- Commute to work.
- Start break.
- Start work, also indicates end of a break if the last reported status was start break.
- End work, also indicates end of a break if last reported status was start break.

### Working time reporting across vehicles or devices

Generally, when reporting working time, the start and the end are performed on the same device, but if drivers and workers use various vehicles and thus devices, the following rules should be considered:

- If a driver moves between PRO navigation devices or Remote LINK devices working times are driver-oriented.
- If a driver moves between GO navigation devices, working times are vehicle-oriented.
- Working sessions have to be closed, with **End work**, before pairing any device with a different device.
- Closing working sessions on a different device than the one with which you have started the working session is only possible, if you use Remote LINK (page 10) with an ID Key or PRO navigation devices.

## Vehicle maintenance

### Maintenance schedule

A maintenance schedule describes a maintenance task that has to be carried out on a vehicle. It defines when the task is scheduled and whether it has to be carried out once at a certain date or vehicle mileage, or if it needs to start at a certain date or vehicle mileage and be repeated at regular intervals.

In accordance with the maintenance schedule, WEBFLEET creates a maintenance task when the task is due. A reminder can also be issued a certain time or distance before the task is due, giving advanced notice of the maintenance task.

### Maintenance task

A maintenance task for a vehicle is created from the respective maintenance schedule. The task appears in WEBFLEET when it is due or the task's reminder is issued.

# Programming Guide

This programming guide is an introduction to using the WEBFLEET.connect interface, how to access the service and how to interpret the output that is returned.

In order to access the WEBFLEET.connect service you need a WEBFLEET account that has WEBFLEET.connect enabled. Otherwise you will not be able to test the integration for your application.

> Please talk to your TomTom Telematics sales contact if you do not have access to a WEBFLEET.connect-enabled account.

## Introduction to WEBFLEET.connect

WEBFLEET.connect is an API that allows you to access the WEBFLEET service through a web-enabled application. These are the primary features accessible through WEBFLEET.connect:

- **Reports** — Retrieve data that correspond to the information contained in the reports generated within WEBFLEET
- **Messaging** — Send text to mobile units and retrieve incoming messages.

  See Message queues (page 35) and Orders (page 93).
- **Addresses** — Insert, update and delete addresses and address groups as well as relations between addresses and address groups.

  See Addresses (page 151).
- **Orders** — Insert, send, update and delete orders and retrieve order status information.

  See Orders (page 93).
- **Drivers** — Insert, update and delete drivers and retrieve driver status information.

  See Drivers (page 130).

**Note**: As WEBFLEET.connect impersonates as the user that is provided as part of the service request authentication, access restrictions set up within WEBFLEET apply. This affects all elements of the WEBFLEET.connect interface. For instance, addresses can only be updated if the user has the respective access right to do so (for example: "Edit access" for "All addresses").

## Preparing for WEBFLEET.connect

WEBFLEET.connect can be made available to every customer with a valid WEBFLEET account. There should be at least one active object to make full use of the functionality offered by WEBFLEET.connect.

## Access to WEBFLEET.connect with API Key

To enable API access for your application, obtain an API key by doing the following:

**For .connect partners**

If you are a .connect partner, you will receive your API key during your partner application process. To request more API keys, complete the online request form on https://uk.support.telematics.tomtom.com/app/ask ( Request API Keys - https://uk.support.telematics.tomtom.com/app/ask).

- In the **Refine search by product model** drop-down menu select **Integration**, then select the desired API.
- In the **Refine search by category** drop-down menu select **API key request**.

**For customers**

If you are a customer and would like to request an API key, complete the online request form on https://uk.support.telematics.tomtom.com/app/ask ( Request API Keys - https://uk.support.telematics.tomtom.com/app/ask).

- In the **Refine search by product model** drop-down menu select **Integration**, then select the desired API.
- In the **Refine search by category** drop-down menu select **API key request**.

## Checking requirements

**Geographic coordinates**

Some functions require geographic coordinates such as longitude and latitude. This includes inserts and updates of addresses and sending orders.

