

**OMNICOMM**

# Omnicom Online

Integration Manual  
26.11.2018

# Contents

## 4 General Information

## 4 Main Features

## 5 Data Downloading Features

## 6 Connection

## 6 Demo access to the web-services

## 6 Description

### 6 SOAP Methods List

6       signIn – authorization

7       getObjectSet – list of objects

8       getSmoothedFuel – smoothed fuel level for the period

9       getFuelConsumption – fuel consumption for the period

9       getEvents – list of events

11      getMileage – mileage for the period

11      getEngineOnTime – engine running time for the period

12      getVehiclesState – VH current status

12      getMileageSpeedExcess – mileage with excess speeding for the period

13      getMovementTime – movement time for the period

14      getEngineOnTimeInMovement – engine running time in movement for the period

14      getEngineOnTimeWithoutMovement – engine running time without movement for the period

16      getEngineOffTime – engine shutdown time for the period

16      getFuelConsumptionInMovement – fuel consumption during movement for the period

17      getFuelConsumptionWithoutMovement – fuel consumption without movement for the period

18       getFuelConsumptionInMotohour – fuel consumption per motor hour  
18       getFuelAtTime – fuel level at a given moment  
20       getUserNotificationsByPeriod – user notifications by the period  
21       getVisitedGeozonesByPeriod – geofences visited by the period  
21       getVehiclesParams – list of parameters available to the user  
22       signOut – session termination  
22       getActiveNotificationRules – profiles of active notifications  
23       setDeviceIdToNotificationRules – assignment of notification profiles to VH  
24       getFuelLevelsByTimeMoment – fuel level at a certain time  
25       getFuelLevelsByPeriod – fuel level by the period of time  
26       getSmoothedFuelLevelsByPeriod – smoothed fuel levels by the period of time  
27       getRefuelingsAndDrainsByPeriod – fuel draining/refueling operations by the period of time  
27       getVehiclesProfiles – VH profiles matching the VH identifiers  
28       getCurrentObjectState – vehicle current status  
29       getReportData – report for auxiliary equipment over the period, TPMS, IQFreeze  
30       getSEOnTime – auxiliary equipment running time for the period  
30       getStatisticsByPeriod – statistics for the period  
32       getTracksByPeriod – VH track for the period  
33       getTrack – track

## 34   REST API Methods List

34       /api/service/geozones/geozones GET – acquisition of data on user geofences  
36       /api/service/geozones/geozone-groups GET – acquisition of data on all user geofence groups

## 36   Errors

## 37   Types of Events

## 40   Work Client Example

# Omnicom Online

Omnicom Online allows the user to control the operation of vehicles and drivers by means of reports being part of it. To access Omnicomm Online only a personal computer is required, connected to the Internet.

Omnicom Online has built-in special-purpose tools to collect the processed data and use them in the accounting documents and in the fleet monitoring systems.

This manual describes the built-in tools operation and the integration with third-party systems.

## General Information

The integration from Omnicomm Online is used to extend functionality and data entry automation from third-party systems.

The data transmission is performed using an open application communication standard – SOAP-based web services. This protocol and the corresponding communication module allow users to collect data from Omnicomm Online across-the-wire.

The Omnicomm web-service communication module should be developed and deployed by the third-party accounting system implementers.

Omnicom does not undertake to delegate specialists or to develop the communication module.

## Main Features

Data, downloaded from the Omnicomm Online system, allows, at the initiation of appropriate functionality of third-party systems, the following tasks to be performed automatically:

- Entering data on mileage and fuel consumption in the waybills
- Accounting for mileage and usage time for calculating the amount of work, driver's salary, etc.
- Accounting for attachable equipment to calculate scope of performed work
- Accounting for mileage, engine hours and auxiliary equipment for maintenance planning
- Write-off of fuel consumed

## Data Downloading Features

- Comparison of documented refills with the actual ones to prevent theft of fuel
- Generation of violation reports
- Plan vs. actual analysis on work done, fuel used, etc.
- Use of the current location to select suitable vehicle to fulfill the order
- Track visualization on the map in a third-party software

Various reports can be generated using the vehicle parameters downloaded from Omnicomm Online.

Data from Omnicomm Online enable quick estimation of the accepted planning system correctness and effective dealing with all the possible machinations of fleet employees, associated with theft of petroleum products and unauthorized use of vehicles.

