The Apartment Service Website

Contents

[Introduction 4](#_Toc211142247)

[Common 4](#_Toc211142248)

[System Overview 4](#_Toc211142249)

[Specifications Used 4](#_Toc211142250)

[Development Process 4](#_Toc211142251)

[Platform 5](#_Toc211142252)

[Server for ApartmentService.com 5](#_Toc211142253)

[Server for TABS Web Service 5](#_Toc211142254)

[Database Server 5](#_Toc211142255)

[Software Stack Design 6](#_Toc211142256)

[Process Functional Design 7](#_Toc211142257)

[Enquiries 7](#_Toc211142258)

[Browsing for an Apartment 7](#_Toc211142259)

[Searching for an Apartment 7](#_Toc211142260)

[Getting Specific Apartment Details 8](#_Toc211142261)

[Booking Process for Public Clients 8](#_Toc211142262)

[Room Allocation Engine 8](#_Toc211142263)

[Availablity of Request Properties 9](#_Toc211142264)

[Payment 9](#_Toc211142265)

[Corporate and Affiliate Bookers 10](#_Toc211142266)

[Authentication 10](#_Toc211142267)

[Authorisation 11](#_Toc211142268)

[Mail 11](#_Toc211142269)

[Audit 11](#_Toc211142270)

[TABS Web Service 12](#_Toc211142271)

[Introduction 12](#_Toc211142272)

[System Overview 13](#_Toc211142273)

[Level 1 – System Layout 13](#_Toc211142274)

[Level 2 – Request Processing 14](#_Toc211142275)

[Level 3 – Message Data Path 15](#_Toc211142276)

[Level 4 – Functional Engine for Availability Search 16](#_Toc211142277)

[Interfaces 17](#_Toc211142278)

[OTA Messages 17](#_Toc211142279)

[Private Messages – Searching 19](#_Toc211142280)

[Private Messages – Booking Engine 20](#_Toc211142281)

[Private Messages – Property Management 20](#_Toc211142282)

[Private Message – Enquiries 24](#_Toc211142283)

[Private Message – Audit 24](#_Toc211142284)

[Functional Engines 24](#_Toc211142285)

[Booking 24](#_Toc211142286)

[Data Feed Agents 25](#_Toc211142287)

[Local 25](#_Toc211142288)

[OTA - InterfaceTech 25](#_Toc211142289)

[Database Entities 26](#_Toc211142290)

[Affliate/Corporate Booker 27](#_Toc211142291)

[Apartment 27](#_Toc211142292)

[Apartment Type 27](#_Toc211142293)

[Booking 28](#_Toc211142294)

[User 28](#_Toc211142295)

[Group 28](#_Toc211142296)

[Enquiry 29](#_Toc211142297)

[Property 29](#_Toc211142298)

[Profile 29](#_Toc211142299)

[Audit 30](#_Toc211142300)

[Website 31](#_Toc211142301)

[Introduction 31](#_Toc211142302)

[Branding and Style 31](#_Toc211142303)

[Pages 31](#_Toc211142304)

[Search and DetaiL 31](#_Toc211142305)

[Booking Manager 34](#_Toc211142306)

[Property Manager 36](#_Toc211142307)

[Dictionaries 36](#_Toc211142308)

[Appendix A – eTrak Form Format 37](#_Toc211142309)

[Fields 37](#_Toc211142310)

[Enumerations 37](#_Toc211142311)

[Appendix B – GDS booking Notes 39](#_Toc211142312)

[Appendix C1 – Work Items for Compliance with Spec 40](#_Toc211142313)

[Appendix C2 – Task List for July 42](#_Toc211142314)

[Appendix D – OTA Usage by XPath 44](#_Toc211142315)

[OTA\_HotelSearchRQ 44](#_Toc211142316)

[OTA\_HotelSearchRS 45](#_Toc211142317)

[OTA\_HotelAvailRQ 46](#_Toc211142318)

[OTA\_HotelAvailRS 47](#_Toc211142319)

[OTA\_HotelResRQ/OTA\_HotelResRS 50](#_Toc211142320)

[Glossary 59](#_Toc211142321)

# Introduction

This Document provides a technical solution to deliver a system which meets the criteria set out in the TD, PD and AC.

It is of note that:

* The proposed design takes into account where possible existing code developed by Sapphire Group
* Consideration is given to later phase development and the impact this has on system architecture.

# Common

## System Overview

In order to understand this system properly you must think of it as not one project but as two. You have:

1. The Apartment Service website (mostly phase 3)
2. The TABS web service (phase 1 & 2)

Each with their own separate database only communicating is through web service calls. This means that for example that if you book something, the website has to ask the web service if it's available, and then get confirmation of the booking from the web service. The reasoning behind this is that because we will be working with lots of agents/affiliates in the future who will want to book directly though the web services and not the website. It also means we can with minimal effort (compared to starting from scratch anyhow) deploy the same website codebase to our other sites just changing the master page and supporting pages

For the most part the system developed in phases 1 & 2 is intended to provide The Apartment Service’s core services via XML web-services; the XML services are to be developed to be consumed by a variety of user agents. In order to test the web services we are to develop a rudimentary web site which demonstrates core functionality.

Phase 3, the development of apartmentservice.com is of course outside the scope of this project but it is of note that the rudimentary site developed in phases 1 & 2 may very well form the basis of apartmentservice.com in phase 3. In fact it may well form the basis of *all* white labelled clients to the system inline with AC 6.4.2

## Specifications Used

* OTA version 2007A
* XHTML Transitional 1.0
* CSS 2.1
* ASP.net 2.0

## Development Process

Best practice is:

* devserver - each developer maintains their own for testing their code prior to commit
* tabstest - master dev server for next version, code deployed here has been checked into SVN and must be buildable
* staging - for testing the next version of live prior to deployment
* live - for live code

## Platform

### Client

* Must support IE 6, IE 7, Firefox, Safari.
* Code

### Server for ApartmentService.com

* Windows 2003 or greater
* IIS v6 or greater
* ASP.net 2.0
* .NET 2.0

### Server for TABS Web Service

* Windows 2003 or greater
* IIS v6 or greater
* ASP.net 2.0
* .NET 2.0

### Database Server

* Windows 2003 or greater
* SQL Server 2005 (express edition or better)
* .NET 2.0

*TABS-Ws*

*Apartment Service.com*

Site

Core Managers

WebService Ref

Private Web Service

Business Logic (Functional Engine)

Local DataAccess

Database

OTA DataFeed

OTA Service

# Software Stack Design

# Process Functional Design

## Enquiries

An enquiry occurs either when the user demands it or when the property they want is unavailable for one reason or another. The enquiry process is a good example of the website operating in a kind of “dumb terminal” mode. All it does is package the enquiry up and sends it to the web-service for processing.

One of the requirements in the specification document is a link into our eTrak system. This requires a plaintext email to be sent to [etrak@apartment.co.uk](mailto:etrak@apartment.co.uk) with the body containing data in the format shown in Appendix A - eTrak form format.

The data will then be collected from the mailbox by the eTrak server (every 10 minutes) and automatically turned into an eTrak enquiry. The eTrak system will also automatically send an acknowledgement to the user.

AC 1.7.3

## Browsing for an Apartment

When we don’t have dates to work with we can show details about an apartment but we cannot confirm prices or availability. To do this rather than using the availability search message we use the search message instead. The user interface will allow them to browse but offer them the opportunity to enter dates and guest numbers to calculate prices (it’s required if they wish to select a room type to move on to booking).

Property Detail Request

ApartmentService.com

TABS-WS

Search Request

Search Response

Detail Response

Booking Request

Booking Response

AC 8.1.5 & 8.1.6 ???

## Searching for an Apartment

Once we have location, dates and the number of persons to work with, we can calculate prices and availability for rooms. We do this by supplying these details to an Availability checking message, which will return the same details as a search message but filtered so that unavailable properties are excluded.

It is possible and even likely that a user will not be sure of the exact parameters of their search. The way we can get around this is via a concept known as tolerance and relevance. Tolerance expands the parameters of their search whilst relevance orders the results by their closeness to the original search parameters.

AC 8.1.5 & 8.1.6 ???