Make sure that you are able to provide valid coordinates, otherwise you won't be able to fully leverage all functionality that WEBFLEET.connect offers. Geographic coordinates used by WEBFLEET always refer to the WGS84 (see WGS84 - https://en.wikipedia.org/wiki/World_Geodetic_System) coordinate system and have different representations.

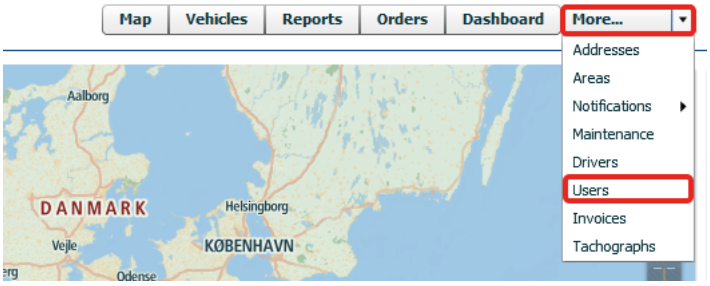
## Creating a user and assigning rights

In order to access WEBFLEET.connect, you must first create a user within your WEBFLEET account and give this user the right to use WEBFLEET.connect.

This procedure is outlined below:

1. Log in to WEBFLEET as account administrator.

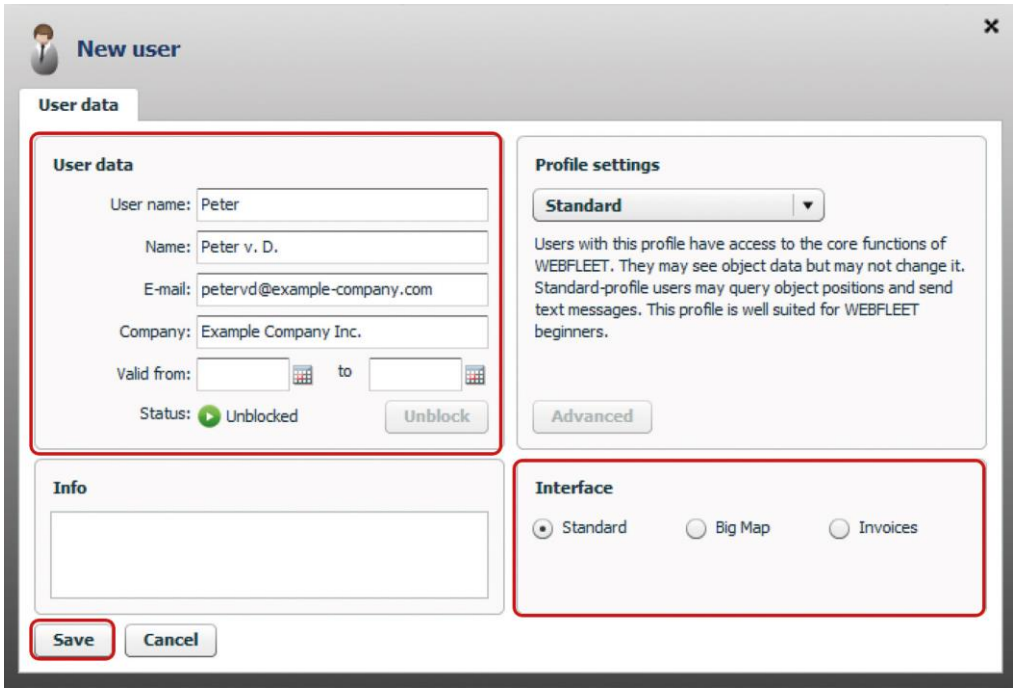
2. Click **More** in the Main Menu and select **Users** from the list.




3. Click **New**.


A pop-up window opens.