The main targets for the integration of the systems are accounting systems, ERP-systems, and branch accounting systems.

## Data Downloading Features

Web services provide for retrieving the following parameters:

- Mileage
- Fuel consumption
- Engine operation time
- Auxiliary equipment operation time
- Fuel volume graph
- Mileage with exceeded speed\*
- Movement duration\*
- Engine operation time in motion\*
- Engine operation without motion\*
- "Engine Off" time\*
- Fuel level at a given moment\*
- Fuel consumption in motion\*
- Fuel consumption without motion\*

## Connection

- Fuel consumption rate per one hour of engine operation\*
- Track\*
- Actual location\*

\* – Parameters marked are available from version 2.4.2.

In addition to these parameters, it is possible to receive all the events displayed in the “Events” report in Omnicomm Online.

## Connection

To obtain the connection address for web services, the client should contact Omnicomm’s Technological Support Team.

Connection to the web-service is performed by the Technical Support Team Ph. 8 800 100-24-42, ext.5

## Demo access to the web-services

If necessary, to test the connection to web-services (to verify the application without using a real Omnicomm Online account, or if there is doubt about the network settings), one can use the web-services demo server of Omnicomm.

Web services connection address (not for 1C system):

<http://demo.omnicomm.ru:8000/AnalyticalServer/v2/ws?wsdl>

Web services connection address for 1C system:

<http://demo.omnicomm.ru:8001/AnalyticalServer/v2/ws?wsdl>

Login: rudemoru Password: rudemo123456

## Description

### SOAP Methods List

For date and time data transmitting the UNIXTIME (in seconds) format is used. Units of other parameters are listed below.

signIn – authorization

## Connection

### Input Values

**String login** – user name in the system

**String password** – password in the system

### Returned Values

Boolean **status** – true/false true in case of successful authorization

String **sessionId** – in case of successful authentication, the session identifier (minimum 16 characters)

Unixtimestamp **dateTimeEnd** – in case of successful authorization, time of the session termination (the time after which you must log in again)

String **error** – error message in case of improper authorization (incorrectly entered username and password, or incorrect data format)

## getObjectSet – list of objects

### Input Values

String **sessionId** – session ID obtained during authorization

## Connection

### Returned Values

Boolean **status** – true/false true in case of successful operation.  
String **error** – error message in case an error occur  
Dataset **objects** – list of vehicles available to the user — owner of the registered session (meaning only those objects which the user has the right to view):  
Integer **id** – object identifier (the same as the identifier of the unit)  
String **objectName** – name of vehicle  
String **objectType** – type of vehicle  
String **GarageNumber** – garage number

getSmoothedFuel – smoothed fuel level for the period

### Input Values

String **sessionId** – session identifier obtained during authorization  
Integer **objectId** – identifier of vehicle / unit  
Unixtimestamp **timeBegin** (seconds) – start time of the interval  
Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** – true/false true in case of successful operation  
Unixtimestamp **timeBegin** (seconds) – start time of the interval  
Unixtimestamp **timeEnd** (seconds) – end time of the interval  
Dataset **fuel** – data set (all the data from the archive for the selected period):  
Unixtimestamp **timeStamp** – time of the registered fuel level  
Double **smoothedFuel** – smoothed value of the fuel, liters, accuracy up to 0.1 l  
String **error** – error message in case an error occur



## Connection

getFuelConsumption – fuel consumption for the period

### Input Values

String **sessionId** - session identifier obtained during authorization

Integer **objectId** - identifier of vehicle / unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) - end time of the interval

### Returned Values

Boolean **status** - true/false true in case of successful operation.

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **fuelConsumption** – fuel consumption for the period, liters, accuracy up to 0.1 l

String **error** – error message in case an error occur

getEvents – list of events

### Input Values

**String sessionId** – session identifier obtained during authorization

**Integer objectId** – identifier of vehicle/unit. Optional parameter, if the identifier is not present, it returns the data for all vehicles.

**Integer type** – type of event, required.

**Unixtimestamp timeBegin** (seconds) – start time of the interval

**Unixtimestamp timeEnd** (seconds) - end time of the interval

## Connection

### Returned Values

Boolean **status** – true/false true in case of successful operation.

