Allmyles API Documentation

Release 2.0-dev02

Allmyles Ltd.

Jun 05, 2017

Contents

This version of the Allmyles API documentation is deprecated and will not be updated in the future. Please visit http://docs.allmyles.apiary.io/ for the new version with autoamtically generated, ready to use code examples

CHAPTER 1

Contents

Introduction

We highly recommend you to read at least *the 'Common Gotchas' section below* before jumping into the API reference, as it contains information about problems often encountered by developers working with our API that are not immediately apparent.

Reading This Documentation

For the most part, it should be rather easy to understand the documentation. Two things that might require explanation are:

- A colon in a URL path denotes a variable; in the path `/flights/:booking_id` the entirety of `:booking_id` should be replaced by a booking ID, as such: `/flights/8986210d-7e2f-4481-a29f-846ab386ddac`
- When declaring the type of a variable, the symbol `[ ]` after a type means that the variable's type is an array, containing items of the type named before the symbol.

  For example, a `String [ ]` variable can look like the following: `["abc", "def", ""]`.

Common Gotchas

- The staging API returns search results from all available airlines, but only the following results are guaranteed to work when booking in staging:
  - Budapest to London & London to Budapest British Airways flights, to filter for these, include `"fromLocation": "BUD", "toLocation": "LON", "preferredAirlines": ["BA"]` in your search request.
  - Budapest to London & London to Budapest Low Cost Carrier flights, to filter for these, include `"fromLocation": "BUD", "toLocation": "LON", "providerType": "OnlyLowCost"` in your search request.

- Workflows expire 17 minutes after a request arrives. Any requests for the same session will return an HTTP 412 error.
- All requests in the booking flow are applied to the last received search request. This means that if a passenger is searching for flights in multiple tabs, they will be able to proceed with booking only in the tab they last searched in.

Request Headers

In addition to standard HTTP headers, the following ones are of interest:

- **Accept**: The format the response data should be sent in. Currently we support JSON, specified as `application/json`, and XML, specified as `application/json`.
- **Cookie**: A unique identifier for the customer's session, this is typically the randomly generated session cookie saved on the customer's computer.
- **X-Auth-Token**: The API key you received after signing up on the Allmyles home page (ex. `12345678-1234`.)

Generic Response Codes

The documentation has a number of different status codes listed as possible reponses to certain requests that are specific to that response. In addition to those, we can also return the following status codes:

- **200 OK**: The request has succeeded.
- **202 Accepted**: The request has been accepted for processing but the processing is not yet completed.
- **301 Moved Permanently**: The resource at the endpoint URI has been moved permanently and the client must use the new address in the future.
- **400 Bad Request**: Invalid request syntax, the error message will in most cases contain the exact reason. The client must not repeat the request in the same form.
- **403 Forbidden**: The client failed to authorize themselves with a valid API key, and the request was rejected.
- **404 Not Found**: Request contains an invalid URI endpoint, or the given resource does not exist.
- **413 Rate Control**: Request limitation threshold violated.
- **500 Internal Server Error**: The server has failed to complete the sent request—please report such errors to our support team at support@allmyles.com!
- **502 Bad Gateway**: The backend servers are down, or an update process is under way.
- **503 Service Unavailable**: Servers are accessible, but overloaded. Try and repeat the request later.

Displaying Errors to Users

When displaying errors to your users you must:

- **Not** directly display our errors on your frontend.
- Show a **meaningful** error message to the user.
- Include the **session cookie** in the error message as an error code or error identifier.

Note: Our error messages are meant to be information rich for the developers and are not optimal for direct reproduction on the frontend. They might contain information that is seen as cryptic to your users which is why we require that you make them more user friendly and fitting for your service.

Asynchronous Calls

Long running requests, such as a flight search use asynchronous calls. This means that the the Allmyles API can—and in most cases, will—respond to the first request with an HTTP 202 status code, and no content. When this happens, the client is expected to periodically send the same request to the API, with a reasonable delay between the requests (5 seconds is a good baseline for this.) The Allmyles API will then reply with an HTTP 202 response code if processing is still underway, or any other status code and a content body if it is done processing the request.

Note: It is a good idea to implement a timeout on the client side that checks for infinite loops in this process, as the server theoretically could fail in a way that it won't stop returning 202 status codes.

Quick Start

Configuration

Create a localrc file with the following:

```
#!/bin/bash
SERVICE_ENDPOINT={ALLMYLES-API-URL-GOES-HERE}
TENANT_KEY={YOUR-TENANTKEY-GOES-HERE}
```

Search Flights

The following script starts a flight search, and then checks if a result is available every 5 or so seconds.

```
#!/bin/bash
source localrc

read -d '' PAYLOAD <<EOF
{
    "fromLocation": "BUD",
    "toLocation": "LON",
    "departureDate": "$(date -v+7d -u +'%Y-%m-%dT%H:%M:%SZ')",
    "resultTypes": "default",
    "returnDate": "$(date -v+14d -u +'%Y-%m-%dT%H:%M:%SZ')",
    "persons": [
        {
            "passengerType": "ADT",
            "quantity": 1
        }
    ],
    "preferredAirlines": ["BA"]
}
EOF
```

```
PAYLOAD=$(echo $PAYLOAD)

echo "Sending search request..."
while true
do
    echo "Checking for search response..."
    STATUS=$(echo "$PAYLOAD" | curl $* \
        -s \
        -H "X-Auth-Token: $TENANT_KEY" \
        -H "Content-Type: application/json" \
        -H "Accept: application/json" \
        -H "Cookie: 12345678-02" \
        -d @- $SERVICE_ENDPOINT/flights \
        -w "%{http_code}" \
        -o /dev/null)
    if ( [ $STATUS == "200" ] )
    then
        break
    fi
    sleep 5
done
echo "Search response received!"

RESPONSE=$(echo "$PAYLOAD" | curl $* \
    -H "X-Auth-Token: $TENANT_KEY" \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cookie: 12345678-02" \
    -d @- $SERVICE_ENDPOINT/flights)

echo BOOKING_ID=$(echo $RESPONSE | sed -n 's/.*\"bookingId\": \"\([A-Za-z1-9\-_]*\)\".
↪*/\1/p') >> localrc
echo $RESPONSE
```

Get Flight Details

```
#!/bin/bash
source localrc

curl $* \
    -H "X-Auth-Token: $TENANT_KEY" \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cookie: 12345678-02" \
    $SERVICE_ENDPOINT/flights/$BOOKING_ID
```

Book a Flight

```
#!/bin/bash
source localrc

read -d '' PAYLOAD <<EOF
{
    "bookingId": "$BOOKING_ID",
    "passengers": [
        {
            "namePrefix": "MR",
            "firstName": "Lajos",
            "lastName": "Kovacs",
            "birthDate": "1911-01-01",
            "gender": "MALE",
            "passengerTypeCode": "ADT",
            "baggage": 0,
            "email": "aaa@gmail.com",
            "document": {
                "type": "Passport",
                "id": "123",
                "issueCountry": "HU",
                "dateOfExpiry": "2015-12-01"
            }
        }
    ],
    "contactInfo": {
        "name": "Kovacs Lajos",
        "address": {
            "countryCode": "HU",
            "cityName": "Budapest",
            "addressLine1": "Xasd utca 13."
        },
        "phone": {
            "countryCode": 36,
            "areaCode": 30,
            "phoneNumber": 1234567
        },
        "email": "lajos.kovacs@example.com"
    },
    "billingInfo": {
        "name": "Kovacs Lajos",
        "address": {
            "countryCode": "HU",
            "cityName": "Budapest",
            "addressLine1": "XBSD utca 23."
        }
    }
}
EOF
echo "$PAYLOAD" | curl $* \
    -H "X-Auth-Token: $TENANT_KEY" \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cookie: 12345678-02" \
    -d @- $SERVICE_ENDPOINT/books
```

Create Your Ticket

```
#!/bin/bash
source localrc

curl $* \
    -H "X-Auth-Token: $TENANT_KEY" \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "Cookie: 12345678-02" \
    $SERVICE_ENDPOINT/tickets/$BOOKING_ID
```

Flights

Summary

The flight booking/ticket creation workflow consists of five steps.

1. *Search*
2. *Details*
3. *Booking*
4. *Payment*
5. *Ticketing*

Additional calls that are available:

- *Ticketing Status*
- *Rules*
- *Get Booking*
- *Cancel Booking*

Search

Request

POST /flights

Searches for flights that match provided criteria.

Note: In most cases you'll want to pass 00:00:00 as time for both your departure and your return date. Time filtering constraints will be very strict otherwise, often resulting in no matches for your query.

Note: If you plan to present the results of *Flexible Date Search* and regular search at the same time to your users, you have two options.

1. You send both requests in one session - you can only send the second request when you already have the results of the first one.
2. You send the two requests in separate sessions - in this case you have to include *Flexible Date Search Reference* in the regular search request, and set *to_be_referenced* to *True* in the flexible date search request.

JSON Parameters

- **fromLocation** (*String*) – departure location, given as IATA code

- **toLocation** (*String*) – destination, given as IATA code
- **departureDate** (*String*) – date of departure, in ISO format, including a time code, even though whole day will be searched by default
- **returnDate** (*String*) – *(optional)* date of return, in ISO format, including a time code, even though whole day will be searched by default
- **persons** (*Person*) – a list of passengers, grouped by type code, containing Persons
- **userData** (*User Data*) – information about the end user
- **fromAirport** (*String*) – *(optional)* departure airport, given as IATA code, must be in the city specified in `fromLocation`
- **toAirport** (*String*) – *(optional)* destination airport, given as IATA code, must be in the city specified in `toLocation`
- **providerType** (*String*) – *(optional)* type of results to retrieve
- **preferredAirlines** (*String [ ]*) – *(optional)* list of airlines to filter results to, given as their two character IATA code
- **extraDays** (*Integer*) – *(optional)* number of days to call *Flexible Date Search* with, between 1-3
- **options** (*Options*) – *(optional)* sorting and filtering options
- **flexible_date_search_reference** (*Flexible Date Search Reference*) – *(only in case of choosing option 2 described in the note above)* data about the flexible date search made with the same parameters as the regular one
- **to_be_referenced** (*Boolean*) – *(optional) True* if this is a flexible date search and a regular search is to be called next with *flexible_date_search_reference*
- **number_of_bags** (*Integer*) – *(optional)* The number of bags to be bundled with the price of LCC flights. **This option has no effect for searches with the default provider, please contact Allmyles for details on alternative providers.**
- **baggage_charges** (*Boolean*) – *(optional)* Wheter or not you would like to receive the baggage price tiers in the search step of LCC flights. Baggage price tiers are always sent in the details step, only request this data if you are using it on search. **This option has no effect for searches with the default provider, please contact Allmyles for details on alternative providers.**
- **check_in_charges** (*Boolean*) – *(optional)* Wheter or not you would like to receive the check in price tiers in the search step of LCC flights. **This option has no effect for searches with the default provider, please contact Allmyles for details on alternative providers.**
- **speedy_boarding_charges** (*Boolean*) – *(optional)* Wheter or not you would like to receive the speedy boarding fee information in the search step of LCC flights. **This option has no effect for searches with the default provider, please contact Allmyles for details on alternative providers.**

Person

JSON Parameters

- **passengerType** (*String*) – one of *PassengerTypes*
- **quantity** (*Integer*) – number of travelers of `passengerType`

PassengerTypes

One of `ADT`, `CHD` or `INF`

Options

JSON Parameters

- **sort** (*String*) – one of Sorting Options
- **filter** (*Filters*) – filtering options

Sorting Options

One of `total_fare`, `-total_fare`, `comfort_score` or `-comfort_score` (*Comfort score*). Reverse-order sorting is indicated with a `-` sign (e.g. `-total_fare` would return the most expensive option first).

Filters

JSON Parameters

- **cabin** (*String*) – one of Cabin types. Filtering for a certain cabin returns combinations that contain at least one leg with the desired cabin type.

Cabin types

One of `economy`, `premium economy`, `business` or `first`

User Data

JSON Parameters

- **ip** (*String*) – the end user's IP address, e.g. `12.123.45.67`.
- **browser_agent** (*String*) – the end user's browser agent based on the User-Agent HTTP header, e.g. `Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0`

Flexible Date Search Reference

JSON Parameters

- **cookie** (*String*) – the Cookie sent in the header of the referenced flexible date search
- **extra_days** (*Integer*) – number of days submitted in **extraDays** in the referenced flexible date search

Response Body

JSON Parameters

- **flightResultSet** (*FlightResult [ ]*) – root container

FlightResult

Warning: The `total_fare` field here does not include the credit card surcharge just yet, as fetching the exact surcharge for a specific flight can require an extra 5-10 second call to the external provider.

This surcharge is retrieved in the FlightDetails call.

Warning: The prices returned in the fields **total_fare** and **ticketing_fee** are converted to HUF by default if the provider returns them in a different currency. When displaying prices to the user, please refer to **price_charged_by_provider** for a more accurate fare, where the total fare is returned in the currency the airline is charging, or **total_fare_in_preferred_currencies** for prices converted from the original currency. **Important**: this price might change later as it is not yet updated with credit card and other surcharges.

JSON Parameters

- **breakdown** (*Breakdown [ ]*) – summary of passenger data per type
- **currency** (*String*) – currency of all prices in response
- **ticketing_fee** (*Float*) – fee charged for ticketing
- **total_fare** (*Float*) – total fare, including service fee and ticketing fee
- **combinations** (*Combination [ ]*) – list of combination objects
- **total_fare_in_preferred_currencies** (*[ ]*) – total fare converted to the client's preferred currencies, including service fee and ticketing fee
  - **currency** (*String*)
  - **total_fare** (*Float*)
- **ticketing_fee_in_preferred_currencies** (*[ ]*) – ticketing fee converted to the client's preferred currencies, including service fee and ticketing fee
  - **currency** (*String*)
  - **ticketing_fee** (*Float*)
- **price_charged_by_provider** (*[ ]*) – fare and ticketing fee in the currency the airline is charging
  - **currency** (*String*)
  - **total_fare** (*Float*)
  - **ticketing_fee** (*Float*)
- **baggageTiers** (*BaggageTier [ ]*) – contains the different options the passenger has for bringing baggages along. May be requested to be included for LCC flights, otherwise not included in the results.

