRPH`** is a unique ID for a service that may be referenced in a **`RoomStay`** element of the HotelReservation

The following examples illustrate each valid combination of the optional **`TimeSpan`** element and its **`Start`** and **`End`** attributes. They are taken from a single 7-night stay (`2026-05-01` → `2026-05-08`, 2 guests).

Example 1: a per-stay welcome package with no **`TimeSpan`** element — the service refers to the whole stay.

```
<Services>
    <!-- Per stay welcome package, no TimeSpan -->
    <Service Inclusive="false" Quantity="1" ServiceRPH="1" ServicePricingType="Per stay"
             ServiceCategoryCode="PACKAGE" ServiceInventoryCode="welcome_pkg"
ID="PACKAGE_welcome_pkg" Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="30" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Welcome package</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

`samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml` - welcome package

Example 2: a per-use bike rental, **`TimeSpan`** with a full date-time including a timezone offset; **`End`** is exclusive (the bike is returned at 17:00 local time).

```
<Services>
    <!-- Per use bike rental, TimeSpan with full date-time and timezone offset;
         End is exclusive (the bike is returned at 17:00 local time) -->
    <Service Inclusive="false" Quantity="1" ServiceRPH="2" ServicePricingType="Per use"
             ServiceCategoryCode="RESOURCE" ServiceInventoryCode="bicycle"
ID="RESOURCE_bicycle" Type="16">
        <ServiceDetails>
            <TimeSpan Start="2026-05-02T09:00:00+02:00" End="2026-05-02T17:00:00+02:00"/>
            <Total AmountAfterTax="25" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Bicycle</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

`samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml` - bike rental

Example 3: a per-night babyphone, **`TimeSpan`** with a date range; **`End`** is the departure date (exclusive),

so the device covers the seven nights from `2026-05-01` through `2026-05-07`.

```
<Services>
    <!-- Per night babyphone, TimeSpan with date range; End = departure date (exclusive),
         so the device covers the seven nights from 2026-05-01 through 2026-05-07 -->
    <Service Inclusive="false" Quantity="7" ServiceRPH="3" ServicePricingType="Per night"
             ServiceCategoryCode="RESOURCE" ServiceInventoryCode="babyphone"
ID="RESOURCE_babyphone" Type="16">
        <ServiceDetails>
            <TimeSpan Start="2026-05-01" End="2026-05-08"/>
            <Total AmountAfterTax="35" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Babyphone</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

`samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml` - babyphone

Example 4: a per-stay fruit basket, **`TimeSpan`** with only **`Start`** — the service refers to that single day.

```
<Services>
    <!-- Per stay fruit basket, TimeSpan with Start only (delivered on 2026-05-03) -->
    <Service Inclusive="false" Quantity="1" ServiceRPH="4" ServicePricingType="Per stay"
             ServiceCategoryCode="FOOD" ServiceInventoryCode="fruit_basket"
ID="FOOD_fruit_basket" Type="16">
        <ServiceDetails>
            <TimeSpan Start="2026-05-03"/>
            <Total AmountAfterTax="20" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Fruit basket</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

`samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml` - fruit basket

Example 5: a per-use couple massage, **`TimeSpan`** with a full date-time including a timezone offset.

```
<Services>
    <!-- Per use couple massage, TimeSpan with full date-time and timezone offset -->
    <Service Inclusive="false" Quantity="1" ServiceRPH="5" ServicePricingType="Per use"
             ServiceCategoryCode="SPA" ServiceInventoryCode="massage_couple"
ID="SPA_massage_couple" Type="16">
        <ServiceDetails>
            <GuestCounts>
                <GuestCount Count="2"/>
            </GuestCounts>
            <TimeSpan Start="2026-05-04T15:00:00+02:00" End="2026-05-04T16:30:00+02:00"/>
            <Total AmountAfterTax="90" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Couple massage</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml - couple massage

Example 6: a per-person-per-night city tax with no **TimeSpan** element. **Quantity** is 2 guests × 7 nights = 14.

```
<Services>
    <!-- Per person per night city tax, no TimeSpan (Quantity = 2 guests * 7 nights) -->
    <Service Inclusive="false" Quantity="14" ServiceRPH="6" ServicePricingType="Per
person per night"
             ServiceCategoryCode="TOURISTTAX" ServiceInventoryCode="city_tax"
ID="TOURISTTAX_city_tax" Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="14" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">City tax</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml - city tax

Example 7: ski lessons on three consecutive days, encoded as one **Service** per occurrence (each **TimeSpan** has only **Start**). This is equivalent to a single **Service** with **Start** = "2026-05-05" and **End** = "2026-05-08".

```
<Services>
    <!-- Per use ski lessons, three consecutive days encoded as one Service per
occurrence
         (each TimeSpan has only Start); equivalent to a single Service with Start and
End -->
    <Service Inclusive="false" Quantity="1" ServiceRPH="7" ServicePricingType="Per use"
             ServiceCategoryCode="ACTIVITY" ServiceInventoryCode="ski_lesson"
ID="ACTIVITY_ski_lesson" Type="16">
        <ServiceDetails>
            <TimeSpan Start="2026-05-05"/>
            <Total AmountAfterTax="60" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Ski lesson</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <Service Inclusive="false" Quantity="1" ServiceRPH="7" ServicePricingType="Per use"
             ServiceCategoryCode="ACTIVITY" ServiceInventoryCode="ski_lesson"
ID="ACTIVITY_ski_lesson" Type="16">
        <ServiceDetails>
            <TimeSpan Start="2026-05-06"/>
            <Total AmountAfterTax="60" CurrencyCode="EUR"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Ski lesson</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
    <Service Inclusive="false" Quantity="1" ServiceRPH="7" ServicePricingType="Per use"
             ServiceCategoryCode="ACTIVITY" ServiceInventoryCode="ski_lesson"
ID="ACTIVITY_ski_lesson" Type="16">
        <ServiceDetails>
            <TimeSpan Start="2026-05-07"/>
            <Total AmountAfterTax="60" CurrencyCode="EUR"/>
            <ServiceDescription>
```

```
                <Text TextFormat="PlainText">Ski lesson</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation-services-matrix.xml - ski lessons

Example 8: a per night bathrobe, **TimeSpan** with date.

```
<Services>
    <!-- Start & End mandatory; ServicePricingType = per use / per person / per night /
per person per night; the same behavior for ServiceCategoryCode = FOOD / BEVERAGE /
ACTIVITY / TRANSFER -->
    <Service Inclusive="false" Quantity="5" ServiceRPH="1" ServicePricingType="Per night"
ServiceCategoryCode="OTHER" ServiceInventoryCode="bathrobe" ID="a123" Type="16">
        <ServiceDetails>
            <Total AmountAfterTax="14.99" CurrencyCode="EUR"/>
            <TimeSpan Start="2026-05-01" End="2026-05-06"/>
            <ServiceDescription>
                <Text TextFormat="PlainText">Bathrobe</Text>
            </ServiceDescription>
        </ServiceDetails>
    </Service>
</Services>
```

Service example - bathrobe

Calculation of AmountAfterTax

The **AmountAfterTax** of the **Total** element of a **Service** is the final price of the service and is not multiplied any further. It is calculated as the price per unit of the service multiplied by the quantity:

AmountAfterTax = price per unit × quantity

For example, a bathrobe at 3 EUR per night, booked for 5 nights, has an **AmountAfterTax** of 15.

The quantity depends on the **ServicePricingType**:

ServicePricingType	quantity
Per night	the maximum number of nights
Per person	the number of guests in the reservation
Per person per night	the number of guests × the maximum number of nights
Per stay	1
Per use	the number selected by the guest

ResGuests

The technically **optional** but recommended **`ResGuests`** element is dedicated to transmit personal information about the guests which will physically stay in the hotel structure.

Nested inside the **`ResGuests`** element there **must** be at least **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.
- The **`ResGuestRPH`** attribute is a reference placeholder, used as an index for this guest in this reservation. It is used by the RoomStay object to indicate which guests are associated with that room stay.

These 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>
```

`files/samples/GuestRequests/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.

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>
                Do you allow pets? sincerely, John Doe
            </Text>
        </Comment>
        <Comment Name="additional info">
            <Text>
                This is a collective request.
            </Text>
        </Comment>
    </Comments>

    <SpecialRequests>
        <SpecialRequest CodeContext="ALPINEBITS" RequestCode="PETS">
            <Text TextFormat="PlainText">two dogs</Text>
        </SpecialRequest>
        <SpecialRequest CodeContext="HOTEL" RequestCode="LATE_CHECKOUT">
            <Text TextFormat="PlainText"></Text>
        </SpecialRequest>
    </SpecialRequests>

    <DepositPayments>
       <!--- optional payments, see below-->
    </DepositPayments>

    <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>
        <!-- The booking ID of the original booking source -->
        <HotelReservationID ResID_Type="14"
                            ResID_Value="XYZ0123456"
                            ResID_Source="booking.com"/>

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

    <Profiles>
        <!--- optional profiles, see below-->
    </Profiles>

    <BasicPropertyInfo HotelCode="123" HotelName="AlpineBits Hotel"/>
</ResGlobalInfo>
```

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

The **mandatory** **`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. In most cases the AlpineBits® client software will just display this to a human hotel employee with no further processing. Each **`ListItem`** element **must** have **`Language`** and **`ListItem`** attributes. At most **one** **`ListItem`** element is allowed for each combination of **`Language`** and **`ListItem`**.
- **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.
- **One** **`Comment`** element (**optional**) with attribute **`Name`** set to `additional info` containing a single **`Text`** element freely filled with additional information NOT entered by the customer. In general this is used by the portal to provide useful information that is not covered by OTA elements.
- **One or more** **`SpecialRequest`** elements (**optional**). For further specification see dedicated chapter below.
- **One** **`DepositPayments`** element (**optional**). For further specification see dedicated chapter below.
- **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**.
  The **`HotelReservationID`** element can be used to track the original booking ID from the system where the guest completed the reservation. This booking ID must be transmitted via the attribute **`ResID_Value`** with **`ResID_Type`**="14". It represents the second identifier of this type, while the primary ID (assigned by the channel manager) is defined in the **`UniqueID`** element (see above).
  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`** to 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** **`Profiles`** element (**optional**). For further specification see dedicated chapter below.
- **One** mandatory **`BasicPropertyInfo`** element with the mandatory **`HotelCode`** and optional **`HotelName`** attributes following the same rules of that used in the corresponding attributes in the request (see section 4.2.1).

SpecialRequest

The **`SpecialRequest`** element has the following mandatory attributes:

- **`CodeContext`** set to `ALPINEBITS` or `HOTEL`.
- **`RequestCode`** set to a distinct value inside of the provided **`CodeContext`**.

SpecialRequests within **`CodeContext`** `ALPINEBITS` must contain a predefined **`RequestCode`** value from the following list:

- `PETS` providing details regarding pets

SpecialRequests within **`CodeContext`** `HOTEL` contain hotel-specific requests, in which the **`RequestCode`** value can be freely defined.

An **SpecialRequest** element can contain a single **Text** element (**optional**) with attribute **TextFormat** set to `PlainText`, stating details regarding the specific request. To express that a **SpecialRequest** element is explicitly not wanted by the customer, the **Text** element **must** be omitted. On the other hand it is **not allowed** to express absence inside of the **Text** element (e.g. “no pets“ or “none” should not be used). An empty **Text** element means that the request is wanted but provides no further details.

The absence of the whole **SpecialRequest** element means that the information about a potential presence of a **SpecialRequest** is not available (i.e. can not be guaranteed their presence nor absence).

Example A: the customer will surely bring pets described as “two dogs”.

```
<SpecialRequests>
    <SpecialRequest CodeContext="ALPINEBITS" RequestCode="PETS">
        <Text TextFormat="PlainText">two dogs</Text>
    </SpecialRequest>
</SpecialRequests>
```

SpecialRequest - PETS (example A)

Example B: the customer will surely bring pets but no further information is given.

```
<SpecialRequests>
    <SpecialRequest CodeContext="ALPINEBITS" RequestCode="PETS">
        <Text TextFormat="PlainText"></Text>
    </SpecialRequest>
</SpecialRequests>
```

SpecialRequest - PETS (example B)

Example C: the customer will surely not have any pet with him.

```
<SpecialRequests>
    <SpecialRequest CodeContext="ALPINEBITS" RequestCode="PETS"></SpecialRequest>
</SpecialRequests>
```

SpecialRequest - PETS (example C)

DepositPayments

The **DepositPayments** element contains one or more **GuaranteePayment** elements and is used to specify the information about the payments made with the booking.

```
<DepositPayments>
   <!-- example online payment -->
   <GuaranteePayment Start="2026-11-21T12:51:49"
GuaranteeCode="129acd49f7db2fabc317db3d9656525">
        <AcceptedPayments>
            <AcceptedPayment GuaranteeTypeCode="46" GuaranteeID="PAYPAL"
PaymentTransactionTypeCode="charge">
                <PaymentCard CardCode="VI" ExpireDate="1128">
                    <CardHolderName>Laura Sample</CardHolderName>
                    <CardNumber Mask="XXXXXXXXXXXX1234"
Token="39acd49f7db2fabc317db3d969592345" TokenProviderID="PAYPAL" />
                </PaymentCard>
```

```
            </AcceptedPayment>
        </AcceptedPayments>
        <AmountPercent Amount="100" CurrencyCode="EUR" />
    </GuaranteePayment>
    <!-- example bank transfer payment -->
    <GuaranteePayment Start="2026-11-21T12:51:49" GuaranteeCode="123456">
        <AcceptedPayments>
            <AcceptedPayment GuaranteeTypeCode="28" GuaranteeID="INTESA">
                <BankAcct>
                    <BankAcctName>Laura Sample</BankAcctName>
                    <BankAcctNumber Mask="IT12XXXXXXXXXXXXXX7990">
                        <PlainText>IT12 3456 7890 1234 5678 90</PlainText>
                    </BankAcctNumber>
                </BankAcct>
            </AcceptedPayment>
        </AcceptedPayments>
        <AmountPercent Amount="100" CurrencyCode="EUR" />
    </GuaranteePayment>
    <!-- example voucher payment -->
    <GuaranteePayment Start="2026-11-21T12:51:49" GuaranteeCode="123456">
        <AcceptedPayments>
            <AcceptedPayment GuaranteeTypeCode="3" GuaranteeID="GETAVO"
PaymentTransactionTypeCode="charge">
                <Voucher ElectronicIndicator="true" Identifier="XVZ-123-14050" />
            </AcceptedPayment>
        </AcceptedPayments>
        <AmountPercent Amount="100" CurrencyCode="EUR" />
    </GuaranteePayment>
    <!-- amount only pre-authorized, not charged yet -->
    <GuaranteePayment Start="2026-11-21T12:52:03" GuaranteeCode="7c1f9ab2e4d05631">
        <AcceptedPayments>
            <AcceptedPayment GuaranteeTypeCode="46" GuaranteeID="PAYPAL"
PaymentTransactionTypeCode="reserve">
                <PaymentCard CardCode="VI" ExpireDate="1128">
                    <CardHolderName>Laura Sample</CardHolderName>
                    <CardNumber Mask="XXXXXXXXXXXX1234"
Token="39acd49f7db2fabc317db3d969592345" TokenProviderID="PAYPAL" />
                </PaymentCard>
            </AcceptedPayment>
        </AcceptedPayments>
        <AmountPercent Amount="30" CurrencyCode="EUR" />
    </GuaranteePayment>
    <!-- part of an earlier charge given back -->
    <GuaranteePayment Start="2026-12-02T09:14:27"
GuaranteeCode="129acd49f7db2fabc317db3d9656525">
        <AcceptedPayments>
            <AcceptedPayment GuaranteeTypeCode="46" GuaranteeID="PAYPAL"
PaymentTransactionTypeCode="refund">
                <PaymentCard CardCode="VI" ExpireDate="1128">
                    <CardHolderName>Laura Sample</CardHolderName>
                    <CardNumber Mask="XXXXXXXXXXXX1234"
Token="39acd49f7db2fabc317db3d969592345" TokenProviderID="PAYPAL" />
                </PaymentCard>
            </AcceptedPayment>
        </AcceptedPayments>
        <AmountPercent Amount="40" CurrencyCode="EUR" />
    </GuaranteePayment>
</DepositPayments>
```

`files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation.xml` - **`DepositPayments`** element

Each **`GuaranteePayment`** element contains:

- **One** **`AcceptedPayments`** element containing exactly **one** **`AcceptedPayment`** element. It has the mandatory **`GuaranteeTypeCode`** which refers to the OpenTravel Code List Payment Type (PMT). We currently support 28 to specify a bank transfer payment, 46 for online payments and 3 if the

payment was done with a voucher. The mandatory attribute **GuaranteeID** is used this to identify the payment service provider and the optional **PaymentTransactionTypeCode** attribute is used this to specify if the amount was already charged, only reserved/pre-authorized or refunden. The allowed values ar either `charge`, `reserve` and `refund`.

Regarding the **PaymentTransactionTypeCode** the following guidelines should be considered: - The payment method and payment instrument must be identical to the previous transaction. - If the amount is identical, the payment is cancelled. If the payment differs (the refund amount must be less than or equal to the previously charged amount), a credit is initiated.

Nested under the **AcceptedPayment** element either an **PaymentCard**, **BankAcct** or **Voucher** element must be specified.

- The **PaymentCard** element should be used to describe online payments made with the booking. The optional attribute **CardCode** is a 2 digit code that references the credit card vendor. The optional attributes **ExpireDate** indicates the expiration date of the credit card. The optional **CardHolderName** element identifies the owner of the card. And the optional **CardNumber** element can be used to specify more detailed data about the payment made. It has the optional attribute **Mask** which is used to transmit a masked representation of the card number and the optional attributes **Token** and **TokenProviderID** which can be used to transmit the data about the credit card tokenization.
- The **BankAcct** element should be used to describe bank transfer payments. The optional **BankAcctName** could be used to specify the name of the bank accounts owner used for the transaction. And the optional **BankAcctNumber** can be used to describe more in detail the bank account used for the payment. It has the optional attribute **Mask** which is used to transmit a masked representation of the bank account number and the optional element **PlainText** which can be used to specify the entire bank account number.
- The **Voucher** element should be used to describe payments made with a voucher. The **ElectronicIndicator** is mandatory and must be always true. We currently support only electronic vouchers. The mandatory **Identifier** element should be used to specify the unique code of the voucher.
  - **One** **AmountPercent** element with the mandatory attributes **Amount** and **CurrencyCode**. The purpose of this element is to specify total amount of the payment.
  - The **mandatory** attribute **GuaranteeCode** is used to specify the unique identifier of the payment transaction
  - The **mandatory** attribute **Start** identifies the timestamp of the payment transaction.

Profiles

The **Profiles** element contains one **Profile** element nested within a **ProfileInfo** element with attribute **ProfileType** set to "4" with information about the booking channel;

```
<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>
```

files/samples/GuestRequests/GuestRequests-OTA_ResRetrieveRS-reservation.xml - **Profiles** element

The **Profile** element contains:

- The **required** **CompanyName** element with attributes **Code** and **CodeContext**.
- The **optional** **AddressInfo**, **TelephoneInfo** and **Email** elements: 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)

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 `X-AlpineBits-Action` field is set to the value `OTA_NotifReport:GuestRequests` and the request body 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 again uses 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.xsd"
        Version="4.000">

    <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"/>
                    <ResGlobalInfo>
                       <BasicPropertyInfo HotelCode="123"/>
                    </ResGlobalInfo>
                </HotelReservation>
```

```
                <HotelReservation>
                    <!-- ACK cancellation with ID="c24e8b15ca469388" -->
                    <UniqueID Type="15" ID="c24e8b15ca469388"/>
                    <ResGlobalInfo>
                       <BasicPropertyInfo HotelCode="123"/>
                    </ResGlobalInfo>
                </HotelReservation>

                <HotelReservation>
                    <!-- ACK quote request with ID="1000000000000001" -->
                    <UniqueID Type="14" ID="1000000000000001"/>
                    <ResGlobalInfo>
                       <BasicPropertyInfo HotelCode="123"/>
                    </ResGlobalInfo>
                </HotelReservation>

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

</OTA_NotifReportRQ>
```

files/samples/GuestRequests/GuestRequests-Acknowledgments-OTA_NotifReportRQ.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 section 2.3 for details.