## Getting Specific Apartment Details

When a customer wishes to look at the details of an apartment it is likely they will want to see summary information first, with an opportunity to drill down for further details. If they haven’t provided us with the date range for their stay or guest counts we will be limited in the amount of detailed information we can give out about a room type and its availability. When a room type is not available within the specified date range we will want to make this clear but still let the user know that it exists so they can bear it in mind for future bookings.

AC 8.1.5 & 8.1.6 ???

## Booking Process for Public Clients

When a public (not corporate or affiliate) client books

1. User selects apartment to book
2. Try to grab a hold transaction on the property with 10-15 minute expiry time.
3. User confirms their details and that of the booking and accepts the terms and conditions of booking.
4. Do transaction with DEFERRED rather than PAYMENT mode to place a shadow on their card.
5. Try and grab a confirmed reservation on the property. If it fails then send ABORT to Protx to clear the shadow from their card otherwise send RELEASE to charge cardholder’s card.

The complexities of the booking process are mostly abstracted away from the website by the web service; however payment is fully the responsibility of the website. The booking is at this point submitted to be saved in the local database and (where appropriate) sent to the OTA and Protx.

The difference with corporate and affiliate bookings is the method of payment. With corporate or affiliate bookings we may offer them a line of credit and then invoice them in bulk.

## Room Allocation Engine

When a booking for one of the properties for which we hold the master availability record comes into the web-service we know the following information from their request:

* Property to book (and thus from our DB the data source of that booking local or remote)
* Apartment type to book
* When to book
* Who to book for. This is the profile of the guests being booked for.
* How much they’re expecting to book for. (rates and total)
* Type of booking, hold or confirm.
* That the person has agreed to our terms and conditions.

From the available rooms in our physical inventory (in future we might have package inventory as well) we need to determine:

* Which room to book (not disclosed to the customer in case we need to move them to another room of the same type)
* Whether what they want to book is available at the price listed when they searched for availability.

It is of course possible that someone has booked the property or the rate offered on it has become unavailable in between the time the customer checked for availability and booking. It is also likely that third parties will send us cached result sets where availability is not real-time. Therefore we must recheck that it is still available at the final moment of the booking. If the apartment is available then we can proceed with the booking, otherwise we must inform the booking client so that they can either look for something else or send an enquiry for a request booking.

If there is more than one apartment of the type selected available we now must decide which room to book. For this phase of the project we shall use a random allocator. We will however consider alternatives, such as choosing the apartment with the smallest difference between the length of the available slot and the length of the stay.

It is worth noting that this problem is known in the computer science field as a “bin packing problem” and is classified as “NP-Hard”. This means that is has been mathematically proven that there is no known optimal solution finding algorithm. We must instead use heuristic algorithms to find the closest possible solution to optimal.

## Availablity of Request Properties

Where a property is a request property we may hold a master property record but not a master availability record. This means we cannot guarantee the booking immediately online and need to contact the property owner to ask if it is available. The easier way to do this is to automatically generate an email which is sent to the property owner. We can then provide a page for them to click back on and enter whether or not the property is available and how long they are willing to hold it for. This information can then be formatted into an email which is sent to the client, allowing them to drop straight into the booking process.

## Payment

Protx is our credit card payment provider, who accepts the card payments on our behalf. Using a payment provider rather than registering the transactions ourselves provides us with several advantages including:

* Scalability and the ability to increase the number of transactions we run rapidly.
* Savings on infrastructure and security.
* Avoidance of PCI-DSS consultants and audits by transferring responsibility to Protx.

When we wish to charge a customer’s card we pass their details (name, address, transaction amount and currency) to Protx. We do not collect any credit card details ourselves or have any interaction with the Credit card companies. Protx will then give us back a transaction reference and a randomly generated security key that we keep secret from the user.

Once the user has completed their interaction with Protx it will send us back a notification to a web page that we have designated. We must then process that notification to confirm the booking and indicate to Protx that we wish the payment to proceed and where we wish them to redirect our customer to.

In phase 4 we will be receiving credit card details via OTA messages as guarantee of bookings and sending them to other businesses via the GDS as guarantee of their bookings. This will mean we will have to move to Protx Direct and get our server setup PCI-DSS certified. For the current phase we're only taking bookings through our own site for this phase we can get away with using Protx Server and thus never touch credit card details. We may be able to avoid the full audit process by never storing the credit card details on our system, instead passing them to Protx Direct for authentication and storage and thus possibly being able to self-certify.

In terms of what we’re actually charging in this phase, we will not be charging the full fee for the properties. Instead we will take a holding fee of between 5 and 15% (which by happy coincidence will be the amount set as our commission amount on the property details AC 1.6.1.4). The customer will pay the remaining balance in line with the supplier’s check-in conditions when they arrive at the property.

## Corporate and Affiliate Bookers

The big difference corporate and affiliate bookers have with other classes of bookers is that they are booking for other people rather than themselves. They will likely want those other people to be able to see details of their booking and make minor amendments to it.

To facilitate this they will need to be able to see all bookings they’ve created, despite them being for another user. We will probably need to produce reports and statistics for our own and their consumption.

Another feature that may prove useful is that of white labelling. Corporate clients will want something they can use for their intranet listing only the properties they’ve approved and contracted for. Affliliates will be primarily be interested in re-branding the site for their own use. We will have to be careful when doing this not to dilute the value of our own site.

## Authentication

In the new design Authentication is split into two parts, one for the website level users and one for the web service level users.

Website level authentication is done against the webservice database using a .NET’s role provider.

Web Service level authentication is not yet implemented but will be done using HTTP basic authentication. Both OTA and private level access must be authenticated.

## Authorisation

Authorisation is a bit haphazard at present. This is also done against the web service database but does not use the .NET authorisation mechanisms.

## Mail

Mail is performed by the webserver. It uses a series of classes that as wrappers around the .NET MailMessage Classes.

There are several reasons we will need to send an email.

|  |  |
| --- | --- |
| To | Reason |
| A | B |

## Audit

All messages coming into the system will be audited at a summary level. Any successful or failed data change needs to be audited.

We will record, when, where and what was changed. The following events are to be logged:

* Profile Creation
* Profiles Updates
* Booking creation
* Booking Updates
* Protx Notifications
* User Login
* Booking accessed by TAS staff
* Property/Apartment Add/Update
* Any other significant events

Whilst implementing this we also need to bear in mind that the user will be able to see all audit trail entries relating to them. TAS staff will be able to see all audit trail entries for all entities.

# TABS Web Service

## Introduction

The TABS web service manages the property, bookings and contact information. All sites that wish to interact with us must make their requests are via the web service (including ApartmentService.com).

## System Overview

### Level 1 – System Layout

The TABS web service is divided into two major subsystems. The OTA Post subsystem is available to our own websites and the other OTA compliant systems we link to. This allows availability searches and booking of our apartments. The site services system is a private part of the system that is available only to our own websites. It contains the services need to run the site including dictionaries and storage of site data.

TABS Web Service

Site Services

OTA

OTAPost

Property Management

Dictionaries

Enquiries

System/User Management

Other Systems

Apartment Service.com

Interface Tech

Search & Avail

Booking

### Level 2 – Request Processing

OTA Interface

Site Services Interface

Booking FE

Prop Management FE

Property Cache FE

User Management FE

SQL Data Feed Agent

OTA Datafeed Agent

Search and Prop Detail FE

### Level 3 – Message Data Path

Web Service

Business Logic Layer

.NET

POS  
Auth

HTTP Auth Handler

De-serialize

F.E.  
(Process)

F.E.  
(Respond)

Request In

Response Out

Serialize

SOAP Endpoint

Audit

Audit

The above diagram describes the path of an OTA message as it flows through the system. An initial request is received by the web server where it comes into contact with our authentication plug-in. The authentication plug-in allows us to validate request from http basic authentication using websites before it even hits out website. Requests are then processed by the Microsoft SOAP web services engine which selects the correct deserializer to turn the XML into a business object and then passes the request on to the correct functional engine for processing. The processed message is returned by the F.E. and once again serialised into XML before being returned to the requestor.