- **speedy_boarding_fee** (*Price*) – Only included in LCC results, and only when requested.
- **check_in_charges** (*Check-in charges [ ]*) – Only included in LCC results, and only when requested.

Breakdown

JSON Parameters

- **fare** (*Float[ ]*) – total price of the tickets for passengers of `type` (including tax)
- **tax** (*Float[ ]*) – total tax on the tickets for passengers of `type`
- **type** (*String*) – type of passengers the breakdown is for, see (see *PassengerTypes*)
- **quantity** (*Integer*) – number of passengers of `type`
- **ticketDesignators** (*TicketDesignator [ ]*) – ticket designators applicable for passengers of `type`
- **fare_in_preferred_currencies** (*[ ]*) – fare converted to the client's preferred currencies - **currency** (*String*) - **fare** (*Float*) - **tax** (*Float*)

TicketDesignator

Ticket designators are the mini-rules for the flight, with entries such as `{"code":  "70|PEN", "extension":  "TICKETS ARE NON-REFUNDABLE|"}`.

JSON Parameters

- **code** (*String*) – ticket designator's code
- **extension** (*String*) – ticket designator's description

Combination

Combinations are the sets of different flight itineraries that can be booked. Every combination in a flight result is guaranteed to have the same total price, but the departure times, arrival times, and transfer locations can differ.

JSON Parameters

- **bookingId** (*String*) – the unique identifier of this combination (this is later used to identify the combination when booking, for example.)
- **firstLeg** (*Leg*) – the outbound leg of the itinerary
- **returnLeg** (*Leg*) – the inbound leg of the itinerary
- **serviceFeeAmount** (*Float*) – ticket designator's description
- **comfortScore** (*Comfort score*) – the comfort score of the combination
- **service_fee_in_preferred_currencies** (*[ ]*) – service fee converted to the client's preferred currencies - **currency** (*String*) - **service_fee** (*Float*)

Leg

Legs are made up of one or more segments, and span from one location the customer searched for to the other.

JSON Parameters

- **elapsedTime** (*String*) – The total time between the leg's first departure, and last arrival (including time spent waiting when transferring). It is given in the format `HHMM`.
- **flightSegments** (*Segment [ ]*) – The list of segments this leg is made up of.

Segment

Segments are the smallest unit of an itinerary. They are the direct flights the passenger will take from one stop to another.

JSON Parameters

- **departure** (*Stop*) – data about the flight's departure
- **arrival** (*Stop*) – data about the flight's arrival
- **aircraft** (*String*) – Planned aircraft scheduled for the specific segment
- **availableSeats** (*Integer*) – the number of seats available for this price tier—the maximum number we can know of is 9, so when 9 is returned, that means 9 or more seats are available.
- **cabin** (*String*) – one of 'economy', 'first', or 'business'
- **class** (*String*) – an airline-specific identifier used in fare pricing. The code related to comfort score is cabin code.
- **marketingAirline** (*String*) – two character IATA code of the marketing airline that publishes and markets the flight booked under its own airline designator and flight number. The marketing airline should be displayed to travelers as the primary airline.
- **operatingAirline** (*String*) – two character IATA code of the airline operating this specific segment
- **marketingAirlineName** (*String*) – The name of the airline that publishes and markets the flight booked under its own airline designator and flight number
- **operatingAirlineName** (*String*) – The airline operating this specific segment
- **flightNumber** (*String*) - the flight number for the specific flight, normally displayed as XXYYYY, where XX is the marketing airline's code, and YYYY is this number

Stop

A stop is either the departure, or the arrival part of a segment.

JSON Parameters

- **dateTime** (*String*) – time of the stop (in ISO format)
- **airport** (*Airport*) – location of the stop
  - **terminal** – the relevant terminal of the airport specified below (this will be `null` is the airport has only one terminal)

    - **name** (*String*) – official airport name of the specific stop
    - **code** – the three letter IATA code of the airport the stop is at
  - **city** (*City*) – location city name of the stop
    - **name** (*String*) – official city name of the specific stop
    - **code** – the three letter IATA code of the city the stop belongs to

Comfort score

Comfort score is a variable that indicates how comfortable each combination option is. It is based on different aspects of the flight, e.g.:

- Total time elapsed from first departure to last arrival
- Number of flight segments (*Segment [ ]*)
- Cabin type
- Passenger capacity of aircrafts
- Red-eye flight status, meaning flight leaves or departs at an inconvenient time
- The time elapsed between flight segments

Check-in charges

JSON Parameters

- **type** (*String*) - Usually "Airport Check-in" or "Web Check-in"
- **currency** (*String*)
- **amount** (*Float*)

Response Codes

- **404 'No flights available'**
- **404 'No flight found for return leg'**
- **404 'Search does not include a required country'** - It is possible to set rules to disallow search queries that don't include a specific country in the itinerary. If a search request doesn't match the set filter, this is returned
- **500 'external provider rejected the request - please try again'**: This is the generic error sent when we receive an unknown error as response from the provider

Examples

Request

JSON:

```
{
  "fromLocation": "BUD",
  "toLocation": "LON",
  "departureDate": "2014-05-15T00:00:00",
  "returnDate": "2014-05-20T00:00:00",
  "persons":[
    {
      "passengerType":"ADT",
      "quantity": 2
    },
    {
      "passengerType":"CHD",
      "quantity": 1
    }
  ],
  "flexible_date_search_reference": {
    "cookie": "1234567asdf",
    "extra_days": 2
  }
}
```

Response

JSON:

```
{
  "flightResultSet": [
    {
      "breakdown": [
        {
          "passengerFare": {
            "fare": 52.8627,
            "tax": 21.1229,
            "ticketDesignators": [],
            "type": "ADT",
            "quantity": 1,
            "fare_in_preferred_currencies": [
              {
                "currency":GBP",
                "fare": 72,
                "tax": 21.1229,
              },
              {
                "currency": "USD",
                "fare": 66,
                "tax": 21.1229,
              }
            ],
          }
        }
      ],
      "currency": "EUR",
      "total_fare": 57.8627,
      "ticketing_fee": 5,
      "total_fare_in_preferred_currencies": [
        {
```

```
        "currency":GBP",
        "total_fare": 72,
      },
      {
        "currency": "USD",
        "total_fare": 66,
      }
    ],
    "ticketing_fee_in_preferred_currencies": [
      {
        "currency":GBP",
        "ticketing_fee": 3.66,
      },
      {
        "currency": "USD",
        "ticketing_fee": 5.74,
      }
    ],
    "price_charged_by_provider": {
      "currency":GBP",
       "ticketing_fee": 3.66,
       "total_fare": 72,
    },
    "combinations": [
      {
        "bookingId": "15_0_0",
        "comfortScore": 47,
        "firstLeg": {
          "elapsedTime": "0230",
          "flightSegments": [
            {
              "operatingAirlineName": "British Airways",
              "marketingAirlineName": "British Airways",
              "aircraft": "Airbus Industries A320",
              "arrival": {
                "airport": {
                  "name": "Stansted",
                  "terminal": null,
                  "code": "STN"
                },
                "city": {
                  "code": "LON",
                  "name": "London"
                },
                "dateTime": "2014-06-05T23:00:00"
              },
              "marketingAirline": "BA",
              "operatingAirline": "FR",
              "departure": {
                "airport": {
                  "terminal": null,
                  "code": "BUD"
                  "name": "Liszt Ferenc Intl",
                },
                "city": {
                  "code": "BUD",
                  "name": "Budapest"
                },
                  "dateTime": "2014-06-05T21:30:00"
                },
                "flightNumber": "867",
                "availableSeats": 9,
                "cabin": "economy",
                "class": "Y",
              }
            ]
          },
          "serviceFeeAmount": 5.0,
          "comfortScore": 50,
          "service_fee_in_preferred_currencies": [
            {
              "currency":GBP",
              "service_fee": 3.66,
            },
            {
              "currency": "USD",
              "service_fee": 5.74,
            }
          ],
        }
      ]
    }
  ]
}
```

Flexible Date Search

Returns the cheapest flight option for all the possible combinations of the departure and arrival dates +/- the number of `extraDays`.

Warning: To proceed with the flight workflow after a flexible date search, a regular search request must be sent with the parameters of the chosen option. It is not possible to make a booking based on booking IDs returned in the flexible date search response! Please include the **flexible_date_search_reference** parameters in the regular search sent after a flexible date search.

JSON Parameters

- **fromLocation** (*String*) – departure location, given as IATA code
- **toLocation** (*String*) – destination, given as IATA code
- **departureDate** (*String*) – date of departure
- **returnDate** (*String*) – date of return
- **id** (*String*) – unique identifier of the result

Examples

Request

JSON:

```
{
  "fromLocation": "BUD",
  "toLocation": "LON",
  "departureDate": "2014-05-15T00:00:00",
  "returnDate": "2014-05-20T00:00:00",
  "persons":[
    {
      "passengerType":"ADT",
      "quantity": 2
    },
    {
      "passengerType":"CHD",
      "quantity": 1
    }
  ],
  "extraDays": 3,
}
```

Response

JSON:

```
{
  "flightResultSet": [
    {
      "flightResult": {
        "_comment": "same as in regular search response"
      },
      "fromLocation": "BUD",
      "toLocation": "LON",
      "departureDate": "2015-04-29T00:00:00Z",
      "returnDate": "2015-05-06T00:00:00Z",
      "id": "0648ae1d-3b48-4a88-b317-a5ca65fd2d67",
    }
  ]
}
```

Details

Request

`GET /flights/:booking_id`

booking_id is the booking ID of the *Combination* to get the details of

Response Body

JSON Parameters

- **flightDetails** (*FlightDetails*) – root container

FlightDetails

Warning: While the `price` field contains the ticket's final price, baggages are not included in that, as the user may be able to choose from different baggage tiers. It is the travel site's responsibility to add the cost of the passenger's baggages themselves as an extra cost.

Note: Providers return prices in the travel site's preferred currency automatically. In the rare case that they might fail to do so, the Allmyles API will convert the prices to the flight fare's currency automatically, based on the provider's currency conversion data.

JSON Parameters

- **rulesLink** (*String*) – link to the airline's rules page (hosted on the airline's website)
- **baggageTiers** (*BaggageTier [ ]*) – contains the different options the passenger has for bringing baggages along. The book request will need to contain the ID of one of these objects in the baggage field.
- **carryOnBaggageTiers** (*CarryOnBaggageTier*) – contains the different options of cabin baggages. The book request will need to contain the ID of one of these objects in the carry-on baggage field.
- **fields** (*Form Fields*) – contains field validation data
- **price** (*Price*) – contains the final price of the ticket (including the credit card surcharge, but not the baggages)
- **result** (*FlightResult*) – contains an exact copy of the result from the *Search* call's response
- **options** (*FlightOptions*) – contains whether certain options are enabled for this flight
- **surcharge** (*Price*) – contains the credit card surcharge for this flight
- **price_in_preferred_currencies** (*Price [ ]*) – contains the final price of the ticket converted to the client's preferred currencies
- **surcharge_in_preferred_currencies** (*Price [ ]*) – contains the credit card surcharge for this flight converted to the client's preferred currencies

BaggageTier

These objects define the passenger's options for taking baggages on the flight. Each passenger can choose one of these for themselves.

Note: Keep in mind that while the tier ID's value may seem closely related to the other fields, it's not guaranteed to contain any semantic meaning at all.

JSON Parameters

- **tier** (*String*) – the ID for this baggage tier (this is used to refer to it when booking)
- **price** (*Price*) – contains the price of the baggage tier

- **max_weights** (*Float [ ]*) – the maximum weight of each piece of baggage a passenger can take in this tier in kg, can be an empty array if there's no limit. Having multiple items in this array means that for the specified price, the passenger can check in as many baggages as there are items in the array. Can be an empty list if data is present in the *total* field.
- **total** – Some airlines don't limit the weights of each bag, only the total weight of all the bags, and the number of bags.
  - **weight** (*Float*) – maximum summed weight of all the bags the passenger can take
  - **number_of_bags** (*Int*) – number of bags that the passenger can take
- **price_in_preferred_currencies** (*Price [ ]*) – contains the price of the baggage tier converted to the client's preferred currencies

CarryOnBaggageTier

These objects define the passenger's options for taking cabin baggages on the flight. Each passenger can choose one of these for themselves.

JSON Parameters

- **tier** (*String*) – the ID for this baggage tier (this is used to refer to it when booking)
- **price** (*Price*) – contains the price of the baggage tier
- **description** (*String*) – A basic description of the carry-on baggage's size, e.g. *Small cabin bag*. Exact dimensions should be checked on the airline's website.
- **price_in_preferred_currencies** (*Price [ ]*) – contains the price of the baggage tier converted to the client's preferred currencies

Form Fields

Form fields define criteria for field validation, making it easy to generate HTML form elements.

JSON Parameters

- **passengers** (*Form Field [ ]*) – contains validation data for Passenger fields
- **contactInfo** (*Form Field [ ]*) – contains validation data for Contact Info fields
- **billingInfo** (*Form Field [ ]*) – contains validation data for Billing Info fields

Form Field

JSON Parameters for `select` fields

- **tag** (*String*) – HTML tag type, in this case `select`
- **options** (*String [ ]*) – value options of the field
- **attributes** (*Attributes [ ]*) – attributes of the field

JSON Parameters for `input` fields

- **tag** (*String*) – HTML tag type, in this case `input`
- **attributes** (*Attributes [ ]*) – attributes of the field

Attributes

JSON Parameters

- **name** (*String*) – one of *Field Names*
- **data-label** (*String*) – user friendly field label
- **type** (*String*) – type of input data (`†ext` or `email`)
- **maxLength** (*Float*)
- **required** (*String*) – if present, field is required
- **pattern** (*String*) – regex pattern of valid data

Field Names

Passenger

- namePrefix
- firstName
- middleName
- lastName
- gender
- birthDate
- document/type
- document/id
- document/issueCountry
- document/dateOfExpiry

Contact and Billing Info

- name
- email
- address/addressLine1
- address/addressLine2
- address/addressLine3
- address/cityName
- address/zipCode
- address/countryCode
- phone/countryCode
- phone/areaCode
- phone/phoneNumber

Price

JSON Parameters

- **amount** (*Float*) – the amount of money in the currency below
- **currency** (*String*) – the currency of the amount specified, can be null when the amount is zero

FlightOptions

{optionName} below refers to the following names:

- seatSelectionAvailable
- travelfusionPrepayAvailable

JSON Parameters

- **{optionName}** (*Boolean*) – whether the option is enabled or not

Response Codes

- **404 'search first'**
- **412 'a request is already being processed'**: This error comes up even when the other request is asynchronous (i.e. when we are still processing a search request). The response for async requests does not need to be retrieved for this error to clear, just wait a few seconds.
- **412 'request is not for the latest search'**: One case where this error is returned is when a customer is using multiple tabs and trying to select a flight from an old result list.