4. Fill in the required details.



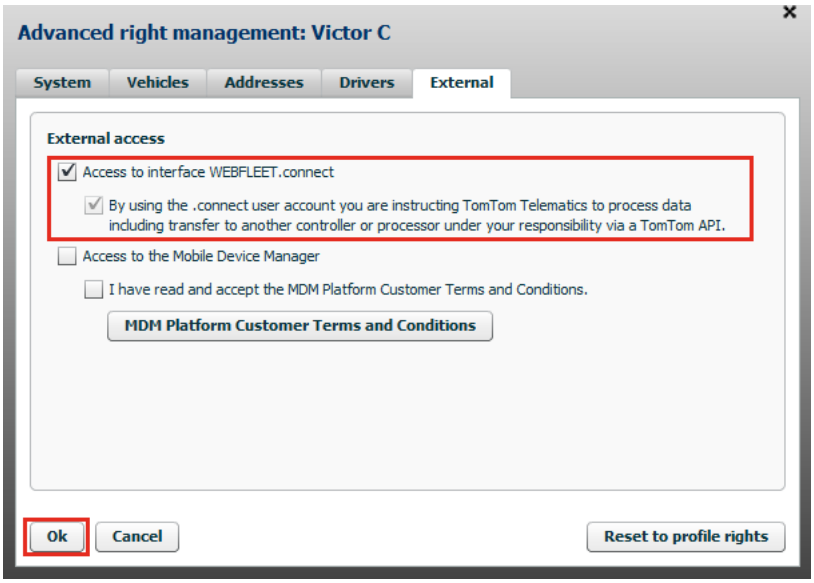

5. Select a security profile under **Profile settings**.

   A good starting point is to use the profile **Standard**.

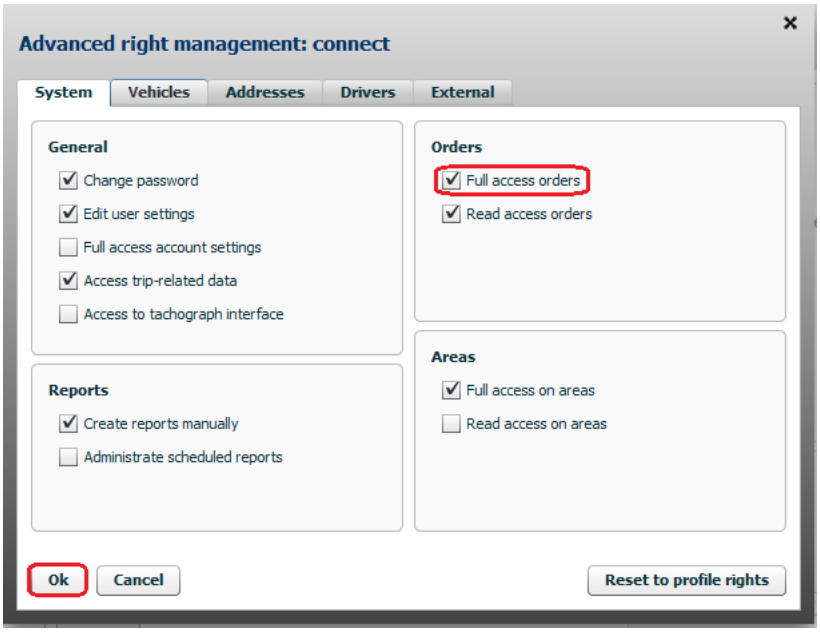
6. Click **Save** to save your changes.
7. Under **Profile settings** click the **Advanced** button to change user rights.

   A pop-up window opens.

