commsignia

User Guide for the Commsignia V2X Software Stack for Onboard Units

Document version: 20.34
Software version: y20.34
Publication date November 20, 2023
Document ID: USR-004

Table of Contents

List of Figures

List of Tables

1. Before connecting to the device

Onboard units (OBUs) are the V2X devices connected to different types of vehicles. They typically relay information to other OBUs and roadside units (RSUs) through wireless communication and also gather information from incoming V2X messages that facilitates the decision making of the driver or the autonomous system responsible for the vehicle.

Onboard units can be integrated to the standard vehicle data connections to gather real-time information about the vehicle itself and then this data can be used to enrich outgoing V2X messages transmitted from it.

Before connecting to the OBU, please ensure the following:

1. All connections have been made according to the provided Quick Start Guide.
2. The OBU is powered on through its 4-pin socket labeled "PWR."
3. Your computer is connected to same network as the OBU, using Ethernet of wireless (Wi-Fi) connection.
4. For the Graphical User Interface (GUI) an internet browser is available on your computer.
5. For the Command Line Interface (CLI) a Secure Shell Protocol (SSH) client is available on your computer.

Commsignia devices are delivered ready-to-use with a basic configuration. In order to configure the features of the Commsignia V2X software stack, a connection needs to be established to at least one V2X device running the stack.

2. Connecting to the OBU

Commsignia devices are delivered ready-to-use with a basic configuration. In order to configure the features of the Commsignia V2X software stack, a connection needs to be established to at least one V2X device running the stack. The device can be accessed via wireless or wired connection using a GUI or a CLI.

Please note that the device has two separate IP addresses for wireless and wired connections. Please ensure that your computer is connected to the same wireless or wired network as the OBU. All passwords are case sensitive.

1. Connecting to the device over wireless (Wi-Fi) network:
   The SSID of the OBU is **ITS-OB4-XXXXXXX**, where XXXXXXX is the last seven digits of the serial number of the OBU, which can be found on the product label. The default Wi-Fi password is **Commsignia**.

   a. In a web browser enter the IP address of the device, which is **192.168.1.54** by default. Use the username **root** and enter the default password, **UK5BJLFZVBPZLIM55Y**, to log in, as shown in Figure 1.

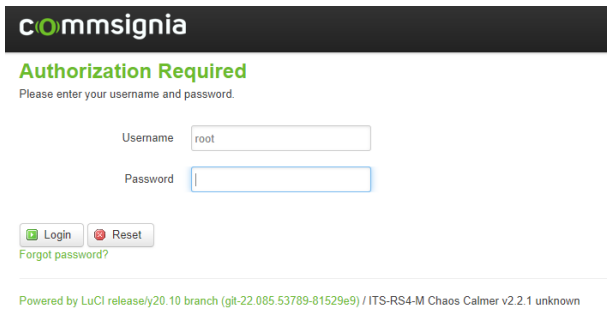

Figure 1. Login screen of the GUI

   b. Alternatively, an SSH connection can be established from the CLI as

```
ssh root@192.168.1.54
```

   and entering the same root password, **UK5BJLFZVBPZLIM55Y**, when prompted.

2. Connecting to the device over wired (Ethernet) connection:

   a. Please ensure that your computer is connected to the same wired network as the OBU. Your computer needs to use the same subnet as the Commsignia default of the OBU.

   b. In a web browser enter the IP address of the device, which is **192.168.0.54** by default. Use the user name **root** and enter the default password, **UK5BJLFZVBPZLIM55Y**, to log in, as shown in Figure 1.

   c. Alternatively, an SSH connection can be established from the CLI as

```
ssh root@192.168.0.54
```

and entering the same root password, **UK5BJLFZVBPZLIM55Y**, when prompted.

3. For troubleshooting purposes the device can be accessed over serial connection:
   a. Connect your computer using the provided microUSB cable to the OBU using the CNSL port.
   b. Identify the appropriate COM port that the device is connected to:
      i. On a Windows computer, open Control panel ⟶ System ⟶ **Device Manager**, and expand **Ports (COM & LPT)**, where two Silicon Labs COM ports needs to appear, one of which provides the actual connection. The exact port can be identified by trying both of them.
      ii. On a Linux computer, use the command

```
dmesg | grep tty
```

      which list the serial ports, one of which provides the actual connection. The exact port can be identified by trying both of them.
   c. Connect to the COM port as follows:
      i. On a Windows computer use a serial console application, such as PuTTY, set the Connection type to Serial, in the Serial line field enter the appropriate port number (e. g. COM4), and in the field Speed enter 115200.
      ii. On a Linux computer, use `picocom` or `minicom` with the appropriate `/dev/tty*` port and a baud rate of 115200.
4. Connecting to multiple devices by connecting them to the same switch on your network after configuring a unique fixed IP address for the devices:
   a. Connect to a device using one of the methods described above.
   b. Set the IP address and gateway of the device under the **Network** → **Interfaces** menu by editing the **eth0** interface, as shown in Figure 2.