Examples

Response

JSON:

```
{
  "flightDetails": {
    "rulesLink": null,
    "baggageTiers": [
        {
            "tier": "0",
            "price": {
                "currency": null,
                "amount": 0.0
            },
            "max_weights": [],
            'total': {
                'weight': None,
                'number_of_bags': None,
            },
            "price_in_preferred_currencies": [
              {
                "currency":GBP",
                "amount": 0.0
              },
              {
                "currency": "USD",
                "amount": 0.0
              }
            ],
        },
        {
            "tier": "1",
            "price": {
                "currency": "HUF",
                "amount": 15427.0
            },
            "max_weights": [15.0],
            'total': {
                'weight': None,
                'number_of_bags': None,
            },
            "price_in_preferred_currencies": [
              {
                "currency":GBP",
                "amount": 10.0
              },
              {
                "currency": "USD",
                "amount": 12.0
              }
            ],
        },
        {
            "tier": "2",
            "price": {
                "currency": "HUF",
                "amount": 37024.8
            },
            "max_weights": [],
            'total': {
                'weight': 45,
                'number_of_bags': 2,
            },
            "price_in_preferred_currencies": [
              {
                "currency":GBP",
                "amount": 20.0
              },
              {
                "currency": "USD",
                "amount": 22.0
              }
            ],
        }
    ],
    "carryOnBaggageTiers": [
        {
            "tier": "1",
            "price": {
                "currency": "null",
                "amount": 0.0
            },
            "description": "Small cabin bag",
        },
        {
            "tier": "2",
            "price": {
                "currency": "HUF",
                "amount": 8000.0
            },
            "description": "Large cabin bag",
        },
        "price_in_preferred_currencies": [
          {
            "currency":GBP",
            "amount": 20.0
          },
          {
            "currency": "USD",
            "amount": 22.0
          }
        ],
    ],
    "fields": {
      "passengers": [
        {
          "tag": "select",
          "options": ["Mr", "Ms", "Mrs"],
          "attributes": [
            {
              "key": "required",
              "value": "required"
            },
            {
              "key": "name",
              "value": "persons/0/namePrefix"
            },
            {
              "key": "data-label",
              "value": "Name Prefix"
            }
          ],
        },
      ],
      "contact_info": [
        {
          "tag": "input",
          "attributes": [
            {
              "key": "maxLength",
              "value": "30"
            },
            {
              "key": "type",
              "value": "text"
            },
            {
             "key": "name",
              "value": "billingInfo/name"
            },
            {
              "key": "data-label",
              "value": "Name"
            }
          ],
        },
      ],
      "billing_info": [
        {
          "_comment": "trimmed in example for brevity's sake"
        },
      ]
    },
    "price": {
      "currency": "EUR",
      "amount": 4464.46
    },
    "result": {
      "_comment": "trimmed in example for brevity's sake"
    },
    "options": {
      "seatSelectionAvailable": false,
      "travelfusionPrepayAvailable": false
    },
    "surcharge": {
      "currency": "EUR",
      "amount": 5.0
      "card_type": "CA",
    },
    "price_in_preferred_currencies": [
      {
        "currency":GBP",
        "amount": 3269
      },
      {
        "currency": "USD",
        "amount": 5162
      }
    ],
    "surcharge_in_preferred_currencies": [
      {
        "currency":GBP",
        "amount": 5.0
        "card_type": "CA",
      },
      {
        "currency": "USD",
        "amount": 5.0
        "card_type": "CA",
      }
    ],
  }
}
```

Booking

Note: When booking LCC flights, there are two possible scenarios. By *default*, the Allmyles API does not send the book request to the external provider until the ticketing call arrives, so there's no response—an HTTP 204 No Content status code is returned. If you have chosen *alternative* providers (you have to contact the Allmyles support about this first), the booking flow of LCC flights is very similar to that of traditional flights. In this case the book response differs just a bit from the traditional book response - please refer to the book response specifications for detailed information.

Request

`POST /books`

JSON Parameters

- **bookBasket** (*String*) – the booking ID of the *Combination* to book
- **billingInfo** (*Billing*) – billing info for ticket creation
- **contactInfo** (*Contact*) – contact info for ticket creation
- **persons** (*Passenger [ ]*) – the list of passengers
- **userData** (*User Data*) – information about the end user
- **tenantReferenceId** (*String*) – *(optional)* ID of the booking on the tenant's side - can be useful for debugging purposes

Contact

JSON Parameters

- **address** (*Address*) – address of the the contact person
- **email** (*String*) – email of the contact person
- **firstName** (*String*)
- **middleName** (*String*) – *(optional)* submission of this parameter is mandatory if the person in question has a middle name
- **lastName** (*String*)
- **phone** (*Phone*) – phone number of the contact person

Billing

JSON Parameters

- **address** (*Address*) – address of the entity in question
- **email** (*String*) – email of the entity in question
- **firstName** (*String*) – name of the entity in question, if the entity is an organization this is the only name field that is required

- **middleName** (*String*) – *(optional)* submission of this parameter is mandatory if the person in question has a middle name and must not be sent in if the entity in question is an organization
- **lastName** (*String*) – *(optional)* submission of this parameter is mandatory if the entity in question is a person and it must not be included if the entity is an organization
- **phone** (*Phone*) – phone number of the entity in question

Address

JSON Parameters

- **addressLine1** (*String*)
- **addressLine2** (*String*) – *(optional)*
- **addressLine3** (*String*) – *(optional)*
- **cityName** (*String*)
- **zipCode** (*String*)
- **countryCode** (*String*) – the two letter code of the country

Phone

JSON Parameters

- **countryCode** (*String*)
- **areaCode** (*String*)
- **phoneNumber** (*String*) - Must be at least 7 characters long.

Passenger

JSON Parameters

- **birthDate** (*String*) – format is `YYYY-MM-DD`
- **document** (*Document*) – data about the identifying document the passenger wishes to travel with
- **email** (*String*)
- **namePrefix** (*String*) – one of `Mr`, `Ms`, or `Mrs`
- **firstName** (*String*)
- **middleName** (*String*) – *(optional)*
- **lastName** (*String*)
- **gender** (*String*) – one of `MALE` or `FEMALE`
- **passengerTypeCode** (*String*) – one of *PassengerTypes*
- **baggage** (*String*) – one of the tier IDs returned in the flight details response
- **carryOnBaggage** (*String*) – one of the tier IDs returned in the flight details response

Document

JSON Parameters

- **id** (*String*) – document's ID number
- **dateOfExpiry** (*String*) – format is YYYY-MM-DD
- **issueCountry** (*String*) – two letter code of issuing country
- **type** (*String*) – one of DocumentTypes

Response Body

Note: Again: **by default, there's no response body for LCC book requests!** An HTTP 204 No Content status code confirms that Allmyles saved the sent data for later use.

Warning: If you have chosen alternative providers - that means there IS a book response for LCC flights, **this is the response that contains the exact final price** that should be shown to the traveler. This price contains the baggage and hand luggage surcharges, if applicable.

Warning: The format of *Contact* and *FlightResult* objects contained within this response might slightly differ from what's described in this documentation as requested. This will be fixed in a later version.

JSON Parameters

- **price** (*Price*) – final price updated with baggage surcharges. **Only in alternative LCC book response!**
- **pnr** (*String*) – the PNR locator which identifies this booking
- **lastTicketingDate** (*String*) – the timestamp of when it's last possible to create a ticket for the booking, in ISO format
- **bookingReferenceId** (*String*) – the ID of the workflow at Allmyles; this is not currently required anywhere later, but can be useful for debugging
- **contactInfo** (*Contact*) – contains a copy of the data received in the *Booking* call
- **flightData** (*FlightResult*) – contains a copy of the result from the *Search* call's response

Response Codes

- **303 'Unable to book this flight - please select a different bookingId'**: This error is returned when the external provider encounters a problem such as a discrepancy between actual flight data and what they returned from their cache before. This happens very rarely, or never in production.
- **404 'search first'**
- **412 'a request is already being processed'**: This error comes up even when the other request is asynchronous (i.e. when we are still processing a search request). The response for async requests does not need to be retrieved for this error to clear, just wait a few seconds.

- **412 'Already booked.'**: This denotes that either us or the external provider has detected a possible duplicate booking, and has broken the flow to avoid dupe payments.
- **412 'already booked'**: This is technically the same as the error above, but is encountered at a different point in the flow. The error messages are only temporarily not the same for these two errors.
- **412 'request is not for the latest search'**
- **500 'could not book flight'**: This is the generic error returned when we encounter an unknown/empty response from the external provider
- **504 'external gateway timed out - book request might very well have been successful!'**: The booking might, or might not have been completed in this case. The flow should be stopped, and the customer should be contacted to complete the booking.
- **504 'Could not retrieve virtual credit card, flight not booked. An IRN should be sent to payment provider now.'**

Examples

Request

JSON:

```
{
  "bookBasket": ["1_0_0"],
  "billingInfo": {
    "address": {
      "addressLine1": "Váci út 13-14",
      "cityName": "Budapest",
      "countryCode": "HU",
      "zipCode": "1234"
    },
    "email": "ccc@gmail.com",
    "name": "Kovacs Gyula",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "phoneNumber": "1234567"
    }
  },
  "contactInfo": {
    "address": {
      "addressLine1": "Váci út 13-14",
      "cityName": "Budapest",
      "countryCode": "HU",
      "zipCode": "1234"
    },
    "email": "bbb@gmail.com",
    "name": "Kovacs Lajos",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "phoneNumber": "1234567"
    }
  },
  "persons": [
    {
      "baggage": "0",
      "carryOnBaggage": "1",
      "birthDate": "1974-04-03",
      "document": {
        "dateOfExpiry": "2016-09-03",
        "id": "12345678",
        "issueCountry": "HU",
        "type": "Passport"
      },
      "email": "aaa@gmail.com",
      "firstName": "Janos",
      "gender": "MALE",
      "lastName": "Kovacs",
      "namePrefix": "Mr",
      "passengerTypeCode": "ADT"
    }
  ]
}
```

Response

JSON:

```
{
  "bookingReferenceId": "req-cfd7963b187a4fe99702c0373c89cb16",
  "contactInfo": {
    "address": {
      "city": "Budapest",
      "countryCode": "HU",
      "line1": "Madach ut 13-14",
      "line2": null,
      "line3": null
    },
    "email": "testy@gmail.com",
    "name": "Kovacs Lajos",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "number": "1234567"
    }
  },
  "flightData": {
    "_comment": "trimmed in example for brevity's sake"
  },
  "lastTicketingDate": "2014-05-16T23:59:59Z",
  "pnr": "6YESST"
}
```

Payment

This is where Allmyles gets the payment data.

Allmyles is a payment platform agnostic solution. When we receive a transaction ID that points to a successful payment by the passenger, we essentially take that money from any Payment Service Provider (PSP), and forward it to the provider to buy a ticket in the *Ticketing* step.

Request

POST /payment

JSON Parameters

- **paymentId** (*String*) – the transaction ID identifying the successful transaction at your PSP
- **basket** (*String[ ]*) – the booking IDs the payment is for

Response Body

N/A:

Returns an HTTP 204 No Content status code if successful.

Response Codes

- **412 ‘a request is already being processed’**: This error comes up even when the other request is asynchronous (i.e. when we are still processing a search request). The response for async requests does not need to be retrieved for this error to clear, just wait a few seconds.
- **412 ‘book request should have been received’**

Examples

Request

JSON:

```
{
  "paymentId": "12345678",
  "basket": ["2_1_0"]
}
```

Ticketing

Two important notes:

1. Call this only when the passenger’s payment completely went through! (That is, after the payment provider’s IPN has arrived, confirming that the transaction did not get caught by the fraud protection filter.)
2. After this call has been made **do not issue refunds** unless the Allmyles API explicitly tells you to. It’s way better to just correct ticketing errors manually than to fire automatic refunds even if the ticket purchase might already be locked in for some reason.

Request

GET /tickets/:booking_id

booking_id is the booking ID of the *Combination* to create a ticket for

Response Body

By default, this is just an abstraction for the book call when buying an LCC ticket (there's no separate book and ticketing calls for those flights). This means the response differs greatly depending on whether the flight is traditional or LCC booked through the *default* providers.

If you have chosen *alternative* providers (you would have to contact the Allmyles support about this first), there **is** a separate book response for LCC flights, but the ticket response is the same as described below.