8. In the **External** tab select **Access to WEBFLEET.connect**.

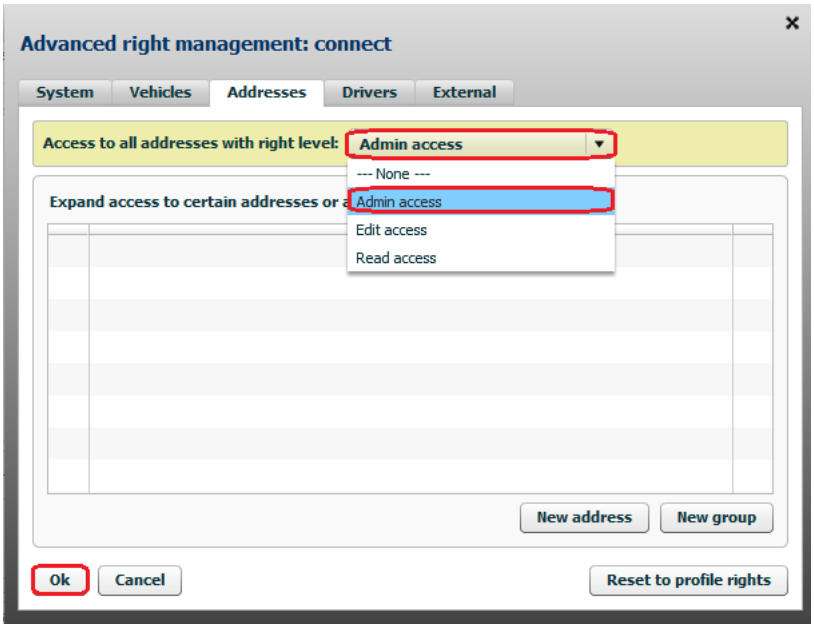



9. Click **Ok** to save your changes.
10. If you want to manage orders select the **System** tab and select **Full access orders** under **Orders**.




11. Click **Ok** to save your changes.

12. If you want to import addresses select the **Addresses** tab and select **Admin access** from the list for **Access to all addresses with right level**.




13. Click **Ok** to save your changes.

## Making requests to WEBFLEET.connect

### Making HTTP requests

This section explains how to use HTTP (see RFC 2616 - Hypertext Transfer Protocol -- HTTP1.1 - http://www.ietf.org/rfc/rfc2616.txt) to issue requests to WEBFLEET.connect.

**Important**: Only HTTPS requests are accepted. Requests using unencrpyted HTTP are rejected.

WEBFLEET.connect generally uses HTTP GET requests as the underlying transport mechanism for requests (POST is supported for few actions only, see below). All requests are made using specific URLs (see RFC 1738 - Uniform Resource Locators (URL) - http://www.ietf.org/rfc/rfc1738.txt), passing parameter names and values as URL parameters. Responses are returned as character-separated values (CSV).You can experiment with WEBFLEET.connect-specific HTTPS requests by entering the request URL into the browser's address bar and submitting the request.

WEBFLEET.connect uses the standardized transport protocol HTTPS 1.1 for which compliance with IETF RFC 2616 (see RFC 2616 - Hypertext Transfer Protocol -- HTTP1.1 - http://www.ietf.org/rfc/rfc2616.txt) is very important. It includes proper evaluation and handling of all HTTP response header values, e.g. character set, content and transfer encoding including chunked transfer encoding.

The sequence of incoming messages may differ from the sequence of messages sent from the device. Use their timestamp to restore the sequence, if the sequence is significant for the application.

We highly recommend to use ISO8601 (see ISO 8601 - https://en.wikipedia.org/wiki/ISO_8601) for all date and time values even if its use is optional with some functions. Date and time values carry timezone information where appropriate.

Character sets and date/time values need translation to local configurations, for example. UTF-8 to ISO-8859-10, UTC to CET. We do not guarantee the character encoding (currently: UTF-8) and timezone (currently: UTC for queue service, else time zone of the account) of the web service response, as all information to properly convert this to local requirements is provided as per the above mentioned standards.

For details about time zones when using ISO8601 in CSV read General parameters (page 29).

**Do not use HTTP authentication**, neither Basic nor Digest. HTTP authentication is not required by the service as username and password are included in the URL of the requests.

**Do not pass parameters with an empty value in a request,** if you do not explicitely want to delete the parameter's value.

**Note**: If the format of the HTTP request is not valid you will get a corresponding error.

#### The base URL

Every HTTP request to WEBFLEET.connect begins with constant elements for

- *host:*

```
csv.telematics.tomtom.com
```

- *path:*

```
extern
```

Therefore, the base URL with the https scheme used is:

https://csv.telematics.tomtom.com/extern (https://csv.telematics.tomtom.com/extern)

**Handling the response**

In case of an error, an error message is returned as plain text. The error message has the following layout:

```
id, description
```

`id` is a numeric value and `description` provides a reason text. The message is either in the language defined by the `lang` parameter or in English if no localised translation is available.