4.2.5. Implementation tips and best practice

The attribute **RatePlanCode** is **mandatory** for reservations and is meant to refer a reservation to the originating RatePlan. If reservations are not originated from an AlpineBits® Rateplan this information may be unavailable. In this case it is recommended to set the value of the **RatePlanCode** attribute to the string "Unknown" (with a capital "U", please note that OTA schemas do not allow an empty value for this attribute).

Both occurrences of **RatePlanCode** in a **RoomStay**, the one on **RatePlan** and the one on **RoomRate**, refer to the same rate plan and **must** carry the same value. Like the rate plan code they refer to (see section 4.4), they are limited to 16 characters.

Every **CurrencyCode** attribute must be consistent with the originating RatePlan. In case of "Unknown" RatePlan, the currency must be one supported by the hotel or previously agreed upon for communication.

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 3/4 board (half board plus afternoon snack, hence a non-standard MealsIncluded combination) to their guests. This may be transmitted as half board, since 3/4 board 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

2. Request or reservation via landing page

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

3. Reservation on a booking platform

a. Bookings on booking portal (e.g. Booking.com)
ResID_Source (utm_source) = booking.com
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)

4.2.6. Requests Push client request

Starting with AlpineBits version 2018-10, a new capability was added (`action_OTA_HotelResNotif_GuestRequests`) to allow a push mechanism for the GuestRequest. If the capability is exposed, the Client is capable of acting as an AlpineBits Server, and the Server is capable of acting as an AlpineBits Client. This means that a few arrangements between the two parties have to be made, and those are not negotiated through AlpineBits, namely the endpoint (URL) where the client can receive the pushed GuestRequests, its credentials, etc.

The `X-AlpineBits-Action` field is set to the value `OTA_HotelResNotif:GuestRequests` and the request body must contain a `OTA_HotelResNotifRQ` document.

Please note that this message does not allow to specify a **HotelCode** or **HotelName** in the root element, this information is however provided in the corresponding attributes within each Request's **BasicPropertyInfo** element. All the Request exchanged with a single message **must** refer to the same Hotel.

The request **must** contain a single **HotelReservations** element with **one or more** **HotelReservation** elements containing the requested information. A client that has no information for the server **must not** initiate a transfer: an empty **HotelReservations** element is rejected by OTA2015A and, since release 2026-10, by the AlpineBits® schema as well. As for the content of the **HotelReservation** element, refer to section 4.2.2.

All the GuestRequests described above (e.g. Booking Requests, Quote Requests, Cancellations) can be pushed by using this mechanism.

4.2.7. Requests Push server response

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

See section 2.3 for details.

In case of success, the `OTA_HotelResNotifRS` response will contain a **single**, empty **Success** element followed by a **HotelReservations** element with **one or more** **HotelReservation** elements. Each **HotelReservation** element **must** contain only a single **UniqueID** element with the **Type** and **ID** attributes referring to one of the GuestRequests the Client sent (quotes, reservations or cancellations).

Each request that could not be processed **must** be listed in a **Warning** element, where the **RecordID** attribute refers to the request **ID** and **Type** and **Code** refer to the reason of the refusal.

If the server could process **none** of the requests, there is nothing to acknowledge: the **HotelReservations** element is then omitted altogether and the response consists of the **Success** element followed by a **Warnings** element listing every request of the batch. This is the **warning** outcome of section 2.3, so the client **must** consider the whole transfer as failed. The element **must not** be omitted when at least one request could be processed, and it is never empty: an empty **HotelReservations** element is rejected by OTA2015A and by the AlpineBits® schema.

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

<OTA_HotelResNotifRS 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_HotelResNotifRS.xsd"
                     Version="6.000">

    <Success/>

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

    <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>

</OTA_HotelResNotifRS>
```

files/samples/GuestRequests/GuestRequests-Push-OTA_HotelResNotifRS.xml

The following example shows a response to the same push in which the server could accept none of the requests: each of them is refused in a **Warning** element and no **HotelReservations** element is present.

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

<OTA_HotelResNotifRS 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_HotelResNotifRS.xsd"
                     Version="6.000">

    <Success/>

    <Warnings>
        <!-- refuse reservation with ID="6b34fe24ac2ff810" -->
        <Warning Type="3" Code="450" RecordID="6b34fe24ac2ff810">Unable to process
reservation</Warning>
        <!-- refuse cancellation with ID="c24e8b15ca469388" -->
        <Warning Type="3" Code="450" RecordID="c24e8b15ca469388">Unable to process
cancellation, unknown reservation</Warning>
    </Warnings>

    <!-- no HotelReservations element: there is nothing to acknowledge -->
```

```
</OTA_HotelResNotifRS>
```

files/samples/GuestRequests/GuestRequests-Push-OTA_HotelResNotifRS-all-refused.xml

4.2.8. Guest Requests status update

Starting with AlpineBits® version 2022-10, clients can send status updates of a given guest request to the server.

The `X-AlpineBits-Action` field is set to the value `OTA_HotelResNotif:GuestRequests_StatusUpdate` and the request body **must** contain a `OTA_HotelResNotifRQ` document.

The request **must** contain a single **`HotelReservations`** element with **one or more** **`HotelReservation`** elements containing the guest request status update.

Each **`HotelReservation`** element **must** contain the **mandatory** attributes **`ResStatus`** and **`CreateDateTime`** (the first timestamp recorded by the client for the guest request). The possible values for the **`ResStatus`** attribute expected by AlpineBits® are:

- Reserved - this is a booking reservation.
- Cancelled - this is a booking cancellation.

A **`HotelReservation`** element **may** contain the **`LastModifyDateTime`** attribute which identifies the timestamp of modification. If the **`LastModifyDateTime`** attribute is not specified, the timestamp of the request may be used instead.

Each **`HotelReservation`** element contains a **mandatory** **`UniqueID`** element which itself contains the **mandatory** **`ID`** and **`Type`** attributes. The **`ID`** attribute must match the unique **`HotelReservation`** identifier previously sent by the server. The **`Type`** attribute is set according to the OTA Unique Id Type (UIT) list.

The **`Type`** attribute value **must** be consistent with the **`ResStatus`** attribute of the surrounding **`HotelReservation`** element:

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

Below is an example of a status update sent from a client for some guest requests.

```
<HotelReservations>

    <HotelReservation ResStatus="Reserved" CreateDateTime="2027-03-21T11:00:00+01:00">
        <UniqueID Type="14" ID="6b34fe24ac999999"/>
    </HotelReservation>

    <HotelReservation ResStatus="Modify" CreateDateTime="2027-04-02T09:30:00+02:00">
        <UniqueID Type="14" ID="6b34fe24ac777777"/>
    </HotelReservation>
    <HotelReservation ResStatus="Cancelled" CreateDateTime="2027-05-12T19:00:00+01:00">
        <UniqueID Type="15" ID="6b34fe24ac244444"/>
    </HotelReservation>

</HotelReservations>
```

files/samples/GuestRequests/GuestRequests-Push-OTA_HotelResNotifRQ-StatusUpdate.xml

The server's response is the same as that described in Chapter 4.2.7.

4.2.9. Tabular representation of OTA_ReadRQ

Level	Element	Attribute	Type	Cardinality
	OTA_ReadRQ		element	1
›	ReadRequests		element	1
› ›	HotelReadRequest		element	1
› › ›	SelectionCriteria		element	0 - 1
		Start	dateTime (\S+)	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.2.10. Tabular representation of OTA_ResRetrieveRS

Level	Element	Attribute	Type	Cardinality
	OTA_ResRetrieveRS		element	1
Start choice				
›	Success		element	1
›	ReservationsList		element	1
› ›	HotelReservation		element	1 - ∞
		CreateDateTime	dateTime (\S+)	1
		LastModifyDateTime	dateTime (\S+)	0 - 1
		ResStatus	string enum (Requested|Reserved|Cancelled|Modify)	1
		RoomStayReservation	boolean (\S+)	0 - 1
› › ›	UniqueID		element	1
		Type	string enum (14|15)	1
		ID	string(1-32)	1
› › ›	RoomStays		element	0 - 1
› › › ›	RoomStay		element	1 - ∞
		RoomStayGroupID	string(1-32)	0 - 1
› › › › [5]	RoomTypes		element	0 - 1
› › › › [6]	RoomType		element	1
		RoomTypeCode	string(1-8)	0 - 1
		RoomClassificationCode		0 - 1
		RoomType	string enum (1|2|3|4|5|6|7|8|9)	0 - 1

Level	Element	Attribute	Type	Cardinality
› › › › 5	RatePlans		element	0 - 1
› › › › 6	RatePlan		element	1
		RatePlanCode	string(1-16)	0 - 1
› › › › 7	Commission		element	0 - 1
		Percent		0 - 1
› › › › 8	CommissionPayableAmount		element	0 - 1
		Amount	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
› › › › 7	MealsIncluded		element	0 - 1
		MealPlanIndicator	string enum (1|true)	1
		MealPlanCodes	string enum (1|3|10|12|14)	1
› › › › 5	RoomRates		element	0 - 1
› › › › 6	RoomRate		element	1
		RatePlanCode	string(1-16)	1
		RoomTypeCode	string(1-8)	1
› › › › 7	Rates		element	1
› › › › 8	Rate		element	1 - ∞
		EffectiveDate	date (\S+)	1
		ExpireDate	date (\S+)	0 - 1
		ExpireDateExclusiveInd	boolean (\S+)	0 - 1
		RateTimeUnit	string enum (Day)	0 - 1
		UnitMultiplier	integer ([0-9]+)	0 - 1
› › › › 9	Base		element	1
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
› › › › 5	GuestCounts		element	0 - 1
› › › › 6	GuestCount		element	1 - 99
		Count		1
		Age		0 - 1
› › › › 5	TimeSpan		element	1
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1

Level	Element	Attribute	Type	Cardinality
		Duration	string pattern P[0-9]+N	0 - 1
› › › › 6	**StartDateWindow**		element	0 - 1
		EarliestDate	date (\S+)	1
		LatestDate	date (\S+)	1
› › › › 5	**Guarantee**		element	0 - 1
› › › › 6	**GuaranteesAccepted**		element	1
› › › › 7	**GuaranteeAccepted**		element	1
› › › › 8	**PaymentCard**		element	1
› › › › 5	**Total**		element	0 - 1
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	1
› › › › 5	**ResGuestRPHs**		element	0 - 1
› › › › 5	**ServiceRPHs**		element	0 - 1
› › › › 6	**ServiceRPH**		element	1 - ∞
		RPH	string pattern [0-9]{1,8}	1
› › ›	**Services**		element	0 - 1
› › › ›	**Service**		element	1 - ∞
		ID	string(1-32)	1
		Type	string enum (16)	1
		ServiceCategoryCode	string enum (BOARD|SUPPLEMENT|SPA|FOOD|BEVERAGE|ACTIVITY|TOURISTTAX|TRANSFER|PACKAGE|RESOURCE|OTHER)	1
		ServiceInventoryCode	string(1-16)	1
		ServiceRPH	string pattern [0-9]{1,8}	0 - 1
		ServicePricingType	string enum (Per night|Per person|Per person per night|Per stay|Per use)	1
		Inclusive		1
		Quantity		1
› › › › 5	**ServiceDetails**		element	1
› › › › 6	**GuestCounts**		element	0 - 1
› › › › 7	**GuestCount**		element	1 - 99
		Age		0 - 1

Level	Element	Attribute	Type	Cardinality
		Count		1
› › › › 6	TimeSpan		element	0 - 1
		Start		1
		End		0 - 1
› › › › 6	Comments		element	0 - 1
› › › › 7	Comment		element	1
› › › › 8	Text		string(1-∞)	1
		TextFormat	string enum (PlainText)	1
› › › › 6	Total		element	1
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	1
› › › › 6	ServiceDescription		element	0 - 1
› › › › 7	Text		string(1-∞)	1
		TextFormat	string enum (PlainText)	1
› › ›	ResGuests		element	0 - 1
› › › ›	ResGuest		element	1 - ∞
		ResGuestRPH	string ([0-9]{1,8})	0 - 1
› › › › 5	Profiles		element	1
› › › › 6	ProfileInfo		element	1
› › › › 7	Profile		element	1
› › › › 8	Customer		element	1
› › ›	ResGlobalInfo		element	0 - 1
› › › ›	Comments		element	0 - 1
› › › › 5	Comment		element	1 - 3
		Name	string enum (included services|customer comment|additional info)	1
Start choice				
› › › › 6	ListItem		string(1-∞)	0 - ∞
		ListItem	integer ([0-9]+)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 6	Text		string(1-∞)	0 - 1
End choice				

Level	Element	Attribute	Type	Cardinality
› › › ›	SpecialRequests		element	0 - 1
› › › › 5	SpecialRequest		element	1 - ∞
		RequestCode	string(1-16)	1
		CodeContext	string enum (ALPINEBITS|HOTEL)	1
› › › › 6	Text		element	0 - 1
		TextFormat	string enum (PlainText)	1
› › › ›	DepositPayments		element	0 - 1
› › › › 5	GuaranteePayment		element	1 - ∞
		Start	dateTime (\S+)	0 - 1
		GuaranteeCode	string(1-32)	0 - 1
› › › › 6	AcceptedPayments		element	1
› › › › 7	AcceptedPayment		element	1
		GuaranteeTypeCode	string enum (3|28|46)	0 - 1
		GuaranteeID	string(1-64)	0 - 1
		PaymentTransactionTypeCode	string enum (charge|reserve|refund)	0 - 1
Start choice				
› › › › 8	PaymentCard		element	0 - 1
End choice				
Start choice				
› › › › 8	BankAcct		element	0 - 1
› › › › 9	BankAcctName		string(1-64)	0 - 1
› › › › 9	BankAcctNumber		element	1
		Mask	string pattern [0-9a-zA-Z]{1,32}	0 - 1
› › › › 10	PlainText		string(1-64)	1
End choice				
Start choice				
› › › › 8	Voucher		element	0 - 1
		ElectronicIndicator	boolean pattern 1|true	1
		Identifier	string(1-64)	1
End choice				
› › › › 6	AmountPercent		element	0 - 1
		Amount	decimal ([0-9]*\.?[0-9]*)	1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1

Level	Element	Attribute	Type	Cardinality
› › › ›	CancelPenalties		element	0 - 1
› › › › [5]	CancelPenalty		element	1
› › › › [6]	PenaltyDescription		element	1
› › › › [7]	Text		string(1-∞)	1
› › › ›	HotelReservationIDs		element	0 - 1
› › › › [5]	HotelReservationID		element	1 - ∞
		ResID_Type		1
		ResID_Value	string(1-64)	0 - 1
		ResID_Source	string(1-64)	0 - 1
		ResID_SourceContext	string(1-64)	0 - 1
› › › ›	Profiles		element	0 - 1
› › › › [5]	ProfileInfo		element	1 - 3
› › › ›	BasicPropertyInfo		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.2.11. Tabular representation of OTA_NotifReportRQ

Level	Element	Attribute	Type	Cardinality
	OTA_NotifReportRQ		element	1
›	Success		element	1
›	NotifDetails		element	0 - 1
› ›	HotelNotifReport		element	1
› › ›	HotelReservations		element	1
› › › ›	HotelReservation		element	1 - ∞
› › › › [5]	UniqueID		element	1
		Type	string enum (14|15)	1
		ID	string(1-32)	1
› › › › [5]	ResGlobalInfo		element	0 - 1
› › › › [6]	BasicPropertyInfo		element	1
›	Warnings		element	0 - 1
› ›	Warning		element	1 - ∞
		Type		1
		Code		1
		RecordID	string(1-64)	1

Level	Element	Attribute	Type	Cardinality
		Status	string enum (ALPINEBITS_SEND_HANDSHAKE|ALPINEBITS_SEND_FREEROOMS|ALPINEBITS_SEND_RATEPLANS|ALPINEBITS_SEND_INVENTORY)	0 - 1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.2.12. Tabular representation of OTA_NotifReportRS

Level	Element	Attribute	Type	Cardinality
	OTA_NotifReportRS		element	1
Start choice				
›	**Success**		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.2.13. Tabular representation of OTA_HotelResNotifRQ

Level	Element	Attribute	Type	Cardinality
	OTA_HotelResNotifRQ		element	1
›	**HotelReservations**		element	1
› ›	**HotelReservation**		element	1 - ∞
		CreateDateTime	dateTime (\S+)	1
		LastModifyDateTime	dateTime (\S+)	0 - 1
		ResStatus	string enum (Requested|Reserved|Cancelled|Modify)	1
		RoomStayReservation	boolean (\S+)	0 - 1
› › ›	**UniqueID**		element	1
		Type	string enum (14|15)	1
		ID	string(1-32)	1
› › ›	**RoomStays**		element	0 - 1
› › › ›	**RoomStay**		element	1 - ∞
		RoomStayGroupID	string(1-32)	0 - 1
› › › › [5]	**RoomTypes**		element	0 - 1
› › › › [6]	**RoomType**		element	1
		RoomTypeCode	string(1-8)	0 - 1