JSON Parameters for traditional flights

- **tickets** (*Ticket [ ]*) – the purchased tickets
  - **passenger** (*String*) – the name of the passenger the ticket was purchased for
  - **passenger_type** (*String*) – one of *PassengerTypes*
  - **ticket** (*String*) – the ticket number which allows the passenger to actually board the plane
  - **price** (*TicketPrice*)
    - **currency** (*String*)
    - **total_fare** (*Float*) – The total amount of money the passenger paid for his ticket, including tax.
    - **tax** (*Float*) – The total amount of tax the passenger had to pay for this ticket.
  - **baggage**
    - **quantity** (*Int*) – The maximum quantity of baggage the passenger can bring along
    - **unit** (*String*) – Units of measurement
  - **price_in_preferred_currencies** (*TicketPrice [ ]*) – the ticket price converted to the client's preferred currencies - **currency** (*String*) - **total_fare** (*Float*) - **tax** (*Float*)
- **flightData** (*FlightResult*) – contains a copy of the result from the *Search* call's response
- **contactInfo** (*Contact*) – contains a copy of the data received in the *Booking* call

JSON Parameters for LCC flights

- **ticket** (*String*) – the ticket number (LCC PNR) for this booking
- **pnr** (*String*) – the PNR locator which identifies this booking
- **bookingReferenceId** (*String*) – the ID of the workflow at Allmyles; this is not currently required anywhere later, but can be useful for debugging
- **contactInfo** (*Contact*) – contains a copy of the data received in the *Booking* call
- **flightData** (*FlightResult*) – contains a copy of the result from the *Search* call's response
- **baggageTiers** (*BaggageTier [ ]*) – the baggage tier option the passenger has chosen
- **carryOnBaggageTiers** (*CarryOnBaggageTier [ ]*) – the carry-on baggage tier option the passenger has chosen

Response Codes

In case of errors (referring to response code 202 and 5xx), the client is expected to either have a correct the ticketing manually, or send periodic *Ticketing Status* requests until a definitive response is given (one of the following statuses: 'successful', 'failed', or 'unknown'.) This should take no longer than 40 minutes. Tickets with an unknown status still require manual intervention.

- **202 'Warning: e-ticket could not be issued due to technical difficulties. Please contact youragent.'**: When this error occurs, the actual ticket is purchased, but an unknown error happens later on in the flow.
- **412 'a request is already being processed'**: This error comes up even when the other request is asynchronous (i.e. when we are still processing a search request). The response for async requests does not need to be retrieved for this error to clear, just wait a few seconds.
- **412 'no payment data given'**
- **412 'book request should have been received'**
- **412 'book response should have been received'**
- **500 'booking failed, cannot create ticket'**: This error is returned if the book response we last received from the provider contained an error.
- **503 'error while creating ticket - please try again later'**: This is the generic error we return when receiving an unknown response for the ticket request. No refund should be sent without manually checking if the ticket has been issued first.
- **504 'ticket creation timed out - but could very well have been successful!'**: Almost the same as above, refunds are definitely not safe in this case.

Examples

Response

JSON for traditional flights:

```
"body": {
  "tickets": [
    {
      "passenger": "Mr Janos kovcas",
      "passenger_type": "ADT",
      "ticket": "125-4838843038",
      "price": {
        "currency": "HUF",
        "total_fare": 26000.0,
        "tax": 17800.0
      }
      "baggage": {
        "quantity": 1,
        "unit": "PC",
      },
      "price_in_preferred_currencies": [
      {
        "currency":GBP",
        "total_fare": 60.48,
        "tax": 41.41
      },
      {
        "currency": "USD",
        "total_fare": 94.84,
        "tax": 64.93
      }
    ],
    },
    {
      "passenger": "Mr Janos kascvo",
      "passenger_type": "ADT",
      "ticket": "125-4838843039",
      "price": {
        "currency": "HUF",
        "total_fare": 26000.0,
        "tax": 17800.0
      }
      "baggage": {
        "quantity": 1,
        "unit": "PC",
      },
      "price_in_preferred_currencies": [
      {
        "currency":GBP",
        "total_fare": 60.48,
        "tax": 41.41
      },
      {
        "currency": "USD",
        "total_fare": 94.84,
        "tax": 64.93
      }
    ],
    }
  ],
  "flightData": {
    "_comment": "trimmed in example for brevity's sake"
  },
  "contactInfo": {
    "address": {
      "city": "Budapest",
      "countryCode": "HU",
      "line1": "Madach ut 13-14",
      "line2": null,
      "line3": null
    },
    "email": "testytesty@gmail.com",
    "name": "Kovacs Lajos",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "number": "1234567"
    }
  }
}
```

JSON for LCC flights:

```
{
  "bookingReferenceId": "req-d65c00dc43ba4ad798e5478803575aab",
  "contactInfo": {
    "address": {
      "city": "Budapest",
      "countryCode": "HU",
      "line1": "Madach ut 13-14",
      "line2": null,
      "line3": null
    },
    "email": "testytesty@gmail.com",
    "name": "Kovacs Lajos",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "number": "1234567"
    }
  },
  "flightData": {
    "_comment": "trimmed in example for brevity's sake"
  },
  "lastTicketingDate": null,
  "pnr": "6YE2LM",
  "ticket": "0XN4GTO",
  "baggageTiers": {
    "tier": "2",
    "max_weights": [15.0, 20.0],
    "price": {
      "amount": 37024.8,
      "currency": HUF
    },
    "price_in_preferred_currencies": [
      {
        "currency":GBP",
        "amount": 10.0
      },
      {
        "currency": "USD",
        "amount": 12.0
      }
    ],
  },
  "carryOnBaggageTiers": {
    "tier": "2",
    "description": "Large cabin bag",
    "price": {
      "amount": 8000.0,
      "currency": HUF
    },
    "price_in_preferred_currencies": [
      {
        "currency":GBP",
        "amount": 10.0
      },
      {
        "currency": "USD",
        "amount": 12.0
      }
    ],
  }
}
```

Ticketing Status

This call enables checking the result of a ticketing request. This is useful when it's unclear whether the ticketing process went through, due to a failure at external providers, in Allmyles' systems, on the client's server, or anywhere in between. The request will identify the correct workflow based on the cookie header's contents, which must match whatever was sent in the ticket request.

If you're using **alternative providers** and an LCC booking returns with the status **pending** or **unknown**, keep in mind that the ticket could still be created successfully in the next 72 hours. You should keep making periodic *Ticketing Status* requests at a reduced rated until a **successful** or **failed** status is returned or the 72-hour period is over.

The periodic checks should be made at most once every 5 minutes.

Available statuses

- **inactive**: this is the status returned when the ticketing process has not been initiated yet, i.e. before a *Ticketing* request is sent
- **pending**: the ticket creation is still in progress
- **successful**: the ticket has been successfully created. PNR data will be passed alongside this status, including the ticket number(s).
- **failed**: the ticket creation failed, and the fare can be refunded (do note that this is the only status in which refunds can be automatically made)
- **unknown**: it is not possible to programmatically determine the outcome of the request. The passenger's money should be held until a human identifies the issue and determines whether the ticket exists or not.

Request

`GET /tickets/:booking_id/status`
booking_id is the booking ID of the *Combination* whose ticket's status we are interested in

Response Body

JSON Parameters

- **status** (*String*) – one of the statuses
- **pnr** (*PNR*) – the pnr object that a *Get Booking* request would return about the flight — this includes the ticket number(s) as well

Examples

Response

JSON for traditional flights:

```
{
    "status": "successful",
    "pnr": {
        "deleted": false,
        "id": "3L4TMN",
        "passengers": [
            {
                "birth_date": "1974-01-01",
                "email": "test@example.com",
                "name": "SMFDETH HYRASESN/MR",
                "traditional_ticket": "125-5249156160",
                "type": "ADT"
            },
            {
                "birth_date": "1974-01-01",
                "email": null,
                "name": "SMIATTASDH OSAJOEONHTDNHO/MR",
                "traditional_ticket": "125-5249156161",
                "type": "ADT"
            }
        ]
    }
}
```

Rules

This call returns the terms and conditions of the flight in question, or a link to them if the raw text isn't available (in case of LCC flights).

Request

`GET /flights/:booking_id/rules`
booking_id is the booking ID of the *Combination* to get the rules of

Response Body

JSON Parameters

- **rulesResultSet** (*RulesResultSet*) – root container
  - **rules** (*Rule [ ]*) – contains the flight rule texts, is returned only for traditional flights
  - **link** (*String*) – contains a link to the airline's rules page, is returned only for LCC flights

Rule

JSON Parameters

- **code** (*String*) - the machine readable identifier code for the given section in the rules
- **title** (*String*) - the human readable section title for the block
- **text** (*String*) - the section's raw rule text body

Response Codes

- **404 'search first'**

- **412 ‘a request is already being processed’**: This error comes up even when the other request is asynchronous (i.e. when we are still processing a search request). The response for async requests does not need to be retrieved for this error to clear, just wait a few seconds.
- **409 ‘request is not for the latest search’**

Examples

Response

JSON (for LCC):

```
{
  "rulesResultSet": {
    "link": "https://www.ryanair.com/en/terms-and-conditions"
  }
}
```

JSON (for traditional):

```
{
  "rulesResultSet": {
    "rules": [
      {
        "code": "OD",
        "text": "NONE UNLESS OTHERWISE SPECIFIED",
        "title": "OTHER DISCOUNTS"
      },
      {
        "code": "SO",
        "text": "STOPOVERS NOT PERMITTED ON THE FARE COMPONENT.",
        "title": "STOPOVERS"
      },
    ]
  }
}
```

Get Booking

This call returns the details of a booking identified by a PNR locator. This makes it possible to re-open an expired session and send a ticketing request based on the PNR locator after the initial session is closed.

Request

`GET /books/:pnr_locator`

pnr_locator is a unique identifier of the booking, received at the book response.

Response Body

JSON Parameters

- **pnr** (*pnr*) – root container
  - **passengers** (*Passenger [ ]*) – the list of passengers

    * **birth_date** (*String*) – format is `YYYY-MM-DD`
    * **traditional_ticket** (*String*) - the ticket number which allows the passenger to actually board the plane (or `null` if flight is LCC)
    * **type** (*String*) – one of *PassengerTypes*
    * **email** (*String*)
    * **name** (*String*) – the name of the passenger the booking was made for
  - **id** (*String*) – the PNR locator which identifies the booking
  - **lcc_ticket** (*String*) – the ticket number which allows the passenger to actually board the plane (or `null` if flight is traditional)

Response Codes

- **404 'PNR not found'**
- **403 'PNR belongs to another auth token'**

Examples

Response

JSON:

```
{
  "pnr": {
    "passengers": [
      {
        "birth_date": "1974-01-01",
        "traditional_ticket": "123-5249155974",
        "type": "ADT",
        "email": "test@gmail.com",
        "name": "KOVACS JANOS/MR"
      }
    ],
    "id": "3KWQUK",
    "lcc_ticket": null
  }
}
```

Cancel Booking

This call cancels the booking identified in the request. Bookings can only be cancelled before a ticket is created. **Only bookings of traditional flights can be cancelled!**

Request

`DELETE /books/:pnr_locator`
pnr_locator is a unique identifier of the booking, received at the book response.

Response Body

N/A:

Returns an HTTP 204 No Content status code if successful.

Response Codes

- **403 ‘PNR belongs to another auth token’**
- **404 ‘PNR not found’**
- **409 ‘Booking already cancelled.’**
- **409 ‘Booked flights can only be cancelled before ticket is created.’**

Hotels

Summary

The hotel booking workflow consists of four or five mandatory steps.

1. *Search*
2. *Details*
3. *Room Details*
4. *Payment* (payment is not required for some hotels)
5. *Booking*

Search

Request

```
POST /hotels
```

Searches for hotels that match provided criteria.

JSON Parameters

- **cityCode** (*String*) – city to search for a hotel in, given as its IATA code
- **arrivalDate** (*String*) – date when the occupants arrive, in ISO format, not including a time code (ex. 2014-12-24)
- **leaveDate** (*String*) – date when the occupants arrive, in ISO format, not including a time code (ex. 2014-12-26)
- **occupancy** (*Integer*) – number of people wanting to book a room

Response Body

JSON Parameters

- **hotelResultSet** (*Hotel [ ]*) – root container

Hotel

JSON Parameters

- **hotel_id** (*String*) –
- **hotel_name** (*String*) –
- **chain_name** (*String*) –
- **amenities** (*Amenities*) – An associative array mapping each amenity listed below to a boolean value based on whether the hotel has given amenity. List of keys: 'restaurant', 'bar', 'laundry', 'room_service', 'safe_deposit_box', 'parking', 'swimming', 'internet', 'gym', 'air_conditioning', 'business_center', 'meeting_rooms', 'spa', 'pets_allowed'
- **latitude** (*Float*) – The latitude component of the coordinates of the hotel
- **longitude** (*Float*) – The latitude component of the coordinates of the hotel
- max_rate (*Price*)
  - amount (*Float*) –
  - currency (*String*) –
- min_rate (*Price*) –
  - amount (*Float*) –
  - currency (*String*) –
- stars (*Integer*) – The amount of stars the hotel has been awarded
- thumbnail (*String*) – Link to a small image representing the hotel

Response Codes

- **404 'No hotels available'**

Examples

Request

JSON:

```
{
  "cityCode": "LON",
  "occupancy": 1,
  "arrivalDate": "2014-09-29",
  "leaveDate": "2014-09-30"
}
```

Response

JSON:

```
{
  "hotelResultSet": [
    {
      "amenities": {
        "air_conditioning": false,
        "bar": true,
        "business_center": false,
        "gym": false,
        "internet": false,
        "laundry": false,
        "meeting_rooms": true,
        "parking": true,
        "restaurant": false,
        "room_service": false,
        "safe_deposit_box": true,
        "spa": true,
        "swimming": false
      },
      "chain_name": "ACCOR HOTELS",
      "hotel_id": "12_2",
      "hotel_name": "MERCURE PARIS PLACE ITALIE 3*",
      "latitude": 48.8303,
      "longitude": 2.35283,
      "max_rate": {
        "amount": 21951.12,
        "currency": "HUF"
      },
      "min_rate": {
        "amount": 18024.3,
        "currency": "HUF"
      },
      "stars": 3,
      "thumbnail": "https://static.allmyles.com/hotels/e4ba87c0/12_2.jpg"
    }
  ]
}
```