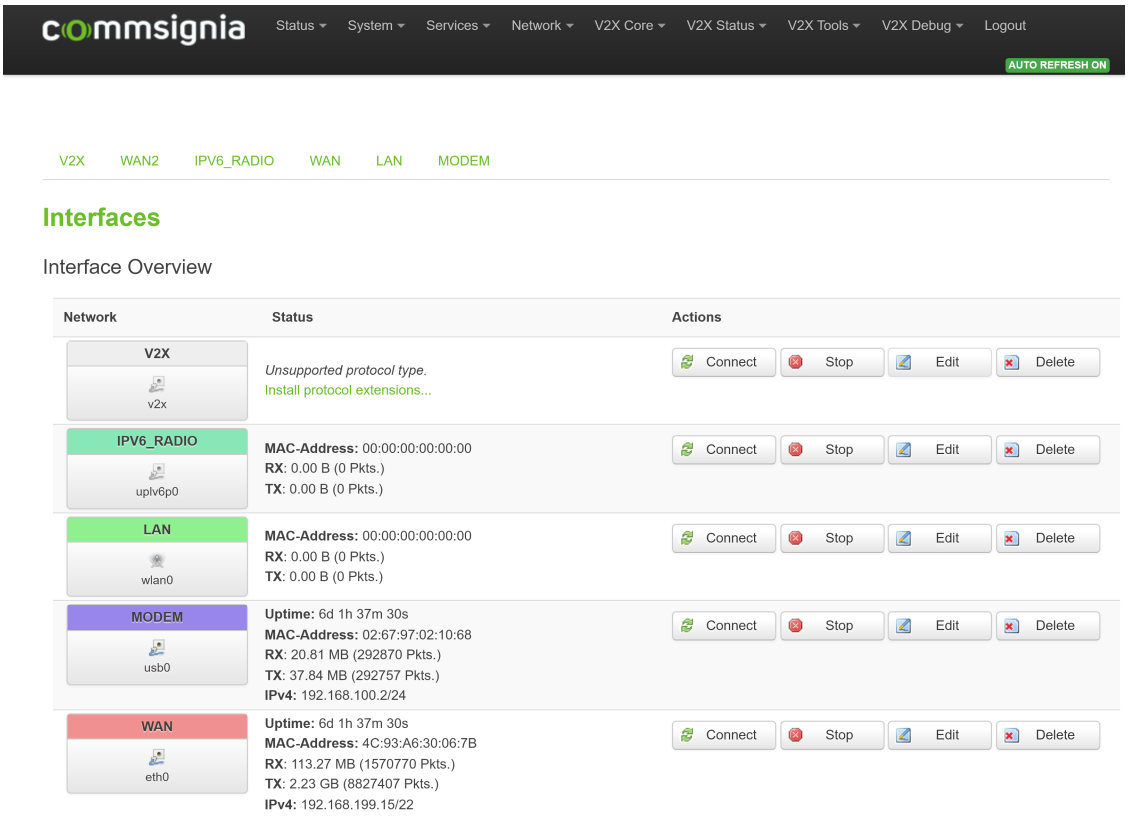

Figure 2. Interfaces page on the GUI

c. Click on the Save & Apply button for the changes to take effect on the device.

d. Repeat the steps above for all devices.

5. Connect to the device using the Foresight Human–machine interface (HMI) application to visualize its data:

   a. Open the Foresight application on your Android device.

   b. The Foresight application opens with the **Connection list** view, as shown in Figure 3, which shows the list of already configured connections to V2X devices.

