Car Rental Software

CRS Booker API Reference

Versione 1.3
Saturday, 14 September 2019

Changelog

Versione	Changes
Versione 1.3	Added subsection 2.2.5 tariff_category block to booker_init result and the tariff_category_id to the booker_quotes parameters
Versione 1.2	API URL update
Versione 1.1	Added details of the booker_submit command
Versione 1.0	Initial document

Table of Contents

1. Introduction 4

2. Commands 5

2.1. booker_init 5

2.1.1. base 5

2.1.2. country 6

2.1.3. user 6

2.1.4. acriss_code 6

2.1.5. tariff_category 6

2.2. booker_quotes 7

2.2.1. acriss_code 8

2.2.2. tariffs 8

2.2.3. insurances 10

2.2.4. accessory 11

2.3. booker_verify 11

2.3.1. quote 11

2.3.2. insurance 12

2.3.3. extras 12

2.3.4. accessory_quotes 12

2.3.5. info 12

2.4. booker_submit_quote 12

2.5. booker_submit 13

1. Introduction

This document contains technical information of the Booking Engine API that is part of the software packet "Car Rental Software" (CRS) developed by My Appy. The API allows authenticated entities to perform bookings for a specific CRS client. The interface exposes its services via the URL:

```
https://carrentalsoftware.myappy.it/web/gateway.php
```

Communication between the caller and gateway works via simple HTTP POST requests to the given URL using the commands described in this document.

Every API call has to be authenticated in two ways:

(a) The CRS user id has to be given as a GET parameter "uuid"

(b) An API key has to be given as a HTTP header field called "Crs-Auth".

In case one of the fields are missing or incorrect, the API call will fail.

Upon a successful request to the API, the requested data can be read from the server’s response. The data will be encoded in the JSON format and contains the following fields:

```
{ "success" : BOOLEAN - true/false
  "error" : STRING - error description
  "key" : value}
```

The "success" field indicates whether the request has been processed successfully (true) or not (false). The "error" field will only be present if an error occurred during the elaboration of the request and contains the error message, respectively. Any other key being present in the server's response contains (some of) the data requested by the caller. These keys are specified for each of the commands described below.

While no special requirements are defined for numeric, string or boolean values (if not specified otherwise), please note that date and time values have to be given in the format

```
YYYY-MM-DD hh:mm:ss
```

specifying the date and time in UTC and NOT in the local timezone. Currency values instead are given in EUR and always taxes excluded.

2. Commands

In this section the various commands of the CRSBooker interface are described. The name of the command has to be specified as another GET parameter that is appended to the API's URL, e.g.

```
https://carrentalsoftware.myappy.it/web/gateway.php?uuid=user_uid&cmd=command
```

In case an invalid command identifier is given for a request, the server will respond with the error code #10000. In order to perform bookings via the CRS Booker interface, it is recommended to perform queries to the system in the same order in which the commands are listed here to ensure the correct functioning of the communication.

2.1. booker_init

The booker_init command has to be used to retrieve general information of the booking engine for the given user, such as a list of available bases and vehicles. The server response for a successful request will contain the following fields as described in this section.

```
{ "success" : true
  "base" : […]
  "country" : […]
  "user" : {…}
  "acriss_code" : […]
  "tariff_category" : […] }
```

2.1.1. base

A base identifies a pick-up or drop-off point for the vehicles, i.e. physical locations where clients can pick up and return the rented vehicle. The list of bases is returned as a JSON array identified by the key "base" of the main JSON object, with each base having the following information:

```
{ "uid" : STRING - unique base identifier
  "active" : NUMERIC - indicates whether the base is active (1) or not (0)
  "address" : MULTILINE STRING - address and other useful information about the base
  "code" : STRING - short base identifier
  "hours_monday" : STRING - opening hours for Mondays
  "hours_tuesday" : STRING - opening hours for Tuesdays
  "hours_wednesday" : STRING - opening hours for Wednesdays
  "hours_thursday" : STRING - opening hours for Thursdays
  "hours_friday" : STRING - opening hours for Fridays
  "hours_saturday" : STRING - opening hours for Saturdays
  "hours_sunday" : STRING - opening hours for Sundays
  "locality" : STRING - additional information about the base's location
  "name" : STRING - the base name }
```

