TomTom LINK.connect integration example

Quick start guide how to install and configure the TomTom LINK.connect integration example

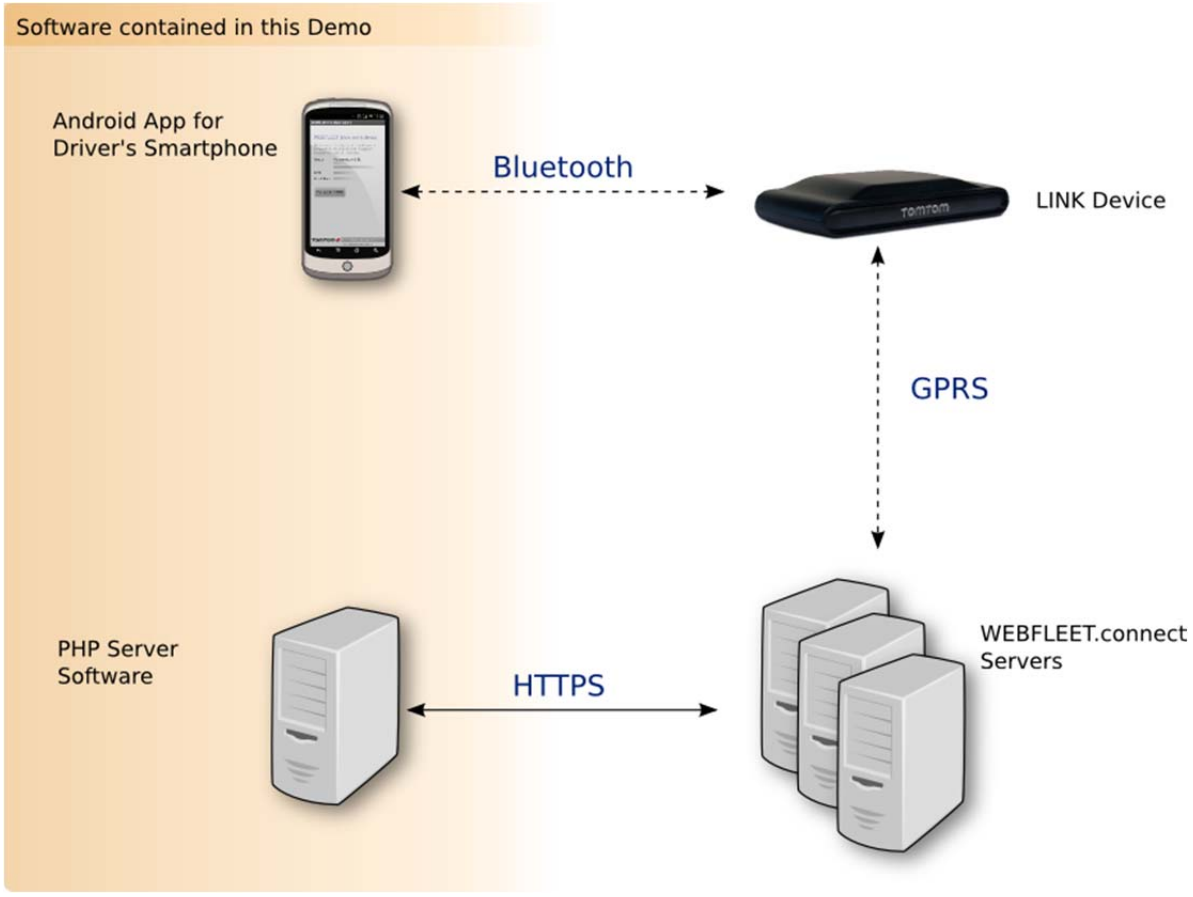


Table of contents

Introduction

This document describes the installation and configuration of the TomTom LINK.connect integration example. For further information regarding TomTom WEBFLEET.connect and TomTom LINK.connect, please refer to the following reference guides available on business.tomtom.com/fleet-management/integration/resources/

System requirements

In order to execute this integration example, the following requirements need to be fulfilled.

On the server side

- PHP 5 or higher.
- Internet access to WEBFLEET.connect. The server itself does not need to be reachable from the internet.