All methods that return data, provide the data as quoted character-separated values (CSV) with one record per line. Those methods' names typically contain verbs indicating data retrieval such as *show...* or *pop...* The Quoting character is ' " ' - if this character is part of the data, it is quoted with ", appearing as "".The ordering of result columns might not always match that of the documentation and is subject to change without notice. It is therefore advisable to use the column names returned in the first response line to identify the data columns by their name. If there is no data to return, an error message is returned, for example:

```
63,document is empty
```

All methods that transmit data, e.g. all *send...* methods, return nothing on successful completion, that is the response is empty.

Error codes and descriptions are also returned in two HTTP response header fields:

- X-WEBFLEET-Errorcode: *<Error code>*
- X-WEBFLEET-Errormessage: *<Error message>*

If there is no error, the header fields are omitted.

**Making HTTP POST requests**

In addition to GET you can use HTTP POST for the following actions only:

- sendDestinationOrderExtern (page 94)
- insertDestinationOrderExtern (page 105)
- updateDestinationOrderExtern (page 101)

WEBFLEET.connect accepts POST requests with Content-Type "*application/x-www-form-urlencoded*". The parameters and values are transferred in the body of the request. The parameter name is separated from the value by ' **=** ' and name/value pairs are separated from each other by ' **&** '. Special characters have to be encoded like query strings in URLs, see RFC 1738 - Uniform Resource Locators (URL) (see RFC 1738 - Uniform Resource Locators (URL) - http://www.ietf.org/rfc/rfc1738.txt).

The default character set is ISO-8859-1. To use UFT-8, you have to specify the character set in the HTTP header "Content-Type".

*Example:*

```
Content-Type: application/x-www-form-urlencoded;charset=utf-8
```

*Code sample 3-1: Example of a complete POST request*

```
POST/extern HTTP/1.1
```

```
Host:csv.telematics.tomtom.com
Connection:keep-alive
Content-Type:application/x-www-form-urlencoded;charset=UTF-8
Content-Length:177
lang=en&account=wfcdevaccount&username=wfcuser&password=yourpwd&apikey=y
ourapikey&action=sendDestinationOrder&objectno=0094&orderid=itn32&ordert
ext=Clean%20streets&longitude=12399200&latitude=51364460&wp=51363230,123
92520,Hamburger%20Str.%2012,1,1
```

**Using JSON**

To use JSON, add the additional parameter *outputformat=json* to the request URL. This will return JSON instead of CSV. All other parameters and functionality stays the same.

The field names in the JSON output are identical to the column names in the CSV format. The returned JSON is an array with a flat representation of the data, which is not grouped nor structured. But there is one exception from the "flat rule": The *surplus_data* member in the result of popQueueMessagesExtern (page 37) is structured JSON.

Empty data is omitted in the result – no "null" members. JSON data types, such as string, number and boolean are used where applicable. If there is no data to return, an empty JSON array is returned.

The HTTP Content-Type is "*application/json;charset=UTF-8*".

## Getting started with HTTP requests

For making HTTP requests, you only need a web browser.

**Preconditions**

- Up-to-date web browser, for example Chrome or Firefox.
- Valid API key and credentials.

**Making an HTTP request with a browser**

1. Simply type (or copy & paste) the full URL into the web browser address bar.

   Here is a simple example URL that will geocode the specified location:

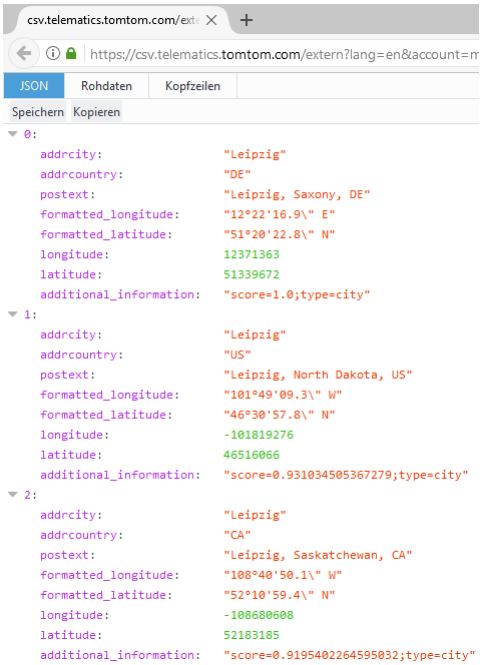
   ```
   https://csv.telematics.tomtom.com/extern?lang=en&account=xxx&username=x
   xx&password=xxx&apikey=xxx&action=geocodeAddress&outputformat=json&free
   text=Leipzig
   ```