While most of the fields should be intuitive a little note on the "hours_" fields. These fields are used to describe the opening hours of a specific base for a specific day of the week. If the field is not set or if it is empty the base has no time restrictions and is operative the whole day (from midnight to midnight). A single dash ("-") as a value indicates that a base is closed for the whole day (from midnight to midnight). If a base has special operating times other than the whole day, these times can be specified as one or more time periods in the format "FROM_TIME - TO_TIME", each on a single line, e.g. a value of

```
{ "hours_monday" : "08:00-13:00\n15:00-19:00" }
```

means that the base is open on Mondays from 8am to 1pm and from 3pm to 7pm.

2.1.2. country

The second JSON array given in the server's response of the "booker_init" command contains a list of countries registered with the system. These are not related in any way to the bases or the country that the actual CRS user is operating in. It rather gives you the names and unique identifiers of all the countries of which one has to be chosen at a later step when submitting the client information for a rental quote or reservation to the server. The list of countries is identified by the key "country" of the main JSON object, with each country having the following information:

```
{ "uid" : STRING - unique country identifier
  "code" : STRING - the country's ISO two-letter ISO code
  "name" : STRING - the country's name
  "nationality" : STRING - letter indicating whether a country is part of the European
                           Union (C) or not (E) }
```

2.1.3. user

The last key "user" gives you another JSON object containing information about the CRS user. These have to be visualized and used during the booking process. In detail, the user object contains the following keys:

```
{ "privacy_file_url" : STRING - URL of the privacy policy document
  "terms_file_url" : STRING - URL of the terms and conditions
  "vat_percent" : NUMERIC - floating point value indicating the default VAT for the user
}
```

The two URLs given in the "user" object are links to the privacy statement and the terms and conditions of the CRS user. These documents have to be visualized by the client during the booking process as they have to be read and accepted before submitting a booking. The booking engine will not ensure that these documents have been read, understood and accepted by the client. It is thus the responsibility of the API user to ensure that the client has been informed.

The "vat_percent" value gives you a numeric factor that let's you calculate the booking price including the value added tax, i.e. if the default VAT percentage for bookings of a user is at 22%, this value will be set to 1.22, e.g. if a booking is indicated as 30€, the price including VAT will be 30€ x 1.22 = 36.60€.

2.1.4. acriss_code

The "acriss_code" key of the main response object of the "booker_init" command contains a list of groups of acriss codes together with a name associated to the group. We are working with groups of acriss codes here to allow to get quotes for more than one acriss code at a time instead of querying the system for each acriss code that you are looking for. Each object of the "acriss_code" array contains the following elements:

```
{ "name" : STRING - name of the group of acriss codes
  "uid" : STRING - comma-separated list of the unique ids of the acriss codes registered
                   with the system }
```

2.1.5. tariff_category

Tariff categories can be used to separate different tariffs (see section 2.2.2) from each other depending on certain parameters. For instance a tariff "Low Season 2018" could be existing in two different versions. One with no kilometres limitation during the rental period and another variant that allows a client to run for example a maximum of 150km per day. These information are stored within the data object (if given) of a tariff category. If no or an empty data object is given, no limitations are defined for the given category. In detail, a category contains the following information:

```
{ "uid" : STRING - unique base identifier
  "name" : STRING - the category name
  "data" :
  {
     "km_max" : NUMERIC - maximum kilometres to be run by a client per day or 0 if not
                          set
     "km_penalty" : CURRENCY - penalty for each kilometer that exceeds the maximum
                               defined for this category and rental period, i.e.
                               days * km_max
  }
}
```

The various CRS tariff categories are being provided within the response object of the booker_init command in order to allow the user to limit further interaction with the booking engine to a single category only if wanted.