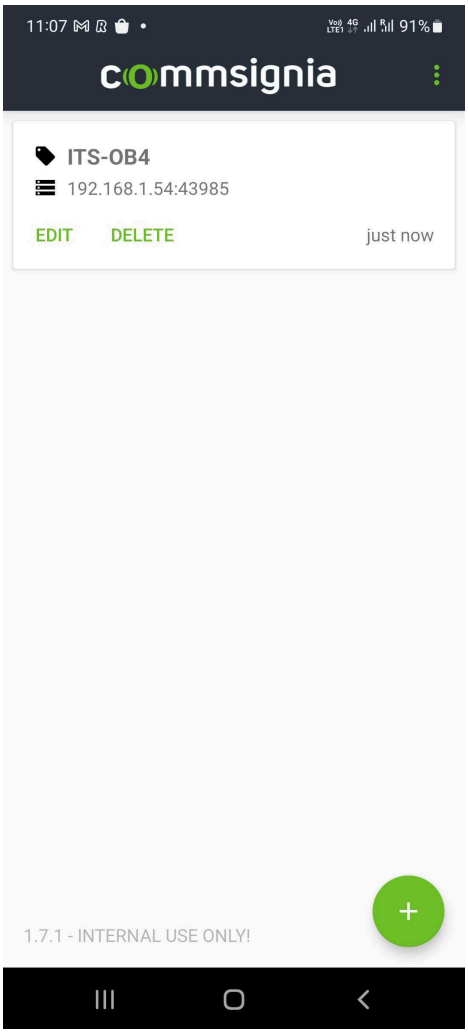

Figure 3. Connection list view in Foresight

c. If the device has an already configured connection, tap on it in the list to connect. Otherwise, tap on the + icon to add a new connection to the list; this opens a **Create new device page**, as shown in Figure 4.

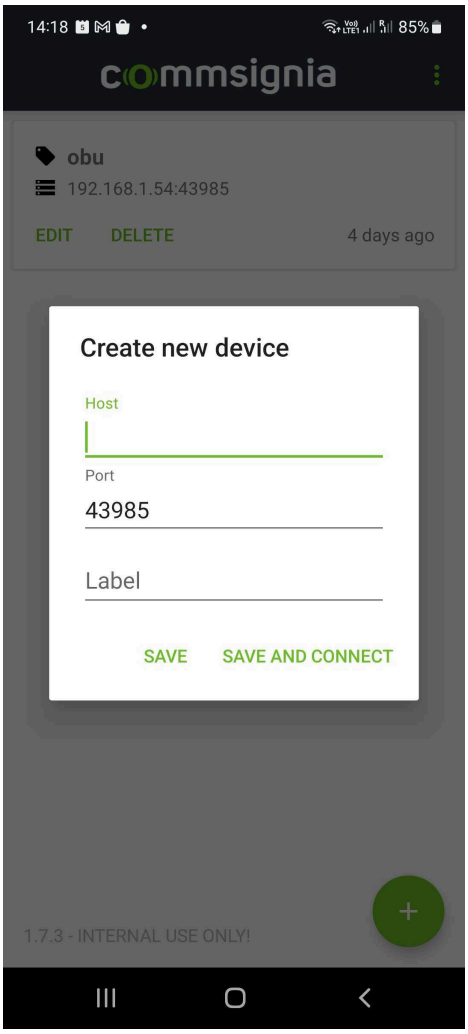

Figure 4. Create new device page

Enter the IP address of the OBU in the field **Host address**. The default port number is **43985**. A custom label can optionally be added to the connection.

6. To save the settings and continue editing, tap on SAVE. To connect immediately to the device and open the **Follow** view, tap on SAVE AND CONNECT.

3. Validating V2X communication

3.1. Using the GUI to check the status of the stack

The recommended method for the validation of the transmission and reception of V2X messages is using the GUI of the software stack. Log into the GUI as described insection "Connecting to the OBU" [2].

The status of the V2X software stack can be monitored under the **V2X Status** → **Status** menu. Scroll down to the **statistics** module, expand it, then expand the appropriate radio and interface modules, as shown in Figure 5.