   **Note**: Please insert your credentials and API key before submitting.

2. Press Enter to submit the request.

   Tip: The output format has been set to JSON. We recommend using the JSON format as output for these kinds of tests as this usually can be displayed inside the web browser as well.

The result output displayed in a web browser:




## HTTP request encoding

Every HTTP/HTTPS request must be a valid URL. That means that only ASCII https://en.wikipedia.org/wiki/ASCII characters are valid characters inside the URL and every other character, such as the German 'ß' character, or a special signs, such as the '@' sign, must be properly encoded inside the URL. There are two different types of encoding available depending on what kind of characters or special signs are needed.

### Percent encoding

This is the basic version of character encoding. It contains only some special characters like for instance the 'ß' character (encoded as %DF) or the '@' sign (encoded as %40).

Find below an example that uses 'Sußanne@work' as WEBFLEET.connect user name and how the special characters and characters inside this user name are correctly percent encoded:

```
https://csv.telematics.tomtom.com/extern?lang=de&account=***&username=Su%DFanne%40work&password=***&apikey=***&action=geocodeAddress&outputformat=json&freetext=Berlin
```

### UTF-8 encoding

For more complex character encoding the UTF-8 encoding is needed as it contains nearly all characters and special signs.

Here is another example. For example, to geocode the Polish town 'Łódź' you have to enable the UTF-8 encoding in the request first by using the following extra parameter:

```
&useUTF8=true
```

All special characters and signs inside the URL must now be encoded using the UTF-8 format.

Here is how the URL should finally look like:

```
https://csv.business.tomtom.com/extern?lang=de&account=***&username=***&p
assword=***&apikey=***&action=geocodeAddress&outputformat=json&useUTF8=tr
ue&freetext=%C5%81%C3%B3d%C5%BA
```

Please note that the word Łódź has been converted to the following UTF-8 encoded string:

```
%C5%81%C3%B3d%C5%BA
```

## Making SOAP requests

This section explains how to use SOAP (see Simple Object Access Protocol (SOAP) - https://www.w3.org/TR/soap/) to issue requests to WEBFLEET.connect. In order to ensure transmission security, it is required to use HTTPS to access the service via SSL.

WEBFLEET.connect supports the SOAP message protocol for issuing requests over an HTTPS connection. The easiest way to use the SOAP interface with your application is to use a SOAP toolkit appropriate for your programming platform. SOAP toolkits are available for most popular languages and platforms.

The files describing the operations and the data types are available at https://soap.telematics.tomtom.com (https://soap.telematics.tomtom.com). Most SOAP toolkits support the automatic generation of routines and classes based on the description.

WEBFLEET.connect uses the MTOM (see SOAP Message Transmission Optimization Mechanism (MTOM) - https://www.w3.org/TR/soap12-mtom/) extension to SOAP in order to provide an optimised transmission of data. Although most modern SOAP toolkits support this extension, your specific toolkit might need an additional support library to enable proper handling of MTOM.

### Using .NET with the WEBFLEET.connect SOAP API

If you are using the .NET to integrate with WEBFLEET.connect we recommend to using C# as the main programming language. If however you are forced to use Visual Basic this requires to apply the additional steps described in Using WEBFLEET.connect SOAP with Visual Basic (page 336) before importing the web service references into your development project.

### Enabling the MTOM encoding support in .NET

.NET 3.5 (and higher) and the underlying Windows Communication Framework support SOAP with the MTOM extension. If you create a default web service project in Visual studio, the MTOM support is not automatically enabled for a new project. To enable it edit the `app.config` file in your code project.

Rename every occurrence of the `textMessageEncoding` element to `mtomMessageEncoding`. See "before" and "after" samples below.