2.2. booker_quotes

The "booker_quotes" command is used to retrieve actual quotes for a certain set of input parameters. These parameters have to be given a POST parameter called "input" which has to be a JSON encoded object with the following keys:

```
{ "checkout_base" : STRING - unique identifier of the base where to pick up the vehicle
  "checkin_base" : STRING - unique identifier of the base where to return the vehicle
  "acriss_code_id" : STRING - comma-separated list of the unique ids of the acriss codes
                              as returned by the booker_init command to identify the kind
                              of vehicles for which you're looking for quotes
  "checkout_time" : TIMESTAMP - pick-up date and time
  "checkin_time" : TIMESTAMP - return date and time
  "tariff_category_id" : STRING - unique identifier of the tariff category that the
                                  search should be limited to or null/not set if not
                                  needed }
```

Please note that except the tariff_category_id, all parameters listed above are required for the booker_quotes command. The tariff_category_id can be left out if you are looking for quotes of all available tariffs no matter what category they belong to.

Once processed by the system, the server will respond with a list "quotes" of quotes for the requested vehicle types and a second list "accessory_quotes" with the quotes for available extras (such as GPS, baby seat, etc.) matching the given input parameters. These lists will be given as JSON encoded arrays identified by the key "quotes" and "accessory_quotes" which will be child elements of the object identified by the key "result" of the main response object, i.e.

```
{ "success" : true,
  "result" :
  {
     "quotes" :
     [
        {
           "checkout_time" : TIMESTAMP - pick-up date and time as given as input
           "checkin_time" : TIMESTAMP - return date and time as given as input
           "acriss_code" : {…}
           "tariffs" : […]
           "insurances" : […]
        }
        …
     ]
     "accessory_quotes" :
     [
        {
           "checkout_time" : TIMESTAMP - pick-up date and time as given as input
           "checkin_time" : TIMESTAMP - return date and time as given as input
           "accessory" : {…}
           "tariffs" : […]
           "max_rate" : CURRENCY - maximum rental fee applicable for this extra
           "rate" : CURRENCY - rental fee for this extra for the given rental period
        }
        …
     ]
  } }
```

Each of the result's list item for both, the "quotes" and the "accessory_quotes" lists will have the values "checkout_time" and "checkin_time" to confirm the values passed to the system as input parameters. For the extra quotes' list items two more simple values are given: The "max_rate" field indicates the maximum rate that is applicable for a certain extra for the rental period. The "rate" field instead indicates the actual rental fee calculated by the booking system. The other fields, "acriss_code", "tariffs", "insurances" and "accessory" are more complex types and are described in the following subsections.

2.2.1. acriss_code

The "acriss_code" object for a specific vehicle quote returned by the server provides information about the vehicle type considered for the quote, respectively. While you can request quotes for various acriss codes via the"booker_quotes" command, every quote returned by the server only applies to one single acriss code which is described by the following parameters:

```
{ "uid" : STRING - unique identifier of the acriss code
  "code" : STRING - actual 4-letter acriss code as defined by the Association of Car
                    Rental Industry Systems Standards(even though variations are
                    possible)
  "name" : STRING - human-readable name and description of the code }
```

2.2.2. tariffs

The "tariffs" key of each quote (vehicle quote and extra quote) contains a list of objects describing the various tariffs that have been considered for the quote. Since CRS tariffs have validity periods that may overlap with other tariffs, it might be necessary to consider more than one tariff for the calculation of a quote.