Level	Element	Attribute	Type	Cardinality
		RoomClassificationCode		0 - 1
		RoomType	string enum (1|2|3|4|5|6|7|8|9)	0 - 1
› › › › 5	RatePlans		element	0 - 1
› › › › 6	RatePlan		element	1
		RatePlanCode	string(1-16)	0 - 1
› › › › 7	Commission		element	0 - 1
		Percent		0 - 1
› › › › 8	CommissionPayableAmount		element	0 - 1
		Amount	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
› › › › 7	MealsIncluded		element	0 - 1
		MealPlanIndicator	string enum (1|true)	1
		MealPlanCodes	string enum (1|3|10|12|14)	1
› › › › 5	RoomRates		element	0 - 1
› › › › 6	RoomRate		element	1
		RatePlanCode	string(1-16)	1
		RoomTypeCode	string(1-8)	1
› › › › 7	Rates		element	1
› › › › 8	Rate		element	1 - ∞
		EffectiveDate	date (\S+)	1
		ExpireDate	date (\S+)	0 - 1
		ExpireDateExclusiveInd	boolean (\S+)	0 - 1
		RateTimeUnit	string enum (Day)	0 - 1
		UnitMultiplier	integer ([0-9]+)	0 - 1
› › › › 9	Base		element	1
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
› › › › 5	GuestCounts		element	0 - 1
› › › › 6	GuestCount		element	1 - 99
		Count		1
		Age		0 - 1

Level	Element	Attribute	Type	Cardinality
› › › › 5	TimeSpan		element	1
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1
		Duration	string pattern P[0-9]+N	0 - 1
› › › › 6	StartDateWindow		element	0 - 1
		EarliestDate	date (\S+)	1
		LatestDate	date (\S+)	1
› › › › 5	Guarantee		element	0 - 1
› › › › 6	GuaranteesAccepted		element	1
› › › › 7	GuaranteeAccepted		element	1
› › › › 8	PaymentCard		element	1
› › › › 5	Total		element	0 - 1
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	1
› › › › 5	ResGuestRPHs		element	0 - 1
› › › › 5	ServiceRPHs		element	0 - 1
› › › › 6	ServiceRPH		element	1 - ∞
		RPH	string pattern [0-9]{1,8}	1
› › ›	Services		element	0 - 1
› › › ›	Service		element	1 - ∞
		ID	string(1-32)	1
		Type	string enum (16)	1
		ServiceCategoryCode	string enum (BOARD|SUPPLEMENT|SPA|FOOD|BEVERAGE|ACTIVITY|TOURISTTAX|TRANSFER|PACKAGE|RESOURCE|OTHER)	1
		ServiceInventoryCode	string(1-16)	1
		ServiceRPH	string pattern [0-9]{1,8}	0 - 1
		ServicePricingType	string enum (Per night|Per person|Per person per night|Per stay|Per use)	1
		Inclusive		1
		Quantity		1
› › › › 5	ServiceDetails		element	1

Level	Element	Attribute	Type	Cardinality
› › › › 6	GuestCounts		element	0 - 1
› › › › 7	GuestCount		element	1 - 99
		Age		0 - 1
		Count		1
› › › › 6	TimeSpan		element	0 - 1
		Start		1
		End		0 - 1
› › › › 6	Comments		element	0 - 1
› › › › 7	Comment		element	1
› › › › 8	Text		string(1-∞)	1
		TextFormat	string enum (PlainText)	1
› › › › 6	Total		element	1
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	1
› › › › 6	ServiceDescription		element	0 - 1
› › › › 7	Text		string(1-∞)	1
		TextFormat	string enum (PlainText)	1
› › ›	ResGuests		element	0 - 1
› › › ›	ResGuest		element	1 - ∞
		ResGuestRPH	string ([0-9]{1,8})	0 - 1
› › › › 5	Profiles		element	1
› › › › 6	ProfileInfo		element	1
› › › › 7	Profile		element	1
› › › › 8	Customer		element	1
› › ›	ResGlobalInfo		element	0 - 1
› › › ›	Comments		element	0 - 1
› › › › 5	Comment		element	1 - 3
		Name	string enum (included services|customer comment|additional info)	1
Start choice				
› › › › 6	ListItem		string(1-∞)	0 - ∞
		ListItem	integer ([0-9]+)	1
		Language	([a-z][a-z])	1
End choice				

Level	Element	Attribute	Type	Cardinality
Start choice				
› › › › 6	Text		string(1-∞)	0 - 1
End choice				
› › › ›	SpecialRequests		element	0 - 1
› › › › 5	SpecialRequest		element	1 - ∞
		RequestCode	string(1-16)	1
		CodeContext	string enum (ALPINEBITS|HOTEL)	1
› › › › 6	Text		element	0 - 1
		TextFormat	string enum (PlainText)	1
› › › ›	DepositPayments		element	0 - 1
› › › › 5	GuaranteePayment		element	1 - ∞
		Start	dateTime (\S+)	0 - 1
		GuaranteeCode	string(1-32)	0 - 1
› › › › 6	AcceptedPayments		element	1
› › › › 7	AcceptedPayment		element	1
		GuaranteeTypeCode	string enum (3|28|46)	0 - 1
		GuaranteeID	string(1-64)	0 - 1
		PaymentTransactionTypeCode	string enum (charge|reserve|refund)	0 - 1
Start choice				
› › › › 8	PaymentCard		element	0 - 1
End choice				
Start choice				
› › › › 8	BankAcct		element	0 - 1
› › › › 9	BankAcctName		string(1-64)	0 - 1
› › › › 9	BankAcctNumber		element	1
		Mask	string pattern [0-9a-zA-Z]{1,32}	0 - 1
› › › › 10	PlainText		string(1-64)	1
End choice				
Start choice				
› › › › 8	Voucher		element	0 - 1
		ElectronicIndicator	boolean pattern 1|true	1
		Identifier	string(1-64)	1
End choice				

Level	Element	Attribute	Type	Cardinality
› › › › 6	AmountPercent		element	0 - 1
		Amount	decimal ([0-9]*\.?[0-9]*)	1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
› › › ›	CancelPenalties		element	0 - 1
› › › › 5	CancelPenalty		element	1
› › › › 6	PenaltyDescription		element	1
› › › › 7	Text		string(1-∞)	1
› › › ›	HotelReservationIDs		element	0 - 1
› › › › 5	HotelReservationID		element	1 - ∞
		ResID_Type		1
		ResID_Value	string(1-64)	0 - 1
		ResID_Source	string(1-64)	0 - 1
		ResID_SourceContext	string(1-64)	0 - 1
› › › ›	Profiles		element	0 - 1
› › › › 5	ProfileInfo		element	1 - 3
› › › ›	BasicPropertyInfo		element	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.2.14. Tabular representation of OTA_HotelResNotifRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelResNotifRS		element	1
Start choice				
›	Success		element	1
›	HotelReservations		element	0 - 1
› ›	HotelReservation		element	1 - ∞
› › ›	UniqueID		element	1
		Type	string enum (14|15)	1
		ID	string(1-32)	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.3. Inventory: room category information

The Inventory request allows up to four values for the `X-AlpineBits-Action` field (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)**).

4.3.1. Inventory/Basic (push) client request

The request body 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`**, an **`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.

Below is an example of the global structure 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="AlpineBits Hotel">
    <FacilityInfo>
      <GuestRooms>

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

        <GuestRoom Code="DOUBLE" MaxOccupancy="3" MinOccupancy="1" MaxChildOccupancy="1"
MaxAdultOccupancy="2">
          <!-- ... see below ... -->
        </GuestRoom>

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

```
        <GuestRoom Code="DOUBLE">
          <TypeRoom RoomID="101"/>
        </GuestRoom>

        <GuestRoom Code="DOUBLE">
          <TypeRoom RoomID="102"/>
        </GuestRoom>

       </GuestRooms>
      </FacilityInfo>

      <TPA_Extensions>
        <AdditionalCodes>
          ...
        </AdditionalCodes>
      </TPA_Extensions>

    </HotelDescriptiveContent>
  </HotelDescriptiveContents>

</OTA_HotelDescriptiveContentNotifRQ>
```

`files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-basic.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 4.4.2 "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.
- **`MaxAdultOccupancy`**: **optional**, must be **`StandardOccupancy`** ≤ **`MaxAdultOccupancy`** ≤ **`MaxOccupancy`**. This attribute can be used to specify how many adults are allowed in a room.
- **`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", "5" means "Pitches", etc…
  - **`RoomType`** (optional) to allow a more exact classification of the guest accommodation (see below)
  - **`Size`** (optional)
- **`Amenities:`** **optional** element containing a list of **`Amenity`** elements, each identified by the **mandatory** attribute **`RoomAmenityCode`** following the OTA list "Room Amenity Type" (RMA) or the AlpineBits "Room Amenity Type" list (ABR).
- **`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.
- **At most one** `MultimediaDescription` element with attribute `InfoCode` = 23 (Pictures) **may be** present.

Regarding the optional attribute `RoomType`, the allowed values are defined by AlpineBits® as follows:

- For rooms: `RoomType` value 1 (should only be used when `RoomClassificationCode` is 42).
- For apartments: `RoomType` value 2 (should only be used when `RoomClassificationCode` is 13).
- For mobile homes: `RoomType` value 3 (should only be used when `RoomClassificationCode` is 13).
- For bungalows: `RoomType` value 4 (should only be used when `RoomClassificationCode` is 13).
- For holiday homes: `RoomType` value 5 (should only be used when `RoomClassificationCode` is 13).
- For camping grounds (place for vehicles in a camping): `RoomType` value 6 (should only be used when `RoomClassificationCode` is 5).
- For pitches (place for tents in a camping): `RoomType` value 7 (should only be used when `RoomClassificationCode` is 5).
- For camping grounds/pitches (place for either tents or vehicles in a camping): `RoomType` value 8 (should only be used when `RoomClassificationCode` is 5).
- For resting places (beds in shared rooms, like hostels or mountain huts): `RoomType` value 9 (should only be used when `RoomClassificationCode` is 42).

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

```
<GuestRoom Code="DOUBLE" MaxOccupancy="3" MinOccupancy="1" MaxChildOccupancy="1"
MaxAdultOccupancy="2">

  <!-- RoomClassificationCode = "42" means Room, 13 Apartment, see OTA table GRI -->
  <TypeRoom StandardOccupancy="2" RoomClassificationCode="42" Size="28"/>

  <Amenities>
    <!-- 26 means Crib, see OTA table RMA -->
    <Amenity RoomAmenityCode="26"/>
    <!-- 7.ABR is Balcony, 59.ABR is Safe, see the AlpineBits list ABR -->
    <Amenity RoomAmenityCode="7.ABR"/>
    <Amenity RoomAmenityCode="59.ABR"/>
  </Amenities>

  <MultimediaDescriptions>

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

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

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

  </MultimediaDescriptions>

</GuestRoom>
```

files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-basic.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>
```

files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-basic.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 a 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="DOUBLE">
  <TypeRoom RoomID="101"/>
</GuestRoom>

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

files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-basic.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 previous 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="AlpineBits Hotel">
  <FacilityInfo>
    <GuestRooms>

      <!-- "SINGLE", formerly known as "EZ" -->

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

      <!-- "DOUBLE", formerly known as "DZ" -->

      <GuestRoom Code="DOUBLE" MaxOccupancy="3" 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".

AdditionalCodes

For exchanging accommodation identification codes the **`TPA_Extensions`** element will be used. This functionality is to comply with national and EU directives (i.e. CIN in Italy introduced in 2024).

```
<HotelDescriptiveContent>

    ...

   <TPA_Extensions>
      <AdditionalCodes>

         <AdditionalCode Type="CIN">IT12345678A1B23456</AdditionalCode>
         <AdditionalCode Type="CIPAT">ABC12345678</AdditionalCode>
```

```
      </AdditionalCodes>
   </TPA_Extensions>
</HotelDescriptiveContent>
```

The **TPA_Extensions** element can contain **one** **AdditionalCodes** with an unlimited number of **AdditionalCode** elements.

For complying with Italian rules the **Type** attribute **should** have **one** of the following values: "CIN" "CIPAT" "CIR".

Other values for **Type** attribute **may** be used in other regions.

This definition is subject to be extended in future.

4.3.2. 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 section 2.3 for details.

4.3.3. Inventory/Basic (pull) client request

The request body 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="AlpineBits Hotel"/>
    </HotelDescriptiveInfos>

</OTA_HotelDescriptiveInfoRQ>
```

files/samples/Inventory/Inventory-Pull-OTA_HotelDescriptiveInfoRQ-basic.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 `X-AlpineBits-Action` field is used by the server to distinguish the two requests.

4.3.4. 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 section 2.3 for details.

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

The **HotelDescriptiveContent** element is the exact same as documented in section 4.3.1 .

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="9.000">

  <Success/>
  <HotelDescriptiveContents>
    <HotelDescriptiveContent HotelCode="123" HotelName="AlpineBits Hotel">

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

    </HotelDescriptiveContents>
  </HotelDescriptiveContents>

</OTA_HotelDescriptiveInfoRS>
```

files/samples/Inventory/Inventory-Pull-OTA_HotelDescriptiveInfoRS-basic.xml - outer part

4.3.5. Inventory/HotelInfo (push) client request

The Inventory/HotelInfo message is used to exchange detailed descriptive information about a property such as:

- Property name / type / category
- Short/long descriptions
- Logos, seasonal pictures, gallery pictures, videos
- City id, latitude, longitude and elevation
- Property amenities
- Review ranking
- Address info and contact info: telephone, email, urls

and descriptive policies info:

- Cancellation
- Extra charges
- Pets
- City tax
- Guarantee and accepted payment methods
- Minimum guest age
- Check-in and check-out range

All **policies** information is intended for **descriptive purposes only** and do not alter room availability and price calculation.

Every transmission must be considered as a **CompleteSet** and will overwrite all previous Inventory/HotelInfo data (although the server will of course **not** replace any Inventory/Basic (push) data it has on record).

The client sends everything it wants to share with the server.

The **server decides what to save and what to ignore** of the Inventory/HotelInfo message, depending on the permissions dictated by the server business logic associated with the client account.

Consider the outer part of the OTA_HotelDescriptiveContentNotifRQ 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="AlpineBits Hotel" AreaID="3181913">

     <HotelInfo>
        <!-- Property type and category -->
        <CategoryCodes> ... </CategoryCodes>

        <!-- Property descriptions, pictures and videos -->
        <Descriptions> ... </Descriptions>

        <!-- 3D position -->
        <Position/>

        <!-- Property amenities -->
        <Services> ... </Services>
     </HotelInfo>

     <!-- Property policies as cancellation, extra charges... -->
     <Policies> ... </Policies>

     <!-- Property review aggregate -->
     <AffiliationInfo> ... </AffiliationInfo>

     <!-- Property address, email, phone, urls... -->
     <ContactInfos> ... </ContactInfos>

    </HotelDescriptiveContent>
  </HotelDescriptiveContents>

</OTA_HotelDescriptiveContentNotifRQ>
```

`files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-hotelInfo.xml` - outer part

Structure of `HotelDescriptiveContent`.

`HotelDescriptiveContent` element **must** have an `HotelCode` attribute with a unique property ID and an `HotelName` attribute with the property name. AlpineBits® allows the following child-element right below `HotelDescriptiveContent`:

Part I: `HotelInfo` element

At most one `HotelInfo` element which contain property classifications and multimedia information. If present, `HotelInfo` holds **at least one** of these elements:

`CategoryCodes` **at most one** element, if present hold **one** `HotelCategory` element. `CategoryCodes` element contain information about the property type and category. `HotelCategory` has **one** `CodeDetail` attribute used to identify the property type and class value.

`CodeDetail` attribute is a string composed by two elements separated by colon: `<CustomPropertyTypeTable_Value>:<ClassValue>`.

- `CustomPropertyTypeTable_Value` substring is a composition of 'CustomPropertyTypeTable' intended as the name of a custom property type table, the _ char used as separator and the value of the table that identify the property type. Partners are free to adopt their own custom tables.
- `ClassValue` substring is intended as the property class value. Es. 4 for 4 stars or `4s` for 4 stars superior properties.

Two examples of **`CodeDetail`** attribute content may be:

- `PCT2017_20:4s` where `PCT2017_20` is the reference to OTA (PCT) Property Class Type Codes (Version 2017) and its relative value, in this case `20` means Hotel. The string `4s` after the colon character is the class value of the property.
- `ASTAT2020_11:4s` where `ASTAT2020_11` is the custom reference property type table name and its relative value. The string `4s` after the colon character is the class value of the property. (In this example, `ASTAT2020` is intended as the property type table defined by the Provincial Statistical Institute of the Autonomous Province of Bolzano for the year 2020).

At most one `Descriptions` element which contains descriptive information like long/short property description, photos and videos. If **`Descriptions`** element is present, hold **one `MultimediaDescriptions`** element, which contain **one or more `MultimediaDescription`** element. **`MultimediaDescription`** element **must** have an **`InfoCode`** attribute with a subset value of `Information Type Code` (INF):

- 1 for a long description
- 17 for a short description
- 23 for pictures
- 24 for videos

If **`InfoCode`** attribute is set to 1 or 17, the inner element **must** be **one `TextItems`**.

`TextItems` **must** have **one or more `TextItem`** with **`SourceID`** and **`CopyrightNotice`** attributes which are **optional**.

`CopyrightNotice` attribute is **optional** and is used to transmit Copyright information.

`TextItem` element **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.

If **`InfoCode`** attribute is set to 23, the inner element **must** be **one `ImageItems`** which contains **one or more `ImageItem`** elements. **`ImageItem`** elements have a **mandatory `Category`** attribute that explain the nature of the image. The value of this attribute is a subset of `Picture Category Code` (PIC).

- 1 for an exterior view
- 2 for a lobby view
- 4 for a restaurant

- 12 for a spa
- 15 for a logo
- 22 for a property amenity

Inside of **ImageItem** element **must** be present **one ImageFormat** element and **zero or more Description** elements. **ImageFormat** element **could** have four optional attributes:

- **CopyrightNotice** attribute describes the copyright information about the image.
- **SourceID** attribute is a unique identifier that may be used by the receiver to verify with the sender if there is the need to download the image.
- **ApplicableStart** and **ApplicableEnd** attributes are used to **suggest** the seasonality of images. The format is `--MM-DD`, where MM is the month (01-12) and DD is the day (01-31).

Inside of **ImageFormat** **one URL** element is present with the reference to the image to download. The **Description** element follows the same rules of **Description** elements described in the long/short description case (**InfoCode** = 1 or 17).

If **InfoCode** attribute is set to 24, the inner element **must** be **one VideoItems** which contains **one or more VideoItem** elements. **VideoItem** elements have a **mandatory Category** attribute that explain the nature of the video. The value of this attribute is a subset of `Picture Category Code` (PIC).

- 1 for exterior view
- 2 for lobby view
- 4 for restaurant
- 12 for spa
- 20 for Miscellaneous
- 22 for property amenity

Inside of **VideoItem** element there **must** be **one VideoFormat** element present and **zero or more Description** elements. The **VideoFormat** element follows the same rules as the **ImageFormat** elements.

At most one Position element which contains geographic coordinates and elevation information of the property. If **Position** element is present it's **mandatory** to have **at least one** of these couple of attributes:

- **Latitude** and **Longitude** coordinates expressed in decimal degrees. The decimal separator is the point symbol.
- **Altitude** and **AltitudeUnitOfMeasureCode**. Where **AltitudeUnitOfMeasureCode** attribute describe the unit of measure. See Unit of Measure Code (UOM) for reference. For Meters use code 3.