**Note**: Only rename the element. The attributes and their values **must not be removed**. Do not forget to **close and re-open the project** and **possibly Visual Studio** after this change, otherwise Visual Studio will not recognise the changes.

The following code samples show when MTOM encoding is supported.

*Code sample 3-2: Example: Before (MTOM encoding NOT supported):*

```
<textMessageEncoding maxReadPoolSize="64" maxWritePoolSize="16"
messageVersion="Soap12" writeEncoding="utf-8">
```

```
<readerQuotas maxDepth="32" maxStringContentLength="8192"
maxArrayLength="16384" maxBytesPerRead="4096"
maxNameTableCharCount="16384" />
</textMessageEncoding>
```

*Code sample 3-3: Example - After (MTOM encoding supported):*

```
<mtomMessageEncoding maxReadPoolSize="64" maxWritePoolSize="16"
messageVersion="Soap12" writeEncoding="utf-8">
<readerQuotas maxDepth="32" maxStringContentLength="8192"
maxArrayLength="16384" maxBytesPerRead="4096"
maxNameTableCharCount="16384" />
</mtomMessageEncoding>
```

#### Using time zones with SOAP requests

With SOAP requests indicate a time zone known to the SOAP service in the general parameters (`gParm`) `timeZone` element. All date time values returned by the output of a SOAP function call will use the time zone thus indicated. Time zones known to the SOAP web service are enumerated in KnownTimeZones (see WSDL (see TomTom SOAP services - https://soap.business.tomtom.com/)). The general parameters (`gParm`) `timeZone` element does not influence the interpretation of date time input data. This means, if a date time value is sent in a SOAP request, the time zone information specified in this value will be used.

## Getting started with SOAP requests

For making SOAP requests, you need a special tool.

We recommend using SoapUI (https://www.soapui.org (see https://www.soapui.org - https://www.soapui.org)), which also offers a basic open source version.

#### Preconditions

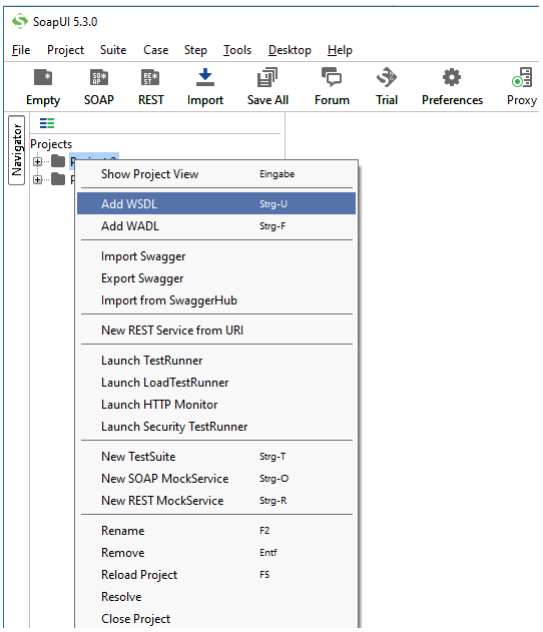
- SoapUI is installed.
- Valid WSDL URL or file.
- Valid API key (page 17) and credentials.

#### Making a SOAP request with SoapUI

To make a SOAP request, do the following:

1. Create an empty project.
2. Right-click on the project

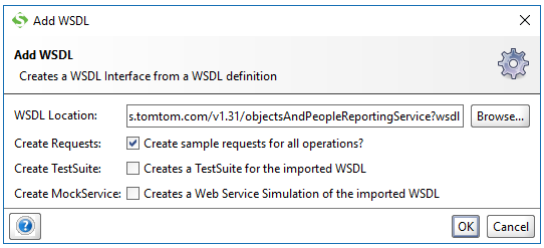
3. Click **Add WSDL**.