Smartphone

- Android based (2.3.3 or higher).
- Bluetooth support.

LINK device

- LINK 300/310/510 with a firmware that supports the LINK.connect feature.

Installation

Smartphone

1. Make sure, applications from other sources than “Google Play” can be installed.

   For this open **Settings**, go to **Application Settings** and enable **Unknown Sources**.

2. Install “btdemo.client-debug.apk” on the Android smartphone.

   You can do this for example by mailing it to the e-mail account of your smartphone or by copying it onto the SD card.

3. Start the App.

   A Bluetooth search for available LINK devices starts. Make sure, your LINK device is within Bluetooth distance, which is usually 10 meters.

4. If multiple devices were found, select your Bluetooth device from the list. If only one LINK device was found, it will be chosen automatically.

5. The App will now show the following information.

Status	Connected to LINK *<Serial number of LINK device>*
	Address: *<Bluetooth address of LINK device>*
IMEI	*<IMEI of the smartphone>*
BT address	*<Bluetooth address of the smartphone>*
BT name	*<Bluetooth “friendly name” of the smartphone>*

   You will need this information to configure the server part.

6. If you want to connect to another LINK device, tap **Menu** and **Discover LINK device** to start a new Bluetooth search.

Server

1. Determine the `objectuid` of the LINK device.

   You can do this for example by executing the following WEBFLEET.connect request.
   `https://csv.business.tomtom.com/extern?lang=en&account=<account>&username=<username>&password=<password>&action=showObjectReportExtern`

   `<account>`: your WEBFLEET account name
   `<username>`: your WEBFLEET.connect username
   `<password>`: your WEBFLEET.connect password

2. Configure the LINK device to allow Bluetooth communication with the smartphone:

   a. Fetch the list of configured Bluetooth peers for the LINK device to make sure, the smartphone is not yet configured and to get the highest `configid`.
   `https://csv.business.tomtom.com/extern?lang=en&account=<account>&username=<username>&password=<password>&action=getRemoteAuxDeviceConfig&objectuid=<objectuid>`

   `<objectuid>`: The object UID of the LINK device from the object-report above.

   b. If the smartphone is not yet configured, add it to the list of allowed peers:
   `https://csv.business.tomtom.com/extern?lang=en&account=<account>&username=<username>&password=<password>&action=configureRemoteAuxDeviceConfig&objectuid=<objectuid>&configid=<configid>&deviceid=<bt-address>&pin=<pin>`

   `<configid>`: If no Bluetooth peers are configured for your LINK device, use `0` (zero). Otherwise use the highest `configid` as returned in the `getRemoteAuxDeviceConfig` and increase its value by `1`.
   `<bt-address>`: The Bluetooth address of the smartphone.
   `<pin>`: The Bluetooth PIN that should be requested by the LINK device when

pairing to the smartphone for the first time. Use a four digit number of your choice.

The smartphone should now be allowed to connect to the LINK device. To check this, execute the `getRemoteAuxDeviceConfig` again.

3. Determine the `driverno` of your driver.
   https://csv.business.tomtom.com/extern?lang=en&account=<account>&username=<username>&password=<password>&action=showDriverReportExtern

   You will need the driver number when editing "config.php" (see below).

4. Unpack **btdemo.server.zip** on your server into a folder of your choice.
5. Change into the directoy **btdemo.server**.
6. Open **config.php** in a text editor and change the following settings as needed:
   a. Enter your WEBFLEET.connect credentials (account, username, password).
   b. Configure a HTTP proxy if required.
   c. Within `$config["drivers"]` enter the IMEI number of your smartphone and the driver number. The smartphone shall be assigned for example to:

      ```
      $config["drivers"] = array("1234567890" => "01");
      ```

      Note: In above example `1234567890` represents the IMEI and `01` represents the driver number.

   d. Save the file.
7. Start the daemon by executing **server.php** at the command line of your server.