At most one Services element which contain a list of property amenities. If the **Services** element is present, **must** contain **one or more Service** elements. Each **Service** element has two mandatory attributes:

- **Code** attribute identifies the facility code. See OTA Hotel Amenity Codes (HAC) or AlpineBits Hotel Amenity Codes (ABH) for reference.
- **ProximityCode** attribute identifies where the service is located. See Proximity code (PRX) for reference.

Inside of **Service** with **Code** attribute set to 47, may be present **at most one Features** element that **must** contain **one or more Feature** elements (a hotel with no accessibility feature to declare simply omits the **Features** element). Each **Feature** elements **must** have an **AccessibleCode** attribute which contain a Disability Feature Code (PHY) value.

There is another special case where half board needs to be seen as 3/4 board when defining **Service** element with **Code** attribute set to 342 (Snack bar) in combination with **MealPlanCode** attribute set to 12 and **Included** attribute set to `true`.

Example of **HotelInfo** element:

```
<HotelInfo>
    <!-- type of propriety like: Hotel, B&B, Chalet... -->
    <CategoryCodes>
        <HotelCategory CodeDetail="ASTAT2020_11:4s"/>
    </CategoryCodes>

    <Descriptions>
        <MultimediaDescriptions>
            <!-- InfoCode="1" for long description -->
            <MultimediaDescription InfoCode="1">
                <TextItems>
                    <TextItem  CopyrightNotice="Other copyright">
                        <Description Language="en" TextFormat="PlainText">A long
description</Description>
                    </TextItem>
                    <TextItem  CopyrightNotice="Other copyright">
                        <Description Language="de" TextFormat="PlainText">erweiterte
Beschreibung</Description>
                    </TextItem>
                    <TextItem  CopyrightNotice="Other copyright">
                        <Description Language="it" TextFormat="PlainText">Descrizione
lunga</Description>
                    </TextItem>
                </TextItems>
            </MultimediaDescription>

            <!-- hotel pictures -->
            <MultimediaDescription InfoCode="23">
                <ImageItems>
                    <!-- example: exterior picture applicable from Sep 30 to Mar 30 for
seasonal pictures -->
                    <ImageItem Category="1">
                        <ImageFormat CopyrightNotice="Image copyright"
SourceID="56757e0211440d37910f407fb6f657fb"
                                     ApplicableStart="--09-30" ApplicableEnd="--03-30">
                            <URL>https://..../HotelExteriorWinterView.jpg</URL>
                        </ImageFormat>
                        <Description TextFormat="PlainText" Language="en">Image
description</Description>
                    </ImageItem>
                </ImageItems>
            </MultimediaDescription>
        </MultimediaDescriptions>
    </Descriptions>

    <!-- geo position -->
    <Position Altitude="200" AltitudeUnitOfMeasureCode="3" Latitude="46.1372647"
Longitude="12.2011353"/>

    <!-- hotel amenities -->
    <Services>
        <!-- hotel level facilities see (HAC) codelist -->
        <Service Code="68"  ProximityCode="2"/>
        <Service Code="47"  ProximityCode="3">
            <Features>
                <!-- See PHY Disability Feature Code-->
                <Feature AccessibleCode="8"/>
            </Features>
        </Service>
        <!-- the AlpineBits list ABH can be used alongside the OTA one: 55.ABH is Garden
-->
        <Service Code="55.ABH"  ProximityCode="1"/>
```

```
    </Services>
</HotelInfo>
```

files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-hotelInfo.xml - HotelInfo element

Part II: `Policies` element

At most one `Policies` element which contain all the property policies. If present, **`Policies`** hold **one or more `Policy`** child-elements which in turn contain **exactly one** of these elements:

At most one `CancelPolicy` element which contain descriptive information about cancellation policies. If present, **`CancelPolicy`** element **must** contain **one `CancelPenalty`** element. **`CancelPenalty`** element contain **one `PenaltyDescription`** element which contain **one or more `Text`** elements. Each **`Text`** element contain a cancellation description for a specific language. **`Text`** elements **must** have **`TextFormat`** and **`Language`** attributes.

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 **`Text`** element with **`TextFormat`** set to HTML is intended as a rich text alternative of a **`Text`** element with **`TextFormat`** set to `PlainText` of the same **`Language`** and makes the latter **mandatory**.
- Please note that an AlpineBits® server is explicitly allowed to **filter, shorten or even skip the HTML content**, therefore the usage of **`Text`** elements with **`TextFormat`** set to HTML is **not** recommended but left as an option for implementers that absolutely need it.

At most one `CheckoutCharges` element containing descriptive information about check-out charge policies. If present, the **`CheckoutCharges`** element **must** contain **one `CheckoutCharge`** element. **`CheckoutCharge`** element may have an **optional `Amount`** attribute. If the **`Amount`** attribute is present, then the **`CurrencyCode`** must be present too. The **`CheckoutCharge`** element contains **one `Description`** element which contains **one or more `Text`** elements. Every **`Text`** element contains check-out descriptive information for a specific language. **`Text`** elements follow the same rules of **`Text`** elements described in **`CancelPolicy`** case.

At most one `PetsPolicies` element which contain descriptive information about pet policies. If present, **`PetsPolicies`** element **must** contain **one `PetsPolicy`** element. **`PetsPolicy`** element may have an **optional `MaxPetQuantity`** attribute and **`NonRefundableFee`**. If **`NonRefundableFee`** attribute is present, then the **`CurrencyCode`** **must** be present too. The **`PetsPolicy`** element contains **one `Description`** element which contains **one or more `Text`** elements. Every **`Text`** element contains pets policies descriptive information for a specific language. The **`Text`** elements follow the same rules of **`Text`** elements described in **`CancelPolicy`** case.

At most one `PolicyInfo` element which describes basic policy information through its attributes and **must not** contain any nested element. The **optional `MinGuestAge`** attribute **must** express the minimum age allowed for a guest at the hotel facility, as a non-negative integer.

At most one `TaxPolicies` element which contains descriptive information about tax policies. If present, the **`TaxPolicies`** element **must** contain **one `TaxPolicy`** element. The **`TaxPolicy`** element has the following attributes:

- **Must** have a **`Code`** attribute with value 3 = City Tax. (See Fee Tax Type FTT CodeList).
- May have an **optional `Amount`** attribute. If the **`Amount`** attribute is present,the **`ChargeFrequency`** attribute with value 1 = Daily (See CHG Charge Type Codes) and the **`ChargeUnit`** attribute with

value 21 = Per Person/night (See CHG Charge Type Codes) as well as the **CurrencyCode** **must** be present too.

For example, in case of a 2€ daily city tax per person, **TaxPolicy** attributes are: **Code** = 3, **Amount** = 200, **ChargeFrequency** = 1, **ChargeUnit** = 21.

The **TaxPolicy** element contains **one** **TaxDescription** element which contains **one or more** **Text** elements. Every **Text** element contains tax descriptive information for a specific language. **Text** elements follow the same rules of **Text** elements described in **CancelPolicy** case.

At most one **GuaranteePaymentPolicy** element which contains descriptive information about payment methods and guarantee policies. If present, **GuaranteePaymentPolicy** element **must** contain **one** **GuaranteePayment** element. **GuaranteePayment** element contain **at most one** **AcceptedPayments** element which contain **one or more** **AcceptedPayment** elements.

Each **AcceptedPayment** element contain **one** of these elements:

- **BankAcct** which **must** contain **BankAcctName** for the name of the bank, **BankAcctNumber** with sub element **PlainText** where insert IBAN number and finally the **BankID** element with sub element **PlainText** for the SWIFT field.
- **Cash** element with a **mandatory** **CashIndicator** attribute with value `true` or `false`.
- **PaymentCard** element with a **mandatory** **CardCode** attribute with an OTA Payment Card Code Type.
  - **At most one** **AmountPercent** element. If present, **AmountPercent** element **must** have a **Percent** attribute set.
  - **At most one** **Deadline** element which defines the timeout for confirming the booking via deposit. If **Deadline** element is present, **OffsetDropTime**, **OffsetTimeUnit** and **OffsetUnitMultiplier** attributes are **mandatory**.

At most one **StayRequirements** element which contains descriptive information about check-in and check-out policies. If present, **StayRequirements** element **must** contain **one or two** **StayRequirement** elements. **StayRequirement** **must** have three attributes:

- **StayContext** **must** be `Checkin` for check-in information or `Checkout` for check-out info.
- **Start** **must** be a time e.g. `15:00:00`.
- **End** **must** be a time e.g. `20:00:00`.

Example of **Policies** element:

```
<!-- Policies: check-in and check_out period, min guest age, ... -->
<Policies>
    <Policy>
        <!-- Check-in and Check-out range -->
        <StayRequirements>
            <StayRequirement StayContext="Checkin" Start="15:00:00" End="20:00:00"/>
            <StayRequirement StayContext="Checkout" Start="06:00:00" End="10:00:00"/>
        </StayRequirements>
    </Policy>
</Policies>
```

files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-hotelInfo.xml - Policies element

Part III: The **AffiliationInfo** element

At most one **AffiliationInfo** element which contains aggregate review information. If present,

AffiliationInfo holds **one Awards** element which contains **one or more Award** elements. Each **Award** elements must have the following attributes:

- **Rating** is **mandatory** and provide the rating value of the aggregate review.
- **Provider** is **mandatory** and provide the review provider.
- **OfficialAppointmentInd** is **optional** and when it's `true` indicate that the receiver has received official permission from the review provider to publish review info. If this attribute is not present it must be considered `false`.

Please note:

- The review scale for **Rating** depends by the service provider specified by **Provider** attribute.
- The transmitter **must** send the **Rating** as it is sent by the service provider.
- The receiver **must** read the documentation of the service provider to know the scale value.
- The **Provider** attribute value **must** be all uppercase. e.g. `TRUSTYOU`, `TRIPADVISOR`, `TRUSTPILOT`

Example of **AffiliationInfo** element:

```
<!-- Scale value depends by the review provider. Check provider's documentation -->
<AffiliationInfo>
    <Awards>
        <Award Rating="95" Provider="TRUSTYOU" OfficialAppointmentInd="false"/>
    </Awards>
</AffiliationInfo>
```

files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-hotelInfo.xml - AffiliationInfo element

Part IV: **ContactInfos** element

At most one ContactInfos element which contains contact information of the property. e.g. address, email, phone numbers, urls. If present, the **ContactInfos** holds **one ContactInfo** element with the attribute **Location** that **must** have value 6 ("Hotel direct contact" according to CON codelist) which contain **at least one** of these child elements:

At most one Addresses element which contains **one or more Address** elements. Each **Address** element **must** have a different **Language** attribute. If present, **Address must** contain at least following attributes: **AddressLine**, **CityName**, **PostalCode** and **CountryName** fields.

At most one Phones element which contains **one or more Phone** elements. Each **Phone** element has a **mandatory PhoneTechType** attribute and a **mandatory PhoneNumber** attribute.

At most one Emails element which contains **one or more Email** elements. Each **Email** element has a **mandatory EmailType** attribute.

At most one URLs element which contain **one or more URL** elements. Each **URL** element, if the **ID** attribute is specified it **must** follow rules described below. The content of the **URL** element **must** be a valid URI * despite its name, the content of **URL** may be any URI, so besides http(s) urls, also other schemes might be specified. URNs are allowed as well and in this case the schema **must** be `id`.

AlpineBits® defines some values for the **ID** attribute that should be recognized by servers and clients:

- `WEBSITE`: for the lodging structure's official website
- `TRUSTYOU`: for the lodging structure's presence on trustyou.com
- `TRIPADVISOR`: for the lodging structure's presence on tripadvisor.com
- `TWITTER`: for the lodging structure's presence on twitter.com

- `FACEBOOK`: for the lodging structure's presence on facebook.com
- `INSTAGRAM`: for the lodging structure's presence on instagram.com
- `YOUTUBE`: for the lodging structure's presence on youtube.com

Partners are allowed to define further values but the value of the attribute **must** be all uppercase. Lodging's official website url **must** be free of referral parameters. The server can delete any parameters not allowed.

At most one `CompanyName` element which contains the name of the company.

Example of **`ContactInfos`** element:

```
<ContactInfos>
    <ContactInfo Location="6">
        <Addresses>
            <Address Language="it">
                <AddressLine>Via Bolzano 63/A</AddressLine>
                <CityName>Bolzano</CityName>
                <PostalCode>39057</PostalCode>
                <StateProv StateCode="BZ"/>
                <CountryName Code="IT"/>
            </Address>
        </Addresses>
        <Phones>
            <Phone PhoneTechType="1" PhoneNumber="+3903720000000"/>
        </Phones>
        <Emails>
            <Email EmailType="5">info@alpinebits.org</Email>
        </Emails>
        <URLs>
            <URL ID="WEBSITE">https://www.alpinebits.org/</URL>
            <URL ID="FACEBOOK">https://www.facebook.com/alpinebits/</URL>
        </URLs>
    </ContactInfo>
</ContactInfos>
```

`files/samples/Inventory/Inventory-Push-OTA_HotelDescriptiveContentNotifRQ-hotelInfo.xml` - ContactInfos element

4.3.6. 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 section 2.3 for details.

4.3.7. Inventory/HotelInfo (pull) client request

The request body 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="AlpineBits Hotel"/>
    </HotelDescriptiveInfos>

</OTA_HotelDescriptiveInfoRQ>
```

files/samples/Inventory/Inventory-Pull-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 `X-AlpineBits-Action` **field is different** and is used to distinguish the two requests.

4.3.8. 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 section 2.3 for details.

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

```
<OTA_HotelDescriptiveInfoRS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            xmlns="http://www.opentravel.org/OTA/2003/05"
                            Version="9.000">

  <Success/>
  <HotelDescriptiveContents>
    <HotelDescriptiveContent HotelCode="123" HotelName="AlpineBits Hotel">