### Level 4 – Functional Engine for Availability Search

Request Object

BRE In

OTA DAL

BRE Out

Response Object

Aggregate

SQL DAL

Here we have the functional engine for the availability search. The request is first processed and possibly modified by the business rules engine. It is then passed to the data access layer that queries the database and returns a result. The received results are passed once again to the BRE and being returned to the requestor.

## Interfaces

### OTA Messages

OTA messages are the path into our database by which XML enabled switches and hubs exchange booking and reservation information with us. On the outbound path they allow us to book properties that are not held in our database. On the inbound path they allow us to display information on those properties.

AC 8.1.7

#### OTA\_PingRQ

The ping operation is designed to allow us to test the operation of the web service. It is unique in that it is the only OTA message we process that doesn’t support the POS element.

|  |
| --- |
| **Supported Parameters** |
| EchoData |
| TimeStamp |

#### OTA\_HotelAvail

The hotel avail RQ has two purposes

1. To search for properties meeting a specific criteria and return information on them including rates and availability.
2. To provide detailed information on a specific property.

Before a search or avail request is sent we need to take into account paramaters outside of the OTA capabilities, e.g. keywords or loose search requirements. For example before we search we know that the clients tolerance is set to high so we return results within a wide radius of the user Point Of Interest (geographical). Then as the results come in we need to arrange them in order of relevance (relevance being calculated from how far the result is from the users original specification). The relevance algorithm is already documented to some extent by Chris G but needs to be worked into this design.

#### OTA\_HotelSearch

Hotel search allows external sites to search us for properties meeting a specific criteria and return information on them excluding rates and availability.

#### OTA\_HotelDescriptiveInfo

This message requests specific information about a given property.

#### OTA\_HotelRes

Allows reservation of an apartment that we hold authoritive availability for; this could be in the local database or forwarded to a remote system we are able to book on via a data feed agent (i.e. Interface Tech). It allows its caller to create a reservation using details the caller has previously collected using other OTA messages. Our GDS linkup provider Interface Tech supports the 2006A version of this message.

Clients of the system are expected to use the OTA\_HotelAvail\_RQ to determine availability and pricing information before attempting to book. When creating a reservation the client system will send the details of the rate it wishes to use and the price it expects. If the rate, price or room is no longer available the server will send back an error indicating this.

#### OTA\_HotelResNotif

This message is one way to cancel a booking. This message supports stateful operation. What we mean by this is whilst a client can just cancel a booking by sending a straight cancel “Commit” operation; it is also possible for them to send an “initiate” message to ask what the penalty is for cancellation. If they dislike the answer then they can cancel the cancellation by sending a “cancel” message, otherwise they can just confirm the cancel by using a “commit” message.

### Private Messages – Searching

#### Search (Basic and Advanced)

The search function is called by the Search Results page. Its job is to provide a list of all properties within the database who match the criteria specified in the database.

AC 1.3 & 1.4

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Location | Name of area hotel is in. |
| Date Range | Date from + to of guests stay |
| Date Position Tolerance | How many days stay may deviate from date range specified. DateFrom+x to DateTo+x. May be positive or negative where P > x > -P |
| Date Length Tolerance | How many days length of stay may deviate from date range specified. DateFrom+x to DateTo+y. May be positive or negative where x+y < L |
| Number of Guests | Number of Adults and Number of Children to stay |
| User | User who is making search if known |
| Amenities | Amenities required |
| Nearby Landmark | Hotel is within x miles of landmark y. |
| Rate Range | High and low limits for prices |

**Information Out**

|  |  |
| --- | --- |
| Field | Description |
| Property Name | Name of the property |
| Description | Brief description of the property |
| Price Range | Range of prices for the property |
| Small Image | A small 70x70 jpeg image of the hotel |
| Booking Channel | Which booking channel this booking came from |
| Property ID | Unique identifier of property |
| Relevance | How relevant the result is |

#### Get Property Detail

This message drives the property detail screen. It requires only two parameters to determine which property to show. If we wish to provide prices and availability information we must provide a host of optional parameters.

When the message is received the booking channel is read and used to determine which data feed agent we should pass it to for further handling.

AC 1.5

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Booking Channel | Which booking channel this booking came from |
| Property ID | Unique identifier of property |
| Date Range | Date from + to of guests stay |
| Date Position Tolerance | How many days stay may deviate from date range specified. DateFrom+x to DateTo+x. May be positive or negative where P > x > -P |
| Date Length Tolerance | How many days length of stay may deviate from date range specified. DateFrom+x to DateTo+y. May be positive or negative where x+y < L |
| Number of Guests | Number of Adults and Number of Children to stay |
| User | User who is making search if known |

**Information Out**

|  |  |
| --- | --- |
| Field | Description |
| Property Name | Name of the property |
| Description | Description of the property |
| Images | Location of images of the property |
| Amenities | Amenities the property has |
| Apartment Types and Rates | List of apartment types available and the prices per night + total (per room or per person) for each type based on the information provided. |

### Private Messages – Booking Engine

#### Create/Update Booking

The booking message is what is used to hold and then confirm bookings.

AC 1.6

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Type of booking | {Hold or Confirmed} |
| Hold Length | If booking is a hold, how long do we wish to request hold for. |
| Booking Channel | Booking channel hotel came from |
| Property ID | Which Hotel to book |
| Booking ID | Present if it’s an update |
| Room Type | Type of room to book |
| Date Range | When they want to stay |
| Guests Details | For each guest, name and age type |

**Information Out**

***Success***

|  |  |
| --- | --- |
| Field | Description |
| Booking ID |  |
| Hold Length | If booking is a hold request, how will the hold last (may not be as long as we requested). |

***Failure***

|  |  |
| --- | --- |
| Field | Description |
| Reason |  |

### Private Messages – Property Management

The property management messages are concerned with managing properties in our own database. They cannot and should not be used to manage properties from other booking sources.

AC: 5

#### Adding/Updating a Property

This message is used for adding, updating and deleting properties from the database. Deletion is a “soft” delete.

**Information In**

|  |  |
| --- | --- |
| Field Name | Description |
| Property ID | If update supply property id | |
| Property Name\* | Minimum of 1 character |
| Description | Description of the property | |
| Address\* | UK addresses must be validated against UK postcode list. Countries and regions ISO standard. Geocode must be generated by ourselves. |
| Images | JPEG or PNG format size limit 500kb |
| Property Level Amenities | Must be in amenties code table |
| Cancellation Policies | If Null inherit umbrella from Vendor Profile. (phase 1, 2 all will inherit) |
| Check in Policies | If Null inherit umbrella from Vendor Profile. (phase 1, 2 all will inherit) |
| Keywords |  |
| Deposit amount percent or fixed | Percent 0-100 |
| Commission amount percent or fixed | Percent range 0-100 |
| Distribution Channels to send on |  |
| Visible (online/offline) | true/false |
| Booking | None/Request/Availability |

***Success***

|  |  |
| --- | --- |
| Field | Description |
| Property ID |  |

***Failure***

|  |  |
| --- | --- |
| Field | Description |
| Reason |  |

#### Adding/Updating an Apartment Type to Property

This message allows the addition and changes to properties apartment types.

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Auth | User | |
| Property Id\* |  |
| Type ID | Type id if updating |
| Type Name\* |  |
| Type Class | OTA Type |
| Amenities |  |
| Keywords |  |
| Rates |  |

**Information out**

***Success***

|  |  |
| --- | --- |
| Field | Description |
| Type ID |  |

***Failure***

|  |  |
| --- | --- |
| Field | Description |
| Reason |  |

#### Adding/updating an Apartment

This allows us to add a set of apartments to a property. Each apartment must be linked to a currently active type on the property.

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Auth | User | |
| Property Id |  |
| Type Id |  |
| Apartment Name |  |

**Information out**

***Success***

|  |  |
| --- | --- |
| Field | Description |
| Apartment IDs |  |

***Failure***

|  |  |
| --- | --- |
| Field | Description |
| Reason |  |

#### Adding an Apartment Availability

This message sets the availability status for a given set of apartments at a given time.