   Note: The server part is a console based PHP application. Therefore it is not intended to be installed into a webserver's htdocs directory. Instead, unpack the ZIP file into a folder of your choice and start the application by executing **server.php** in a shell/command line. For example:

   Windows (see http://www.php.net/manual/en/install.windows.commandline.php)

   ```
   C:\PHP5\php.exe -f server.php
   ```

   Linux (see "man php")

   ```
   php server.php
   ```

8. The program should output some logging information similar to this:

   ```
   [2012-11-05 12:22:11] connecting to https://csv.business.tomtom.com/extern
   [2012-11-05 12:22:11] create message queue if needed
   [2012-11-05 12:22:11] created message queue
   [2012-11-05 12:22:11] start server loop, interval: 10 seconds, press <CTRL><C> to stop
   [2012-11-05 12:22:11] fetched 0 messages
   ```

   If you see any errors, verify the following:

   a. You have entered the correct login credentials in **config.php.**
   b. The user is allowed to use WEBFLEET.connect. You can configure this in the WEBFLEET user management.
      1. In WEBFLEET go to the main menu and click **More**.
      2. Select **Users** from the list.
      3. Select a user and click **Edit**.

4. Select the **User data** tab, find **Profile settings** and click **Advanced**.
5. Select the **System** tab, find **WEBFLEET.connect** and select **Access to interface WEBFLEET.connect**.

c. Your server has internet access to WEBFLEET.connect.

d. Your PHP installation has SSL support, which is required for HTTPS connections.

   On Linux you have to install the package **php5-openssl**. The package name may vary on your distribution.

   For Windows read http://php.net/manual/en/openssl.installation.php

Using the integration example

When you have successfully completed all installation steps you can tap the button **Assign me to LINK** within the Android app. You should see a dialogue **Sending request to LINK**. After this dialogue disappears, the **Status** label should be set to **Submitted request to LINK, waiting for server response**. After a few seconds you should see some lines in the server console, similar to the following:

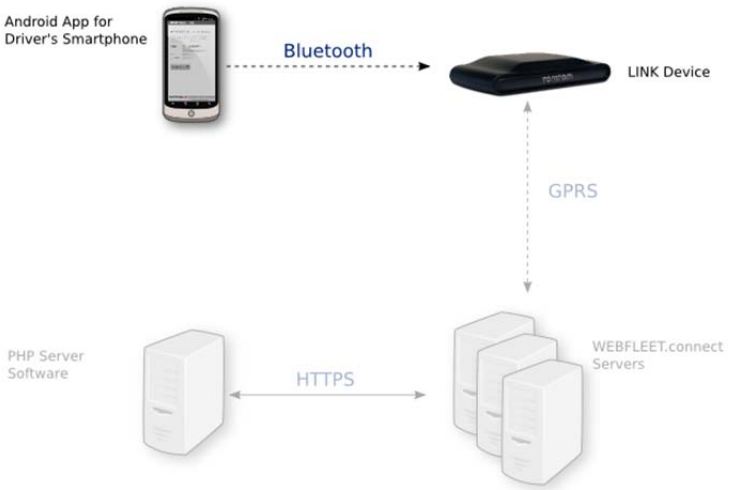
```
[2012-11-05 15:04:07] try to assign driverno <no> to objectuid <uid>

[2012-11-05 15:04:07] assign driver <no> to object <uid>

[2012-11-05 15:04:07] sending AUX data to objectuid <uid>, device: <address>, payload: successfully assigned as driver no: <no>
```

The server now sends an acknowledge message back to the LINK device. On the smartphone the status text should be updated to **Got server response. successfully assigned as driver no: <no>**.

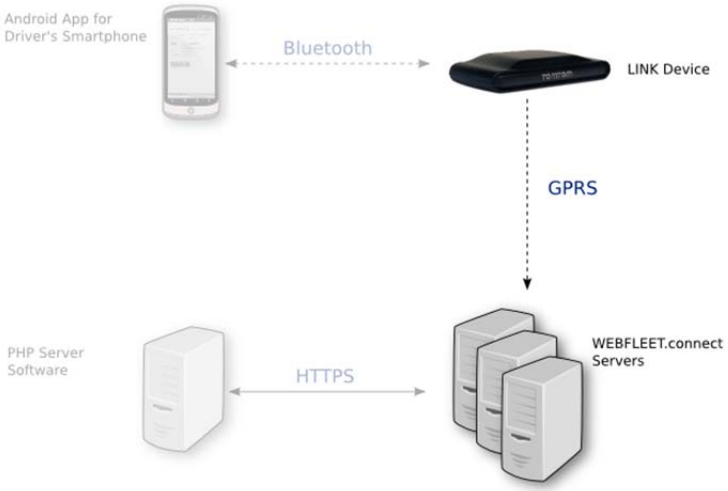
If you tap the button again without un-assigning the driver before you should see the status text **driver <no> already assigned**.