String **error** - error message in case an error occur

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Dataset **objectEvents** – data set for each event:

Unixtimestamp **timeStamp** – date and time of the event

Integer **objectId** – identifier of vehicle/ unit

String **type** – type of event

String **parameters** – parameters of the event

String **eventAddress** – address of the event, if available

String **iButton** – iButton code in HEX. Only for events such as 'Driver'

String **name** – name of geofence. Only for entry and exit to or from Geofence events

## Connection

getMileage – mileage for the period

### Input Values

String **sessionId** – session identifier during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) - end time of the interval

### Returned Values

Boolean **status** -- true/false true in case of successful operation.

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **Mileage** – mileage in km for the specified interval, accuracy 0.1 km

String **error** – error message in case an error occur

getEngineOnTime – engine running time for the period

### Input Values

String **sessionId** - session identifier obtained during authorization

Integer **objectId** - identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) - end time of the interval

## Connection

### Returned Values

Boolean **status** - true/false true in case of successful operation  
Unixtimestamp **timeBegin** (seconds) – start time of the interval  
Unixtimestamp **timeEnd** (seconds) – end time of the interval  
Double **engineOnTime** – total running time of the engine, in seconds  
String **error** – error message in case an error occur

## getVehiclesState – VH current status

### Input Values

String **sessionId** – session identifier obtained during authorization  
VehiclesType **vehicles** – list of VH IDs

### Returned Values

Boolean **status** – operation status  
String **error** – error message in case an error occur  
vehicleStatesType **states** – list of parameters describing each VH status

## getMileageSpeedExcess – mileage with excess speeding for the period

### Input Values

## Connection

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** - true/false true in case of successful operation.

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **mileageSpeedExcess** – mileage with speeding in km for the specified interval, accuracy 0.1 km

String **error** – error message in case an error occur

## getMovementTime – movement time for the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – the identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

## Connection

### Returned Values

Boolean **status** - true/false true in case of successful operation.  
Unixtimestamp **timeBegin** (seconds) – start time of the interval  
Unixtimestamp **timeEnd** (seconds) – end time of the interval  
Double **movementTime** – movement time, seconds  
String **error** – error message in case an error occur

getEngineOnTimeInMovement – engine running time in movement for the period

### Input Values

String **sessionId** – session identifier obtained during authorization  
Integer **objectId** – identifier of vehicle/unit  
Unixtimestamp **timeBegin** (seconds) – start time of the interval  
Unixtimestamp **timeEnd (seconds)** – end time of the interval

### Returned Values

Boolean **status** - true/false true in case of successful operation  
Unixtimestamp **timeBegin** (seconds) – start time of the interval  
Unixtimestamp **timeEnd** (seconds) – end time of the interval  
Double **engineOnTimeInMovement** – engine running time during movement, seconds  
String **error** – error message in case an error occur

getEngineOnTimeWithoutMovement – engine running time without movement for the period

## Connection

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** – true/false true in case of successful operation

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **engineOnTimeWithoutMovement** – engine running time without movement, seconds

String **error** – error message in case an error occur

## Connection

getEngineOffTime – engine shutdown time for the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** - true/false true in case of successful operation

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **engineOffTime** – engine shutdown time for the period, seconds

String **error** – error message in case an error occur

getFuelConsumptionInMovement – fuel consumption during movement for the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval



## Connection

### Returned Values

Boolean **status** - true/false true in case of successful operation

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **fuelConsumptionInMovement** – fuel consumption during movement for the period, liters, accuracy 0.1 liters

String **error** – error message when an error occurs

getFuelConsumptionWithoutMovement – fuel consumption without movement for the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** – true/false true in case of successful operation

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **fuelConsumptionWithout Movement** – fuel consumption without movement for the period, litres, accuracy 0.1 l

String **error** – error message when an error occurs

## Connection

getFuelConsumptionInMotohour – fuel consumption per motor hour

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** - true/false true in case of successful operation

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **fuelConsumptionIn Motohour** – average fuel consumption for the engine hour for the period, liters, accuracy 0.1 l

String **error** – error message when an error occurs

getFuelAtTime – fuel level at a given moment

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **time** (seconds) – moment of time

## Connection

### Returned Values

Boolean **status** - true/false true in case of successful operation

Unixtimestamp **time** (seconds) – moment of time

Double **fuelAtTime** – fuel level at a given moment, litres, accuracy 0.1 l

String **error** – error message in case an error occur

## Connection

getUserNotificationsByPeriod – user notifications by the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds. If not defined, end time = system time of the request execution by the server

Integer **page** – requested page number. If not defined, first page containing perPage records is returned