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Auth | User |
| Apartment IDs |  |
| Date range |  |
| Status |  |

**Information out**

Success

***Failure***

|  |  |
| --- | --- |
| Field | Description |
| Reason |  |
| Booking ID | If failure is due to collision with booking(s) then this the ref number(s) |

#### List Properties

The function of this message is to populate the property list page with a list of properties that the user logged in is authorised to access.

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Auth |  |
| Supplier ID Filter | Optional Allows user to filter properties by supplier id. |

**Information Out**

|  |  |
| --- | --- |
| Field | Description |
| Name |  |
| ID |  |

#### Get Property Details

This method drives the property management details screen. It deliberately uses the same structure as the add/update method.

**Information In**

|  |  |
| --- | --- |
| Field | Description |
| Auth | User |
| Property ID | Unique identifier of property |

**Information Out**

|  |  |
| --- | --- |
| Field Name | Description |
| Property Name\* | Minimum of 1 character |
| Description | Description of the property | |
| Address\* | UK addresses must be validated against UK postcode list. Countries and regions ISO standard. Geocode must be generated by ourselves. |
| Images | JPEG or PNG format size limit 500kb |
| Property Level Amenities | Must be in amenties code table |
| Keywords |  |
| Apartment Types | List of apartment types |
| Cancellation Policies | If Null inherit umbrella from Vendor Profile. (phase 1, 2 all will inherit) |
| Check in Policies | If Null inherit umbrella from Vendor Profile. (phase 1, 2 all will inherit) |
| Deposit amount percent or fixed | Percent 0-100 |
| Commission amount percent or fixed | Percent range 0-100 |
| Distribution Channels to send on |  |
| Visible (online/offline) | true/false |
| Bookable | true/false |

#### Get Apartment Type Details

**Information in**

|  |  |
| --- | --- |
| Field | Description |
| Auth | User | |
| Type ID | Type id |

**Information Out**

|  |  |
| --- | --- |
| Field | Description |
| Type Name\* |  |
| Type Class | OTA Type |
| Amenities |  |
| Keywords |  |
| Rates |  |
| Apartments |  |

#### Get Apartment Details

**Information in**

|  |  |
| --- | --- |
| Field | Description |
| Auth | User | |
| Apartment Id | Unique identifier of apartment |

**Information Out**

|  |  |
| --- | --- |
| Field | Description |
| Apartment Name | Name of apartment |
| Availability | Availability of property |
| Bookings | List of bookings attached to the property |

### Private Message – Enquiries

#### Save Enquiry

This component will be reused from the original design.

AC 1.7

#### Get Enquiry

This component will be reused from the original design.

#### List Enquiries

This component will be reused from the original design.

### Private Message – Audit

#### List Audit Messages

This component will be reused from the original design.

#### GET Audit Message Detail

This component will be reused from the original design.

#### Add Audit Message

This component will be reused from the original design.

## Functional Engines

### Booking

#### Search

The de-serialised search object will first be parsed for validity (you can never be too careful) against our own business validation rules (I’m referring to the hard coded ones for phase 1 not the soft-coded BRE). The authenticated and validated query will now be passed asynchronously to the data feed agents. It is important to remember that there is no guarantee that the data feed agents will return valid results or that they will return them within a reasonable time-frame. As a result of this we need to set a timeout on result return and go with what results we have if it is exceeded.

The data feed agents responses are aggregated and returned to the requestor. Each result will be tagged with a booking channel marker so that we know where it came from. If the same result came twice from two booking channels we will remove the one which came from the booking channel with the lowest priority (priority is usually inversely proportional to cost).

This list is automatically filtered depending on which user is searching and which site is retrieving the information from the web service. For example, roomspace.com would be filtered so that only Roomspace properties appear. You can filter the feed by:

* property type
* location
* chain code

#### Reservation

It uses the booking channel field to select which data feed agent to tell about the booking and then passes the remaining information on and waits for a response.

## Data Feed Agents

### Local

This is how we manipulate the data in our local SQL server database. The SQL server is Microsoft SQL Server 2005 with at least SP2. Authentication is done by use of the Windows Authentication using the Web Server’s machine account.

Phase 1

### OTA - InterfaceTech

This uses OTA messages to communicate with an external database. The specific dialect in this case is “OTA2006 – InterfaceTech ReservIT variant” specified in their interfacing document. Authentication is done by means of HTTP Basic Authentication over a SSL secure channel.

Phase 4

## Database Entities



### Affliate/Corporate Booker

An affiliate somewhat like a travel agent is someone who books properties through our website on behalf of other people. They share much with corporate bookers, who book on behalf of members of their firm. The difference between the two is that we may split the commission between us and the affiliate, whereas depending on our agreement with corporate we will likely pass it back to them as a saving.

**Data Held**

|  |  |
| --- | --- |
| Field | Notes |
| Profile ID | Foreign Primary Key |
| Class | Enum { Affiliate, Corporate } |

### Apartment

An apartment is the part of a property which a client has to themselves and that can therefore be booked as a unit.

**Data Held**

*Core*

|  |  |
| --- | --- |
| Field | Notes |
| Apartment ID | Unique id |
| Property ID | Property it is part of |
| Name |  |
| Availability |  |

**Web Services**

* S

### Apartment Type

A type is a group of properties that share certain similarities.

**Data Held**

*Core*

|  |  |  |
| --- | --- | --- |
| Field | Notes | |
| Type ID | Unique id | |
| Property ID | Property it is part of | |
| Name |  | |
| Type Class | OTA Type | |
| Rates |  | |
| Keywords |  |
| Amenities |  | |

*Rates*

Rates are always specified per night.

|  |  |
| --- | --- |
| Field | Notes |
| Amount |  |
| Currency |  |
| What amount is for | (per person/per room) |
| Tax inclusive |  |
| Tax amount |  |
| Tax percent |  |
| Tax Description |  |

**Web Services**

* SaveDirectBookingRegistration

### Booking

A booking for an Apartment is registered against a specific client profile.

**Data Held**

|  |  |
| --- | --- |
| Field | Notes |
| Booking ID |  |
| Booker | Client Profile of primary booker, (optionally) Profile of Affiliate who booked on their behalf. |
| Who | List of all guests on this booking |
| What | ID of Apartment booked, Type of Apartment Booked |
| When | Date Apartment booked (relates to an availability record) |
| Booking Source | Direct, GDS, etc |
| Price | Price breakdown of booking |
| Status | {cancelled}, {provisional}, {confirmed}, {checked in} |
| Guests | List of Guests with ages and optional names |

**Web Services**

•

### User

A user is someone who has access to our website via a username and password. They could be an internet client; an affiliate; or a supplier.

**Data Held**

|  |  |
| --- | --- |
| Field | Notes |
| UserIDCode |  |
| Username |  |
| Password |  |
| Enabled |  |

**Web Services**

* DisableUser

### Group

Users are members of groups which define a group of people with similar access requirements.

### Enquiry

An enquiry is produced when a customer fills out the enquiry form. It is intended for situations when a client cannot find what they are looking for.

**Data Held**

**Web Services**

**List**: GetEnquiries  
**Save**: SaveEnquiry  
**Get**: EnquiryGet

### Property

A property is a group of apartments in the same complex or building.

**Data Held**

|  |  |  |
| --- | --- | --- |
| Field | Notes | |
| Property ID |  | |
| OwnerID |  | |
| Property Name |  | |
| Address |  | |
| Images |  | |
| Amenities |  | |
| Keywords |  | |
| Distance from Near |  | |
| Cancellation Policies | If Null inherit umbrella from Vendor Profile. (phase 1, 2 all will inherit) |
| Check in Policies | If Null inherit umbrella from Vendor Profile. (phase 1, 2 all will inherit) |
| Active |  | |

### Profile

There are several different kinds of profiles.