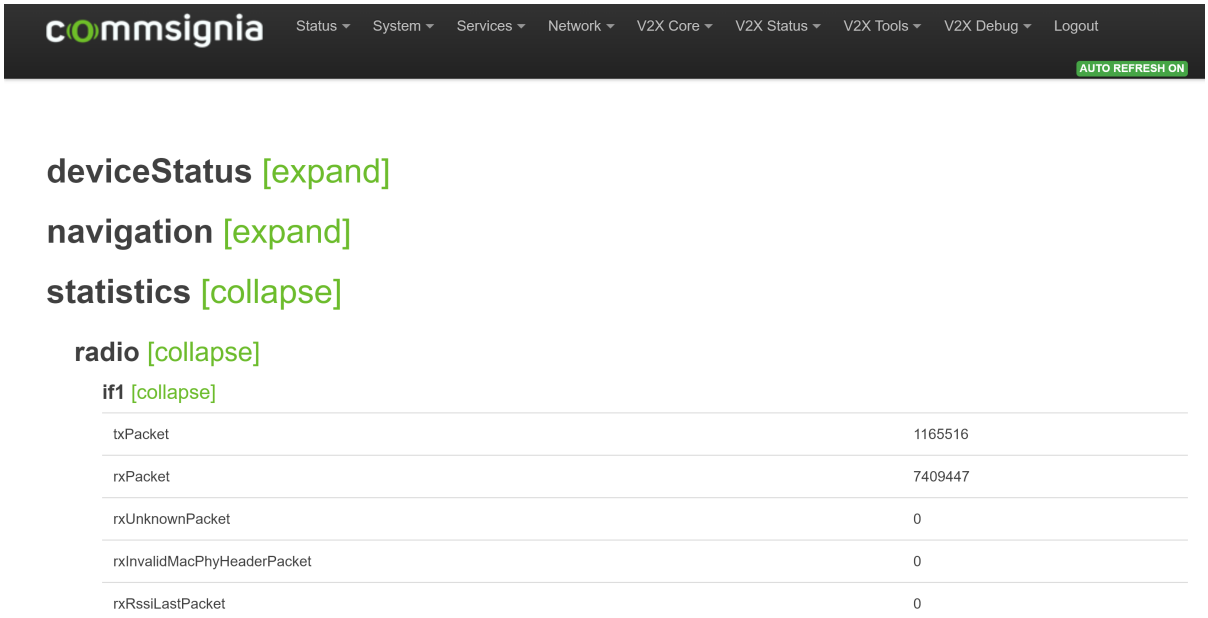

Figure 5. V2X stack status page

After refreshing the page, the transmission and reception of the packets can be verified by the increase of the values of the transmitted (txPacket) or received (rxPacket) packets. If the counters do not change, refer to the section Troubleshooting V2X communication.

3.2. Using the CLI to generate a status report

A status report of the running software stack can be generated using the CLI. Log into the device using SSH as described in section "Connecting to the OBU" [2].

Use the following command to generate a status log:

```
v2x-status-json-gen
```

The `j` utility can be used to filter the required values, for example the command `v2x-status-json-gen| jq '.statistics.radio.if1'` list the counter values related to radio interface 1 only:

```
{
  "txPacket": 1151711,
  "rxPacket": 7196345,
  "rxUnknownPacket": 0,
  "rxInvalidMacPhyHeaderPacket": 0,
  "rxRssiLastPacket": 0
}
```


If the values of the transmitted (txPacket) or received (rxPacket) are not increasing, please refer to the section Troubleshooting V2X communication

3.3. Using the Foresight application to visualize data

Open the Foresight HMI application on your tablet for the visualization of the connected V2X devices. For more information about setting up a Foresight connection, please refer to the section "Connecting to the OBU" [2].

The "Follow" mode of the application shows the surrounding area on a map view and indicate the triggered safety applications with icons. For more information, please refer to the section Safety applications for OBUs.

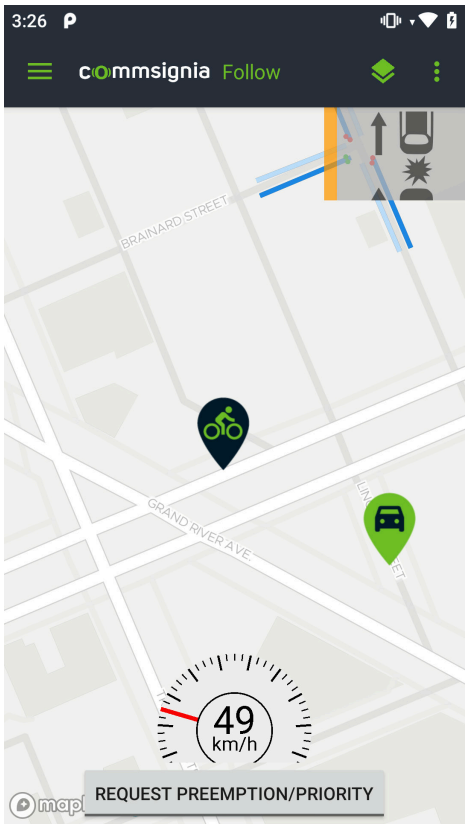

Figure 6. Follow view in the Foresight application

3.4. Creating Packet Capture files on the OBU

For a comprehensive analysis, Packet Capture (PCAP) files can be saved on the OBU using the optional Commsignia Capture Protocol (C2P) module, which generates a User Datagram Protocol (UDP) data stream from the sent and received V2X packets, This data stream can be locally saved on the device into a PCAP file, that can then be copied to a computer using SSH, and the packets can be parsed using the optional packet analyzer *Capture Application* by Commsignia.

The C2P module can be enabled and configured on the OBU RSU using the GUI or the `muci` tool.