Details

Request

GET /hotels/:hotel_id

hotel_id is the ID of the *Hotel* to get the details of

Response Body

JSON Parameters

- **hotel_details** (*HotelDetails*) – root container

HotelDetails

JSON Parameters

- **chain_code** (*String*)
- **chain_name** (*String*)
- **hotel_code** (*String*)
- **hotel_name** (*String*)
- **location** (*HotelLocation*) – contains info about the hotel's location.
- **points_of_interest** (POI *[ ]*) – contains a list of notable locations around the hotel
- **description** (*String*) – A short text describing the hotel
- **contact_info** (*HotelContactInfo*) –
  - **phone_numbers** (*String [ ]*)
  - **email** (*String*)
  - **website** (*String*)
- **price** (*PriceRange*) – contains the lowest and highest rates available for a room at this hotel
  - **minimum** (*Float*) – Rate of the cheapest room at the hotel
  - **maximum** (*Float*) – Rate of the most expensive room at the hotel
  - **currency** (*String*)
- **thumbnail** (*String*) – Contains a URL pointing to a small image of the hotel
- **photos** (*String [ ]*) – Contains an array of URLs pointing to a larger photos of the hotel
- **amenities** (*Amenities*) – Contains an associative array, mapping each amenity listed below to a boolean value based on whether the hotel has given amenity. List of keys: 'restaurant', 'bar', 'laundry', 'room_service', 'safe_deposit_box', 'parking', 'swimming', 'internet', 'gym', 'air_conditioning', 'business_center', 'meeting_rooms', 'spa', 'pets_allowed'
- **stars** (*Integer*) – Contains the amount of stars this hotel has been awarded.
- **rules** (*Rules*) – Contains an associative array, mapping each rule type listed below to the relevant text. List of keys: 'guarantee', 'safety', 'extra_occupants', 'policy', 'charges', 'deposit', 'meals', 'stay', 'tax'
- **rooms** (*Room [ ]*) – contains the available rooms

HotelLocation

JSON Parameters

- **country** (*String*)
- **state** (*String*)
- **city** (*String*)
- **address** (*String*)
- **zip_code** (*String*)
- **area** (*String*) – one of: 'north', 'east', 'south', 'west', 'downtown', 'airport', 'resort'

- **recommended_transport** (*String*) – one of: 'boat', 'coach', 'train', 'free', 'helicopter', 'limousine', 'plane', 'rental car', 'taxi', 'subway', 'walking'

Room

JSON Parameters

- **room_id** (*String*) – ID of the room in question
- **booking_id** (*String*) – ID to use when booking this room
- **price** (*RoomPrice*) – Contains data about the price of the room
  - **amount** (*Float*) –
  - **covers** (*String*) – One of 'day' or 'trip', specifies which duration the price covers
  - **rate_varies** (*Boolean*) – True if the rate is not going to be the same for each day during the occupant's stay. In this case, the above given amount is the highest one during the trip.
- **room_type** (*Traits*) – Contains the traits of the given room, including the category, bed/shower availability, whether smoking is allowed, and whether it is a suite. The keys are the following: 'bath', 'shower', 'nonsmoking', 'suite', 'category'. The first four have boolean values, while 'category' can be one of: 'minimum', 'standard', 'moderate', 'superior', 'executive'
- **bed_type** (*String*) – One of: 'single', 'double', 'twin', 'king size', 'queen size', 'pull-out', 'water bed'
- **description** (*String*) – Contains a short text about the room
- **quantity** (*Integer*) – Contains the amount left to be booked of this room

Examples

Response

JSON:

```
{
  "hotel_details": {
    "amenities": {
      "air_conditioning": false,
      "bar": true,
      "business_center": false,
      "gym": true,
      "internet": false,
      "laundry": false,
      "meeting_rooms": true,
      "parking": true,
      "restaurant": false,
      "room_service": false,
      "safe_deposit_box": true,
      "spa": true,
      "swimming": false
    },
    "category": "tourist",
    "chain_code": "RT",
    "chain_name": "ACCOR HOTELS",
    "contact_info": {
      "phone_numbers": [
        "33/1/40851919",
        "33/1/40859900"
      ]
    },
    "description": "the ibis paris gennevilliers hotel boasts an ideal␣
↪location just outside paris just a stone's throw away from the les␣
↪agnettes metro stop, you'll find yourself in the center of paris in just␣
↪over 15 minutes this 3-star hotel has everything you need foran enjoyable␣
↪stay: fully equipped rooms, gourmet restaurant, 24-hour bar, conference␣
↪rooms and an ideal location with shops nearby and a shopping center␣
↪opposite the hotel.",
    "hotel_code": "GVL",
    "hotel_name": "Ibis paris gennevilliers.",
    "location": {
      "address": "32 36 rue louis calmel.",
      "area": "downtown",
      "city": "PAR",
      "country": "FR",
      "recommended_transport": "taxi",
      "state": "",
      "zip_code": "92230"
    },
    "photos": [
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_EXT_01.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_EXT_02.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_LOUNGE_01.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_LOUNGE_02.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_REST_01.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_REST_02.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_CONF_01.jpg",
      "https://static.allmyles.com/hotels/81bf3a6c/55_0_REC_01.jpg"
    ],
    "points_of_interest": {
      "airports": [
        {
          "airport_code": "CDG",
          "airport_name": "CHARLES DE GAULLE",
          "direction": "NE",
          "distance": "14.9",
          "unit": "MI"
        },
        {
          "airport_code": "ORY",
          "airport_name": "ORLY",
          "direction": "S",
          "distance": "21.7",
          "unit": "MI"
        }
      ],
      "city_center": {
        "distance": "0.4",
        "unit": "MI"
      },
      "miscellaneous": [
        {
          "direction": "NE",
          "distance": "1.8",
          "name": "EIFFEL TOWER",
          "type": "tourist",
          "unit": "KM"
        },
        {
          "direction": "W",
          "distance": "1.0",
          "name": "LE LOUVRE",
          "type": "tourist",
          "unit": "KM"
        }
      ]
    },
    "price": {
      "currency": "HUF",
      "maximum": 20308.52,
      "minimum": 14634.08
    },
    "rooms": [
      {
        "bed_type": "twin",
        "booking_id": "55_0/85_0",
        "description": "STANDARD ROOM WITH 2 SINGLE BEDS",
        "price": {
          "amount": 14634.08,
          "covers": "trip",
          "rate_varies": false
        },
        "quantity": 2,
        "room_id": "85_0",
        "room_type": {
          "bath": true,
          "category": "standard",
          "nonsmoking": false,
          "shower": true,
          "suite": false
        }
      }
    ],
    "rules": {
      "charges": "FAX CHARGE: -INCOMING FAX COMPLIMENTARY : COMPLIMENTARY -
↪OUTGOING FAX COMPLIMENTARY : COMPLIMENTARY",
      "deposit": "NO DEPOSIT REQUIRED",
      "extra_occupants": null,
      "guarantee": "FROM 26:10:2006 UNTIL 31:12:2050 MONDAY TUESDAYWEDNESDAY␣
↪THURSDAY FRIDAY SATURDAY SUNDAYHOLD TIME: 19:00GUESTS ARRIVING AFTER 19:00␣
↪(LOCAL TIME) MUST PROVIDE A GUARANTEE.ACCEPTED FORM OF GUARANTEE -␣
↪26:10:2006 - 31:12:2050 CREDIT CARDCREDIT CARD ACCEPTED FOR GUARANTEE AX -␣
↪CA - DC - EC - IK - VINO GUARANTEE REQUIREDFROM 24:10:2006 UNTIL␣
↪31:12:2050CANCELLATION POLICIES:CANCEL BY 19:00(24 HOUR CLOCK) ON DAY OF␣
↪ARRIVAL,LOCAL HOTEL TIMECANCEL 0 DAY BEFORE ARRIVALNO CANCELLATION CHARGE␣
↪APPLIES PRIOR TO 19:00(LOCAL TIME) ON THE DAY OF ARRIVAL. BEYOND THAT TIME,
↪ THE FIRST NIGHT WILL BE CHARGED.",
      "meals": null,
      "policy": "CHECK-IN TIME: 12:00CHECK-IN TIME 12:00CHECK-OUT TIME:␣
↪12:00CHECK-OUT TIME 12:00NO SPECIAL CONDITIONS FOR CHILDREN.ACCEPTED FORM␣
↪OF PAYMENT - 26:10:2006 - 31:12:2050 CREDIT CARDCREDIT CARD ACCEPTED FOR␣
↪PAYMENT AX - CA - DC - EC - IK - VI",
      "safety": "-SAFE DEP BOX              -SMOKE DETECTOR-FIRE SAFETY       ␣
↪       -ELEC GENERATOR-FIRE DETECTORS-EMERG LIGHTING           -SAFE",
      "stay": null,
      "tax": "CITY TAX 1.00 EUR PER PERSON PER NIGHT -FOOD & BEVERAGE TAX␣
↪PER ROOM PER NIGHTINCLUSIVE - COUNTRY TAX PER ROOM PER NIGHTINCLUSIVE"
    },
    "stars": 3,
    "thumbnail": "https://static.allmyles.com/hotels/81bf3a6c/55_0.jpg"
  }
}
```

Room Details

Request

GET /hotels/:hotel_id/rooms/:room_id

hotel_id is the ID of the *Hotel* the room belongs to, **room_id** is the ID of the *Room* to get the details of.

Response Body

JSON Parameters

- **hotel_room_details** (*HotelRoomDetails*) – root container

HotelRoomDetails

JSON Parameters

- **rules** (*Rules*) – Contains an associative array, mapping each rule type listed below to the relevant text, or a relevant boolean value. List of keys: ‘cancellation’, ‘notes’, ‘needs_guarantee’, ‘needs_deposit’
- **price** (*RoomPrice*) –
  - **total** (*Float*) – The total cost of booking the hotel for the guest. This includes the charge we require right now.
  - **charge** (*Float*) – The amount of money we need to charge the guest to complete the booking. If this amount is zero, no transaction needs to be made and you can go on to booking straight away.
- **includes** (*String [ ]*) – Contains what services or extras are included in the price.

Examples

Response

JSON:

```
{
  "hotel_room_details": {
    "price": {
      "amount": "12887.08",
      "includes": [
        "Extra Adult",
        "Value Added Tax"
      ]
    },
    "rules": {
      "cancellation": "CANCEL LATEST BY 01-MAR-15 12PM TO AVOID PENALTY OF␣
↪36.00<br>",
      "needs_deposit": false,
      "needs_guarantee": true,
      "notes": "NON SMOKING DOUBLE EN SUITE<br>MAX OCCUPANCY 2 ADULTS<br>1␣
↪DOUBLE BED<br> BAR FLEXIBLE RATE<br>GUARANTEE IS MANDATORY,AX,CA,MC,TG,VI
↪<br>A DEPOSIT IS NOT REQUIRED.<br>Minimum Duration, 1, Days<br>Maximum␣
↪Duration, 28, Days<br>"
    }
  }
}
```

Payment

If payment is required—that is, if the room's charge field was not zero—this is where Allmyles gets the payment data.

The only supported payment provider at the moment is PayU. When we receive a transaction ID that points to a successful payment by the passenger, we essentially take that money from PayU, and forward it to the provider to book a hotel room in the *Booking* step.

Request

POST /payment

JSON Parameters

- **payuId** (*String*) – the transaction ID identifying the successful transaction at PayU
- **basket** (*String[ ]*) – the booking IDs the payment is for

Response Body

N/A:

Returns an HTTP 204 No Content status code if successful.

Response Codes

- **412 'a request is already being processed'**: This error comes up even when the other request is asynchronous (i.e. when we are still processing a search request). The response for async requests does not need to be retrieved for this error to clear, just wait a few seconds.

Examples

Request

JSON:

```
{
  "payuId": "12345678",
  "basket": ["2_1_0"]
}
```

Booking

Request

POST /books

JSON Parameters

- **bookBasket** (*String [ ]*) – an array containing only the booking ID of the *Room* to book
- **billingInfo** (*Contact*) – billing info for the booking
- **contactInfo** (*Contact*) – contact info for the booking
- **persons** (*Person [ ]*) – the list of occupants

Contact

JSON Parameters

- **address** (*Address*) – address of the entity in question
- **email** (*String*) – email of the entity in question
- **name** (*String*) – name of the entity in question
- **phone** (*Phone*) – phone number of the entity in question

Address

JSON Parameters

- **addressLine1** (*String*)
- **addressLine2** (*String*) – *(optional)*
- **addressLine3** (*String*) – *(optional)*
- **cityName** (*String*)
- **zipCode** (*String*)
- **countryCode** (*String*) – the two letter code of the country

Phone

JSON Parameters

- **countryCode** (*Integer*)
- **areaCode** (*Integer*)
- **phoneNumber** (*Integer*)

Person

JSON Parameters

- **birthDate** (*String*) – format is `YYYY-MM-DD`
- **email** (*String*)
- **namePrefix** (*String*) – one of `Mr`, `Ms`, or `Mrs`
- **firstName** (*String*)
- **lastName** (*String*)
- **gender** (*String*) – one of `MALE` or `FEMALE`

Response Body

JSON Parameters

- **confirmation** (*String*) – the ID of the booking, this is what the occupant can use at the hotel to refer to his booking
- **pnr** (*String*) – the PNR locator of the record in which the booking was made

Examples

Request

JSON:

```
{
  "bookBasket": ["1_0/2_0"],
  "billingInfo": {
    "address": {
      "addressLine1": "Váci út 13-14",
      "cityName": "Budapest",
      "countryCode": "HU",
      "zipCode": "1234"
    },
    "email": "ccc@gmail.com",
    "name": "Kovacs Gyula",
    "phone": {
      "areaCode": 30,
      "countryCode": 36,
      "phoneNumber": 1234567
    }
  },
  "contactInfo": {
    "address": {
      "addressLine1": "Váci út 13-14",
      "cityName": "Budapest",
      "countryCode": "HU"
    },
    "email": "bbb@gmail.com",
    "name": "Kovacs Lajos",
    "phone": {
      "areaCode": 30,
      "countryCode": 36,
      "phoneNumber": 1234567
    }
  },
  "passengers": [
    {
      "birthDate": "1974-04-03",
      "email": "aaa@gmail.com",
      "firstName": "Janos",
      "gender": "MALE",
      "lastName": "Kovacs",
      "namePrefix": "Mr"
    }
  ]
}
```

Response

JSON:

```
{
  "confirmation": "305863919",
  "pnr": "6JT3ZB"
}
```

Car Rentals

Summary

The car rental workflow consists of three mandatory steps.