* Client – Ordinary persons who have booked an apartment for their own use.
* Affiliate – Other companies who wish to book apartments on behalf of other clients.
* Suppliers – Apartment owners who wish to have their apartments listed on apartmentservice.com

**Data held**

*Common to all*

|  |  |
| --- | --- |
| Field | Notes |
| ProfileID |  |
| UserID | Optional |
| Name |  |
| Address |  |
| Email Addresses |  |
| Telephone Numbers |  |
| Created |  |
| Updated |  |

*On-site Suppliers*

People who supply us with properties. Basically anyone who uses the property manager to sell their properties through our site.

|  |  |
| --- | --- |
| Field | Notes |
| ProfileID |  |
| TermsConds |  |
| Cancellation Policy |  |

*Payees (Both Suppliers and affliliates)*

This is people who we have to pay. It’s rather unlikely we will have to pay Suppliers as we take commission at source and the client pays the balance when they check in.

|  |  |
| --- | --- |
| Field | Notes |
| ProfileID |  |
| Settlement Method |  |
| Banking Details |  |

### Audit

Auditing primarily takes place in the functional processing layer.

**Data Held**

|  |  |
| --- | --- |
| Field | Notes |
| Who Did it (username, ip address) |  |
| What they Did |  |
| When |  |
| Profile it relates to if any |  |
| Booking it relates to if any |  |
| Whether the user can see the entry |  |

# Website

## Introduction

The website is not just a frontend to the TABS web service. It manages the workflow of the booking process from beginning to end.

AC 1

## Branding and Style

All branding and style on pages will be customisable using CSS templates. The master page will be able to change which CSS template is loaded at runtime so that we are able to “skin” the site’s look.

AC 6.4.2

## Pages

### Search and Detail

#### Basic Search Box

Search and Detail Pages

Basic Search Box (Master Page)

Advanced Search

Search Results without Avail or Prices

Search Results showing Avail & Prices

Prop Detail without Avail or Prices

Prop Detail showing Avail & Prices

Got Nos. & Dates?

Book Property

Yes

No

Nos. & Dates supplied

Nos. & Dates supplied

This isn’t actually a page in its own right but rather part of the master page. It allows users to search for properties with the minimum of information.

AC 1.3

#### Advanced Search

From here we can specify all kinds of details about what we want in a property. Many of the filters used in this page will be applicable on the Search Results page to allow a user to ‘filter’ their results.

Page will have:

* Country
* Destination
* Keyword
* Arrival Date
* Return Date
* No of Travellers
* Min Price
* Max Price (with currency dropdown)
* Amenities

AC 1.3

#### Search Results

This page is a grid view showing the properties that match the users search criteria. If the user has specified dates and guest numbers then it will show restrict itself to showing properties that are available and will list their prices. If on the other hand the user has not specified this, then it will present all properties that are open for booking and the price ranges. It will also give them the opportunity to provide a date range and guest numbers so as to refine their search. There may also be other factors the user wishes to search on, the page will allow the user to refine their search using these as well.

AC1.4

#### Property Detail

This shows the details of a property and the apartment types within the property that are available for enquiries and booking.

AC 1.5

### Booking Manager

Can’t grab hold reservation?

Confirm Guest Details & Login/ Create Account

Grab Hold & Redirect to Protx

Confirm Booking

Thank You

Protx Payment

Booking Engine

Fail?

Booking Failure

Fail?

Payment Failure

Success

Yes

Yes

No

No

#### Confirm Guest Details and Booker Login

In order to pass the booking to a Property we need the guest names. We gather them at this page. If the booker is not currently logged in to the site we need to them to do so at this point. If the user does not have a user id then we should allow them to create a new account directly from this page. Once we have all these details we can place a hold booking on the property and attempt to pass their details to Protx.

#### Recieving Page from Protx – Confirmation Page

This is one of the possible pages that Protx will redirect a user back to after having completed entering their card details (we decide which one when Protx sends a notification message to us). If the payment has been successful it will be this confirmation page. The page will display a summary of the booking and once the user clicks confirm on this page we will actually make the final booking. There is of course the possibility that the booking will fail in which case we will redirect the user to a page asking them if they wish to send a manual enquiry.

#### Recieving Page from PRotx – Failure Page

If the user gets to this page then something has gone wrong with the card they were attempting to pay with (failed security check or they clicked cancel). The Protx notification message will have already caused us to cancel the hold on the property. Here we simply display our apologies and redirect them back to our home page.

#### Booking Failure Page

If the user gets to this page, then unfortunately something has happened to their booking that we can’t recover from. It could be that the property has been booked in the interim, or that the price we offered them has expired. Here we simply display our apologies and redirect them back to our home page.

### Property Manager

Type Details

Property Manager

Property List

Property Details

Basic Details

Apart Type List

Basic Details

Prices List

Apartment List

Apartment Details

Availability

Bookings

Images

Amenities

Amenities

#### Property List

This page is a simple grid view with a list of all the properties that the user ID logged in has access to. If the user logged in is the site administrator then they have access to all properties.

#### Property Details

This page shows the details of an individual property such as name and amenities. From here a user can drill down into the details of an apartment type attached to that property. It also contains an image management system which allows users to add/edit/delete pictures to the property.

#### Apartment Type Details

Here we have the common shared features of a group of apartments in the same property.

#### Apartment Details

This page doesn’t actually show many details about the apartment itself. It rather shows the bookings and availability of the apartment.

## Dictionaries

Dictionaries are lists of items that are used to populate dropdowns and similar.

# Appendix A – eTrak Form Format

<fieldname>: <data><cr><lf>

### Fields

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Default |
| source | Website id for Marketing Source | roomspace |
| realname |  | roomspace265@roomspace.com |
| author |  | roomspace265@roomspace.com |
| titl | Title |  |
| fname | Forename |  |
| lname | Surname |  |
| company | Company |  |
| emad | Email Address |  |
| tel1 | Main Telephone number (home/office/mobile) |  |
| fax | Fax number |  |
| apdetail |  |  |
| town | Town of apartment |  |
| location | Specific location within town |  |
| arrd | Arrival Date |  |
| eta | Arrival Time |  |
| depd | Departure Date |  |
| nights | Nights (departure-arrival) |  |
| adult | Adults staying |  |
| child | Children staying |  |
| pax | Total staying (adults+children) |  |
| mysubmit |  | Submit Request |
| dummy |  | dummy |
| apartment | Type of apartment |  |
| budget | Budget of apartment |  |
| country | Country of Apartment |  |
| reason | Reason for stay |  |
| comments | Freetext comments |  |

### Enumerations

|  |
| --- |
| Apartment Type |
|  |
|  |
|  |
|  |

|  |
| --- |
| Budget |
|  |
|  |
|  |

|  |
| --- |
| Reason for Trip |
| Business trip |
| Relocation |
| Leisure |

# Appendix B – GDS booking Notes

The GDS system charges a flat fee per reservation made. However, commission is paid directly to the agent through whom the reservation was booked. There are services that will collect commissions for us direct from hotels such as that which Pegasus offers.

If a client makes a reservation through the GDS then wish to cancel it this must be done by them through the GDS they cannot cancel with us directly. This is because otherwise the GDS will have no way to know that the booking been cancelled and therefore will still charge us the fee for it anyhow.

This causes complications when we have “no-shows”. Because they’ve not cancelled their booking the GDS still charges us the fee. We can get around this by taking a credit card guarantee or taking it into account when charging a deposit.

A 'chain code' is a two-letter code used by the GDS to identify what chain hotels belong to.

# Appendix C1 – Work Items for Compliance with Spec

DISCLAIMER

This list is not comprehensive as it contains only the issues I’ve found so far. Given the fragile nature of the code base I expect to find more as we try and fix these.

High Priority

1. Separate out the public OTA messages from the private ones and move the public ones to ota.asmx. Each message must have its own SOAP action type and be handled in SOAP messaging format not RPC in order to comply with the OTA transport specification.