Example: Consider a tariff T1 that is valid for the whole month of December and another tariff T2 that is valid only for the Christmas holidays from Dec 24 to Dec 27. If you ask for a quote for a 10-days rental from Dec 20 to Dec 30, the system will consider both tariffs, T1 and T2, for the calculation of the rental fees. More precisely, it will split the whole rental period into sub periods and apply the various tariffs to these to get the rental fees for every single subperiod. At the end it sums up the fees for the subperiods to get the full amount. For the given example, the system would apply the tariffs as follows:

(a) Dec 20 - Dec 23: apply tariff T1 for four days

(b) Dec 24 - Dec 27: apply tariff T2 for four days

(c) Dec 28 - Dec 30: apply tariff T1 for two days

In this way it is easy to see why the server provides a list of tariffs for each quote. Each of the list's item describes the sub period and the applied tariff and is defined by the following parameters:

```
{ "date_start" : TIMESTAMP - date and time of the beginning of the sub period for which
                             the tariff was applied
  "date_end" : TIMESTAMP - date and time of the end of the sub period for which the
                           tariff was applied
  "days" : NUMERIC - number of days for which the tariff was applied
  "tariff" :
  {
     "uid" : NUMERIC - unique identifier of the tariff
     "name" : STRING - tariff name
     "date_start" : TIMESTAMP - date and time of the beginning of the tariff's validity
                                period
     "date_end" : TIMESTAMP - date and time of the end of the tariff's validity period
     "tariff_category" :
     {
        "uid" : STRING - unique identifier of the tariff category
        "name" : STRING - name of the tariff category
        "data" : STRING - JSON encoded information about the tariff category
     }
     "group" :
     {
        "uid" : STRING - unique identifier of the tariff's group
        "name" : STRING - name of the tariff's group
        "json_data" : STRING - JSON encoded information about the tariff group }
     }
  }
  "vn" : NUMERIC - number of active vehicles being assigned to the acriss code
  "vb" : NUMERIC - number of vehicles already blocked in this period
  "vab" : NUMERIC - number of active vehicles that are available for the whole rental
                    period and located at the pick-up base at the time of the pick-up
  "va" : NUMERIC - number of active vehicles that are available for the whole rental
                   period (whether or not located at the pick-up base)
  "rate" : CURRENCY - the calculated rental fees for the given sub period and tariff }
```

As you can see, the server provides the necessary details for each sub period, such as it's start date and time, end date and time, the applied tariff and the actual rate or rental fees calculated for the sub period, respectively. In addition, information are given about how many vehicles (in general and for the actual acriss code) are available for the chosen rental period. Please note at this point that it is not possible to make a reservation for an acriss code for which no vehicles are available any more, i.e. "va" is equal zero.

Each CRS tariff is divided into one or more groups that contain the actual pricing information and to which are assigned the various acriss codes. For example, a tariff "Low Season 2018" might contain a group "Group A" to indicate the prices for motorcycles, "Group B" to indicate small and medium-sized vehicles, ecc. Hence, for each tariff that is applied to a sub period of the whole rental period, only one group of the tariff is actually considered, i.e. the group that has the quote's acriss code applied to it.

As seen above, a group has a unique identifier a name and "json_data" object that contains the actual pricing information. The "json_data" object is a JSON encoded string and has thus to be decoded separately in case you want to retrieve more information about it. Thus after decoding the string, you'll get something like

```
"json_data" :
{
  …
  "image_url" : STRING - URL of the image of a car/group of cars/ecc representing
                         this group
  …
}
```

Since the system already provides the rate calculated for the given sub period and most of the information of the group's "json_data" object depend on the actual tariff, no information are provided about these information in this place. The only thing worth mentioning is the presence of an "image_url" field that contains the URL of an image that can be shown to visualise an example of a vehicle of the actual group.

Apart from the different groups that are assigned to a tariff, the tariff itself might also be assigned to a tariff category as described in section 2.1.5. If this is the case, the "tariff_category" item provides further information about this category. It contains the same information as already provided in the result of the booker_quotes command, except that the data object is given as a JSON encoded string. If necessary decode the string and you will see the elements that have been described earlier.