1. *Search*
2. *Payment*
3. *Booking*

Additional calls that are available:

- *Details*

Search

Request

`POST /cars`

Searches for cars that match provided criteria.

JSON Parameters

- **airport_code** (*String*) – the IATA code of the airport to find available cars at
- **start_date** (*String*) – pickup date and time, in ISO format ex. 2014-12-24T12:00:00Z)
- **end_date** (*String*) – return date and time, in ISO format (ex. 2014-12-26T12:00:00Z)

- **filters** (*Filter*) – *(optional)* search filter for different car properties

Filter

JSON Parameters

- **type** (*String [ ]*) – one of the *Available Car Types* listed below

Response Body

JSON Parameters

- **car_results** (*Car [ ]*) – root container

Car

JSON Parameters

- **vehicle_id** (*String*) – The ID that identifies the car for getting its details or booking it
- **vendor_id** (*String*) – The ID to use for finding more results by this vendor
- **vendor_name** (*String*)
- **vendor_code** (*String*)
- **available** (*Boolean*) – Whether the car is available to book
- **traits** (*Car Traits*) – Certain properties of the car
  - **class** (*String*) – One of the *Available Car Classes* listed below
  - **type** (*String*) – One of the *Available Car Types* listed below
  - **transmission** (*String*) – One of 'manual' or 'automatic'
  - **air_conditioning** (*Boolean*) – Whether the car has AC or not
- **price** (*Price*)
  - **amount** (*Float*)
  - **currency** (*String*)
- **unlimited** (*Boolean*) – Whether the booking fee covers unlimited usage for the car for the given number of days.
- **overage_fee** (*Overage Fee*) – These fields are relevant only if 'unlimited' has a value of False above. Details the distance usage limitations and overage fees to be paid if the passenger goes over the given distance limit
  - **unit** (*String*) – The distance unit that the value of the 'included_distance' field's amount is given in, and also the unit of distance that the 'amount' field is valid for
  - **included_distance** (*Integer*) – The distance that is included in the car's current price. Once this is reached, the passenger has to pay the per mile or per kilometer overage fee set below
  - **amount** (*Float*) – The amount of money the passenger has to pay per unit of distance (mile or kilometer as given above) once they hit the given limit
  - **currency** (*String*)

To show some examples calculating the overage fee (developers using the API will rarely need to really do this, but passengers need to be informed about how this works, hence the detailed description here):

If 'overage_fee' has the values `{"unit": "Mile", "included_distance": 1000, "amount": 0.05, "currency": "EUR"}` and 'price' has the values `{"amount": 150.0, "currency": "EUR"}`:

- If the passenger drives 800 miles, they only have to pay the base price of 150 EUR.
- If the passenger drives 1400 miles, they pay the pay fee of 150 EUR plus 400 miles times 0.05 EUR per mile, which comes out to 170 EUR in total.

Please note that the unit above could be kilometers as well.

Available Car Classes

- mini
- mini elite
- economy
- economy elite
- compact
- compact elite
- intermediate
- intermediate elite
- standard
- standard elite
- full-size
- full-size elite
- premium
- premium elite
- luxury
- luxury elite
- oversize
- special

Available Car Types

- 2 door car
- 2/4 door car
- 4 door car
- coupe
- SUV
- crossover
- motor home
- open air all terrain
- commercial van/truck
- limousine
- monospace
- roadster
- pick up regular cab
- pick up extended cab
- recreational vehicle
- sport
- convertible

- passenger van
- wagon/estate
- special
- 2 wheel vehicle
- special offer car

Response Codes

- **410 'Location is closed at either the arrival or the departure time.'**

Examples

Request

JSON (regular search):

```
{
  "airport_code": "LHR",
  "start_date": "2015-03-01T10:00:00Z",
  "end_date": "2015-03-04T10:00:00Z",
  "filters": {
    "type": [
      "crossover"
    ]
  }
}
```

Response

JSON:

```
{
  "car_results": [
    {
      "available": true,
      "traits": {
        "transmission": "manual",
        "air_conditioning": true,
        "type": "2/4 door car",
        "class": "mini"
      },
      "vehicle_id": "1_0_0",
      "vendor_name": "NATIONAL",
      "overage_fee": {
        "currency": "EUR",
        "amount": null,
        "unit": null,
        "included_distance": null
      },
      "price": {
        "currency": "EUR",
        "amount": "75.48"
      },
      "vendor_id": "0",
      "unlimited": true,
      "vendor_code": "ZL"
    },
    {
      "available": true,
      "traits": {
        "transmission": "manual",
        "air_conditioning": true,
        "type": "4-5 door car",
        "class": "compact"
      },
      "vehicle_id": "2_1_0",
      "vendor_name": "EUROPCAR",
      "overage_fee": {
        "currency": "EUR",
        "amount": "0.14",
        "unit": "Mile",
        "included_distance": 300
      },
      "price": {
        "currency": "EUR",
        "amount": "98.90"
      },
      "vendor_id": "1",
      "unlimited": false,
      "vendor_code": "EP"
    }
  ]
}
```

Details

Request

GET /cars/:vehicle_id
vehicle_id is the ID of the *Car* to get the details of

Response Body

JSON Parameters

- **car_details** (*CarDetails*) – root container

CarDetails

JSON Parameters

- **locations** (*Location [ ]*) – The list of the vendor's pick up/drop off locations.
- **car_model** (*String*) – The most exact name of the car that is available to us.
- **included** (Package *[ ]*) – A list of things that are already included in the price, and are mandatory (this includes insurance fees, taxes, surcharges, etc.)

- **extras** (Package *[ ]*) – A list of extras that the passenger is going to be able to buy when picking up the car.
- **rules** (*String*) – A string including longform text with the rules for renting given car.

Location

JSON Parameters

- **city** (*String*)
- **address** (*String*)
- **phone** (*String*)
- **fax** (*String*)
- **opens_at** (*String*) – In the format 'HH:MM'
- **closes_at** (*String*) – In the format 'HH:MM'

Package

JSON Parameters

- **price** (*Price*)
  - **amount**
  - **currency**
- **type** (*String*) – The category that the package is in, one of 'surcharge', 'tax', 'coverage', 'coupon' for included packages, or 'child seat', 'child seat (<1 year)', 'child seat (1-3 years)', 'child seat (4-7 years)', 'baby stroller', 'navigation system', or 'extra coverage' for extras.
- **period** (*String*) – The period that the given price applies to, one of: 'day', 'week', 'month', or 'rental' - renting a car for 5 days means that adding an extra with a day period set here for the entire trip is going to add five times the 'price' amount to the total price.
- **description** (*String*) – The exact name of the package, such as type of insurance.

Examples

Response

JSON:

```
{
  "car_details": {
    "included": [
      {
        "price": {
          "currency": "EUR",
          "amount": "0.00"
        },
        "type": "surcharge",
        "period": "day",
        "description": "DAMAGE LIABILITY WAIVER"
      },
      {
        "price": {
          "currency": "EUR",
          "amount": "11.99"
        },
        "type": "coverage",
        "period": "day",
        "description": "CDW - COLLISION DAMAGE WAIVER"
      },
      {
        "price": {
          "currency": "GBP",
          "amount": "20.00"
        },
        "type": "tax",
        "period": "rental",
        "description": "VALUE ADDED TAX"
      }
    ],
    "car_model": "FIAT 500 OR SIMILAR",
    "extras": [
      {
        "price": {
          "currency": "EUR",
          "amount": "13.98"
        },
        "type": "extra coverage",
        "period": "day",
        "description": "MCP"
      }
    ],
    "rules": "BASE RATE INCLUDES SURCHARGES\nBASE RATE INCLUDES TAXES\nPRICE␣
↪INCLUDES TAX SURCHARGE INSURANCE. 0.00 GBP\nDAMAGE LIABILITY WAIVER␣
↪ALREADY INCLUDED.\nIATA NBR NOT ON FILE QUEUE AGENCY INFO TO ZL\nALLOWED -␣
↪RETURN TO SPECIFIED LOCATION ONLY\nA MINIMUM OF 3 DAYS WILL BE CHARGED",
    "locations": [
      {
        "closes_at": "23:59",
        "city": "GB",
        "fax": null,
        "phone": "44 08713843410",
        "address": "EUROPCAR AND NATIONAL HEATHROW NORTHER",
        "opens_at": "00:00"
      }
    ]
  }
}
```

Payment

Request

POST /payment

JSON Parameters

- **payuId** (*String*) – the transaction ID identifying the successful transaction at PayU
- **basket** (*String [ ]*) – contains the booking IDs the payment was made for (this array will normally have only one item in it)

Response Body

N/A:

Returns an HTTP 204 No Content status code if successful.

Examples

Request

JSON:

```
{
  "payuId": "12345678",
  "basket": ["1_0_0"]
}
```

Booking

Request

POST /books

JSON Parameters

- **bookBasket** (*String [ ]*) – an array containing only the booking ID of the *Room* to book
- **billingInfo** (*Contact*) – billing info for the booking
- **contactInfo** (*Contact*) – contact info for the booking
- **persons** (*Person [ ]*) – the list of occupants

Contact

JSON Parameters

- **address** (*Address*) – address of the entity in question
- **email** (*String*) – email of the entity in question
- **name** (*String*) – name of the entity in question
- **phone** (*Phone*) – phone number of the entity in question

Address

JSON Parameters

- **addressLine1** (*String*)
- **addressLine2** (*String*) – *(optional)*
- **addressLine3** (*String*) – *(optional)*
- **cityName** (*String*)
- **zipCode** (*String*)
- **countryCode** (*String*) – the two letter code of the country

Phone

JSON Parameters

- **countryCode** (*Integer*)
- **areaCode** (*Integer*)
- **phoneNumber** (*Integer*)

Person

JSON Parameters

- **birthDate** (*String*) – format is `YYYY-MM-DD`
- **email** (*String*)
- **namePrefix** (*String*) – one of `Mr`, `Ms`, or `Mrs`
- **firstName** (*String*)
- **lastName** (*String*)
- **gender** (*String*) – one of `MALE` or `FEMALE`
- **document** (*Document*) – data about the identifying document the person wishes to travel with

Document

JSON Parameters

- **id** (*String*) – document's ID number
- **dateOfExpiry** (*String*) – format is YYYY-MM-DD
- **issueCountry** (*String*) – two letter code of issuing country
- **type** (*String*) – one of DocumentTypes

Response Body

JSON Parameters

- **confirmation** (*String*) – the ID of the booking, this is what the occupant can use at the car vendor to refer to his booking
- **pnr** (*String*) – the PNR locator of the record in which the booking was made

Examples

Request

JSON:

```
{
  "billingInfo": {
    "address": {
      "addressLine1": "Váci út 13-14",
      "cityName": "Budapest",
      "countryCode": "HU",
      "zipCode": "1234"
    },
    "email": "ccc@gmail.com",
    "name": "Kovacs Gyula",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "phoneNumber": "1234567"
    }
  },
  "bookBasket": [
    "33_0_0"
  ],
  "contactInfo": {
    "address": {
      "addressLine1": "Váci út 13-14",
      "cityName": "Budapest",
      "countryCode": "HU",
      "zipCode": "1234"
    },
    "email": "ccc@gmail.com",
    "name": "Kovacs Gyula",
    "phone": {
      "areaCode": "30",
      "countryCode": "36",
      "phoneNumber": "1234567"
    }
  },
  "persons": [
    {
      "birthDate": "1974-04-03",
      "document": {
        "dateOfExpiry": "2016-09-03",
        "id": "12345678",
        "issueCountry": "HU",
        "type": "Passport"
      },
      "email": "aaa@gmail.com",
      "firstName": "Janos",
      "gender": "MALE",
      "lastName": "Kovacs",
      "namePrefix": "Mr"
    }
  ]
}
```

Response

JSON:

```
{
  "confirmation": "1647353336COUNT",
  "pnr": "6KSSY3"
}
```

Masterdata

Summary

The masterdata component consists of the following two endpoints:

- *Search* is for general usage; this will search through all location data and find appropriate entries for a given keyword, intended mostly to be used for autocompleting the location field for users.
- *Retrieval* is the call used to retrieve all data of a certain type. This should not be used very often—data is cached for 24 hours, but downloading these once or twice a month should be sufficient.

Search

Request

GET /masterdata/search

Searches for locations whose name starts with the provided keyword.

GET Parameters

- **keyword** – the string to find locations for
- **limit** – *(optional)* the number of results to retrieve (default: 10, maximum: 100)
- **locales** – *(optional)* the languages to search in in addition to English (which is on by default.) Multiple locale codes can be given, separated by commas (ex. `?locales=hu-HU,cs-CZ`) See the available locale codes below.

Locale Codes

- it-IT
- pl-PL

- ja-JP
- es-ES
- en-GB
- cs-CZ
- zh-CN
- tr-TR
- ro-RO
- lt-LT
- kk-KZ
- ru-RU
- hu-HU
- fr-FR
- el-GR
- fi-FI
- nl-BE
- hr-HR
- pt-PT
- ko-KR
- sk-SK
- de-DE
- sq-AL

Response Body

JSON Parameters

- **locationSearchResult** (*SearchResult [ ]*) – root container
  - **canonicalName** – the complete name of the airport/multiairport
  - **htmlFragment** – the canonical name, preformatted by bolding the searched substring. You can inject this string directly into your HTML source.
  - **iataCode** – the code identifying the matched location—either an airport's, or a city's IATA code
  - **category** – one of the following: airport, multiairport, locality, state, country
  - **cityName**
  - **countryCode**
  - **countryName**

Examples

Response

JSON:

```
{
  "locationSearchResult": [
    {
      "canonicalName": "Budapest, HU - Liszt Ferenc Intl (BUD)",
      "category": "airport",
      "cityName": "Budapest",
      "countryCode": "HU",
      "countryName": "Hungary",
      "htmlFragment": "<strong>Bud</strong>apest, HU - Liszt Ferenc Intl (
↪<strong>BUD</strong>)",
      "iataCode": "BUD"
    }
  ]
}
```

Retrieval

Request

GET /masterdata/:category

category is the data repo you'd like to retrieve. It can be one of the following:

- airlines
- airplanes
- airports
- categories
- cities
- localised_cities
- countries
- states
- hotel_chains
- hotels
- rule_links
- eticket_rules

Response Body

The response will have a root container that is unique to the requested data repo. This is an array, containing objects that are, again, unique.