Public messages are:

* 1. PingRQ
  2. CancelRQ
  3. HotelAvailRQ
  4. HotelResRQ
  5. HotelSearchRQ (low priority unused in ApartmentService.com)
  6. HotelDescriptiveInfoRQ
  7. ProfileCreateRQ

Private:

1. OTA\_HotelRatePlanNotifRQ
2. OTA\_HotelInvCountNotifRQ
3. OTA\_UpdateRQ
4. OTA\_DeleteRQ
5. Ensure all OTA messages have the correct message version tags.
6. Go through each and every public OTA message and verify that its xml parameters are being used in the fashion indicated in the HTNG/ExplIT specification. (BIG JOB)
7. Remove user management from OTA\_ProfileCreateRQ. This needs to be separately handled, you CANNOT use a <givenname> tag to store a username and password if you plan to work with any other 3rd party implimentation.
8. Make apartments represented by roomtypes on a hotel (property) not individual hotels when sending out OTA data.
9. POS being sent back on responses is not really meaningful. We need to look at these and decide on what to send back for each apartment.
10. Apartments should default to not avail rather than avail. This is a fairly big job, and is due to an architectural flaw where all availability data is represented by bookings.
11. It is possible to cancel a cancellation operation even if it's been committed. This is a mistake.
12. Cancellation operations do not return properties cancellation penalty rules.
13. Loading the audit trail via the Intranet is painfully slow. It loads the whole thing rather than presenting it 10 items at a time.
14. Currency.
    1. All Price fields should include a currency field which is passed with them across procedure calls, having just the amount is insufficient.
    2. All pages that show prices must show currency that those prices are in.
15. Protx. System is currently stuck on charging £10 GBP for every transaction.
    1. Type of payment method must be included in property table and used to decide on how much to charge for deposit via Protx. For ERS properties option must be given to pay the balance as well.
16. Shift address, currency, geocode, termsconditions from apartment to property.
17. HTTP Basic Authentication against TAS DB is specified in TD but not implemented yet.
18. Mechanism to add images to property.
19. Mechanism to add amenities to both properties and apartments.

Medium Priority

1. Stop adding entire xml messages to the audit trail; instead add them to a separate debug log. Audit is for information useful to users, not developer information.
2. Normalise affiliate and supplier information out of the client table. Replace with a view to keep code working. In future the test for whether someone is an affiliate or a supplier should look to see whether they have a record in the normalised affiliate or supplier table.
3. Strongly type all datasets passed on the web service. Passing untyped datasets is prone to mistakes.
4. Convert disable user to being a UpdateSecurityInfo operation. Architectural consistency.
5. SaveDirectBookingRegistration doesn’t allow changing all aspects of booking registration by extranet/bookingdetails.aspx. (Note that Extranet/bookingdetails.aspx doesn’t allow saving of changes to all aspects of the booking anyhow.)
6. Geocode is not ISO compliant as the spec says it should be.
7. No mechanism is included to apply geocodes to new addresses.
8. Affiliates aren’t able to truely book properties on behalf of other persons as indicated by spec. Whilst they can put another persons name on the booking that is all they can do.  The person they book for can’t then view that booking via the booking viewing interface on the site, nor is the booking linked to that person properly in the database.  It is instead linked to the affiliates’ clientid which means we don’t’ have a full profile for them.

Low Priority

1. According to the spec “To allow for future foreign language features, data will be stored in the database in Unicode”. This has not been done consistently.
2. Business Rules Engine.

Wishlist

1. Data feed engine for fivestar.

# Appendix C2 – Task List for July

|  |  |  |
| --- | --- | --- |
| Task 5 | Apartments defaulting to not avail rather than avail | 2-3 working days |
| Status: Done | | |
| Possible solution – When we create a new apartment we can add a record to the directbookingregistartions table that apartment is unavailable from the moment of creation and for a long time. Later apartment owner can delete this record and say that apartment is available | | |
| 1. we need to change property/apartments maintenance 2. we need to change datatransfer | | |

|  |  |  |
| --- | --- | --- |
| Task 8 | Allow more fields to be edited on a booking | 1 working day |
| Status: TODO | | |
| the fields I want to be able to edit on ApartmentService.Site.Extranet.BookingDetails::SaveDirectBookingRegistration() are:  Address  DepositPrice  EMail  GuestCount  ClientName  Phone  TotalPrice  Town | | |
| Edit ApartmentService.Site.Extranet.BookingDetails::SaveDirectBookingRegistration() | | |

|  |  |  |
| --- | --- | --- |
| Task 9 | Move Search filtering from website to webservice | 5 working days |
| Status: ??? | | |
|  | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Task 11 | OTA message fixes | 10-15 working days |
| OTA messages need to be moved from website to web service and brought into compliance with the interface tech dialect. | | |
|  | | |

# Appendix D – OTA Usage by XPath

## OTA\_HotelSearchRQ

|  |  |
| --- | --- |
| **XPath** | **Meaning** |
| / |  |
| @EchoToken | Allows sender to uniquely identify a transaction. Must be copied and returned with response. |
| @TimeStamp | ISO timestamp |
| @Target | {“Test”,”production”} |
| @Version | OTA version being used |
| @PrimaryLangID | (optional) ISO language code. If specified only give responses in that language |
| @MaxResponses |  |
| /POS | Point of Sale Object – Contains information on the originating system & company that sent the request |
| /POS/Source | Source of request |
| @RequestorID |  |
| @Type | BookingChannelType OTA code |
| @ID | Unique ID assigned by ReservIT to each trading partner. 6 chars long |
| @CompanyName | Name of the company doing the sale to the customer |
| /Criteria | Parameters that are used to filter the search for hotels |
| /Criteria/Criterion | An individual filter parameter, you can only specify one per Criterion tag |
| /Criteria/Criterion/HotelRef | Hotel identification codes (ReservIT require at least one of these criterions per search) |
| @ChainCode | Chain code in ReservIT |
| @AreaID | Internal ReservIT zone id code. Could be World, continent, country, area, city or district |
| @HotelName | Specific hotel name |
| /Criteria/Criterion/RoomStayCandidates | Room occupancies (how we plan to fill rooms with our guests) |
| /Criteria/Criterion/RoomStayCandidates/ RoomStayCandidate/GuestCounts/ |  |
| /Criteria/Criterion/RoomStayCandidates/ RoomStayCandidate/GuestCounts/GuestCount | Number of persons staying in room |
| @Age | Age of Guest. Use this to let find which age qualifying code matches which age for this hotel. Response will return a guestcount with age qualifying code replacing this element. |
| @AgeQualifyingCode | Age Qualifying Code ‘7 for infants, ‘8’ for Children, ‘9’ for juniors, ‘10’ for adults, ’11’ for old aged pensioners |
| @count | Number of persons for this age or age qualifying code |
| /Criteria/Criterion/StayDateRange |  |
| @start |  |
| @end |  |
| /Criteria/Criterion/Address |  |
| @cityname |  |
| /Criteria/Criterion/Position |  |
| @longitude |  |
| @latitude |  |
| /Criteria/Criterion/Radius |  |
| @distance |  |
| @distancemeasure |  |
| /Criteria/Criterion/TPA\_Extensions |  |
| /Criteria/Criterion/TPA\_Extensions/Airport |  |
| @Cityname |  |
| @airportcode |  |
| /Criteria/Criterion/TPA\_Extensions/Keyword |  |

## OTA\_HotelSearchRS

|  |  |
| --- | --- |
| **XPath** | **Meaning** |
| /Success | Indicates search was successful even if no hotels are returned |
| /Warnings | Search raised some warnings |
| /Warnings/Warning | Each warning generated produces one of these elements |
| @code | Warning code |
| @type | Type of warning code |
| /Properties |  |
| /Properties/Property | An individual successful search result |
| @hotelcode |  |
| @chaincode |  |
| /Properties/Property/RateRange |  |
| @maxrate |  |
| @minrate |  |
| @currencyCode |  |