Turning back to the discussion about multiple tariffs being applied to a vehicle quote, please note that in order to calculate the full amount of rental fees for the specified vehicle type, it is necessary to sum up all the rates of the specified tariffs for the listed sub periods. As mentioned in the introduction, all currency values provided by the booking engine are specified in EUR and taxes excluded. In order to comunicate the correct price to the client, it is thus necessary to multiply the currency values with the "vat_percent" value from the "booker_init" response. In case a client needs a different VAT percentage applied to his booking, it might be more appropriate to book directly with the CRS user or to get in contact with the CRS user after performing the booking.

2.2.3. insurances

For each vehicle type (identified by an acriss code) for which a quote was returned by the server, one or more insurance options may exist at different rates. For example there might be an insurance option with a basic coverage and high excess rates at a lower cost or for free and other options with a wider coverage and lower excess rates at a higher cost. All of the available options are listed in the insurances array for a single quote and contains the following information:

```
{ "insurance_class" :
  {
     "uid" : STRING - unique identifier of the insurance class
     "excess" : CURRENCY - the deposit that the rental company usually asks the client
                           to leave when picking up the car
     "excess_damage" : CURRENCY - the excess to be paid by the client in case of any
                                  damages caused during the rental period
     "excess_theft" : CURRENCY - the excess to be paid by the client in case of theft
                                 of the vehicle during the rental period
```

```
    "excess_info" : STRING - textual description of the various excess rates
    "name" : STRING - insurance class name
  }
  "tariffs" : […]
  "rate" : CURRENCY - price of the insurance option }
```

The calculation of the price of an insurance option depends on the same principle of concatenation of various tariffs similar to the calculation of the actual rental fees. As is the case the rental fees, also here, the system provides the whole rate for a specific insurance options and thus no other information are given about the information about the insurance class tariffs list at this place.

2.2.4. accessory

The "accessory" field of extra quotes provide further information about a certain extra that can be booked together with a vehicle. It contains the following fields:

```
{ "uid" : STRING - unique extra identifier
  "name" : STRING - the extra name }
```

2.3. booker_verify

The "booker_verify" command should be used to validate the input parameters that you are going to send to the booking engine to perform an actual booking. Although the same validation will take place during the placement of a booking this command allows you to pre-validate all the entered information in order to avoid later problems, e.g. with cancelled payments, etc. due to wrong input data during the booking process. If the validation succeeds, the booking engine will simply reply with a

```
{ "success" : true }
```

Otherwise, if an error occurred, also the error field will be set appropriately.

The "booker_verify" command takes one POST argument called "input" containing the following information in a JSON encoded string:

```
{ "quote" : {…}
  "insurance" : {…}
  "extras" : […]
  "accessory_quotes" : […]
  "info" : {…}
  "checkout_base" : STRING - unique identifier of the base where to pick up the vehicle
  "checkin_base" : STRING - unique identifier of the base where to return the vehicle
  "checkout_time" : TIMESTAMP - pick-up date and time
  "checkin_time" : TIMESTAMP - return date and time }
```

The fields "checkout_base", "checkin_base", "checkout_time" and "checkin_time" are the same fields that were submitted for the "booker_quotes" command and do not need any further explanation. The other fields are described in the following subsections.

2.3.1. quote

The "quote" field specifies the selected quote for the booking engine. The object has to be one of the quotes from the "quotes" list of the response from the "booker_quotes" command. If the object is modified before being submitted to the booking engine, the booking will result in failure.

2.3.2. insurance

The "insurance" field tells the booking engine which insurance option should be included in the booking. If specified, the object has to be one of the insurance objects from the "insurances" list of the selected quote of the "booker_quotes" response. If the object is modified before being submitted to the booking engine, the booking will result in failure.

2.3.3. extras

The "extras" field expects an array containing the unique identifiers of the extras that should be included in the booking.

2.3.4. accessory_quotes

The "accessory_quotes" list should contain all the extras whose unique identifiers have been listed in the "extras" list from above. It can thus be considered a copy of the "accessory_quotes" of the response of the "booker_quotes" command, except that non-selected extras can be omitted.

