Mobile API Quick Start Guide

> ⚠ **Work in Progress**
> This page is a work in progress, for the time being you should use this in conjunction with the included documentation.

The mobile API can perform actions similar to the current desktop and mobile reservation site. It can log a member into the system, check vehicle availability, make reservations, and change reservations. The API contains many functions that may be confusing to those who are not familiar with the Autovera system, this guide distills the important calls, information on how to communicate with the server, and provides a potential workflow for designing a replacement or supplement to the reservation site.

Accessing the API

The URL for the API is CSO specific, however the format is the same for everyone.

URL

```
https://[CSO reservation site]/webservices/index.php/WSUser/WSDocumentation
```

URL - Example

```
https://demo.autovera.com/webservices/index.php/WSUser/WSDocumentation
```

All requests to the API must be in the form of a HTTP GET or POST with the arguments as a query string (See below).

Calling and Authenticating with the API

Every request to the API must contain the following pieces of data in the form of a query string. This section lists every key that is required, an explanation, how to format the value if necessary, and an example. This section also explains how to send function specific arguments. Afterwards is an example query using the API.

Required Keys

action

Description: The API function you would like to call.

Example: `makeReservationGetReservation`

billcode

Description: This is used to determine the source of the request, one of the following should be used

- ivr: This reservation was done from an IVR/phone based reservation system
- mobile: A generic mobile site
- iphone: An iPhone app/site
- android: An Android app/site
- service: A generic service request

Example: `android`

hash

Description: A sha1 hash comprising of the following pieces of data

- a sha1 hash of the member's password
- The current time in a Unix timestamp
- The API function you would like to call

The hash should be generated from the above in the form of a single string in that order without any spaces.

Example: Assuming the member's password is `1234`, the current timestamp is `1389798916` and the function you are calling is `makeReservationGetReservation`, the string to hash and the hash produced would be the following

```
String to hash:
7110eda4d09e062aa5e4a390b0a572ac0d2c02201389798916makeReservationGetReservation
Hash: c9465da51afef896dab9860ea0454027c7bc0977
```

time

Description: The current time in the form of a Unix timestamp

Example: `1389798916`

user

Description: The member ID.

Example: `999`

Function Specific Arguments

Many functions in the API require additional arguments to return valid data, these are also passed as part of the GET or POST request in the query string. The name of the attribute is the name of the variable in the documentation. Variables that are objects should be treated as a key value array, the available keys for these objects can be found in the documentation.

Example: Using the function `resultsFromStackFilter` to return available vehicles. This function requires a boolean (`includeStack`), and a "WSStackFilter" (`aStackFilter`). As a result the following will be appended to the query string.

```
includeStack=true&aStackFilter[startTime]=1389886200&aStackFilter[endTime]
=1389888000
```

Example

From the above, the entire query for a `resultsFromStackFilter` request will be the following. Please note that this call will not actually work as the time is probably expired by the time you read this.

URL

```
https://demo.autovera.com/webservices/WSUser/WSRest?action=resultsFromStackFilter&user
=999&hash=c9465da51afef896dab9860ea0454027c7bc0977&time=1389798916&billcode=android&in
cludeStack=true&aStackFilter[startTime]=1389886200&aStackFilter[endTime]=1389888000
```

Response

The response will always be in XML. The below is a response when the `amenities` function is used.

```
<?xml version="1.0"?>
<methodResponse>
    <WSAmenity>
        <name>air_conditioning</name>
        <description>A/C</description>
    </WSAmenity>
    <WSAmenity>
        <name>dog_free</name>
        <description>Pet Friendly</description>
    </WSAmenity>
    <WSAmenity>
        <name>fold_rear_seat</name>
        <description>Folding Rear Seats</description>
    </WSAmenity>
    <WSAmenity>
        <name>bike_rack</name>
        <description>Bike Rack</description>
    </WSAmenity>
</methodResponse>
```

Responses that are a single value will simply have the value within the `<methodResponse>` tags. Some of the more important responses will be discussed below.

Errors

Errors will always be returned in the following format

```
<?xml version="1.0"?>
<methodResponse>
    <fault>
        <value>
            <struct>
                <member>
                    <name>faultCode</name>
                    <value>
                        <int>0</int>
                    </value>
                </member>
                <member>
                    <name>faultString</name>
                    <value>
                        <string>Authentication Failed!</string>
                    </value>
                </member>
            </struct>
        </value>
    </fault>
</methodResponse>
```

The useful bits in this XML response are the two `member` tags.