## OTA\_HotelAvailRQ

|  |  |
| --- | --- |
| **XPath** | **Meaning** |
| / |  |
| @EchoToken | Allows sender to uniquely identify a transaction. Must be copied and returned with response. |
| @TimeStamp | ISO timestamp |
| @Target | {“Test”,”production”} |
| @Version | OTA version being used |
| @PrimaryLangID | (optional) ISO language code. If specified only give responses in that language |
| @BestOnly |  |
| @SummaryOnly | Exclude booking and cancellation policies |
| @RateRangeOnly |  |
| /POS | Point of Sale Object – Contains information on the originating system & company that sent the request |
| /POS/Source | Source of request |
| @RequestorID |  |
| @Type | BookingChannelType OTA code |
| @ID | Unique ID assigned by ReservIT to each trading partner. 6 chars long |
| @CompanyName | Name of the company doing the sale to the customer |
| /AvailRequestSegments/ |  |
| /AvailRequestSegments/AvailRequestSegment |  |
| @AvailReqType | Mandatory “Room” – for normal request “NonRoom” for promos and packages. |
| /AvailRequestSegments/AvailRequestSegment/ StayDateRange |  |
| @start |  |
| @end |  |
| @Duration |  |
| @datewindowrange | Mandatory but not used by ReservIT, non-empty string value should be sent such as “dummy” |
| /AvailRequestSegments/AvailRequestSegment/ RateRange | Only can be used if /@RateRangeOnly=”True” |
| @MinRate |  |
| @MaxRate |  |
| @CurrencyCode |  |
| /AvailRequestSegments/AvailRequestSegment/ RoomStayCandidates/RoomStayCandidate/ |  |
| @Quantity | Number of rooms requested |
| /AvailRequestSegments/AvailRequestSegment/ RoomStayCandidates/RoomStayCandidate/ GuestCounts/ | Room occupation. If not present then all prices returned will be base prices and not real prices. |
| @Age | Age of Guest. Use this to let find which age qualifying code matches which age for this hotel. Response will return a guestcount with age qualifying code replacing this element. |
| @AgeQualifyingCode | Age Qualifying Code ‘7 for infants, ‘8’ for Children, ‘9’ for juniors, ‘10’ for adults, ’11’ for old aged pensioners |
| @count | Number of persons for this age or age qualifying code |
| /AvailRequestSegments/AvailRequestSegment/ HotelSearchCriteria/Criterion |  |
| @HotelRef |  |
| @ChainCode |  |
| @HotelCode | Mandatory |

## OTA\_HotelAvailRS

|  |  |
| --- | --- |
| **XPath** | **Meaning** |
| / |  |
| /Success |  |
| /Warning |  |
| /RoomStays |  |
| /RoomStays/RoomStay |  |
| /RoomStays/RoomStay/RoomTypes/ RoomType | When interfacing with interface tech we’re limited to one roomtype per roomstay |
| @RoomTypeCode |  |
| /RoomStays/RoomStay/RoomTypes/ RoomType/RoomDescription |  |
| #TEXT | Room Type Name |
| @Language | ISO Language code for name |
| /RoomStays/RoomStay/RoomTypes/ RoomType/AdditionalDetails/AdditionalDetail |  |
| @type | **with** **interface tech** always “12” |
| /RoomStays/RoomStay/RoomTypes/ RoomType/AdditionalDetails/AdditionalDetail/DetailsDescription |  |
| #TEXT | Detailed description of room type |
| @Language | ISO Language code for room type description |
| /RoomStays/RoomStay/RoomRates | Collection of all rates available for specific room type |
| /RoomStays/RoomStay/RoomRates/RoomRate | Each rate will have its own RoomRate element |
| @effectivedate | Start date for room stay period |
| @expiredate | End date for room stay period |
| @NumberOfUnits | Number of rooms available for this room type for this rate |
| @RatePlanCode | Identifies rate plan. If not specified Rack rate used |
| @RatePlanCategory | 6 Possible Values: RAC for rack rate; EB for early booking; LM for last minute; PRO for promotion; SPE for special rate; PKG for package |
| /RoomStays/RoomStay/RoomRates/RoomRate/Rates/Rate | Repeated for each element of the room rate I.E. 1/1 to 3/1 £50 per night 3/1 to 5/1 £60 per night. |
| @effectivedate |  |
| @expirydate |  |
| @ratetimeunit | **with** **interface tech** always “day” |
| @unitMultiplier | Number of nights covered by rate |
| /RoomStays/RoomStay/RoomRates/RoomRate/Rates/Rate/Base |  |
| @AmountAfterTax |  |
| @AmountBeforeTax |  |
| @CurrencyCode |  |
| /RoomStays/RoomStay/RoomRates/RoomRate/Rates/Rate/RateDescription |  |
| #TEXT | Description of rate if not rack |
| @Language |  |
| RoomStays/RoomStay/RoomRates/RoomRate/ Rates/Rate/TPA\_Extensions | **Interface tech tpa extensions** |
| RoomStays/RoomStay/RoomRates/RoomRate/ Rates/Rate/TPA\_Extensions/ BookingCancelPolicyRPH |  |
| @RPH | booking and cancel policies identifier used for this rate referring to a  BookingCancelPolicy element |
| RoomStays/RoomStay/RoomRates/RoomRate/ Rates/Rate/TPA\_Extensions/SpecialRateInfo | The inclusion of SpecialRateInfo conforms to interfacetech’s xsd but not their docs |
| **TODO** |  |
| RoomStays/RoomStay/RoomRates/RoomRate/ RoomRateDescription |  |
| #text | Name of rate if not Rack (optional) |
| @language |  |
| RoomStays/RoomStay/GuestCounts |  |
| @isperroom | **Interface tech** always true (mandatory) |
| RoomStays/RoomStay/GuestCounts/ GuestCount |  |
| @AgeQualifyingCode | Age Qualifying Code ‘7 for infants, ‘8’ for Children, ‘9’ for juniors, ‘10’ for adults, ’11’ for old aged pensioners |
| @count | Number of persons for this age or age qualifying code |
| RoomStays/RoomStay/TimeSpan | Start and end period for the room stay. (This has to be here because the date windowing mechanism allows us to be slightly flexible with dates). |
| @start |  |
| @end |  |
| RoomStays/RoomStay/Total | Total cost for roomstay. If multiple rates are included we go with the RACK rate. |
| @amountaftertax |  |
| @AmountBeforeTax |  |
| @currencycode |  |
| RoomStays/RoomStay/Total/Taxes | Breakdown of the taxes included in amountaftertax |
| @amount |  |
| @currencycode |  |
| RoomStays/RoomStay/Total/Taxes/Tax |  |
| @percent |  |
| RoomStays/RoomStay/Total/Taxes/Tax/ TaxDescription |  |
| #TEXT | Tax name |
| @language |  |
| RoomStays/RoomStay/BasicPropertyInfo |  |
| @ChainCode |  |
| @HotelCode |  |
| RoomStays/RoomStay/TPA\_Extensions | **TODO** |
| /TPA\_Extensions | **Mandatory** |
| /TPA\_Extensions/BookingCancelPolicy |  |
| @RPH | Identifier for booking cancel policy in list |
| /TPA\_Extensions/BookingCancelPolicy/ BookingPolicy |  |
| #TEXT |  |
| @Language |  |
| /TPA\_Extensions/BookingCancelPolicy/ CancelPolicy |  |
| #TEXT |  |
| @Language |  |

## OTA\_HotelResRQ/OTA\_HotelResRS

With this message the response message is virtually identical to the response message.