Integer **perPage** – quantity of records per page, if not defined, quantity is not limited

### Returned Values

Boolean **status** – operation status. True in case of successful operation

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** – start time of the interval (UTC) , seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds

userNotificationsType **userNotifications** – array of returning parameters sets

Integer **notificationsCount** – total number of notifications for all pages. If no notification found, error code 10 is returned

## Connection

getVisitedGeozonesByPeriod – geofences visited by the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds

vehicleIdsType **vehicleId** – list of the VH IDs. If absent, all available vehicles are used for request

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds

visitedGeozone **geozoneVisits** – block of arrays of visited geofences parameters

getVehiclesParams – list of parameters available to the user

### Input Values

String **sessionId** – session identifier obtained during authorization

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Vehicle **vehicles** – VH parameters that the current user is entitled to view

## Connection

### signOut – session termination

#### Input Values

String **sessionId** – session identifier obtained during authorization

#### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

String **sessionId** – terminated session ID

### getActiveNotificationRules – profiles of active notifications

#### Input Values

String **sessionId** – session identifier obtained during authorization

#### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

rulesType **rules** – parameters for each notification profile

## Connection

setDeviceIdToNotificationRules – assignment of notification profiles to VH

### Input Values

String **sessionId** – session ID obtained during authorization

String **deviceId** – device ID

String **deviceTypeId** – device type identifier

rulesType **rules** – identifiers of notification profiles to which it is necessary to assign VH

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

## Connection

getFuelLevelsByTimeMoment – fuel level at a certain time

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeMoment** – moment of time (UTC), seconds

vehicleAndTankIdsType

**vehicleAndTankIds** – list of vehicles and fuel tanks IDs.

In the absence of the list, the request will be executed for all vehicles and tanks available to the user

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Unixtimestamp **timeMoment** – moment of time (UTC), seconds

fuelData **fuelDataSet** – list of parameters for each VH:

int **vehicleId** – vehicle identifier;

int **tankNumber** – fuel tank number;

fuelLevelsType **fuelLevels** – fuel level data;

activityPeriodsType **activityPeriods** – engine operation data;

ignitionOffListType **ignitionOffList** – ignition-OFF data;

ignitionOnListType **ignitionOnList** – ignition-ON data;

ItsFailurePeriodsType

**ItsFailurePeriods** – data on fuel level sensor failures



## Connection

getFuelLevelsByPeriod – fuel level by the period of time

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds. If the end time is not indicated, the end time of the period = the system time of the beginning of the Server-side request processing

VehicleAndTankIdsType

**vehicleAndTankIds** – list of vehicles and fuel tanks IDs.

In the absence of the list, the request will be executed for all vehicles and tanks available to the user

Int **reduce** – thinning:

0 = not required

1 = required

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** – start time of the period (UTC), seconds

Unixtimestamp **timeEnd** – end time of the period (UTC), seconds

fuelData **fuelDataSet** – list of parameters for each vehicle:

int **vehicleId** – VH identifier;

int **tankNumber** – fuel tank number;

fuelLevelsType **fuelLevels** – fuel level data;

activityPeriodsType **activityPeriods** – engine operation data;

ignitionOffListType **ignitionOffList** – ignition-OFF data;

ignitionOnListType **ignitionOnList** – ignition-ON data;

ItsFailurePeriodsType

**ItsFailurePeriods** – data on fuel level sensor failures

## Connection

getSmoothedFuelLevelsByPeriod – smoothed fuel levels by the period of time

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – object identifier

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

vehicleAndTankIdsType **vehicleAndTankIds** – list of vehicles and fuel tanks IDs. In the absence of the list, the request will be executed for all vehicles and tanks available to the user

Int **reduce** – thinning:

0 = not required

1 = required

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** – start time of the period (UTC), seconds

Unixtimestamp **timeEnd** – end time of the period (UTC), seconds

fuelData **fuelDataSet** – list of parameters for each VH:

int **vehicleId** – VH identifier;

int **tankNumber** – fuel tank number;

fuelLevelsType **fuelLevels** – fuel level data;

activityPeriodsType **activityPeriods** – engine operation data;

ignitionOffListType **ignitionOffList** – ignition-OFF data;

ignitionOnListType **ignitionOnList** – ignition-ON data;

ItsFailurePeriodsType

**ItsFailurePeriods** – data on fuel level sensor failures

## Connection

getRefuelingsAndDrainsByPeriod – fuel draining/refueling operations by the period of time