2.3.5. info

The "info" object is used to communicate the personal information about the client that the booking is made for to the booking engine and contains the following fields:

```
{ "name" : STRING - name of the client
  "surname" : STRING - surname of the client
  "phone" : STRING - phone number of the client
  "email" : STRING - email address of the client
  "address" : STRING - address of the client
  "city" : STRING - city/municipality where the client resides
  "zip_code" : STRING - zip code of the city where the client resides
  "country_id" : NUMERIC - unique identifier of the country where the client resides
  "voucher" : STRING - optional field that allows the API user to communicate the number
                       of a voucher, etc. CRS user }
```

All fields, except the "voucher" field are obligatory and must not be left out. The "country_id" field must be set to the unique identifier of one of the countries that were listed in the "booker_init" response.

2.4. booker_submit_quote

The command "booker_submit_quote" can be used to confirm a quote requested via the API before. The command expects the same input as is the case for "booker_verify". Upon successful validation of the input, a new record will be created for the entered client information and a quote with the presented information will be saved to the records of the CRS user. Finally, the client as well as the CRS user will be informed via email about the newly created quote. Quotes that are created in this way usually have a validity of one week. Hence the presented rental fee will be blocked and can be accepted only for this period of time.

Please note, that while the quote document sent to the client will contain information about all the insurance options and extras without highlighting the selected items (because a quote is not yet a fully confirmed booking and is supposed to inform about alternatives as well), Car Rental Software will indeed highlight the selected items for its operators. No selection will thus be lost.

In case the quote could be saved successfully, the server will provide the number of the quote as well as its unique identifier and the unique identifier of the newly created client record for you. More precisely, the response of the server will look like this:

```
{ "success" : true
  "result" :
  {
     "client_reservation":
     {
        "uid" : STRING - the unique identifier of the newly created quote
        "ref_id_quote" : STRING - humans readable number of the new quote
     }
     "client":
     {
        "uid": STRING - the unique identifier of the newly created client record
     }
  } }
```

2.5. booker_submit

Other than the "booker_submit_quote" command, the "booker_submit" command performs an actual booking instead of just saving a selected quote. In this case a vehicle of the selected type will be blocked to guarantee its availability for the client's booking. The command expects almost the same input as is the case for "booker_verify":

```
{ "quote" : {…}
  "insurance" : {…}
  "extras" : […]
  "accessory_quotes" : […]
  "info" : {…}
  "payment" :
  {
     "broker_id" : STRING - unique identifier of the broker performing the booking
     "voucher_no" : STRING - unique identifier of the voucher identifying the payment
     "confirmation_no" : STRING - confirmation number of the payment
  }
  "checkout_base" : STRING - unique identifier of the base where to pick up the vehicle
  "checkin_base" : STRING - unique identifier of the base where to return the vehicle
  "checkout_time" : TIMESTAMP - pick-up date and time
  "checkin_time" : TIMESTAMP - return date and time }
```

As can be seen, the parameters are the same like before, except the "payment" block. In fact, in order to block a vehicle for the client and thus to perform an actual booking, a payment has to be registered. Usually, this payment is received by the appropriate broker that is registering the reservation. Hence, the broker's unique identifier (can be found on the broker's page of the CRS control panel), the voucher number and the number of the payment confirmation has to be given.

Upon a successful elaboration of the request, the server's response will be similar to the response of the "booker_submit_quote" command:

```
{ "success" : true
  "result" :
  {
     "client_reservation":
     {
        "uid" : STRING - the unique identifier of the newly created quote
        "ref_id_reservation" : STRING - humans readable number of the new reservation
     }
     "client":
     {
        "uid": STRING - the unique identifier of the newly created client record
     }
  } }
```

The only difference is that instead of getting the number of the new quote, this time, the number of the new reservation is returned.