Client-Server communication described in detail

1. The Android smartphone sends data to the LINK device using a Bluetooth RFCOMM channel.

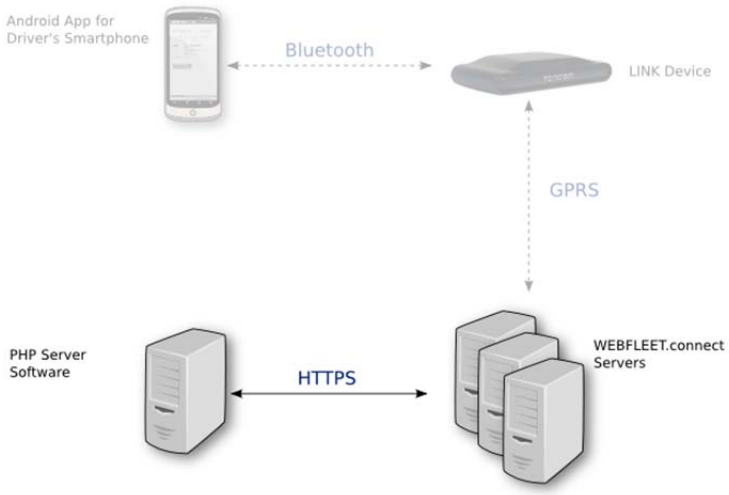


2. The LINK device forwards the received data to the WEBFLEET servers using GPRS. The payload of the smartphone is not interpreted by WEBFLEET - neither by the LINK nor by the WEBFLEET servers. It's passed unmodified to the message queues of WEBFLEET.connect.

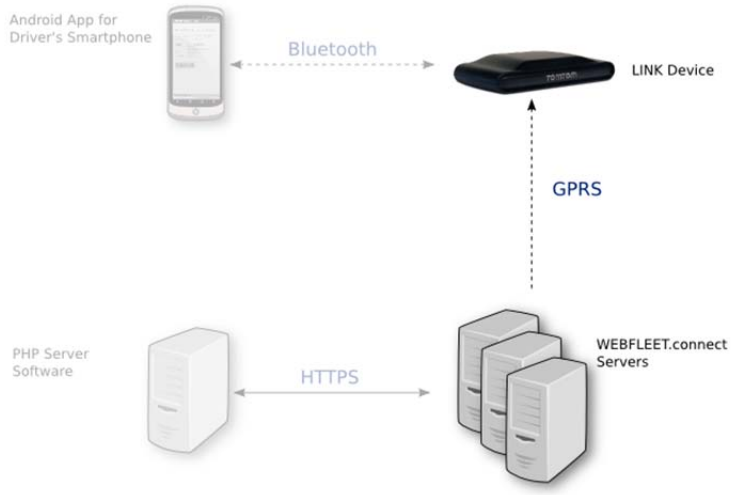


3. The PHP server regularly fetches new messages from WEBFLEET.connect using `popQueueMessagesExtern`. If a message contains matching payload, it's parsed by the PHP server to determine the IMEI of the smartphone. If there's a driver number assigned to the IMEI in **config.php**, the server executes the action `assignDriverToVehicle` to assign this driver to the LINK device the message came from.

After that, the server executes the action `sendAuxDeviceData` to notify the smartphone about the driver assignment. The server encodes the payload using Base64.



4. The WEBFLEET.connect servers pass the received payload back to the LINK device.



5. The Android smartphone receives the payload using a Bluetooth RFCOMM channel and shows it on the screen.