Member name	Explanation
faultCode	The error code
faultString	The plain English description of the error

Important Concepts

There are some terminology and concepts in Autovera that are used in the API that may not be easily understood. The explanation below is by no means exhaustive, but does cover every bit that is used for the API.

Optional Adjustments

Description: Optional adjustments are normally used for per reservation damage deductible waiver programs. Each optional adjustment is referred by its ID.

Example: A damage deductible program that costs $1/hour has an ID of 23

Stacks (Stack ID)

Description: Every reservation in Autovera is based off of a stack. In Autovera members do not reserve a vehicle with an id, but rather a stack. A stack is represented by an ID and is simply the representation of a vehicle in a lot at a given time. Once a vehicle is moved from a lot, the stack ID will change.

Example: A Red Honda Civic with the id 5, in lot 10 at Yonge/Eglinton may be given a stack ID of 20.

Timestamp and Reservation Time

Description: Time stamps in Autovera are always in a Unix timestamp. When searching or making reservations, the timestamp must be divisible by the trip time resolution. The trip time resolution is the interval of times that trips can start or end on. Usually this is either 15 or 30 minutes.

Get Writing and Useful API Functions

The API contains many convenience functions that may be intimidating to one new to the API. Below are the most functions that you will most likely end up using in your application. This documentation is sorted by workflow, you should do the following when starting out with the API. The workflow is essentially as follows:

1. Logging in with `isLoggedIn`
2. Searching for availability with `resultsFromStackFilter`
3. Make a reservation using `makeReservationWithOptionalAdjustmentIdsGetReservation`

isLoggedIn()

If the supplied username and password is correct, true is returned. Otherwise an error will be returned.

This function does not actually "log in" the user since every request does require the same authentication information, but this function is useful to ensure that the username and password is correct and your communication methods are correct.

resultsFromStackFilter(aStackFilter, includeStack)

aStackFilter
Type: WSStackFilter

A list of filters to apply when searching for vehicles. A WSStackFilter is simply an array with the key as one of the options. When including this parameter as part of an HTTP GET/POST request, it should be formatted like below:

```
aStackFilter[optionA]=false&aStackFilter[optionB]=123
```

Possible filter options are

Key	Type	Description
latitude	Float	Latitude
longitude	Float	Longitude
vehicleTypeIds	Integer array	A list of vehicle type IDs
amenities	WSAmenity array	A list of amenities
startTime	Integer	The start time of the reservation in a Unix timestamp
endTime	Integer	The end time of the reservation in a Unix timestamp
timeIsFlexable	boolean	Set whether or not the start/end time can be modified slightly
showNonMatches	boolean	Set whether or not unavailable vehicles will be returned
marketId	Integer	The market id you want to restrict results for

includeStack
Type: boolean

Should a DBEntityStack be returned with the search results, this is generally a good thing

Returns a list of availability for vehicles matching the search filters. Results will be sorted by best match. A maximum of 30 results will be returned.

Response
A DBRankedStacks

makeReservationWithOptionalAdjustmentIdsGetReservation(stackId, startTime, endTime, optionalAdjustmentIds, memo)

stackId
Type: integer

The stack ID that you want to get an estimate of.

startTime
Type: integer

The Unix timestamp for the start of the hypothetical trip. This timestamp must be divisible by the trip time resolution.

endTime
Type: integer

The Unix timestamp for the end of the hypotehtical trip. The timestamp must be divisible by the trip time resolution.

optionalAdjustmentIds
Type: integer array
Optional

An array of optional adjustments that will be considered for this estimate

memo
Type: string
Optional

A note to the driver or member about this reservation.

Returns a string that contains the estimate for this trip. This function should be used rather than the other make reservation functions. This value is in a string as the currency unit is included in the request.

Response
A DBEntityReservation

Related functions
makeReservationGetReservation

tripEstimateAndOptionalAdjustments(stackId,startTime,endTime,optionalAdjustmentIds)

stackId
Type: integer

The stack ID that you want to get an estimate of.

startTime
Type: integer

The Unix timestamp for the start of the hypothetical trip. This timestamp must be divisible by the trip time resolution.

endTime
Type: integer

The Unix timestamp for the end of the hypotehtical trip. The timestamp must be divisible by the trip time resolution.

optionalAdjustmentIds
Type: integer array

An array of optional adjustments that will be considered for this estimate

Returns a string that contains the estimate for this trip. This value is in a string as the currency unit is included in the request.

Example

Here's some example PHP code, our new top secret Textmode Autovera UI. This example does the following:

- Search availability
- Create reservation
- Virtual Swipe

Please note that this code is NOT production ready, and should most definitely not be used in production apps. If you agree, click the link to the right I will not use this code in production systems!