Note: A small cosmetic deficiency in the XML output is that the tags of the child elements are generated from the root tag, by a not-so-intelligent block of word singularizing code. This can lead to things such as a <Cities> root containing <Citie> elements. When the root doesn't end with the letter S, the XML generator just defaults to calling the children <item>s.

Examples

Response

JSON:

```
{
  "Airlines": [
    {
      "Active": "true",
      "AirLineCode": "01",
      "AirLineName": "RailEasy",
      "ProviderType": "TravelFusion2Provider"
    },
    {
      "Active": "true",
      "AirLineCode": "08",
      "AirLineName": "Air Southwest",
      "CountryCode": "GB",
      "ProviderType": "ERetailWebFareProvider"
    }
  ]
}
```

XML:

```
<Airlines>
  <Airline>
    <Active>true</Active>
    <AirLineCode>ZY</AirLineCode>
    <ProviderType>AmadeusProvider;SkyProvider</ProviderType>
    <AirLineName>Sky Airlines</AirLineName>
  </Airline>
  <Airline>
    <Active>false</Active>
    <AirLineCode>ZZ</AirLineCode>
    <ProviderType>AmadeusProvider</ProviderType>
    <AirLineName>Airline Service</AirLineName>
  </Airline>
</Airlines>
```

Allmyles PHP SDK

Summary

The Allmyles PHP SDK is a PHP package aiming to simplify integration of the Allmyles API into PHP projects. The SDK is designed to be as forgiving as possible, often accepting multiple types as input, handling conversion between different (at least somewhat sensible) types automatically.

You can find the project's source on GitHub.

Installation

Just add this in your composer.json file and run `composer install`:

```
"require": {"allmyles/allmyles-sdk-php": "1.*"}
```

Or, if for some reason you can't use a package manager, you can also just download the SDK's source in a ZIP archive from GitHub and handle your dependencies manually. You *really* shouldn't be doing that, though.

Example Code

The allmyles-sdk-php repository contains several examples in /doc/examples/. You might want to try those to get a general feel for how the SDK and the ticketing process work in general before you delve into the details down below.

There's code for three different ways to use the API—these are completely intercompatible, though. We recommend you to read through the simple booking flow first, as that contains everything you need to successfully create a ticket. The complex flow is kind of a showcase for all the advanced ways you can use the API, you will most likely not need everything from there. (It might be a good idea to start reading the API Reference once you're about to start experimenting with these advanced features.)

The raw examples take a lot of responsibility off the shoulders of the SDK; the requests are written more or less manually in those. This should be used by advanced users who don't feel comfortable deferring the task of crafting the requests to the SDK, and would rather build arrays themselves instead of using the SDK's query classes. Raw requests might also come in handy if the Allmyles API has new features deployed that aren't supported by the SDK just yet.

Initialization

Initialization is done by creating an `Allmyles\Client` object with the API access details given as arguments. Each request will be made via a method call to this object.

Initialization will look something like this:

```
$allmyles = new Allmyles\Client('https://api.allmyles.com/v2.0', 'api-key');
```

API Reference

class `Allmyles\`**`Client`**

`__construct` (*$baseUrl*, *$authKey*)
Initializes your Allmyles client.

Parameters

- **`$baseUrl`** (*`string`*) – The URL of the Allmyles API you're using (ex. `https://api.allmyles.com/v2.0`).
- **`$authKey`** (*`string`*) – Your API key for the Allmyles API.

Returns A `Client` object.

`searchFlight` (*$parameters*[, *$async = true*, *$session = null*])
Sends a flight search request to the Allmyles API.

Parameters

- **`$parameters`** (*`mixed`*) – This should be either a `Flights\SearchQuery` object, or an associative array containing the raw data to be sent off based on the *Search* API docs.
- **`$async`** (*`boolean`*) – Whether the SDK should defer asyncronosity to your code or not. Setting this to false will make the method call block until a search response arrives (which can take around 30-40 seconds.) If it's true you will need to periodically call the response's retry method yourself until the results arrive.
- **`$session`** (*`string`*) – If you want to manually set the session cookie for the workflow, specify it here. The SDK automatically handles sessions, though, so feel free to leave this out.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an array of `Flights\FlightResult` objects.

`searchHotel` (*$parameters*[, *$session = null*])
Sends a hotel search request to the Allmyles API.

Parameters

- **$parameters** (*mixed*) – This should be either a `Hotels\SearchQuery` object, or an associative array containing the raw data to be sent off based on the *Search* API docs.
- **$session** (*string*) – If you want to manually set the session cookie for the workflow, specify it here. The SDK automatically handles sessions, though, so feel free to leave this out.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an array of `Hotels\Hotel` objects.

searchCar (*$parameters*[, *$session = null*])

Sends a car search request to the Allmyles API.

Parameters

- **$parameters** (*mixed*) – This should be either a `Cars\SearchQuery` object, or an associative array containing the raw data to be sent off based on the *Search* API docs.
- **$session** (*string*) – If you want to manually set the session cookie for the workflow, specify it here. The SDK automatically handles sessions, though, so feel free to leave this out.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an array of `Cars\Car` objects.

searchLocations (*$parameters*[, *$session = null*])

Sends a masterdata search request to the Allmyles API.

Parameters

- **$parameters** (*array*) – This should be an associative array containing the raw data to be sent off based on the *Search* API docs.
- **$session** (*string*) –

Returns An associative array containing the location search results.

retrieveMasterdata (*$repo*[, *$session = null*])

Sends a masterdata retrieval request to the Allmyles API.

Parameters

- **$repo** (*string*) – This should be the name of one of the data repos served by Allmyles (ex. 'airports').
- **$session** (*string*) –

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an array.

Warning: The methods below are only documented for the sake of completeness. Only use them if you really, *really* need to. The methods of the `Flights\Combination` class handle using all these automatically.

getFlightDetails (*$bookingId*[, *$session = null*])

Gets the details of the given booking ID from the Allmyles API. In almost all cases, this should not be directly called, use `Flights\Combination::getDetails()` instead.

Parameters

- **$bookingId** (*string*) –

- **$session** (*string*) –

Returns A `Curl\Response` object.

bookFlight (*$parameters*[, *$session = null*])
Sends a book request to the Allmyles API. In almost all cases, this should not be directly called, use `Flights\Combination::book()` instead.

Parameters

- **$parameters** (*array*) –
- **$session** (*string*) –

Returns A `Curl\Response` object.

addPayuPayment (*$parmaeters*[, *$session = null*])
Sends a PayU transaction ID to the Allmyles API to confirm that payment was successful. In almost all cases, this should not be directly called, use `Flights\Combination::addPayuPayment()` instead.

Parameters

- **$parameters** (*array*) – Contains 'payuId' and 'basket'
- **$session** (*string*) –

Returns A `Curl\Response` object.

createFlightTicket (*$bookingId*[, *$session = null*])
Gets a ticket for the given booking ID from the Allmyles API. In almost all cases, this should not be directly called, use `Flights\Combination::createTicket()` instead.

Parameters

- **$bookingId** (*string*) –
- **$session** (*string*) –

Returns A `Curl\Response` object.

getHotelDetails (*$hotel*)
Gets the details of the given hotel from the Allmyles API. In almost all cases, this should not be directly called, use `Hotels\Hotel::getDetails()` instead.

Parameters

- **$hotel** (*object*) – This should be a `Hotels\Hotel` object

Returns A `Curl\Response` object.

getHotelRoomDetails (*$room*)
Gets the details of the given room from the Allmyles API. In almost all cases, this should not be directly called, use `Hotels\Room::getDetails()` instead.

Parameters

- **$room** (*object*) – This should be a `Hotels\Room` object

Returns A `Curl\Response` object.

bookHotel (*$parameters*[, *$session = null*])
Sends a book request to the Allmyles API. In almost all cases, this should not be directly called, use `Hotels\Room::book()` instead.

Parameters

- **`$parameters`** (*`array`*) –
- **`$session`** (*`string`*) –

Returns A *`Curl\Response`* object.

`getCarDetails` (*$bookingId*[, *$session = null*])
Gets the details of the given car from the Allmyles API. In almost all cases, this should not be directly called, use *`Cars\Car::getDetails()`* instead.

Parameters

- **`$bookingId`** (*`object`*) – A string, one of the vehicle_id values for the search results.

Returns A *`Curl\Response`* object.

`bookCar` (*$parameters*[, *$session = null*])
Sends a book request to the Allmyles API. In almost all cases, this should not be directly called, use *`Cars\Car::book()`* instead.

Parameters

- **`$parameters`** (*`array`*) –
- **`$session`** (*`string`*) –

Returns A *`Curl\Response`* object.

class `Allmyles\Curl\`**`Response`**
A response from the Allmyles API. Methods that are for internal use only are excluded from this documentation (such as __`construct()`.)

`get`()
Processes the received response, and returns the processed result.

Returns Varies per call, check the notes next to the return values in this documentation to find out what `get()` will return.

`retry`()
Retries the request that resulted in this response. This comes in handy when making async calls.

Returns A new `Curl\Response` object.

class `Allmyles\Flights\`**`SearchQuery`**
This is the object you can pass to a *`Allmyles\Client::searchFlight()`* call to simplify searching.

`__construct` (*$fromLocation*, *$toLocation*, *$departureDate*[, *$returnDate = null*])
Starts building a search query. Searches for a one way flight if no `$returnDate` is given.

Parameters

- **`$fromLocation`** (*`string`*) – A location's three letter IATA code.
- **`$toLocation`** (*`string`*) – A location's three letter IATA code.
- **`$departureDate`** (*`mixed`*) – Either an ISO formatted timestamp, or a `DateTime` object.
- **`$returnDate`** (*`mixed`*) – Either an ISO formatted timestamp, or a `DateTime` object.

Returns A *`SearchQuery`* object.

`addPassengers` (*$adt*[, *$chd = 0*, *$inf = 0*])
Adds the number of passengers to your search query. This is required for your search request to go through.

Parameters

- **`$adt`** (*`integer`*) – The number of adults wanting to travel.
- **`$chd`** (*`integer`*) – The number of children wanting to travel.
- **`$inf`** (*`integer`*) – The number of infants wanting to travel.

`addProviderFilter`(*$providerType*)

Adds a filter to your query that restricts the search to a specific provider.

Parameters

- **`$providerType`** (*`string`*) – The provider to filter to. Use the following contants:

constant `PROVIDER_ALL`

All providers

constant `PROVIDER_TRADITIONAL`

Traditional flights only

constant `PROVIDER_LOWCOST`

LCC flights only

`addAirlineFilter`(*$airlines*)

Adds a filter to your query that restricts the search to specific airlines.

Parameters

- **`$providerType`** (*`mixed`*) – Either a two letter IATA airline code as a string, or an array of multiple such strings.

class `Allmyles\Flights\`**`FlightResult`**

property `combinations`

An associative array of booking ID to *`Combination`* key-value pairs.

Contains the combinations the passenger can choose from in this result.

property `breakdown`

An associative array.

Contains a breakdown of fares per passenger type. See *Breakdown*.

property `totalFare`

A `Common\Price` object.

Contains the fare total to the best of our knowledge at this point.

class `Allmyles\Flights\`**`Combination`**

This is the bookable entity, and these methods are where most of the magic happens.

property `flightResult`

A *`FlightResult`* object.

Contains the parent flight result.

property `bookingId`

A string.

Contains the booking ID associated with this combination.

property `providerType`

A string.

Contains the provider that returned this result.

property `legs`
An array of *`Leg`* objects.

Contains the legs that this combination consists of.

property `serviceFee`
A `Common\Price` object.

Contains the service fee for this combination.

`getDetails`()
Sends the flight details request for this flight.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API in it. See *Details*.

`book`(*$parameters*)
Sends the book request for this flight.

Parameters

- **`$parameters`** (*`mixed`*) – Either a *`BookQuery`* object, or an associative array containing the raw data to be sent off based on the *Booking* API docs.

Returns Either `true` when booking LCC, or a `Curl\Response` object for traditional flights. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API in it. See *Booking*.

`addPayuPayment`(*$payuId*)
Sends the PayU transaction ID to confirm that payment for the ticket has been completed for this flight.

Parameters

- **`$payuId`** (*`string`*) – The PayU transaction ID to confirm payment with.

Returns `true`

`createTicket`()
Sends the ticket creation request for this flight.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API in it. See *Ticketing*.

class `Allmyles\Flights\`**`Leg`**

property `combination`
A *`Combination`* object.

Contains the parent combination.

property `length`
A `DateInterval` object

Contains the length of the leg in hours and minutes.

property `segments`
An array of *`Segment`* objects.

Contains the segments of this leg.

class `Allmyles\Flights\`**`Segment`**

property `leg`
A *Leg* object.

Contains the parent leg.

property `arrival`
A *Stop* object

Contains details about the arrival stop.

property `departure`
A *Stop* object.

Contains details about the departure stop.

property `airline`
A string.

Contains the two character IATA code of the affiliated airline.

property `flightNumber`
A string.

Contains the flight's number.

property `availableSeats`
An integer.

Contains the number of seats left at this price.

property `cabin`
A string.

Contains which cabin the passenger will get a ticket to on this segment.

class `Allmyles\Flights\Stop`

property `segment`
A *Segment* object.

Contains the parent segment.

property `time`
A `DateTime` object

Contains the time of the arrival or departure.

property `airport`
A string.

Contains the three letter IATA code of the airport where the arrival or departure is going to take place.

property `terminal`
A string, or `null`.

Contains the terminal of the airport where the arrival or departure is going to take place, or `null` if the airport only has one terminal.

class `Allmyles\Flights\BookQuery`
This is the object you can pass to a `Flights\Combination::book()` call to simplify booking.