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

    </HotelDescriptiveContents>
  </HotelDescriptiveContents>

</OTA_HotelDescriptiveInfoRS>
```

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

4.3.9. Implementation tips and best practice

Please note that section 4.3 has been changed multiple times since its creation. See appendix B for compatibility information.

4.3.10. Tabular representation of OTA_HotelDescriptiveContentNotifRQ

Level	Element	Attribute	Type	Cardinality
	`OTA_HotelDescriptiveContentNotifRQ`		element	1
›	`HotelDescriptiveContents`		element	1

Level	Element	Attribute	Type	Cardinality
› ›	HotelDescriptiveContent		element	1
		HotelCityCode	string(1-8)	0 - 1
		AreaID	string ([0-9]{1,8})	0 - 1
› › ›	HotelInfo		element	0 - 1
		HotelStatusCode		0 - 1
› › › ›	CategoryCodes		element	0 - 1
› › › › [5]	HotelCategory		element	1
		Code	string ([0-9A-Z]{1,3}(\.[A-Z]{3}(\.X){0,1}){0,1})	0 - 1
		CodeDetail	string(1-128)	1
› › › ›	Descriptions		element	0 - 1
› › › › [5]	MultimediaDescriptions		element	1
› › › › [6]	MultimediaDescription		element	1 - ∞
		InfoCode	integer ([0-9]+) enum (1|17|23|24)	0 - 1
Start choice				
› › › › [7]	TextItems		element	0 - 1
› › › › [8]	TextItem		element	1 - ∞
		SourceID	string(1-32)	0 - 1
		CopyrightNotice	string(1-64)	0 - 1
› › › › [9]	Description		string(1-∞)	1 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › [7]	ImageItems		element	0 - 1
› › › › [8]	ImageItem		element	1 - ∞
		Category	integer ([0-9]+) enum (1|2|4|12|15|22)	1
› › › › [9]	ImageFormat		element	1
		CopyrightNotice	string(1-64)	0 - 1
		SourceID	string(1-32)	0 - 1
		Title	string(1-64)	0 - 1

Level	Element	Attribute	Type	Cardinality
		ApplicableStart	string pattern --[0-1][0-9]-[0-3][0-9]	0 - 1
		ApplicableEnd	string pattern --[0-1][0-9]-[0-3][0-9]	0 - 1
› › › › 10	URL		URL (https?://.+)	1
› › › › 9	Description		string(1-∞)	0 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 7	VideoItems		element	0 - 1
› › › › 8	VideoItem		element	1 - ∞
		Category	integer ([0-9]+) enum (1|2|4|12|20|22)	1
› › › › 9	VideoFormat		element	1
		CopyrightNotice	string(1-64)	0 - 1
		SourceID	string(1-32)	0 - 1
		Title	string(1-64)	0 - 1
		ApplicableStart	string pattern [0-1][0-9]-[0-3][0-9]	0 - 1
› › › › 10	URL		URL (https?://.+)	1
› › › › 9	Description		string(1-∞)	0 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
› › › ›	Position		element	0 - 1
		Altitude	decimal ([+-]?[0-9]{1,6}(\.[0-9]{1,8})?)	0 - 1
		AltitudeUnitOfMeasureCode	string enum (3)	0 - 1
		Latitude	decimal ([+-]?[0-9]{1,6}(\.[0-9]{1,8})?)	0 - 1
		Longitude	decimal ([+-]?[0-9]{1,6}(\.[0-9]{1,8})?)	0 - 1
› › › ›	Services		element	0 - 1
› › › › 5	Service		element	1 - ∞

Level	Element	Attribute	Type	Cardinality
		Code	string ([0-9]{1,3}(\.[A-Z]{3})?)	1
		MealPlanCode	string enum (12)	0 - 1
		ProximityCode		0 - 1
		Included	boolean (\S+)	0 - 1
› › › › 6	**Features**		element	0 - 1
› › › › 7	**Feature**		element	1 - ∞
		AccessibleCode		1
› › ›	**FacilityInfo**		element	0 - 1
› › › ›	**GuestRooms**		element	1
› › › › 5	**GuestRoom**		element	0 - ∞
		Code	string(1-8)	1
		MaxOccupancy	integer ([0-9]+)	0 - 1
		MinOccupancy	integer ([0-9]+)	0 - 1
		MaxChildOccupancy	integer ([0-9]+)	0 - 1
		MaxAdultOccupancy	integer ([0-9]+)	0 - 1
		ID	string(1-8)	0 - 1
› › › › 6	**TypeRoom**		element	1
		StandardOccupancy	integer ([0-9]+)	0 - 1
		RoomClassificationCode		0 - 1
		RoomID	string(1-16)	0 - 1
		Size	integer ([0-9]+)	0 - 1
		RoomType	string enum (1|2|3|4|5|6|7|8|9)	0 - 1
› › › › 6	**Amenities**		element	0 - 1
› › › › 7	**Amenity**		element	1 - ∞
		RoomAmenityCode	string ([0-9]{1,3}(\.[A-Z]{3})?)	0 - 1
› › › › 6	**MultimediaDescriptions**		element	0 - 1
› › › › 7	**MultimediaDescription**		element	1 - ∞
		InfoCode	integer ([0-9]+) enum (1|23|25)	0 - 1
Start choice				
› › › › 8	**TextItems**		element	0 - 1

Level	Element	Attribute	Type	Cardinality
› › › › 9	TextItem		element	1
› › › › 10	Description		string(1-∞)	1 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 8	ImageItems		element	0 - 1
› › › › 9	ImageItem		element	1 - ∞
		Category		1
› › › › 10	ImageFormat		element	1
		CopyrightNotice	string(1-64)	0 - 1
› › › › 11	URL		URL (https?://.+)	1
› › › › 10	Description		string(1-∞)	0 - ∞
		TextFormat	string enum (PlainText)	1
		Language	([a-z][a-z])	1
End choice				
› › ›	Policies		element	0 - 1
› › › ›	Policy		element	1 - ∞
Start choice				
› › › › 5	CancelPolicy		element	1
› › › › 6	CancelPenalty		element	1
› › › › 7	PenaltyDescription		element	1
› › › › 8	Text		string(1-∞)	1 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 5	GuaranteePaymentPolicy		element	1
› › › › 6	GuaranteePayment		element	1
› › › › 7	AcceptedPayments		element	1
› › › › 8	AcceptedPayment		element	1 - ∞
› › › › 9	BankAcct		element	0 - 1
› › › › 10	BankAcctName		string(1-64)	1

Level	Element	Attribute	Type	Cardinality
› › › › 10	BankAcctNumber		element	1
› › › › 11	PlainText		string(1-64)	1
› › › › 10	BankID		element	1
› › › › 11	PlainText		string(1-64)	1
› › › › 9	Cash		element	0 - 1
		CashIndicator	boolean (\S+)	1
› › › › 9	PaymentCard		element	0 - 1
		CardCode	string ([A-Z]{1,2})	0 - 1
› › › › 10	CardType		string(1-∞)	0 - 1
› › › › 7	AmountPercent		element	0 - 1
		Percent		1
› › › › 7	Deadline		element	0 - 1
		OffsetDropTime	string enum (BeforeArrival|AfterBooking|AfterConfirmation|AfterArrival|AfterDeparture)	1
		OffsetTimeUnit	enum (Year|Month|Week|Day|Hour|Second|FullDuration|Minute)	1
		OffsetUnitMultiplier		1
End choice				
Start choice				
› › › › 5	CheckoutCharges		element	1
› › › › 6	CheckoutCharge		element	1
		Amount	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
		DecimalPlaces	integer ([0-9]+)	0 - 1
› › › › 7	Description		element	1
› › › › 8	Text		string(1-∞)	1 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 5	PolicyInfo		element	1

Level	Element	Attribute	Type	Cardinality
		MinGuestAge	integer ([0-9]+)	0 - 1
End choice				
Start choice				
› › › › 5	TaxPolicies		element	1
› › › › 6	TaxPolicy		element	1
		Amount	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
		DecimalPlaces	integer ([0-9]+)	0 - 1
		Code	string enum (3)	0 - 1
		ChargeFrequency	string enum (1)	0 - 1
		ChargeUnit	string enum (21)	0 - 1
› › › › 7	TaxDescription		element	1
› › › › 8	Text		string(1-∞)	1 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 5	PetsPolicies		element	1
› › › › 6	PetsPolicy		element	1
		MaxPetQuantity		0 - 1
		NonRefundableFee	decimal ([0-9]*\.?[0-9]*)	0 - 1
		ChargeCode		0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
		DecimalPlaces	integer ([0-9]+)	0 - 1
› › › › 7	Description		element	1
› › › › 8	Text		string(1-∞)	1 - ∞
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	1
End choice				
Start choice				
› › › › 5	StayRequirements		element	1
› › › › 6	StayRequirement		element	1 - 2
		StayContext	string enum (Checkin|Checkout)	0 - 1

Level	Element	Attribute	Type	Cardinality
		Start	string pattern [0-2][0-9]:[0-5][0-9]:[0-5][0-9]	0 - 1
		End	string pattern [0-2][0-9]:[0-5][0-9]:[0-5][0-9]	0 - 1
End choice				
› › ›	**AffiliationInfo**		element	0 - 1
› › › ›	**Awards**		element	1
› › › › [5]	**Award**		element	1 - ∞
		Rating	decimal ([0-9]*\.?[0-9]*)	1
		Provider	string(1-∞)	1
		RatingSymbol	string enum (Diamond|Star)	0 - 1
		OfficialAppointmentInd	boolean (\S+)	0 - 1
› › ›	**ContactInfos**		element	0 - 1
› › › ›	**ContactInfo**		element	1
		Location	string enum (6)	0 - 1
› › › › [5]	**Addresses**		element	0 - 1
› › › › [6]	**Address**		element	1 - ∞
		Language	([a-z][a-z])	0 - 1
› › › › [7]	**AddressLine**		string(1-255)	1
› › › › [7]	**CityName**		string(1-64)	1
› › › › [7]	**PostalCode**		string(1-16)	1
› › › › [7]	**StateProv**		element	0 - 1
		StateCode	string(2-8)	1
› › › › [7]	**CountryName**		element	0 - 1
		Code	string pattern [A-Z][A-Z]	1
› › › › [5]	**Phones**		element	0 - 1
› › › › [6]	**Phone**		element	1 - ∞
		PhoneTechType	string ([0-9A-Z]{1,3}(\.[A-Z]{3}(\.X){0,1}){0,1})	1
		PhoneNumber	string pattern \+?[0-9]+string (1-32)	1
› › › › [5]	**Emails**		element	0 - 1
› › › › [6]	**Email**		string(1-∞)	1 - ∞
		EmailType	string ([0-9A-Z]{1,3}(\.[A-Z]{3}(\.X){0,1}){0,1})	1

Level	Element	Attribute	Type	Cardinality
› › › › [5]	URLs		element	0 - 1
› › › › [6]	URL		element	1 - ∞
		ID	string(0-32) ([A-Z]+)	0 - 1
› › › › [5]	CompanyName		string(0-∞)	0 - 1
› › ›	TPA_Extensions		element	0 - 1
› › › ›	AdditionalCodes		element	1
› › › › [5]	AdditionalCode		string(1-∞)	1 - ∞
		Type	string(1-∞)	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.3.11. Tabular representation of OTA_HotelDescriptiveContentNotifRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelDescriptiveContentNotifRS		element	1
Start choice				
›	Success		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.3.12. Tabular representation of OTA_HotelDescriptiveInfoRQ

Level	Element	Attribute	Type	Cardinality
	OTA_HotelDescriptiveInfoRQ		element	1
›	HotelDescriptiveInfos		element	1
› ›	HotelDescriptiveInfo		element	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.3.13. Tabular representation of OTA_HotelDescriptiveInfoRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelDescriptiveInfoRS		element	1
Start choice				
›	Success		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.4. RatePlans

If the `X-AlpineBits-Action` field is `OTA_HotelRatePlanNotif:RatePlans` the client sends information about rates and related rules.

4.4.1. Client request

The request body 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 the currency in which all amounts are expressed, it must be one of the ISO 4217 currency codes.
- **`RatePlanCode`** is the rate plan ID. It is limited to **16 characters**: the same code has to travel back in every reservation that originates from this rate plan (see section 4.2.5), and 16 characters is what OTA2015A allows there.

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.

```
<!-- a promotion the hotel presents itself: no RatePlanCategory -->
<RatePlan RatePlanNotifType="New" CurrencyCode="EUR" RatePlanCode="EARLYBIRD-2027"
          RatePlanType="12">
    ...
</RatePlan>

<!-- campaigned by a third party: RatePlanCategory holds a value defined by them -->
<RatePlan RatePlanNotifType="New" CurrencyCode="EUR" RatePlanCode="BIKEWEEKS-2027"
          RatePlanType="12" RatePlanCategory="SUEDTIROL-BIKE-HOTELS">
    ...
</RatePlan>
```

`files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-special-offer.xml` - the two kinds of special offer

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 about 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.xsd"
        Version="7.001">

    <RatePlans HotelCode="123" HotelName="AlpineBits Hotel">

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

            <BookingRules>
                <BookingRule Start="2027-03-03" End="2027-04-17">
                    ...
                </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>
```

`files/samples/RatePlans/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 Ski & 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 4.3) 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), the only exception being the pets' booking rules described below, which pair the fixed **Code** value ANY with **CodeContext** ALPINEBITS-PETS. 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. It is the responsibility of the **client** to check that - if the attribute **MinMaxMessageType** sets a value for SetMinLOS, it **must** be ≤ than SetMaxLOS, similarly SetForwardMinStay **must** be ≤ SetForwardMaxStay.

```
<BookingRule Start="2027-07-03" End="2027-08-28">
    <LengthsOfStay>
        <LengthOfStay Time="4"  TimeUnit="Day" MinMaxMessageType="SetForwardMinStay"/>
        <LengthOfStay Time="14" TimeUnit="Day" MinMaxMessageType="SetForwardMaxStay"/>
    </LengthsOfStay>
    <RestrictionStatus Restriction="Master" Status="Open"/>
</BookingRule>
```

files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-overlay.xml - the forward variants of the length of stay restrictions

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

- The criteria of the booking rule (if any) that **applies to the arrival day** (**Start** ≤ arrival day ≤ **End**): SetMinLOS, SetMaxLOS, arrival DOW, master status Open.
- The criteria of the booking rule (if any) that **applies to the departure day** (**Start** ≤ departure day ≤ **End**): departure DOW.
- The 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 2027-03-03 ≤ arrival day ≤ 2027-04-17 to a duration between 5 and 7 nights.

```
<BookingRule Start="2027-03-03" End="2027-04-17">
    <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 room of the `DOUBLE` category (note the **Code** attribute) to arrive and depart on a Thursday or Saturday.

```
<BookingRule Start="2027-01-01" End="2028-12-31" 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="2027-01-01" End="2028-12-31" 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="2027-08-01" End="2027-08-31" 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.

Another specific **CodeContext** for **BookingRule** is `ALPINEBITS-PETS` which denotes a special **BookingRule** that restricts the applicability of the rate plan to the presence of pets in the group. It applies to the whole rate plan and to every category specified in it. **Code** must have the fixed value `ANY` (to express that it refers to any kind of animal). It must not overlap with other `ALPINEBITS-PETS` **BookingRule** elements and must only contain a **RestrictionStatus** element with **Restriction** set to `Master` and a **Status** attribute set either to `Close` to disallow pets for the given time period or `Open` to allow them (which is equivalent to no restriction). Releases 2022-10 and 2024-10 spell this value `ALPINEBITSEXTRA-PETS`; it was shortened in this release because the same value is also carried by the **InvType** attribute of a **Supplement**, where OTA allows at most 16 characters (see the Supplements section later in this chapter).

```
<BookingRules>
   ...
   <!-- pets are accepted in autumn ... -->
   <BookingRule Start="2027-09-09" End="2027-11-30"
                CodeContext="ALPINEBITS-PETS" Code="ANY">
      <RestrictionStatus Restriction="Master" Status="Open"/>
   </BookingRule>

   <!-- ... and not accepted during the Christmas period -->
   <BookingRule Start="2027-12-01" End="2027-12-31"
                CodeContext="ALPINEBITS-PETS" Code="ANY">
      <RestrictionStatus Restriction="Master" Status="Close"/>
   </BookingRule>
</BookingRules>
```

files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-pets.xml - pets' booking rules

Check the Supplements' section later in this chapter for further details about pricing of pets.

Rates

AlpineBits® classifies Rate elements into two kinds: static **Rate** elements and date-dependent **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:

- Contains static information that is valid for the whole Rateplan and is not date dependent.
- **Must** be transmitted in every message, regardless if **RatePlanNotifType** set to `New` or `Overlay`.
- 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.
In overlay messages, this information is only used for control purposes.
The server should compare the transferred price type with that of the rate plan
already registered. If there is a discrepancy, the data should not be processed
and a corresponding error (ideally with a description of the discrepancy) should
```

```
be returned in response.
-->

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

files/samples/RatePlans/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-dependent rates.

The date-dependent rates (just "called" rates from here on) must have an **InvTypeCode** attribute that links them to room categories (see section 4.3).

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 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 after taxes; all amounts must be expressed in the currency specified by the **RatePlan CurrencyCode** attribute and must follow its currency convention of decimal digit amount after the decimal sign. **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="2027-03-03" End="2027-03-08">
<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>
```

files/samples/RatePlans/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 positive non zero decimal value (zero **is not** allowed) which must follow the currency convention of decimal digit amount after the decimal sign for the specified currency,
- **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 the minimum number of guests that ought to pay the full rate (see below for the definition). 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 positive decimal value (zero **is** allowed) which must follow the currency convention of decimal digit amount after the decimal sign for the specified currency.

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, **at most one AdditionalGuestAmount** element with a **AgeQualifyingCode** set to 10 ("adult") **may** 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 `ALPINEBITS` is not currently used, but is reserved for a future shared list of common **InvCode** values. Releases 2022-10 and 2024-10 spell this value `ALPINEBITSEXTRA`; it was renamed in this release so that the two AlpineBits®-specific values of this attribute share one prefix. The **InvType** value `ALPINEBITS-PETS` is used for transmitting pets' supplements. See end of section for further details.

Supplements, analogously to Rates are split into static and date-dependent **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-dependent **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
  - `18` - Per room per stay
  - `19` - Per room per night
  - `20` - Per person per stay
  - `21` - Per person per night
  - `24` - Item (per stay, see below).

    Note that when **ChargeTypeCode** is 1 or 24, if **MandatoryIndicator** is true, their amount should be considered as "1 per day" and "1 per stay" respectively; otherwise, for an optional Supplement, 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-dependent** supplements:

- The attribute **Amount** indicates the cost of the supplement after taxes; amounts must be expressed in the currency specified by the **RatePlan CurrencyCode** attribute and must follow its currency convention of decimal digit amount after the decimal sign; 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-dependent data, the attribute **Amount** and the sub element **PrerequisiteInventory** are **optional**. No further Element or Attribute is allowed.

Multiple date-dependent **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 the responsibility of the client to check that different amounts 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-dependent 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="2027-10-01"
            End="2027-10-11">
</Supplement>

</Supplements>
```

files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ.xml - supplement

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

Date-dependent 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 18, 20 and 24 (see above, all supplements that are *per stay*) the cost of a supplement may vary in the period of the stay. In this case, the total cost of the supplement **must** be calculated using the following algorithm (assuming a 3-night stay with a cost of € 80 for the first two days and € 85 for last two (including the departure day)):

- Calculate the number of days where the supplement applies (the supplement must not be applied to the departure day, hence the result is **3**).
- Sum the applicable price for each day (80 + 80 + 85 = 245 €).
- Divide the result for the number of days obtained at step 1 and round the result at the second decimal place (245 / 3 = 81.67 €).

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.

A specific kind of supplements can be sent with **InvType** value ALPINEBITS-PETS in order to transmit supplements regarding pets' pricing (called "Pets' Supplements" in the following paragraphs). These special supplements, if present, follow the same rules just mentioned for the regular **Supplement** elements with the addition that they should be defined only for Open time periods of **BookingRule** elements with **CodeContext** value ALPINEBITS-PETS (called "Pets' BookingRules" in the following paragraphs).

A best practice is to dedicate a separate **RatePlan** for managing categories that host pets: in fact if a "Pets' BookingRule" defines a period for which pets are allowed, every category without an associated "Pets' Supplement" will not charge any extra for the pets since no **Supplement** can be added to the total cost, as stated above in the **Supplements** definition.

Defining at least one of "Pets' BookingRules" or "Pets' Supplements" implies that the information about pets' acceptance and prices is managed. If both are undefined it means the information is unknown.

```
<Supplements>
   ...
   <Supplement InvType="ALPINEBITS-PETS" InvCode="pet" AddToBasicRateIndicator="true"
               MandatoryIndicator="false" ChargeTypeCode="1">
      <Description Name="title">
         <Text TextFormat="PlainText" Language="en">Pet surcharge</Text>
         <Text TextFormat="PlainText" Language="de">Zuschlag für Haustiere</Text>
         <Text TextFormat="PlainText" Language="it">Supplemento animali al seguito</Text>
      </Description>
   </Supplement>

   <Supplement InvType="ALPINEBITS-PETS" InvCode="pet" Amount="15"
               Start="2027-09-09" End="2027-11-15"/>

   <Supplement InvType="ALPINEBITS-PETS" InvCode="pet" Amount="20"
```

```
                Start="2027-11-16" End="2027-11-30">
        <PrerequisiteInventory InvType="ROOMTYPE" InvCode="DOUBLE"/>
    </Supplement>
    ...
</Supplements>
```

files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-pets.xml - pets' supplements

Offers

Offers are considered **static data** and can **only** be transmitted in complete/full 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 **AgeQualifyingCode** attribute set to 10 ("adult") and an **optional MinAge** attribute (integer value > 0)
- **Zero or one Occupancy** element with the **AgeQualifyingCode** attribute 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 **AgeQualifyingCode** 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 **AgeQualifyingCode** 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>
```

`files/samples/RatePlans/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.

Family offers are **only possible** if the **first** **`OfferRule`** element allows children (see the rules regarding the **`Occupancy`** elements above): the **`Occupancy`** element with **`AgeQualifyingCode`** set to `10` ("adult") **must** have a **`MinAge`** attribute and the **`Occupancy`** element with **`AgeQualifyingCode`** set to 8 ("child") **must** be present.

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 content
- 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.