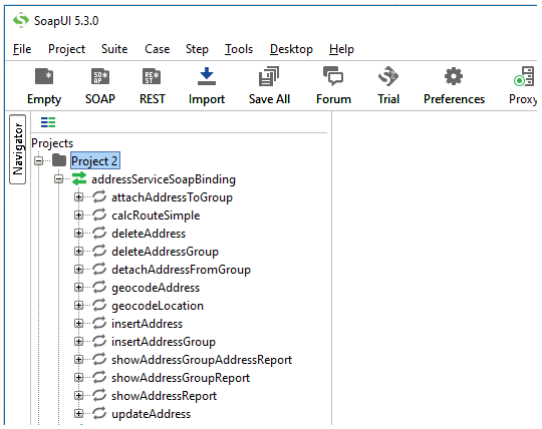



The **Add WSDL** dialogue opens.

4. Enter a valid WSDL URL that points to a WEBFLEET.connect SOAP service.
   All available actions for this SOAP service are displayed.

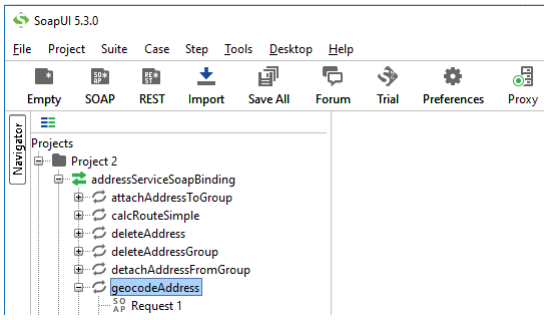
   *Example:*
   `https://soap.telematics.tomtom.com/v1.33/objectsAndPeopleReportingService?wsdl`



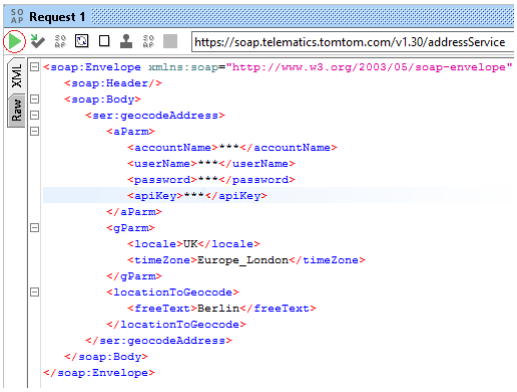




5. Click to select an action, for example **geocodeAddress**.
6. Create a new SOAP request.

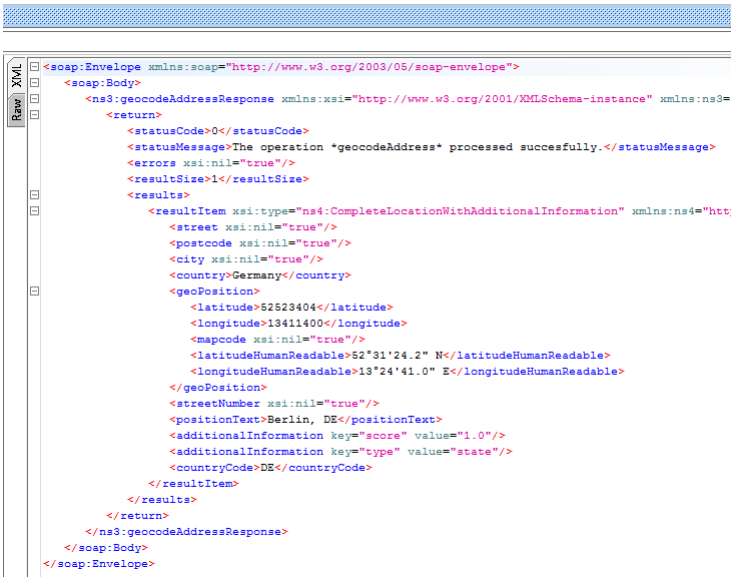
The request window opens.




7. Enter all required parameters.
8. Click the submit button.




Here is the result for the example SOAP request:




## Request limits

The number of requests that can be issued is limited. If the number of requests executed exceeds this limit, WEBFLEET.connect will return an error message and not process requests again until there were no further requests within the limit monitoring interval. Limits are defined by a maximum number of requests allowed in a certain time period. For more information about request limits, read the chapter for the function you want to use.