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

vehicleAndTankIdsType **vehicleAndTankIds** – list of vehicles and fuel tanks IDs. In the absence of the list, the request will be executed for all vehicles and tanks available to the user

Integer **page** – number of the requested page with data

Integer **perPage** – number of entries per page; if it is not preset, it will be taken as unlimited one

String **sortname** – field by which it is necessary to sort out the return parameters

String **sortorder** – sort order:

**asc** – ascending

**desc** – descending

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds

Integer **entriesCounter** – total number of entries by the period of time

RefuelingsAndDrainsType

**RefuelingsAndDrains** – list of parameters for each VH

getVehiclesProfiles – VH profiles matching the VH identifiers

## Connection

### Input Values

String **sessionId** – session identifier obtained during authorization

VehiclesType **vehicles** – list of VH IDs

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

**Vehicles** – list of parameters for each VH

## getCurrentObjectState – vehicle current status

### Input Values

String **sessionId** – session ID obtained during authorization

Integer **objectId** – object identifier

## Connection

### Returned Values

Boolean **status** – true/false. True in case of successful operation.

String **error** - error message in case an error occur

String **lastGPS** – latest valid coordinates. Contains latitude and longitude values separated by a semi-colon

Integer **lastGPSDir** – movement direction, degrees from 0 to 359

Double **currentSpeed** – current speed at the given moment, in kph, accuracy 0.1 kph

Double **currentFuel** – current fuel level, in litres, accuracy 0.1 l

Boolean **currentIgn** – ignition status. True if the ignition is ON

Boolean **speedExceed** – speed threshold exceed. True in case of speed threshold exceeded

Integer **lastGPSSat** – number of satellites with the last valid coordinates

Double **currentInputValue** – actual value of universal input. Attributes: Integer number – UI number, String name – UI name

getReportData – report for auxiliary equipment over the period, TPMS, IQFreeze

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – object identifier

Long **timeBegin** – start time of the interval (UTC), seconds

Long **timeEnd** – end time of the interval (UTC), seconds

String **reportTemplateID** – identifier of report template in Omnicomm Online. Possible values: addEquipment, TPMS, refState, refWork

## Connection

### Returned Values

Boolean **status** – true / false. True in case of successful operation

String **error** – error message in case an error occur

ReportDataType **reportData** – array including report data

getSEOnTime – auxiliary equipment running time for the period

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – identifier of vehicle/unit

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) - end time of the interval

### Returned Values

Boolean **status** - true/false true in case of successful operation

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Double **sEOnTime** (seconds) – auxiliary equipment running time for every connected UI.

Attributes: Integer number – UI number, String name – UI name

getStatisticsByPeriod – statistics for the period

## Connection

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds. If not defined, end time = system time of the request execution by the server.

int **objectType** – type of object:

0=vehicle;

1=driver;

If the type does not exist, error code 12 is returned.

objectIdsType **objectIds** – array of the type objectIdsType, containing list of objectId parameters of int type. In case of non-existence, the query is performed for all the objectId of the corresponding type, available to the user.

requiredStatParamsType **requiredStatParams** – list of the required subgroups of the 'Statistics' report parameters. If the list is empty, all the subgroups with all parameters are returned

## Connection

### Returned Values

movingAndWorkingParamsType **movementAndWorkingParams** – subgroup of VH movement and operation parameters. If the list is empty, all the parameters of the subgroup are returned, otherwise only the enumerated parameters are returned. If the subgroup is not transmitted, the whole subgroup will not be returned.

fuelParamsType **fuelParams** – subgroup of parameters for fuel, draining and refuellings. If the list is empty, all the parameters of the sub-group are returned, otherwise only the enumerated parameters are returned. If the subgroup is not transmitted, the whole subgroup will not be returned.

optionalEquipmentParamsType **optionalEquipmentParams** – subgroup of auxiliary equipment parameters. If the list is empty, all the parameters of the subgroup are returned, otherwise only the enumerated parameters are returned. If the subgroup is not transmitted, the whole subgroup will not be returned.