|  |  |
| --- | --- |
| **XPath** | **Meaning** |
| / |  |
| /POS | Point of Sale Tag |
| /POS/Source |  |
| /POS/Source/RequestorID |  |
| @type | **InterfaceTech.**  Always set to “2” |
| @ID | **InterfaceTech.**  Unique id of the trading partner assigned by ReservIT. This attribute is also called the Partner identifier (partid). When the message is sent by ReservIT system, this attribute will specify the direct partner code (ie the partner code for the chain) which creates the booking. For exemple: the chain 12 is a partner for the chain 2 (the part code is 4). If a booking is created by the chain 12 in an hotel from chain 2, the ID attribute will be set to the partner code 4 |
| /POS/Source/RequestorID/CompanyName/ #text | Name of the partner that performed the reservation |
| /POS/Source/RequestorID/BookingChannel |  |
| @type |  |
| /POS/Source/RequestorID/BookingChannel/ CompanyName |  |
| #text |  |
| @code | **InterfaceTech**. Specifies the chain identifier. But ReservIT server will set this attribute in order to specify the source of the booking (chain). For exemple: the chain 12 is a partner for the chain 2. If a booking is created by the chain 12 in an hotel from chain 2, the Code attribute in CompanyName element will be set to 12. The Company name value (a string) will be set with the partner name (if the booking was created a partner) or with the chain name. |
| /HotelReservations/HotelReservation |  |
| @RoomStayReservation | *Mandatory* **InterfaceTech** always set to true. |
| @CreateDateTime | Used only in a booking creation. |
| @LastModifyDateTime | Used only in a booking update. |
| @CreatorID | Used only in booking creation. Specified only when a callcentre agent made the reservation. |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay | Number of rooms of this type booked |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/RoomTypes/ RoomType |  |
| @NumberOfUnits |  |
| @RoomTypeCode |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/RoomRates/ RoomRate/ |  |
| @effectivedate | Start date for room stay period |
| @expiredate | End date for room stay period |
| @NumberOfUnits |  |
| @RatePlanCode | If not RACK rate then specify rate code here. If RACK do not specify. |
| @RatePlanCategory | 6 Possible Values: RAC for rack rate; EB for early booking; LM for last minute; PRO for promotion; SPE for special rate; PKG for package |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/RoomRates/ RoomRate/Rates/Rate | Repeated for each element of the room rate I.E. 1/1 to 3/1 £50 per night 3/1 to 5/1 £60 per night. |
| @effectivedate |  |
| @expirydate |  |
| @AgeQualifyingCode |  |
| @ratetimeunit | **with** **interface tech** always “day” |
| @unitMultiplier | Number of nights covered by rate |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/RoomRates/ RoomRate/Rates/Rate/Base |  |
| @AmountAfterTax |  |
| @AmountBeforeTax |  |
| @CurrencyCode |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/RoomRates/ RoomRate/Rates/Rate/Base/TPA\_Extensions | **TODO** |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/GuestCounts/ GuestCount |  |
| @AgeQualifyingCode |  |
| @age | Age value if qualifying code not speced |
| @count |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/TimeSpan |  |
| @start |  |
| @end |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/Total | Total cost for roomstay. If multiple rates are included we go with the RACK rate. |
| @amountaftertax |  |
| @AmountBeforeTax |  |
| @currencycode |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/Total/Taxes | Breakdown of the taxes included in amountaftertax |
| @amount |  |
| @currencycode |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/Total/Taxes/Tax |  |
| @percent |  |
| @amount |  |
| @Type | **Interface Tech** possible value “inclusive” |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/Total/Taxes/Tax/ TaxDescription |  |
| #TEXT | Tax name |
| @language |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/BasicPropertyInfo |  |
| @ChainCode |  |
| @HotelCode |  |
| /HotelReservations/HotelReservation/  RoomStays/RoomStay/ServiceRPHs/ ServiceRPH |  |
| @IsPerRoom |  |
| @RPH | Reference Place Holder (index) for service associated with this roomstay |
| /HotelReservations/HotelReservation/  Services/Service | “Stuff like breakfast or daily newspaper” |
| @ServiceRPH | Index of this service |
| @Quantity |  |
| @ServiceInventoryCode |  |
| @ServicePricingType | { “Per Stay”, “Per person”, “Per Night”, “Per person per night” } |
| /HotelReservations/HotelReservation/  Services/Service/Price/Base |  |
| @amountaftertax |  |
| @AmountBeforeTax |  |
| @currencycode |  |
| /HotelReservations/HotelReservation/  Services/Service/Price/Base/Taxes | Breakdown of the taxes included in amountaftertax |
| @amount |  |
| @currencycode |  |
| /HotelReservations/HotelReservation/  Services/Service/Price/Base/Taxes/Tax |  |
| @percent |  |
| @amount |  |
| @type | **InterfaceTech** possible value “Inclusive” |
| /HotelReservations/HotelReservation/  Services/Service/ServiceDetails/TimeSpan | Only relevant to “per night” and “per person per night”. |
| @Duration | ISO-8601 format |
| /HotelReservations/HotelReservation/  Services/Service/TPA\_Extensions | TODO |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest |  |
| @ResGuestRPH |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer |  |
| @currencycode |  |
| @gender |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/PersonName |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/PersonName/ NamePrefix/#TEXT |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/PersonName/GivenName/ #TEXT |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/PersonName/Surname/ #TEXT |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Telephone |  |
| @CountryAccessCode | I.E. +44 |
| @PhoneNumber |  |
| @PhoneTechType | Undocumented by interfacetech always “3”? |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Email/#TEXT |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Address | (not Mandatory) |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Address/AddressLine/  #TEXT |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Address/CityName/  #TEXT | Postal Town |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Address/PostalCode/  #TEXT |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/Address/CountryName |  |
| #TEXT |  |
| @code | ISO country code |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/CustLoyalty |  |
| @membershipid |  |
| @programid |  |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Profiles/ProfileInfo/ Profile/Customer/TPA\_Extensions | **TODO** |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Comments | **TODO** |
| /HotelReservations/HotelReservation/  ResGuests/ResGuest/Comments/Comment | Comment on individual guest |
| #TEXT | The Actual Comment |
| @GuestViewable | (mandatory) InterfaceTech always set to ‘1’ |
| @CommentOriginatorCode | (mandatory) InterfaceTech always set to ‘0’ |
| /HotelReservations/HotelReservation/  ResGlobalInfo |  |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Comments/Comment | Comment on reservation as a whole |
| #TEXT | The Actual Comment |
| @GuestViewable | (mandatory) InterfaceTech always set to ‘1’ |
| @CommentOriginatorCode | (mandatory) InterfaceTech always set to ‘0’ |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee | GDS will only create bookings with Credit Card Guarantee |
| @HoldTime | **InterfaceTech**. If set GuaranteeType will not be set nor will details of credit card be provided. |
| @GuaranteeType | {“CC/DC/Voucher”,”Deposit”,”PrePay”} |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee/DepositPayments |  |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee/DepositPayments/ RequiredPayment |  |
| @Amount |  |
| @Percent |  |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee/Deadline |  |
| @AbsoluteDeadline |  |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee/ GuaranteesAccepted/ |  |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee/ GuaranteesAccepted/GuaranteeAccepted/ PaymentCard |  |
| @CardType | Type of Card {“AX” – Amex, “VI” – Visa, “DN” – Diners, “DC” – Diners, “DS” – Discover, “JC” – JCB, “MC” – Mastercard, “IK” – Mastercard, “CA” – Mastercard, EC– Mastercode |
| @CardNumber | The 16 digit card number (Mandatory for this element) |
| @ExpireDate | Card Expiry Date |
| @SeriesCode | CVV number |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Guarantee/ GuaranteesAccepted/GuaranteeAccepted/ PaymentCard/CardHolderName/#TEXT | Card holder name |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Total | Booking Total Price including all services. **InterfaceTech** Not read for request but set in response. |
| @AmountAfterTax |  |
| @CurrencyCode |  |
| /HotelReservations/HotelReservation/  ResGlobalInfo /Total/Taxes/Tax |  |
| @percent |  |
| @amount |  |
| @type | **InterfaceTech** possible value “Inclusive” |
| /HotelReservations/HotelReservation/  ResGlobalInfo /HotelReservationIDs | Must be set for update or cancel. Will definitely be set in a successful response message. |
| /HotelReservations/HotelReservation/  ResGlobalInfo /HotelReservationIDs/ HotelReservationID |  |
| @ResID\_Source | **InterfaceTech** to specify ReservIT ID this must be set to “ReservIT” |
| @ResID\_SourceContext | **InterfaceTech** Set in response message with Chain code for which booking is created/updated. |
| @ResID\_Value |  |
| /HotelReservations/HotelReservation/  TPA\_Extensions | **TODO** |

# Glossary

|  |  |
| --- | --- |
| Rate Plan |  |
| Base Rate | A rate plan that contains its own pricing information |
| Rack Rate |  |
| Derived Rate | A rate plan that is based upon another rate for its pricing. For example a senior citizen rate might be priced at a 10% discount relative to the rack rate. |
| PAX | Simply put a person of any age. |
| PMS | Property Management System. A system that manages hotel room inventory, guest billing. |
| CRS | Central Reservation System. |
| GDS | Global Distribution System. These are large legacy systems used in the travel industry for passing inventory to travel agents, tour operators etc. Interface Technologies acts as an onward distributor for the major GDS’s as they are not directly accessible via the www |
| OTA | The Open Travel Alliance. They provide a ‘loose’ XML distribution standard for the travel Industry |
| HTNG | Hotel Technology Next Generation. Among other things they describe a particular implementation of the OTA standard |