```
<!-- the master: exactly one per RatePlanID has RatePlanQualifier="true" -->
<RatePlan RatePlanNotifType="New" CurrencyCode="EUR" RatePlanCode="STD-HB"
          RatePlanID="STD-2027" RatePlanQualifier="true">
    <BookingRules> ... </BookingRules>
    <Rates>
        <Rate RateTimeUnit="Day" UnitMultiplier="1">
            <BaseByGuestAmts>
                <BaseByGuestAmt Type="7"/>
            </BaseByGuestAmts>
            <MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
        </Rate>
        ...
    </Rates>
    <Offers> <!-- taken from the master when joining --> </Offers>
    <Description Name="title"> ... </Description>
</RatePlan>

<!-- the alternative version: same RatePlanID, RatePlanQualifier="false" -->
<RatePlan RatePlanNotifType="New" CurrencyCode="EUR" RatePlanCode="STD-BB"
          RatePlanID="STD-2027" RatePlanQualifier="false">
    <BookingRules> <!-- taken from the alternative version --> </BookingRules>
    <Rates>
        <Rate RateTimeUnit="Day" UnitMultiplier="1">
            <BaseByGuestAmts>
                <BaseByGuestAmt Type="7"/>
            </BaseByGuestAmts>
            <!-- MealsIncluded is the static rate information that is not inherited -->
            <MealsIncluded MealPlanIndicator="true" MealPlanCodes="3"/>
        </Rate>
        ...
    </Rates>
    <Offers> ... </Offers>
</RatePlan>
```

`files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-ratePlanJoin.xml` - the same stay offered with half board and with bed and breakfast

4.4.2. 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 guests that are declared as children in the request: an array of integers **c**. Whether such a guest actually counts as a "child" or as an "adult" is **not** decided by the request, but by the rate plan: the first **`OfferRule`** element defines the cut-off age (see step 0 below).
- The requested room category (i.e. **`InvTypeCode`** / **`Code`**): **code**.
- The arrival date **arr** and the departure date **dep** where **dep** > **arr**.
- The booking date **book** (needed only if the first **`OfferRule`** element has a **`MinAdvancedBookingOffset`** or a **`MaxAdvancedBookingOffset`** attribute).

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 4.3):

- 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**,
- (Optional) The value of **`MaxAdultOccupancy`** **max_adults**, such that **std** ≤ **max_adults** ≤ **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** = 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, in this order:

- first the guests are classified as adults or children according to the rate plan (step 0),
- then **all** the checks that decide whether the rate plan is bookable for the stay are performed (steps 1 to 3), on the numbers of adults and children resulting from step 0 and **before** any transformation,
- then the further transformations are applied: *family offers* first (step 4), then children to adults (step 5),
- and finally the cost is computed (step 6).

Step 0 (guest classification)

Apply the definition of "adult" and "child" given by the first **`OfferRule`** element (see the Offers section):

- If the **`Occupancy`** element with **`AgeQualifyingCode`** `10` ("adult") has **no** **`MinAge`** attribute, **all** guests are adults: increment **n** by the length of **c** and empty the array **c**.
- Otherwise, every guest whose age is ≥ **`MinAge`** is an adult: remove every element ≥ **`MinAge`** from **c** and increment **n** by the number of removed elements.

Example:
The request is for 2 adults and one child aged 15; the first **OfferRule** element contains only the **Occupancy** element with **AgeQualifyingCode** 10 and no **MinAge**. After this step n = 3 and c is empty: the stay is priced for 3 adults. Had that **Occupancy** element carried **MinAge** 16, the 15-year-old would have remained a child (n = 2, c = [15]). Had it carried **MinAge** 14, the 15-year-old would have been classified as an adult (n = 3, c is empty).

Step 1 (occupancy check)

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

Step 2 (offer rule check)

Verify that the stay is consistent with **all** the static rules of the first **OfferRule** element explained in the Offers section:

- If c is not empty, the **Occupancy** element with **AgeQualifyingCode** 8 ("child") **must** be present and the age of **every** element of c must lie in the interval given by its **MinAge** and **MaxAge** attributes, if present.
- n must respect the **MinOccupancy** / **MaxOccupancy** attributes of the "adult" **Occupancy** element and the length of c must respect those of the "child" **Occupancy** element, if present.
- arr, dep and book must respect the **LengthOfStay**, **ArrivalDaysOfWeek** and **DepartureDaysOfWeek** elements and the **MinAdvancedBookingOffset** / **MaxAdvancedBookingOffset** attributes, if present.

Unless **all** these conditions hold, the rate plan is **not** bookable for the stay and **no** cost can be computed. It is **not** allowed to promote children to "adults" (that is remove elements from c and increment n) to force a match: a guest who is a child according to step 0 remains a child for the purpose of these checks.

Step 3 (booking rule 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 4 (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).

Family offers are applied **before** the transformation of step 5, so that their conditions (**MinCount**, **MaxAge**) are evaluated on the children as classified in step 0 and a child who stays free is never transformed into an adult. As a consequence, a *family offer* may reduce the number of guests paying the full rate below minfull: in a room with std = 2, a stay of one adult and one child matching a "first child goes free" offer is paid by the adult alone (at the per-person amount for 2 guests, see step 6). A hotel that wants the rebate only when the full-rate payers are already present has to say so with **MinOccupancy** on the "adult" **Occupancy** element of the first **OfferRule** (which makes the rate plan not bookable at all for such a stay, see step 2).

Step 5 (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. Children removed from c in step 4 are not

available for this transformation.

Note that, unlike step 0, this is a pricing transformation: the guests are still children as far as the rate plan is concerned (the checks of steps 1 to 3 are **not** repeated on the transformed values), they merely pay the adult amount.

Step 6 (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 checks in steps 0 and 2.
- 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")

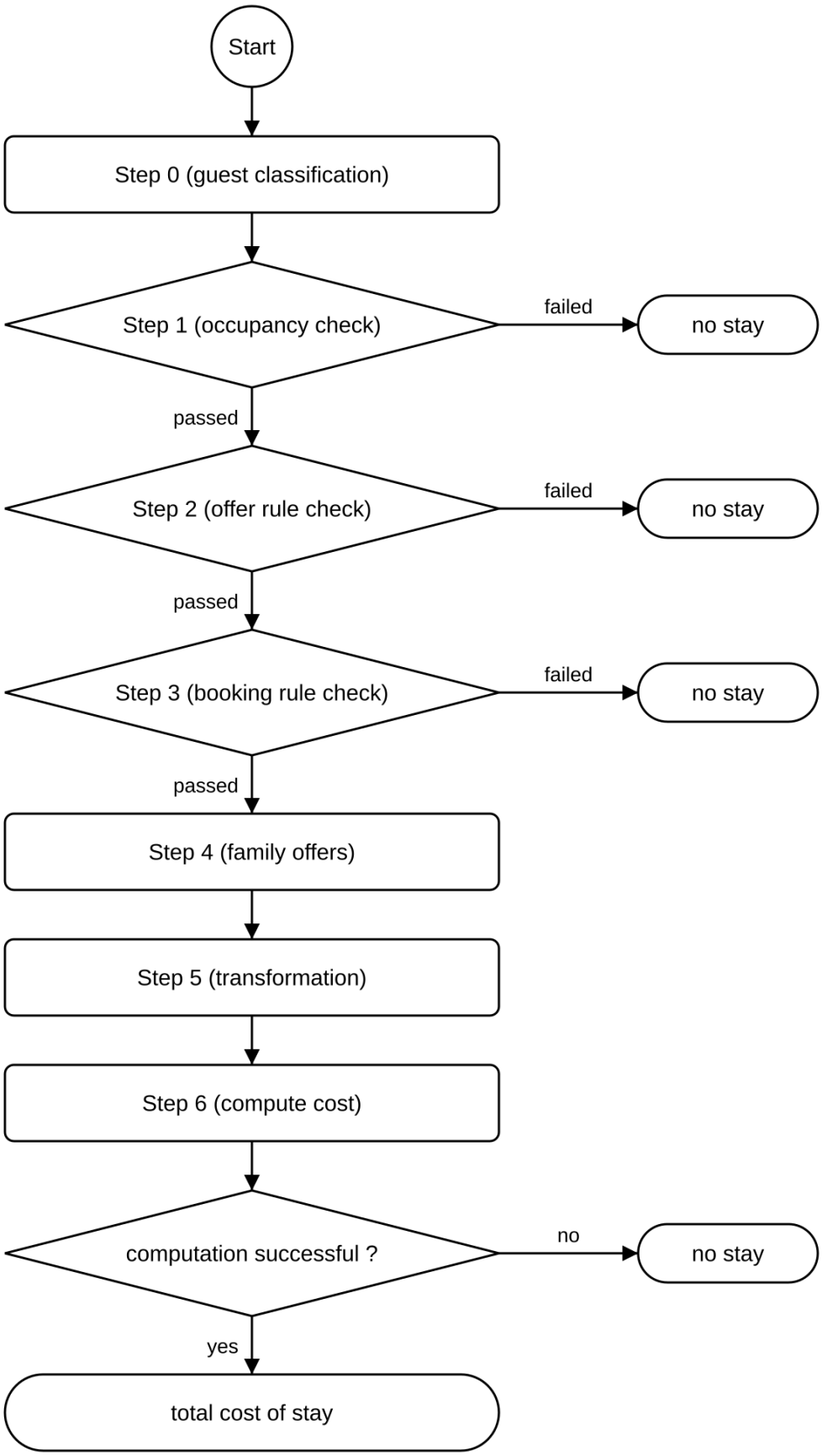
    If the correct **`BaseByGuestAmt`** element is not available, the stay as a whole is not possible (the rate plan is incomplete and cannot be applied).
- The remaining guests among the `n` - if any - **each** pay an amount given by the **`AdditionalGuestAmount`** elements with **`AgeQualifyingCode`**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



4.4.3. Synchronization

When it comes to exchange information about rates, AlpineBits® offers two possibilities for different use cases. One for **complete information** and one for **deltas**.

AlpineBits® uses the `RatePlanNotifType` attribute in each `RatePlan` element to define exactly how

the transmitted data has to be handled and processed.

- Complete/Full/New: In order to transmit new rate plans or to replace them completely, **`RatePlanNotifType`** = New **must** be used.
- Delta: To transmit deltas (changes) a **`RatePlanNotifType`** value of `Overlay` must be used. This is useful to keep the total amount of data to be processed in check.

In specific, the message types are defined as follows:

- **`RatePlanNotifType`** = New

  A message with the attribute **`RatePlanNotifType`** = New contains the complete/full RatePlan information, either as new data or as a replacement for all existing information.

The server receives and stores the rate plan as a whole. The capability `OTA_HotelRatePlanNotif_accept_full` has to be set if this type of message is supported. If a rate plan with the same **`RatePlanCode`** already exists, it is replaced. If it doesn't exist, it has to be created.

+ The message **must** contain all information regarding the rate plan, in specific the following requisites have to be met:

- Static rates **must** be sent within this message.
- At least **one** **`Description`** element **must** be present in the rate plan.
- In case of supplements, all static data **must** be sent within this message.
- Offers are always considered static and hence **must** also be sent within this message.
- The attributes **`RatePlanID`** and **`RatePlanQualifier`** - if supported by the server - **might only** be sent within this message.

  AlpineBits® requires each client and server to handle this message type.

  - **`RatePlanNotifType`** = `Overlay`

    This is a delta message and not a replacement. The **`RatePlanNotifType`** = `Overlay` is used to update existing rate plans identified by **`RatePlanCode`**. A server **must** set the `OTA_HotelRatePlanNotif_accept_overlay` capability if it supports this type of message.

    Static rates **must** be sent also within this message for control purposes. Since this information is valid for the whole RatePlan, independent of date or room type, the server should compare the transferred price type with that of the rate plan already registered. In case of discrepancy, the rates should not be processed and an appropriate error should be returned.

    All data regarding a specific rate plan **must** be contained in one **`RatePlan`** element.

    There cannot be multiple **`RatePlan`** elements for the same **`RatePlanCode`**.

    Elements that are not transmitted **must not** be touched by the server, elements that are transmitted **must** be completely replaced, including all subelements.

    Empty elements in AlpineBits® always do replace existing elements. Therefore, if a received message contains empty elements, those elements **must** be deleted.

    In case of rates or supplements, only date-dependent data may be sent within this message. Empty date-dependent rate elements **must** be deleted.

```
<RatePlan RatePlanNotifType="Overlay" CurrencyCode="EUR" RatePlanCode="STD-HB">

    <!-- an empty BookingRule deletes the rule the server holds for that period -->
    <BookingRules>
        <BookingRule Start="2027-09-01" End="2027-09-30"/>
    </BookingRules>
```

```
    <Rates>

        <!-- repeated for control only: it must match what the server already
stored -->
        <Rate RateTimeUnit="Day" UnitMultiplier="1">
            <BaseByGuestAmts>
                <BaseByGuestAmt Type="7"/>
            </BaseByGuestAmts>
            <MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
        </Rate>

        <!-- an empty date-dependent Rate deletes the prices stored for that period
-->
        <Rate InvTypeCode="DOUBLE" Start="2027-09-01" End="2027-09-30"/>

    </Rates>

</RatePlan>
```

`files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-overlay.xml` - a delta that updates one period and deletes another

Offers cannot be changed with an Overlay message. In order to update offers, the whole RatePlan has to be sent again with the complete/full set, using `New`.

If the server has no rate plan with the given **`RatePlanCode`** it must return a warning.

- **`RatePlanNotifType`** = `Remove`

  With this message type, a client can tell the server to delete rate plans. The rate plan **must** be empty with no child elements. The server deletes the rate plan identified by **`RatePlanCode`**. If the server has no rate plan with the given **`RatePlanCode`** it **must** accept the message and may give an advisory in the success response.

```
<RatePlans HotelCode="123" HotelName="AlpineBits Hotel">

    <RatePlan RatePlanNotifType="Remove" CurrencyCode="EUR" RatePlanCode="EARLYBIRD-
2027"/>

</RatePlans>
```

`files/samples/RatePlans/RatePlans-OTA_HotelRatePlanNotifRQ-remove.xml` - deleting a rate plan

Generally sending a rate plan, **`Rate`** and **`BookingRule`** elements are always updated as a whole, single sub-elements (such as only the **`LengthOfStay`** restrictions) cannot be updated individually. Sending empty **`BookingRule`** or empty **`Rate`** elements will delete them including all sub-elements.

Messages limits

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

- **`RatePlanNotifType`** = `New`

  Every RatePlan **must** be transmitted in a separate message.

- **`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 limits. In case of doubt, the updates should be

transmitted for one RatePlan at a time.

CompleteSet

There is the special case of rate plan messages that contain a **`UniqueID`** element with attribute **`Instance`** set to `CompleteSet`.

It is used by the client to send the codes of all rate plans. The server receives a complete and updated list of existing rate plans.

Hence the server **must** consider all rate plans it has on record **that are not contained in the current message** as expired and **must** delete them.

The **`RatePlan`** element will **not** have any **`RatePlanNotifType`** and **no** child elements must be present. Only the **`RatePlanCode`** **must** be present.

If the client wishes to reset all rate plans for a given hotel in a simple way, it can send a single empty **`RatePlan`** element. Sending no **`RatePlan`** element at all would violate OTA and AlpineBits® validation.

If a client intends to re-synchronize the RatePlans the following procedure is recommended:

1. Each RatePlan is transferred in a dedicated complete/full message, using the **`RatePlanNotifType`** = `New`.
2. A CompleteSet is transferred, containing all rate plan codes.

In this case the server is aware of the updated situation and can purge exceeding and/or outdated rate plan data.

Whether the server supports only `New`, `Overlay` or both, `CompleteSet` is mandatory and **must** be implemented.

4.4.4. 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 section 2.3 for details.

4.4.5. 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-dependent prices
- Examples: cleaning fees, parking, New Year’s Eve dinner
- 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.

4.4.6. Tabular representation of OTA_HotelRatePlanNotifRQ

Level	Element	Attribute	Type	Cardinality
	OTA_HotelRatePlanNotifRQ		element	1
›	UniqueID		element	0 - 1
		Type	string enum (16)	1
		ID	string(1-32)	1
		Instance	string enum (CompleteSet)	1
›	RatePlans		element	1
› ›	RatePlan		element	1 - ∞
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1
		RatePlanNotifType	string enum (Overlay|New|Remove)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
		RatePlanCode	string(1-16)	0 - 1
		RatePlanType	string enum (12)	0 - 1
		RatePlanCategory	string(1-64)	0 - 1
		RatePlanID	string(1-64)	0 - 1
		RatePlanQualifier	boolean (\S+)	0 - 1
› › ›	BookingRules		element	0 - 1
› › › ›	BookingRule		element	1 - ∞
		CodeContext	string enum (ROOMTYPE|ALPINEBITS-PETS)	0 - 1
		Code	string(1-8)	0 - 1

Level	Element	Attribute	Type	Cardinality
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1
› › › › 5	LengthsOfStay		element	0 - 1
› › › › 6	LengthOfStay		element	1 - ∞
		Time	decimal ([0-9]*\.?[0-9]*)	1
		TimeUnit	string enum (Day)	1
		MinMaxMessageType	string enum (SetMinLOS|SetForwardMinStay|SetMaxLOS|SetForwardMaxStay)	1
› › › › 5	DOW_Restrictions		element	0 - 1
› › › › 6	ArrivalDaysOfWeek		element	0 - 1
› › › › 6	DepartureDaysOfWeek		element	0 - 1
› › › › 5	RestrictionStatus		element	0 - 1
		Restriction	string enum (Master)	0 - 1
		Status	string enum (Open|Close)	0 - 1
› › ›	Rates		element	0 - 1
› › › ›	Rate		element	1 - ∞
		MinGuestApplicable		0 - 1
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1
		RateTimeUnit	string enum (Day)	0 - 1
		UnitMultiplier		0 - 1
		Mon	boolean (\S+)	0 - 1
		Tue	boolean (\S+)	0 - 1
		Weds	boolean (\S+)	0 - 1
		Thur	boolean (\S+)	0 - 1
		Fri	boolean (\S+)	0 - 1
		Sat	boolean (\S+)	0 - 1
		Sun	boolean (\S+)	0 - 1
		Duration	string pattern P[0-9]+N	0 - 1
		InvTypeCode	string(1-8)	0 - 1
› › › › 5	BaseByGuestAmts		element	0 - 1
› › › › 6	BaseByGuestAmt		element	1 - ∞
		NumberOfGuests		0 - 1

Level	Element	Attribute	Type	Cardinality
		AmountAfterTax	decimal ([0-9]*\.?[0-9]*)	0 - 1
		CurrencyCode	string(3-3) ([a-zA-Z]{3})	0 - 1
		Type	string enum (7|25)	0 - 1
		AgeQualifyingCode		0 - 1
› › › › 5	AdditionalGuestAmounts		element	0 - 1
› › › › 6	AdditionalGuestAmount		element	1 - ∞
		Amount	decimal ([0-9]*\.?[0-9]*)	0 - 1
		AgeQualifyingCode		0 - 1
		MinAge		0 - 1
		MaxAge		0 - 1
› › › › 5	RateDescription		element	0 - 1
		Name	string enum (included services)	1
› › › › 6	ListItem		string(1-∞)	1 - ∞
		ListItem	integer ([0-9]+)	1
		Language	([a-z][a-z])	1
› › › › 5	MealsIncluded		element	0 - 1
		Breakfast	boolean (\S+)	0 - 1
		Lunch	boolean (\S+)	0 - 1
		Dinner	boolean (\S+)	0 - 1
		MealPlanCodes	string enum (1|3|10|12|14)	0 - 1
		MealPlanIndicator	string enum (1|true)	0 - 1
› › ›	Supplements		element	0 - 1
› › › ›	Supplement		element	1 - ∞
		AddToBasicRateIndicator	string enum (1|true)	0 - 1
		ChargeTypeCode	string enum (1|12|18|19|20|21|24)	0 - 1
		Amount	decimal ([0-9]*\.?[0-9]*)	0 - 1
		InvType	string enum (EXTRA|ALPINEBITS|ALPINEBITS-PETS)	1
		InvCode	string(1-16)	1
		MandatoryIndicator	boolean (\S+)	0 - 1

Level	Element	Attribute	Type	Cardinality
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1
Start choice				
› › › › 5	**PrerequisiteInventory**		element	0 - 1
		InvCode	string(1-16)	1
		InvType	string enum (ALPINEBITSDOW|ROOMTYPE)	1
End choice				
› › › › 5	**Description**		element	0 - 5
		Name	string enum (title|intro|description|gallery|codelist)	1
Start choice				
› › › › 6	**ListItem**		string(1-∞)	1
End choice				
Start choice				
› › › › 6	**Image**		URL (https?://.+)	1
End choice				
Start choice				
› › › › 6	**Text**		string(1-∞)	1
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	0 - 1
End choice				
Start choice				
› › › › 6	**URL**		URL (https?://.+)	1
End choice				
› › ›	**Offers**		element	0 - 1
› › › ›	**Offer**		element	1 - 3
› › › › 5	**OfferRules**		element	0 - 1
› › › › 6	**OfferRule**		element	1
		MinAdvancedBookingOffset	string pattern P[0-9]+D	0 - 1
		MaxAdvancedBookingOffset	string pattern P[0-9]+D	0 - 1
› › › › 7	**LengthsOfStay**		element	0 - 1

Level	Element	Attribute	Type	Cardinality
› › › › 8	LengthOfStay		element	1 - ∞
		Time	decimal ([0-9]*\.?[0-9]*)	1
		TimeUnit	string enum (Day)	1
		MinMaxMessageType	string enum (SetMinLOS|SetMaxLOS)	1
› › › › 7	DOW_Restrictions		element	0 - 1
› › › › 8	ArrivalDaysOfWeek		element	0 - 1
› › › › 8	DepartureDaysOfWeek		element	0 - 1
› › › › 7	Occupancy		element	1 - 5
		AgeQualifyingCode	string enum (8|10)	1
		MinAge		0 - 1
		MaxAge		0 - 1
		MinOccupancy		0 - 1
		MaxOccupancy		0 - 1
› › › › 5	Discount		element	0 - 1
		Percent	string enum (100)	1
		NightsRequired	integer ([0-9]+)	0 - 1
		NightsDiscounted	integer ([0-9]+)	0 - 1
		DiscountPattern	string pattern 0*1*string (1-32)	0 - 1
› › › › 5	Guests		element	0 - 1
› › › › 6	Guest		element	1
		AgeQualifyingCode		1
		MaxAge		1
		MinCount	integer ([0-9]+)	1
		FirstQualifyingPosition	string enum (1)	1
		LastQualifyingPosition	integer ([0-9]+)	1
› › ›	Description		element	0 - 5
		Name	string enum (title|intro|description|gallery|codelist)	1
Start choice				
› › › ›	ListItem		string(1-∞)	1
End choice				

Level	Element	Attribute	Type	Cardinality
Start choice				
› › › ›	**Image**		URL (https?://.+)	1
End choice				
Start choice				
› › › ›	**Text**		string(1-∞)	1
		TextFormat	string enum (PlainText|HTML)	1
		Language	([a-z][a-z])	0 - 1
End choice				
Start choice				
› › › ›	**URL**		URL (https?://.+)	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.4.7. Tabular representation of OTA_HotelRatePlanNotifRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelRatePlanNotifRS		element	1
Start choice				
›	**Success**		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.5. BaseRates

If the `X-AlpineBits-Action` field 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.

4.5.1. Client request

The request body 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:

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

      <DateRange Start="2027-12-25" End="2028-01-03" />

      <RatePlanCandidates>
        <RatePlanCandidate RatePlanCode="DailyRate-HB" RatePlanID="DailyRate"/>
        <RatePlanCandidate RatePlanCode="SPECIAL"/>
      </RatePlanCandidates>

      <HotelRef HotelCode="123" HotelName="AlpineBits Hotel" />

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

`files/samples/BaseRates/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:

   The server will answer with all the rate plans it has on record. Note that the **`Rate`** elements are omitted, the response contains just the **`RatePlan`** elements with the **`Description`** whose **`Name`** attribute is set to `Title`.

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** whose **Name** attribute is set to `Title`.

4.5.2. 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 section 2.3 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:

```xml
<OTA_HotelRatePlanRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="4.000">
    <Success/>
    <RatePlans HotelCode="123" HotelName="AlpineBits Hotel">
        <RatePlan RatePlanCode="Base" CurrencyCode="EUR">
            <Rates>
                <Rate RateTimeUnit="Day" UnitMultiplier="1">
                    <BaseByGuestAmts>
                        <BaseByGuestAmt Type="7"/>
                    </BaseByGuestAmts>
                    <MealsIncluded MealPlanIndicator="true" MealPlanCodes="12"/>
                </Rate>
                <Rate Start="2027-12-29" End="2027-12-31" InvTypeCode="SINGLE">
                    <BaseByGuestAmts>
                        <BaseByGuestAmt NumberOfGuests="1"
                                        AmountAfterTax="130.00"/>
                    </BaseByGuestAmts>
                </Rate>
                <Rate Start="2027-12-29" End="2027-12-31" InvTypeCode="DOUBLE">
                    <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>
```

files/samples/BaseRates/BaseRates-OTA_HotelRatePlanRS.xml

4.5.3. Tabular representation of OTA_HotelRatePlanRQ

Level	Element	Attribute	Type	Cardinality
	OTA_HotelRatePlanRQ		element	1
›	**RatePlans**		element	1
› ›	**RatePlan**		element	1

Level	Element	Attribute	Type	Cardinality
› › ›	**DateRange**		element	0 - 1
		Start	date (\S+)	0 - 1
		End	date (\S+)	0 - 1
› › ›	**RatePlanCandidates**		element	0 - 1
› › › ›	**RatePlanCandidate**		element	1 - ∞
		RatePlanCode	string(1-16)	0 - 1
		RatePlanID	string(1-64)	0 - 1
› › ›	**HotelRef**		element	1

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.5.4. Tabular representation of OTA_HotelRatePlanRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelRatePlanRS		element	1
Start choice				
›	**Success**		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.6. ActivityData

If the `X-AlpineBits-Action` field is `OTA_HotelPostEventNotif:EventReports` the client intends to send information about hotel internal activities made available to their guests.

4.6.1. Exchange of hotel activities

The request body contains an `OTA_HotelPostEventNotifRQ` document.

Here is the global structure of the document:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelPostEventNotifRQ
        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_HotelPostEventNotifRQ.xsd"
        Version="3.000">

    <EventReports>
        <EventReport>
            <EventSites>
                <EventSite HotelCode="123" HotelName="AlpineBits Hotel" >
                    <Event_ID Type="18" ID="41654" ID_Context="providerNameSrl" />
                    <!--...-->
                </EventSite>
            </EventSites>

            <GeneralEventInfo Type="4" URL="https://example.com/event-permalink"
Acronym="YO-CES" >
                <EventContacts>
                    <!--...-->
                </EventContacts>

                <AttendeeInfo TotalQuantity="50" PreRegisteredQuantity="0"/>

                <Dates>
                    <!-- ... -->
                </Dates>

                <Comments>
                     <Comment Name="Title"> <!-- ... --> </Comment>
                     <Comment Name="Category"> <!-- ... --> </Comment>
                     <Comment Name="Gallery"> <!-- ... --> </Comment>
                     <Comment Name="Description"> <!-- ... --> </Comment>
                </Comments>
            </GeneralEventInfo>
        </EventReport>
    </EventReports>
</OTA_HotelPostEventNotifRQ>
```

`files/samples/ActivityData/ActivityData-OTA_HotelPostEventNotifRQ.xml` - outer part

Each document contains one **`EventReports`** element which MUST contain at least one and at most 99 **`EventReport`** elements. Each **`EventReport`** element contains two elements **`EventSites`** and **`GeneralEventInfo`**, both mandatory.

`EventSites` requires exactly one **`EventSite`** element.

For the attributes **`HotelCode`** and **`HotelName`**, inside the **`EventSite`** element, the rules are the same as for room availability notifications (section 4.1.1). While it is possible to send more than one **`EventReport`** element, AlpineBits® requires that all **`EventReport`** elements refer to the same hotel. Hence exactly one hotel can be dealt with in a single request.

An AlpineBits® server must return an error if it receives a request referring to more than one hotel.

The mandatory **Event_ID** element has no content and three attributes, all mandatory:

- **Type** must have the value 18.
- **ID** an identifier of the activity as set by the provider.
- **ID_Context** the context referring to the ID. Usually it's the name of the provider for the activity.

The pair **ID**,**ID_Context** is the unique identifier of the activity: it MUST appear only once in a single message and MUST be used as reference for further synchronization messages.

An AlpineBits® server must return an error if the same activity (i.e. multiple conflicting **Event_ID** elements) are sent in the same message.

The **GeneralEventInfo** holds the information about the activity. It contains the following attributes:

- **Type**.
- **URL**. Optional, a URL where the activity can be accessed in the providing system.
- **Acronym**. Optional, an acronym of the activity. If specified, it must be comprehensible for humans.

Further, the **GeneralEventInfo** element contains the following 4 elements:

- **EventContacts**. Optional. It must contain at least one **EventContact** sub-element.
- **AttendeeInfo**. Optional. See below for explanation.
- **Dates**. Optional. It must contain at least one and at most 99 **Date** sub-elements.
- **Comments**. Mandatory. See below for explanation.

An example of the complete structure of the **EventContact** element is the following:

```
<EventContact Role="chief">
    <PersonName>
        <GivenName>Werner</GivenName>
        <Surname>Call</Surname>
    </PersonName>

    <URL Type="Image">https://example.com/image.jpg</URL>
    <EmployeeInfo EmployeeId="Chief cook"/>
</EventContact>
```

files/samples/ActivityData/ActivityData-OTA_HotelPostEventNotifRQ.xml - inner part

Each **EventContact** element describes one contact that pertains to the activity. Its optional attribute **Role**, due the fact that of not being localized, must be agreed upon by the AlpineBits® partners and is used to transmit the Role of this person with regard to this activity (The same person might have different roles in different activities, for instance the "Guide" in an Hiking activity and "Instructor" in a skiing activity).

Further, the **EventContact** element contains the following elements:

- The optional **PersonName** element describes the referer person for the activity, surname and first name are sent using the mandatory **Surname** element and the optional **GivenName** element respectively.
- The optional **URL** element with the mandatory **Type** attribute set to `Image`, contains a link to a picture of the contact.
- The optional **EmployeeInfo** element contains in the mandatory **EmployeeId** is used to assign a unique Identifier to the contact person. This could be useful for further synchronization. (Be aware

that this attribute is limited to a maximum length of 16 character)

The optional element **AttendeeInfo** contains the **TotalQuantity** attribute that is used to specify the maximum number of guests that can participate to the activity. The attribute **PreRegisteredQuantity** contains the number of slots that are already busy, hence if the two attributes hold the same value the activity can be considered "sold out". Please note that AlpineBits® currently does not specify a way to book or reserve a slot for activities.

The **Dates** element contains at least one and up to 99 **Date** elements, which are the dates when the activity is scheduled. The main structure of the **Date** element is the following:

```
<Date Start="2027-02-23T09:30:00+02:00" End="2027-02-23T11:30:00+02:00">
    <EndDateWindow LatestDate="2027-02-23T09:30:00+02:00"/>

    <LocationCategories>
        <!-- optional location id -->
        <Location AreaID="1" />
        <Category Language="en">Wellness Area</Category>
        <Category Language="de">Wellnessbereich</Category>
        <Category Language="it">Area wellness</Category>

    </LocationCategories>
</Date>
```

files/samples/ActivityData/ActivityData-OTA_HotelPostEventNotifRQ.xml - inner part

The attributes **Start** and **End** define the beginning and the end of the activity, both may also contain time information if available. The **Start** attribute is mandatory, the **End** attribute is optional and its absence means that the event is open-ended. The optional **EndDateWindow** element contains the mandatory **LatestDate** attribute: the deadline by which guests SHOULD subscribe to the activity.

The optional **LocationCategories** element is used to describe the meeting place / location where the activity is planned to happen. The optional **Location** element allows the exchange of a unique identifier for the meeting place through its mandatory **AreaID** attribute. At least one **Category** element must be specified and contain the mandatory **Language** attribute set to a two-letter lowercase language abbreviation according to ISO 639-1. At most **one Category** element is allowed for each **Language**. Each **Category** element is used to transmit the localized name of the location.

The Comment section is the part where descriptive elements are defined.

Here is a sample:

```
<Comments>
    <Comment Name="Title">
        <Text Language="en">Yoga with Cesare</Text>
        <Text Language="de">Yoga mit Cesare</Text>
        <Text Language="it">Yoga con Cesare</Text>
    </Comment>
    <Comment Name="Category">
        <Text>WELLNESS-MENTAL</Text>
        <Text Language="en">Mental wellness</Text>
        <Text Language="de">mentales Wohlbefinden</Text>
        <Text Language="it">Benessere mentale</Text>
    </Comment>
    <Comment Name="Gallery">
        <Image>https://example.com/image1.jpg</Image>
        <Image>https://example.com/image2.jpg</Image>
    </Comment>
    <Comment Name="Description">
        <Text Language="en">Yoga Lorem Ipsum ...</Text>
```

```
        <Text Language="de">Yoga Lorem Ipsum ...</Text>
        <Text Language="it">Yoga Lorem Ipsum ...</Text>
    </Comment>
</Comments>
```

files/samples/ActivityData/ActivityData-OTA_HotelPostEventNotifRQ.xml - inner part

Inside the **Comments** element there are up to 4 **Comment** elements.

The **Comment** element with the attribute **Name** set to `Title` is mandatory and it contains at least one **Text** element with the mandatory attribute **Language** set to a two-letter lowercase language abbreviation according to ISO 639-1. Multiple **Text** elements can be specified, but each must specify a different **Language**: these are the title of the activity in the different languages.

The **Comment** element with the attribute **Name** equal to `Category` is mandatory and it contains at least one **Text** element with the optional attribute **Language** set to a two-letter lowercase language abbreviation according to ISO 639-1. Multiple **Text** elements can be specified, but each must specify a different **Language**: These are the categories of the activity in the different languages. At most one **Text** element can be specified without the **Language** attribute: this holds a unique identifier for the category, not meant for displaying to humans.

The **Comment** element with **Name** equal to `Gallery` is optional and it contains at least one **Image** element without attributes. Each **Image** element contains a valid url to an image that can be used to display the activity.

The **Comment** element with **Name** equal to `Description` is optional and it contains at least one **Text** element with the mandatory attribute **Language** set to a two-letter lowercase language abbreviation according to ISO 639-1. Multiple **Text** elements can be specified, but each must specify a different **Language**: These are the descriptions of the activity in the different languages.

4.6.2. ActivityData: Synchronization and deletion

Upon receiving an activity (identified by the given **Event_ID**) an AlpineBits® server MUST replace the previous activity stored with the same identificative in its entirety. This means that in case of rescheduling or cancellation of specific dates of an activity the whole activity must be sent again with up to date information.

In case of the complete cancellation of an activity, with all its dates, a client can notify to the server a deletion sending one **EventReport** element formed according to the following rules:

- The **EventSites** element MUST contain the same data of the original message for new activity.
- the **GeneralEventInfo** element MUST be empty (self closed tag).

Here is an example:

```
<EventReports>
    <EventReport>
        <EventSites>
            <EventSite HotelCode="123" HotelName="AlpineBits Hotel" >
                <Event_ID Type="18" ID="41654" ID_Context="providerNameSrl" />
            </EventSite>
        </EventSites>

        <GeneralEventInfo />
    </EventReport>
</EventReports>
```

files/samples/ActivityData/ActivityData-OTA_HotelPostEventNotifRQ-deletion.xml - inner part

4.6.3. Server response

The server will send a response indicating the outcome of the request. The response is a `OTA_HotelPostEventNotifRS` document. Any of the four possible AlpineBits® server response outcomes (success, advisory, warning or error) are allowed. See section 2.3 for details.

Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelPostEventNotifRS
        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_HotelPostEventNotifRS.xsd"
        Version="3.000">
    <Success/>
</OTA_HotelPostEventNotifRS>
```

files/samples/ActivityData/ActivityData-OTA_HotelPostEventNotifRS-success.xml

4.6.4. Tabular representation of OTA_HotelPostEventNotifRQ

Level	Element	Attribute	Type	Cardinality
	OTA_HotelPostEventNotifRQ		element	1
›	**EventReports**		element	1
› ›	**EventReport**		element	1 - 99
› › ›	**EventSites**		element	1
› › › ›	**EventSite**		element	1
› › › › [5]	**Event_ID**		element	1
		ID	string(1-32)	1
		ID_Context	string(1-32)	1
		Type	string enum (18)	1
› › ›	**GeneralEventInfo**		element	1
		Type	string ([0-9A-Z]{1,3}(\.[A-Z]{3}(\.X){0,1}){0,1})	0 - 1
		URL	URL (https?://.+)	0 - 1
		Acronym	string(1-32)	0 - 1
› › › ›	**EventContacts**		element	0 - 1
› › › › [5]	**EventContact**		element	1 - 99
		Role	string(1-32)	0 - 1

Level	Element	Attribute	Type	Cardinality
› › › › 6	PersonName		element	0 - 1
› › › › 7	GivenName		string(1-64)	0 - 1
› › › › 7	Surname		string(1-64)	1
› › › › 6	URL		string(1-∞)	0 - 1
		Type	string enum (Image)	1
› › › › 6	EmployeeInfo		element	0 - 1
		EmployeeId	string (1-16)	0 - 1
› › › ›	AttendeeInfo		element	0 - 1
		TotalQuantity	integer ([0-9]+)	1
		PreRegisteredQuantity	integer ([0-9]+)	1
› › › ›	Dates		element	0 - 1
› › › › 5	Date		element	1 - 99
		Start		1
		End		0 - 1
› › › › 6	EndDateWindow		element	0 - 1
		LatestDate		1
› › › › 6	LocationCategories		element	0 - 1
› › › › 7	Location		element	0 - 1
		AreaID	string ([0-9]{1,8})	1
› › › › 7	Category		string(1-∞)	1 - 99
		Language	([a-z][a-z])	1
› › › ›	Comments		element	0 - 1
› › › › 5	Comment		element	1 - ∞
		Name	string enum (Title|Category|Gallery|Description)	1
› › › › 6	Text		string(1-∞)	0 - ∞
		Language	([a-z][a-z])	0 - 1
› › › › 6	Image		string(1-∞)	0 - ∞

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

4.6.5. Tabular representation of OTA_HotelPostEventNotifRS

Level	Element	Attribute	Type	Cardinality
	OTA_HotelPostEventNotifRS		element	1
Start choice				
›	**Success**		element	1
End choice				

See file `files/schema-xsd/alpinebits.xsd` in the AlpineBits® documentation kit.

A. AlpineBits® developer resources

The AlpineBits® development home page is at https://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://gitlab.com/alpinebits/. Contributions are welcome (any programming language).

B. Protocol Version Compatibility

This chapter documents some of the major caveats between AlpineBits® clients and servers. The provided list of incompatibilities is not meant to be exhaustive and it is **mandatory** that partners use the same version of AlpineBits® for exchanging data. Most notably a client **is highly discouraged** from using different AlpineBits® versions for different `actions` when interacting with the same server.

For a complete list of modifications see changelog .

B.2026-10 Major overhaul in version 2026-10

Protocol changes

Version 2026-10 drops the `multipart/form-data` *Content-Type* of the POST request in favour of a plain XML body. Two **breaking changes** follow:

- The action is no longer a form parameter. It **must** be sent in the new `X-AlpineBits-Action` HTTPS header field, whose value is one of the action strings already in use (e.g. `OTA_Ping:Handshaking`).
- The XML document is no longer the `request` form parameter. It **must** be sent as the body of the POST request, with the header `Content-Type` set to `application/xml`.

Everything else stays as it was: the action strings, the XML documents, the authentication, the gzip compression and the `X-AlpineBits-ClientProtocolVersion` and `X-AlpineBits-ClientID` header fields are unchanged.

The change removes the need for a multipart parser, which is not available in every language or environment and does not always combine well with gzip compression, and it makes requests easier to build, log and debug.

To keep the migration manageable, a server **may** keep serving earlier versions from the same URL and tell the two apart by the *Content-Type* of the request, so that existing partners need no reconfiguration. Section 2.1 describes the rules. A server doing so keeps announcing those earlier versions in the handshake, so a single handshake still describes everything the URL offers, including a migration where only some actions have moved to 2026-10.

The new *Content-Type* applies to the handshaking action as well, and this is the one case the handshake cannot negotiate on its own: a client speaking 2026-10 cannot be understood by a server that supports no version from 2026-10 on, and gets an ERR0R response it cannot interpret. Section 3.5 suggests how a client should read that answer.

See section 2.1 for the complete description and a cURL example.

RatePlans

In version 2024-10, the value New for the **`RatePlanNotifType`** attribute has been replaced by Full. Since Full is not specified in the OTA standard, we are reverting this change and returning to New by keeping the capability name `OTA_HotelRatePlanNotif_accept_full`.

The steps of the algorithm describing the computation of the cost of a stay (see section 4.4.2) have been reordered and renumbered: a new step 0 classifies the guests as adults or children according to the first **`OfferRule`** element, the former steps 1, 1b and 4a are now the checks of steps 1, 2 and 3 (all performed before any transformation), the former steps 3, 2 and 4b are now steps 4, 5 and 6: *family offers* (step 4) are now applied **before** the transformation of children into adults (step 5), where the transformation came first up to 2024-10. This new order only makes a difference when a *family offer* grants a child a free stay before the subsequent transformation (step 5) could turn that child into an adult to reach the minimum number of full-rate payers. This is a purely theoretical scenario: in practice, a *family offer* is unlikely to ever be meant for such a child.

B.2024-10 Major overhaul in version 2024-10

RelaxNG

RelaxNG has been removed completely as schema for AlpineBits®. XSD remains as the only schema.

HotelCode and HotelName

The following breaking changes have been introduced:

- Attribute **`HotelCode`** is now mandatory while attribute **`HotelName`** has become **optional**. In practice this is no real change as probably no implementation used the **`HotelName`** to identify a structure.
- Attributes **`HotelCode`** and **`HotelName`** must not be case sensitive anymore. This restriction could not be fulfilled by case-insensitive implementations.

Inventory

New AlpineBits Code Lists for Hotel and Room Amenities.

New custom AlpineBits Code Lists have been added in order to be more precise when transmitting hotel and room amenities. Those new code lists (ABH and ABR) can be used in addition to the already existing OTA code lists (HAC and RMA). Please note that the introduction of this new feature required the addition of a **breaking change** to the schema. Specifically, the data type for `HotelInfo>Services>Service>Code` and `FacilityInfo>GuestRooms>GuestRoom>Amenities>Amenity>RoomAmenityType` was modified from int to string.

For redundancy reasons a breaking change has been introduced by removing attributes **`CheckInTime`** and **`CheckOutTime`** in element **`PolicyInfo`**. Element **`StayRequirement`** can be used.

RatePlans

For better interpretation in **`RatePlanNotifType`** the value `Full` has been added to replace the value `New`. All functionality remains the same. Implementations have to accept `Full` but may still accept also `New` for compatibility reasons.

B.2022-10 Major overhaul in version 2022-10

GuestRequests

The **`SpecialRequests`** attribute **`Name`** has been replaced with **`RequestCode`** and a new **`CodeContext`** attribute has been added.

The special case which allowed to specify alternative periods by adding one optional **`RoomStay`** element with only one **`TimeSpan`** element has been removed in favor of the new **`RoomStayGroupID`** attribute.

FreeRooms

In version 2022-10 a server might not handle CompleteSet messages for `OTA_HotelInvCountNotif:FreeRooms` but it **MUST** provide the capability `OTA_HotelInvCountNotif_accept_complete_set` if it does.

B.2020-10 Major overhaul in version 2020-10

Currency

The version 2020-10 adds support for all currency codes from the ISO 4217 in all actions where amounts are used.

Copyright and License information on multimedia objects

Guidelines to exchange copyright and license information on multimedia objects like images have been

added to the chapter data exchange.

GuestRequests

The version 2020-10 introduces some new options:

- special requests regarding pets (**`SpecialRequests`**)
- **`RoomType`** attribute now is mandatory only for reservations

FreeRooms

In version 2020-10 the FreeRooms message has been changed from `OTA_HotelAvailNotif` to `OTA_HotelInvCountNotif`. The chapter has been completely re-written and `OTA_HotelAvailNotif` is to be considered deprecated. Even all relative capabilities and action names have been changed.

OTA_HotelAvailNotif (old)	OTA_HotelInvCountNotif (new)
action_OTA_HotelAvailNotif	action_OTA_HotelInvCountNotif
OTA_HotelAvailNotif_accept_rooms	OTA_HotelInvCountNotif_accept_rooms
OTA_HotelAvailNotif_accept_categories	OTA_HotelInvCountNotif_accept_categories
OTA_HotelAvailNotif_accept_deltas	OTA_HotelInvCountNotif_accept_deltas
OTA_HotelAvailNotif_accept_BookingThreshold	OTA_HotelInvCountNotif_accept_out_of_market
-	OTA_HotelInvCountNotif_accept_out_of_order
-	OTA_HotelInvCountNotif_accept_closing_seasons

The new message offers the same options as the one used previously, but introduces the following new options:

- sending of closing seasons
- rooms out-of-order `OTA_HotelInvCountNotif_accept_out_of_order`

RatePlans

A new mandatory attribute **`CurrencyCode`** of the **`RatePlan`** element has been introduced to support all ISO 4217 currency codes.

Inventory/HotelInfo

The specification of Inventory/Hotelinfo has been expanded and provides an explicit structure for the exchange of hotel information. The version 2020-10 now defines the acceptable structure of **`HotelDescriptiveContent`** and its sub-elements. This action is no longer considered experimental.

Activities

This new action has been introduced with 2020-10. It is used to exchange information about hotel internal activities.

B.2018-10 Major overhaul in version 2018-10

Housekeeping / Handshaking

The `Housekeeping` action has been renamed to `Handshaking` and the relative chapter has been completely rewritten introducing **breaking changes**. Through the introduction of the `OTA_Ping:Handshaking` action it is now possible for clients to expose the capabilities they support.

New action GuestRequests (Push)

This functionality was introduced in version 2018-10.

It is now possible to push `GuestRequests` instead of polling them. Please note that this functionality relies on a few arrangements between the two parties that are not negotiated through AlpineBits®, namely the endpoint (URL) where the client can receive the pushed `GuestRequests`, its credentials, etc.

New attribute RoomType

This functionality was introduced in version 2018-10.

For `Inventory` and `GuestRequests`, in addition to the **`RoomClassificationCode`** attribute, the optional attribute **`RoomType`** has been added in order to allow a more precise classification of the guest accommodation.

Reference to the online presence of the lodging structure

This functionality was introduced in version 2018-10.

For `Inventory`, the element **`ContactInfos`** can be used to transmit URIs where the lodging structure has a presence (like social networks, review aggregators, etc.).

Review price calculation

These changes were made to the price calculation algorithm:

- The **`AdditionalGuestAmount`** element with attribute **`AgeQualifyingCode`** equal to 10 becomes optional.
- The **`AdditionalGuestAmounts`** element can be transmitted also if the **`MaxOccupancy`** attribute is equal to the **`StandardOccupancy`**.

Interaction with channel managers

- For `BaseRates`, a new case is described that allows servers to exchange information with clients that are only able to send RatePlans overlay information.
- For `GuestRequests` of type `Reservation`, a recommendation to set the value of the mandatory **`RatePlanCode`** attribute to the string "Unknown" has been added to ease the integration with channel managers that do not provide a reference to the AlpineBits® `RatePlan` which the `GuestRequest` actually refers to.

Server request for complete data set

In case of a suspected misalignment of data, servers can now request the client to send a full data set.

RatePlans

Version 2018-10 introduced optional **`AdditionalGuestAmount`** elements. Since they were mandatory in version 2017-10, a communication between both versions is only possible if the request from 2018-10 is gapless.

B.2017-10 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 section 2.3.

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 section 2.3.

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. The 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 send additional descriptive content.
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 4.4.2): 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 stay) 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.

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

Version 2015-07b is a maintenance release. The section 4.4 about rate plans has been mostly rewritten with more precise and strict information about how to handle corner cases, especially regarding rebates.

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

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

B.2015-07 Major overhaul in version 2015-07

Inventory

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

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

B.2014-04 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 vice versa. Here is a list of major changes in 2014-04.

HTTPS layer

The possibility of compression with gzip has been added.

FreeRooms

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

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

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

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

GuestRequests

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

SimplePackages

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

Inventory and RatePlans

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

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

Housekeeping

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

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

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

FreeRooms

There is no compatibility problem in the request part: the client will not send partial information (deltas), since it is simply not aware of the existence of the feature. Please be aware that the lack of this feature

(obviously) causes more data to be sent to the server, something not all companies that run servers will be happy with. The server response might contain a **`Warning`** element the client cannot process. If the client - as it should - carefully parses the response, it will treat this as an error situation and act accordingly. So basically the client is expected to treat the warnings as error, which might be an issue and this should be tested.

GuestRequests

No compatibility problems are expected.

SimplePackages

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

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

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

Housekeeping

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

FreeRooms

The client will not send partial information (deltas), since the server does not export the `OTA_HotelAvailNotif_accept_deltas` capability.

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

GuestRequests

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

SimplePackages

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

C. External links

[0] **Creative Commons BY SA license:**
https://creativecommons.org/licenses/by-sa/3.0/

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

[2] **OpenTravel Alliance:**
https://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"):**
https://www.pilotfishtechnology.com/modelviewers/OTA/

D. Code Lists

Index of AlpineBits Code Lists

ABH	AlpineBits Hotel Amenity Code
ABR	AlpineBits Room Amenity Type

ABH: AlpineBits Hotel Amenity Code

Value	Description
1.ABH	Agreement with golf course
2.ABH	Agreement with indoor pool
3.ABH	Agreement with outdoor pool
4.ABH	Animation
5.ABH	Baby change
6.ABH	Babysitter
7.ABH	Bank card/Maestro
8.ABH	Bar
9.ABH	Barbecue area
10.ABH	Barbecue facilities
11.ABH	Barbeque area
12.ABH	Barrier-free accomodation
13.ABH	Beauty farm
14.ABH	Bicycle rental
15.ABH	Bike depot
16.ABH	Bread delivery service
17.ABH	Bread roll delivery service
18.ABH	Breakfast buffet
19.ABH	Breakfast on request
20.ABH	Central location
21.ABH	Changing table
22.ABH	Chargeable Wi-Fi in room / apt.
23.ABH	Charging station for electric vehicle
24.ABH	Child care
25.ABH	Choice of menus possible
26.ABH	Close to a bus stop (< 100 m)
27.ABH	Close to public transport
28.ABH	Coach tours
29.ABH	Conf./event organisation possible
30.ABH	Conference and event facilities
31.ABH	Continental breakfast
32.ABH	Continental breakfast/Brunch
33.ABH	Credit card
34.ABH	Cuisine without glutine

Value	Description
35.ABH	Cycling
36.ABH	Dietary cuisine
37.ABH	Directly at the cross-country track
38.ABH	Directly at the lake
39.ABH	Directly at the lift
40.ABH	Directly on the ski slope
41.ABH	Directly on the slope
42.ABH	Doctor or medical care
43.ABH	Dogs allowed
44.ABH	Dogs and pets on request
45.ABH	Drying room
46.ABH	E-bike rental
47.ABH	Elevator
48.ABH	Entertainment evenings
49.ABH	Families
50.ABH	Fitness room
51.ABH	Free access to lake/private beach
52.ABH	Free WiFi in room / apt.
53.ABH	Fresh bread service
54.ABH	Fruit growing farm
55.ABH	Garden
56.ABH	Gluten-free dishes
57.ABH	Golf
58.ABH	Guided cycling tours
59.ABH	Guided ski tours
60.ABH	Guided tours and hikes
61.ABH	Hairdresser
62.ABH	Handicapped accessible
63.ABH	Hiking equipment
64.ABH	Hiking trail
65.ABH	Horse riding
66.ABH	Hot showers
67.ABH	Hotel information channel
68.ABH	Hotel skipass service
69.ABH	Indoor pool
70.ABH	Infrared cabin

Value	Description
71.ABH	In-house info channel
72.ABH	International cuisine
73.ABH	Internet access in room/apartment
74.ABH	Jogging trail
75.ABH	Lactose-free dishes
76.ABH	Lactose-free food
77.ABH	Laptop on request
78.ABH	Laptop upon request
79.ABH	Large pets
80.ABH	Laundry/Laundry service
81.ABH	Library
82.ABH	Livestock farm
83.ABH	Lounge
84.ABH	Luggage transfer
85.ABH	Massages
86.ABH	Mediterranean cuisine
87.ABH	Midsize pets
88.ABH	Miniature golf
89.ABH	Motorbikes welcome
90.ABH	Motorcycle depot
91.ABH	Mountain biking/ bike trail
92.ABH	Mountain climbing
93.ABH	MTB rental
94.ABH	Natural, local remedies
95.ABH	Near a skiing and hiking area
96.ABH	Near bus stop
97.ABH	Near ski bus stop
98.ABH	No pets or domestic animals
99.ABH	Open car park
100.ABH	Outdoor pool
101.ABH	Own horse riding stable
102.ABH	Panoramic location
103.ABH	Pick-up service
104.ABH	Ping Pong
105.ABH	Pizzeria
106.ABH	Playground

Value	Description
107.ABH	Playroom
108.ABH	Polo
109.ABH	Public bar
110.ABH	Quiet location
111.ABH	Racing bike rental
112.ABH	Reception 24 h
113.ABH	Relaxation area
114.ABH	Residence bar
115.ABH	Restaurant
116.ABH	River rafting
117.ABH	Rock climbing
118.ABH	Roofed car park
119.ABH	Sales of home-made products
120.ABH	Salt grotto
121.ABH	Sanitary cabins for rent
122.ABH	Sauna
123.ABH	Self service laundry
124.ABH	Seminar room
125.ABH	Seniors
126.ABH	Shared kitchen
127.ABH	Shop / Shopping possibility
128.ABH	Shop/shopping facility
129.ABH	Shopping facility 200m away
130.ABH	Shuttle service
131.ABH	Sightseeing tours
132.ABH	Ski bus / Ski shuttle
133.ABH	Ski depot
134.ABH	Ski rental
135.ABH	Ski service
136.ABH	Sledge rental
137.ABH	Small pets allowed
138.ABH	Smoking room
139.ABH	Snack bar
140.ABH	Snacks / small in-between meals
141.ABH	Snacks/Small dishes in-between
142.ABH	Snacks/tidbits between meals

Value	Description
143.ABH	Snowboard
144.ABH	Snow-shoe rental
145.ABH	Solarium
146.ABH	Spa licence (state license/agreement)
147.ABH	Spa treatments
148.ABH	Special agreement golf course
149.ABH	Special agreement indoor pool
150.ABH	Special agreement outdoor pool
151.ABH	Sports animation
152.ABH	Steam bath
153.ABH	Stroller rental
154.ABH	Supermarket at < 200m
155.ABH	Surf point
156.ABH	Tennis court
157.ABH	Terrace
158.ABH	Thermal baths
159.ABH	Toilet cubicles
160.ABH	Tumble drier
161.ABH	Underground car park
162.ABH	Vegan cuisine
163.ABH	Vegetarian cuisine
164.ABH	Visa
165.ABH	Volleyball
166.ABH	Walking routes
167.ABH	Washing machine
168.ABH	Welcome drink
169.ABH	Wellness / Spa
170.ABH	Whirlpool
171.ABH	WiFi
172.ABH	Windsurf
173.ABH	Wine cellar
174.ABH	Wine growing farm
175.ABH	Wine tasting

ABR: AlpineBits Room Amenity Type

Value	Description
1.ABR	2 or more bathrooms
2.ABR	2 or more bedrooms
3.ABR	Accessible rooms
4.ABR	Additional bed
5.ABR	Air conditioning
6.ABR	Baby cot
7.ABR	Balcony
8.ABR	Barrier-free
9.ABR	Bath towels
10.ABR	Bathrobe
11.ABR	Bathroom amenities
12.ABR	Bathtub
13.ABR	Bed linen
14.ABR	Bidet
15.ABR	Bunk bed
16.ABR	Cleaning upon request
17.ABR	Coffee machine
18.ABR	Crockery
19.ABR	Daily cleaning included
20.ABR	Dependance
21.ABR	Desk
22.ABR	Dishwasher
23.ABR	Dormitory
24.ABR	Double bed
25.ABR	Double sofa bed
26.ABR	Eat-in Kitchen
27.ABR	Electric stove
28.ABR	Electrical adaptors available
29.ABR	Final cleaning included
30.ABR	Free Wifi
31.ABR	Fully equipped eat-in kitchen
32.ABR	Garden
33.ABR	Gas connection
34.ABR	Gas stove

Value	Description
35.ABR	Hairdryer
36.ABR	Hi-Fi system
37.ABR	Hob
38.ABR	Hotel annex
39.ABR	Hypoallergenic room
40.ABR	Induction hob
41.ABR	Infrared cabin
42.ABR	Internet access
43.ABR	Iron
44.ABR	Ironing board
45.ABR	Kettle
46.ABR	Lake view
47.ABR	Main building
48.ABR	Microwave
49.ABR	Minibar
50.ABR	Mountain view
51.ABR	Non-smoking
52.ABR	Oven
53.ABR	Panorama view
54.ABR	Pets
55.ABR	Pets on request
56.ABR	Refrigerator
57.ABR	Room For Allergy-sufferers
58.ABR	Room with connecting door
59.ABR	Safe
60.ABR	Satellite/cable TV
61.ABR	Sauna
62.ABR	Sea view
63.ABR	Seasonal rent possible
64.ABR	Separate kitchen
65.ABR	Separate living area
66.ABR	Separate WC
67.ABR	Shower On The Floor
68.ABR	Shower/WC
69.ABR	Single bed
70.ABR	Single room

Value	Description
71.ABR	Single sofa bed
72.ABR	Snack bar
73.ABR	South Side
74.ABR	Stove
75.ABR	Suite
76.ABR	Tea/Coffee machine
77.ABR	Telephone
78.ABR	Terrace
79.ABR	Tiled stove/firplace
80.ABR	Towels
81.ABR	TV
82.ABR	TV upon request
83.ABR	Twin room
84.ABR	Washing machine
85.ABR	Water boiler
86.ABR	Water/waste connection
87.ABR	WC on the floor
88.ABR	Whirlpool
89.ABR	With hotel service