`__construct`([*$passengers = null*, *$contactInfo = null*, *$billingInfo = null*])
Starts building a book query.

Parameters

- **`$passengers`** (*`array`*) – The details of the people wanting to travel. See *Passenger* in the API docs.
- **`$contactInfo`** (*`array`*) – The contact details to book the flight with. See *Contact* in the API docs.
- **`$billingInfo`** (*`array`*) – The billing details to book the flight with. See *Contact* in the API docs.

Returns A *`BookQuery`* object.

`addPassengers`(*$passengers*)

Adds passengers to your book query.

Parameters

- **`$passengers`** (*`array`*) – Either an associative array containing data based on *Passenger* in the API docs, or an array of multiple such arrays.

`addContactInfo`(*$address*)

Adds contact info to your book query.

Parameters

- **`$address`** (*`array`*) – The contact details to book the flight with. See *Contact* in the API docs.

`addBillingInfo`(*$address*)

Adds billing info to your book query.

Parameters

- **`$address`** (*`array`*) – The billing details to book the flight with. See *Contact* in the API docs.

class `Allmyles\Hotels\`**`SearchQuery`**

This is the object you can pass to a *`Allmyles\Client::searchHotel()`* call to simplify searching.

`__construct`(*$location*, *$arrivalDate*, *$leaveDate*[, *$occupants = 1*])

Starts building a search query.

Parameters

- **`$location`** (*`string`*) – A location's three letter IATA code.
- **`$arrivalDate`** (*`mixed`*) – Either an ISO formatted date (ex. 2014-12-24), or a `DateTime` object.
- **`$leaveDate`** (*`mixed`*) – Either an ISO formatted date (ex. 2014-12-24), or a `DateTime` object.
- **`$occupants`** (*`integer`*) – The number of occupants looking for a hotel.

Returns A *`SearchQuery`* object.

class `Allmyles\Hotels\`**`BookQuery`**

This is the object you can pass to a `Flights\Combination::book()` call to simplify booking.

`__construct`([*$occupants = null*, *$contactInfo = null*, *$billingInfo = null*])

Starts building a book query.

Parameters

- **`$occupants`** (*`array`*) – The details of the people wanting to travel. See *Passenger* in the API docs.

- **`$contactInfo`** (*`array`*) – The contact details to book the hotel with. See *Contact* in the API docs.
- **`$billingInfo`** (*`array`*) – The billing details to book the hotel with. See *Contact* in the API docs.

Returns A *`BookQuery`* object.

`addOccupants`(*$occupants*)

Adds occupants to your book query.

Parameters

- **`$occupants`** (*`array`*) – Either an associative array containing data based on *Passenger* in the API docs, or an array of multiple such arrays.

`addContactInfo`(*$address*)

Adds contact info to your book query.

Parameters

- **`$address`** (*`array`*) – The contact details to book the flight with. See *Contact* in the API docs.

`addBillingInfo`(*$address*)

Adds billing info to your book query.

Parameters

- **`$address`** (*`array`*) – The billing details to book the flight with. See *Contact* in the API docs.

class `Allmyles\Hotels\`**`Hotel`**

This contains data about entire hotels.

property `hotelId`

A string.

property `hotelName`

A string.

property `chainName`

A string.

property `thumbnailUrl`

A string.

Contains a link to a small image representing this hotel.

property `stars`

An integer.

Contains the amount of stars this hotel has been awarded.

property `priceRange`

A `Common\PriceRange` object.

Contains the available rates for this hotel (for the cheapest and the most expensive room).

property `location`

A `Common\Location` object.

Contains the coordinates of the hotel.

property amenities

An associative array, maps strings to booleans.

Contains whether the hotel has any of the listed amenities.

- restaurant
- bar
- laundry
- room_service
- safe_deposit_box
- parking
- swimming
- internet
- gym
- air_conditioning
- business_center
- meeting_rooms
- spa
- pets_allowed

getDetails()

Sends the hotel details request for this hotel.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API (see *Details*.), and also an array of bookable room objects in the 'rooms' key.

class `Allmyles\Hotels\`**Room**

This contains data about a room in a hotel.

property hotel

A *Hotel* object.

Contains the parent hotel.

property hotelId

A string.

property bookingId

A string.

property price

A `Common\Price` object.

Contains the total price for this room. Make sure to take the values of the two attributes below into consideration when working with this value.

property priceVaries

A boolean.

If this is true, then the hotel has a different rate for at least one of the nights. The given price is the rate of the most expensive day.

property priceScope

A string.

Either 'day', or 'trip'. The given price covers this scope.

property stars

An integer.

Contains the amount of stars this hotel has been awarded.

property `traits`
An associative array.

Contains the traits of the given room, including the category, bed/shower availability, whether smoking is allowed, and whether it is a suite.

property `bed`
A string.

Specifies the type of the bed in the room. Can be one of the values below.

- single
- double
- twin
- king size
- queen size
- pullout
- water bed

property `description`
A string.

Contains a short text about the room.

property `rules`
An associative array.

Contains each rule type listed below with the relevant text, or a relevant boolean value.

Keys: 'cancellation', 'notes', 'needs_guarantee', 'needs_deposit'

Available only after *`Room::getDetails()`* has been called.

property `charge`
A `Common\Price` object.

Contains the amount that needs to be charged on the guest's credit card to book this room.

Available only after *`Room::getDetails()`* has been called.

property `includes`
An array of strings.

Contains what services or extras are included in the price.

Available only after *`Room::getDetails()`* has been called.

property `quantity`
An integer.

Contains the amount left to be booked of this room.

`getDetails`()
Sends the hotel room details request for this room and updates the object with new attributes.

Returns The currently used *`Allmyles\Hotels\Room`* object.

`addPayuPayment`(*$payuId*)
Sends the PayU transaction ID to confirm that payment for the room has been completed.

Does not need to be called if the required charge was zero

Parameters

- **`$payuId`** (*`string`*) – The PayU transaction ID to confirm payment with.

Returns `true`

`book` (*$parameters*)
Sends the book request for this room.

Parameters

- **`$parameters`** (*`mixed`*) – Either a *`BookQuery`* object, or an associative array containing the raw data to be sent off based on the *Booking* API docs.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API. See *Booking*.

class `Allmyles\Cars\`**`SearchQuery`**
This is the object you can pass to a *`Allmyles\Client::searchCar()`* call to simplify searching.

`__construct` (*$location*, *$startDate*, *$endDate*)
Starts building a search query.

Parameters

- **`$location`** (*`string`*) – An airport's three letter IATA code.
- **`$startDate`** (*`mixed`*) – Either an ISO formatted date (ex. 2014-12-24), or a `DateTime` object.
- **`$endDate`** (*`mixed`*) – Either an ISO formatted date (ex. 2014-12-24), or a `DateTime` object.

Returns A *`SearchQuery`* object.

class `Allmyles\Cars\`**`BookQuery`**
This is the object you can pass to a `Cars\Car::book()` call to simplify booking.

`__construct` ([*$persons = null*, *$contactInfo = null*, *$billingInfo = null*])
Starts building a book query.

Parameters

- **`$occupants`** (*`array`*) – The details of the people wanting to travel. See *Person* in the API docs.
- **`$contactInfo`** (*`array`*) – The contact details to book the hotel with. See *Contact* in the API docs.
- **`$billingInfo`** (*`array`*) – The billing details to book the hotel with. See *Contact* in the API docs.

Returns A *`BookQuery`* object.

`addPersons` (*$persons*)
Adds people to your book query.

Parameters

- **`$persons`** (*`array`*) – Either an associative array containing data based on *Person* in the API docs, or an array of multiple such arrays.

`addContactInfo` (*$address*)
Adds contact info to your book query.

Parameters

- **`$address`** (*`array`*) – The contact details to book the flight with. See *Contact* in the API docs.

`addBillingInfo`(*$address*)

Adds billing info to your book query.

Parameters

- **`$address`** (*`array`*) – The billing details to book the flight with. See *Contact* in the API docs.

class `Allmyles\Cars\`**`Car`**

This contains data about cars. See explanation of field values in the API docs, in the Car chapter's *Search* section.

property `vehicleId`

A string.

property `price`

A `Common\Price` object.

property `vendor`

A `Cars\Vendor` object.

property `isAvailable`

A boolean value.

property `isUnlimited`

A boolean value.

property `overageFee`

An associative array.

Keys are 'includedDistance' as an integer, 'unit' as a string, and 'price' as a `Common\Price` object.

property `traits`

An associative array.

`getDetails`(*$parameters*)

Sends the car details request for this hotel.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API (see *Details*.)

`book`(*$parameters*)

Sends the book request for this car.

Parameters

- **`$parameters`** (*`mixed`*) – Either a *`BookQuery`* object, or an associative array containing the raw data to be sent off based on the *Booking* API docs.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns an associative array with the response from the Allmyles API. See *Booking*.

class `Allmyles\Cars\`**`Vendor`**

This contains data about a vendor.

property `id`

A string.

This is what Allmyles identifies the vendor by internally.

property `name`

A string.

property `code`

A string.

searchCars()

Sends a search request that will return more results from this vendor.

Returns A `Curl\Response` object. Calling `Curl\Response::get()` on this returns a new array of `Cars\Car` objects.

class `Allmyles\Common\`**`Price`**

property **`amount`**

A floating point number.

Contains the amount of money in the given currency that the price entails.

property **`currency`**

A string.

Contains the currency that the amount is in.

class `Allmyles\Common\`**`PriceRange`**

property **`minimum`**

A floating point number.

Contains the minimum amount of money in the given currency that the price range entails.

property **`maximum`**

A floating point number.

Contains the maximum amount of money in the given currency that the price range entails.

property **`currency`**

A string.

Contains the currency that the amounts are in.

class `Allmyles\Common\`**`Location`**

property **`latitude`**

A floating point number.

Contains the latitude of the location.

property **`longitude`**

A string.

Contains the longitude of the location.

Certification Requirements

This document details the list of things to check on client's website before allowing them to use our API in production.

General

The site must send all required HTTP headers with correct values:

- Content-Type should accurately reflect the sent data's format.

- Cookie should be unique for all booking sessions, but persistent in each. There must be one cookie per user, and all of a user's search requests must be sent using that cookie. The cookie must be the same in all tabs of one browser.
- User-Agent should contain an identifier for the client's software. This would preferably be unique to a specific version of what the backend is using to communicate with the Allmyles API, so that the server side can block requests from user agents that are known to misbehave in case of an emergency. (Bad: Java - Good: allmyles-api.py v0.7.2-dev)
- X-Auth-Token should always contain the client's API token. While there's no way for us to check for this via just clicking through a site, we expect this token to be easily configurable even by someone with little technical experience, to be able to react as quickly as possible if this token is compromised.
- All requests must be relayed through the site's servers, so that the API token doesn't get exposed in the browser, as it would if calls were made with AJAX request right in the passenger's browser.
- Requests should not be made to a nonexistent endpoint (HTTP 404 or HTTP 405), and requests with malformed bodies should be avoided as much as possible. If the request fails to validate on the Allmyles API, the error's description should be shown to the passenger.

Flights

- The asynchronous flight search calls should be handled properly as described in the documentation; infinite loops must never happen.
- There should be a reasonable amount of time spent sleeping between each flight search call for one search.
- The flight search request must contain the user data as it is explained in the documentation.
- Within one client session only one flight search should be going on at a time. Parallel periodic requests must not be sent for different flight searches. When the processing of a flight search call is in progress (Allmyles API is responding with 202) and a new search request is submitted with different parameters in the same client session, a HTTP 412 error will be returned.
- After the Allmyles API returned a status other than 202 with a content body in response to a search request, and a new search request with different parameters is submitted in the same client session, periodic requesting of the result of the previous search should stop.
- Once the flight result is retrieved once, it shouldn't be requested again later in the same workflow.
- Flight details calls must only be made once the passenger has explicitly selected one of the results, no prefetching is allowed. (However, making multiple details calls in one workflow is allowed.)
- When the retrieved flight details are shown, the flight's updated price should be displayed to the passenger. This updated price must be clearly visible.
- When the passenger is sent to pay, the amount should match the sum of the total price given in the flight details call, and any extra fees that the passenger selected, such as for baggages.
- Ticket requests must be made only after payment has been confirmed to have been successful. (This is unrelated to the payment call, we are referring to the payment from the passenger to the travel site.)
- The passenger's transaction must not be refunded unless the Allmyles API explicitly says that it is okay to do so.

Displaying Errors to Users

- Error messages on the frontend must not be the same as the message received from our API.
- Error messages must be meaningful for the end-user.

- Error messages must include an identifier for debugging purposes, which must be the same as the session cookie.

Note: For example, an error during the search must return a message like this on the frontend, regardless of the error you have received from our API:

```
An error has occured during the processing of your search. Session id:  12345
```

Attachments of the Certification

You must include screenshots of the following on order to certify successfully:

- Your booking page which includes fields for the first, middle and last names of each passenger.
- Message displayed to users when booking has timed out.
- Message displayed to users when the external gateway has timed out (http error code 504).
- Message displayed to users when the flight is already booked (http error code 412).
- Message displayed to users when you receive an undefined error from the provider (http error code 500).

Masterdata

- The number of results retrieved and the number of locales searched for a search keyword must be reasonable.
- Masterdata repos should not be retrieved more than once a day.

This documentation provides information about the Allmyles travel services, API including flight, hotel, car rental search and booking. The documentation contains a general API overview, detailed API reference and common use case examples.

CHAPTER 2

Intended Audience

The intended audience of this documentation is mainly the developers and system integrators who are looking to use the Allmyles provided services in their web, mobile, or other business solutions. To ease the integration process and shorten adoption time, the services are reachable through a standard REST API (explained in later chapters) which can be easily fit with most common platforms used nowadays, including Ruby, node.js, Python, PHP (incl. Drupal), Java, C#/.Net, and so on.

CHAPTER 3

Wrappers and Example Code

- PHP Allmyles API SDK

CHAPTER 4

Indices and tables

- genindex
- search

HTTP Routing Table

PHP Namespace Index

Index