CANDataParamsType **CANDataParams** – subgroup of CAN parameters. If the list is empty, all the parameters of the subgroup are returned, otherwise only the enumerated parameters are returned. If the subgroup is not transmitted, the whole subgroup will not be returned.

addDataParamsType **addDataParams** – subgroup of statistics additional parameters: TPMS, iQFreeze, etc.

getTracksByPeriod – VH track for the period

### Input Values



## Connection

### Input Values

String **sessionId** – session identifier obtained during authorization

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds. If the end time is not indicated, the end time of the period = the system time of the beginning of the Server-side request processing

VehiclesType **vehicles** – list of vehicle ID

Int **reduce** – thinning:

0 = not required

1 = required

### Returned Values

Boolean **status** – operation status

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** – start time of the interval (UTC), seconds

Unixtimestamp **timeEnd** – end time of the interval (UTC), seconds

trackDataSetType **trackDataSet** – parameters of the track for every vehicle:

**trackPoint** – parameters of the track point by one vehicle

unixtimestamp **timestamp** – time of the event when the coordinates have been fixed

Integer **latitude** – latitude with accuracy of 0.0000001 degree

Integer **longitude** – longitude with accuracy of 0.0000001 degree

Integer **direction** – direction, degrees

Integer **sattelitesCount** – number of satellites

Double **speed** (km/hour) – speed

Long **timeStamp** (seconds) – time of event. (UTC)

getTrack – track

## Connection

### Input Values

String **sessionId** – session identifier obtained during authorization

Integer **objectId** – object identifier

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

### Returned Values

Boolean **status** – true/false true in case of successful operation

String **error** – error message in case an error occur

Unixtimestamp **timeBegin** (seconds) – start time of the interval

Unixtimestamp **timeEnd** (seconds) – end time of the interval

Dataset **trackEvents** – array of track points:

String **gpsPos** – event coordinates. Contains latitude and longitude values separated by a semicolon

Integer **gpsDir** – movement direction, degrees from 0 to 359

Integer **sattelitesCount** – number of satellites

Double **speed** – speed, in km/hour with accuracy up to 0.1 km/hour

Unixtimestamp **timeStamp** – point date and time

## REST API Methods List

/api/service/geozones/geozones GET – acquisition of data on user geofences

The user geofences are divided into pages. It is possible to specify the desired page and the number of geofences per page in the request form.

## Connection

### Input Values

String **Bearer {SessionID} or JWT {JWT}** – session identifier received during the authorization.

For example, for SessionID: Bearer 611ed1554ced2e10d13938bbd18482b060cb20f1

Integer **page** – the number of the page with user's geofences. If the value is not specified, the first page is returned

Integer **pageSize** – number of geofences on the page. Possible values: from 1 to 50. Default value: 20

### Returned Values

Integer **total** – total number of geofences

Integer **page** – number of the current page with user geofences

Integer **pageSize** – number of geofences on the page

Dataset **rows** – list of geofences:

String **name** – name of the geofence

Integer **id** – geofence identifier

Integer **rootGroupId** – root group of the geofence

Integer **geozoneTypeId** – geofence type identifier

string **geozoneTypeName** – geofence type

Number **radius** – radius of a "circle" type geofence

Integer **geometryTypeId** – geofence shape identifier. A polygon is used for a "circle" type geofence

String **geometryTypeName** – shape of the geofence

Integer **lineWidth** – width of a "line" type geofence

Integer **status** – status identifier (active/inactive)

String **statusName** – status

String **uuid** – uuid geofence identifier

Number **latitude** – latitude of the geographic center point of a "circle" type geofence

Number **longitude** – longitude of the geographic center point of a "circle" type geofence

Integer **countPoints** – number of points in a geofence

Dataset **points** – points of a geofence:

Integer **pointId** – point identifier

Number **latitude** – latitude

Number **longitude** – longitude

## Connection

/api/service/geozones/geozone-groups GET – acquisition of data on all user geofence groups

Input Values
String <b>Bearer {SessionID} or JWT {JWT}</b> – session identifier received during the authorization. For example, for SessionID: Bearer 611ed1554ced2e10d13938bbd18482b060cb20f1
Returned Values
<b>geozonelds</b> – identifiers of geofences that are not part of any group Integer <b>geozonelds</b> – geofence identifier Dataset <b>groups</b> – list of geofence groups: <div><div>Integer <b>Id</b> – group identifier Integer <b>parentGroupId</b> – parent group identifier String <b>name</b> – name of the group <b>geozonelds</b> – identifiers of geofences that are part of any group Integer <b>geozonelds</b> – geofence identifier</div></div>

## Errors

List of returned errors:

- 0: No errors** – there are no errors
- 1: Signing in failed** – incorrect Login / Password entered
- 2: Authorization required** – authorization is required to access the data
- 3: Dead session number** – session has expired, re-authorization is required
- 4: Bad interval** – incorrect time interval entered
- 5: Bad object** – there is no object with this identifier
- 6: Admin login** – someone is trying to log in as Admin User
- 7: Unusable object** – the value cannot be calculated for the object with this identifier
- 8: Bad event type** – there is no event type with this identifier
- 9: Access denied** – no authorization to access the object

## Connection

**10: Data not found** – no data for the corresponding input values

**11: Blocked interval** – the requested interval contains data blocking periods

**12: Bad object type** – the specified object type does not exist

**13: Invalid format** – the format is incorrect

**14: Undefined error** – the error is unspecified

**15: 404** – page not found

## Types of Events

Event type number	Meaning	Parameter values, comments
1	Start of refuelling (for refuelling tankers – fueling)	Value of refuelling, accuracy 0.1 litres
2	End of refuelling (for refuelling tankers – fueling)	Value of refuelling, accuracy 0.1 litres
3	Beginning of draining	Value of draining, accuracy 0.1 litres
4	End of draining	Value of draining, accuracy 0.1 litres
5	Ignition ON	Time from the last ignition switch off, minutes
6	Ignition OFF	
7	External power supply ON	
8	Battery power ON	
9	Driver authorisation	iButton key code in HEX

## Connection

Event type number	Meaning	Parameter values, comments
10	Transition to roaming	
11	Exit from roaming	
12	Instant speeding	Maximum speed value, accuracy 0.1 kph
13	Idle time	
14	Beginning of speeding	
15	Beginning of transaction groups (for refuelling tankers)	Fuel volume before the beginning of transaction groups, accuracy 0.1 litres
16	End of transactions group (for refuelling tankers)	Fuel volume after finishing transactions group, accuracy 0.1 litres
17	Start of fueling during transactions group (for refuelling tankers)	
18	End of fueling during transactions group (for refuelling tankers)	
19	Start of draining during transactions group (for refuelling tankers)	
20	End of draining during transactions group (for refuelling tankers)	
21	Connection established	

## Connection

Event type number	Meaning	Parameter values, comments
22	Additional equipment turned ON	
23	Additional equipment turned OFF	
24	Exceeding the auxiliary equipment max limits	
25	Return to auxiliary equipment normal values	
31	Pressing the panic button	
32	Overload of auxiliary equipment	
33	Power ON	
34	Instant RPM exceeding	
35	Entering the geofence	
36	Exiting the geofence	
38	Power OFF	
42	Beginning of stop	
43	End of stop	
44	Beginning of acceleration	

## Connection

Event type number	Meaning	Parameter values, comments
45	End of acceleration	
46	Digital input ON	
47	Digital input OFF	
48	Instant acceleration	
49	Unknown driver	
52	Device tampering	
53	Driver authorization finished	
54	iButton applied	
55	Deleted driver registration	

## Work Client Example

### Import of Interfaces

wsimport -d bin -s src <http://demo.omnicomm.ru:8000/AnalyticalServer/ws?wsdl>

Java code:

It is necessary to change strings "user" and "pass" by real values.

```
package ru.omnicomm.test.client;

import ru.omnicomm.analyticalserver.*;

import java.net.MalformedURLException;

import java.net.URL;
```



## Connection

```
import java.util.List;

public class ExampleClient {
    public static void main(String[] args) throws MalformedURLException {
        AnalyticalServer = new AnalyticalServer(new URL("http://demo.omnicomm.ru:8000,

        AnalyticalServerWS port = AnalyticalServer.getAnalyticalServerPort();

        AuthResponseEntry auth = port.signIn("user", "pass");

        String sessionId = auth.getSessionId ();

        System.out.println("auth sessionId: " + sessionId);

        ObjectSetResponseEntry objects = port.getObjectSet(sessionId);
        List<Vehicle> vehicles = objects.getVehicleList();

        for (Vehicle vehicle : vehicles) {
            System.out.printf("vehicle: %d / %s\n", vehicle.getVehicleID(), vehicle.g

        }
    }
}
```

**OMNICOMM**

[info@omnicomm-world.com](mailto:info@omnicomm-world.com)

[www.omnicomm-world.com](http://www.omnicomm-world.com)