3.4.1. Configuring the C2P module

3.4.1.1. Using the GUI for configuring the C2P module

To use the GUI for configuring the C2P module, proceed as follows:

1. Log into the GUI. For more information, refer to section "Connecting to the OBU" [2].

2. Open the **V2X Core** → **Core stack** menu item, check the box near the option **Commsignia Capture Protocol (C2P)**, and expand it, as shown in Figure 7.

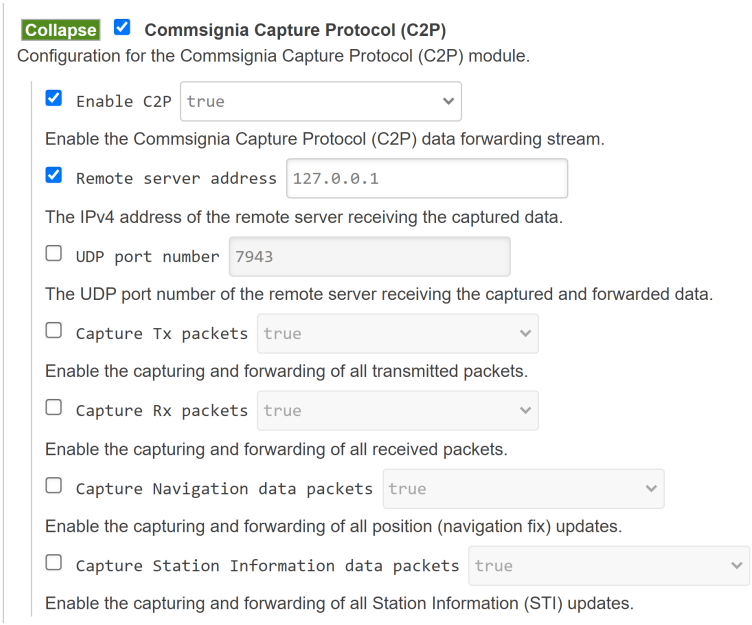

Figure 7. Commsignia Capture Protocol (C2P) settings for localhost

3. To enable the C2P data stream, check the box next to `Enable C2P` and set its value to `true`.

4. Check the box next to `Remote server address` and specify the IP address of the localhost, 127.0.0.1.

5. Click on the Save & Apply button for the changes to take effect on the device.

3.4.1.2. Using the muci tool for configuring the C2P module

If the `muci` tool is available on the device (only ion OB4/RS4 devices) it can be used for configuring the C2P module as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Use the following command to enable the C2P data stream:

```
muci its set capture.enable true
```

3. Specify the IP address of the localhost, 127.0.0.1 as follows:

```
muci its set capture.address 127.0.0.1
```

4. Restart the stack using the command

```
unplugged-rt-restart.sh
```

3.4.2. Writing data stream into PCAP files

To write the data stream into a PCAP file the `tcpdump` command line program can be used. To save a PCAP file and copy to the computer running the Capture Application, proceed as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Use the following command to start writing the data stream into a PCAP file in the `/tmp` directory:

```
tcpdump -v -pi lo -w /tmp/c2p-$HOSTNAME-$(date +%Y%m%d%H%M%S).pcap port 7943
```

   The `tcpdump` program starts writing the data into a PCAP file with a filename containing the hostname and the date and time of the capture.

   

   Please note that the `/tmp` directory is erased after rebooting the V2X device.

3. After the necessary number of packet were captured, terminate the `tcpdump` program by pressing *Ctrl + C*.

4. On the computer running the Capture Application, enter the directory where the PCAP file needs to be downloaded from the V2X device and use the following SCP command in a terminal window:

```
scp root@<IP address of the V2X device>:/tmp/c2p-HOSTNAME-
DATE.pcap <name>.pcap
```

   Alternatively, on a Windows computer an SCP client, such as Win SCP can be used.

5. Open the copied PCAP file using a packet analyzer, such as the optional *Capture Application* by Commsignia.

4. Basic settings of the OBU

Commsignia OBUs are delivered with a default configuration preset and V2X message transmission, and thus, reception works automatically. However, several configuration options are available to customize the behavior of OBUs.

4.1. Changing the passwords

To ensure the secure operation of the device, it is strongly recommended to change both its WiFi and login passwords after the first login. The passwords can be changed by using either the GUI or the CLI of the deice.

4.1.1. Changing the WiFi password

Please note that the WiFi password must be at least 8 characters long.

4.1.1.1. Changing the WiFi password using the GUI

To change the WiFi password of the device using the GUI, proceed as follows:

1. Log into the GUI. For more information, refer to section "Connecting to the OBU" [2].

2. Open the **Network** → **Wifi** menu and select the SSID of the device on the **Wireless Overview** group, as shown in Figure 8.

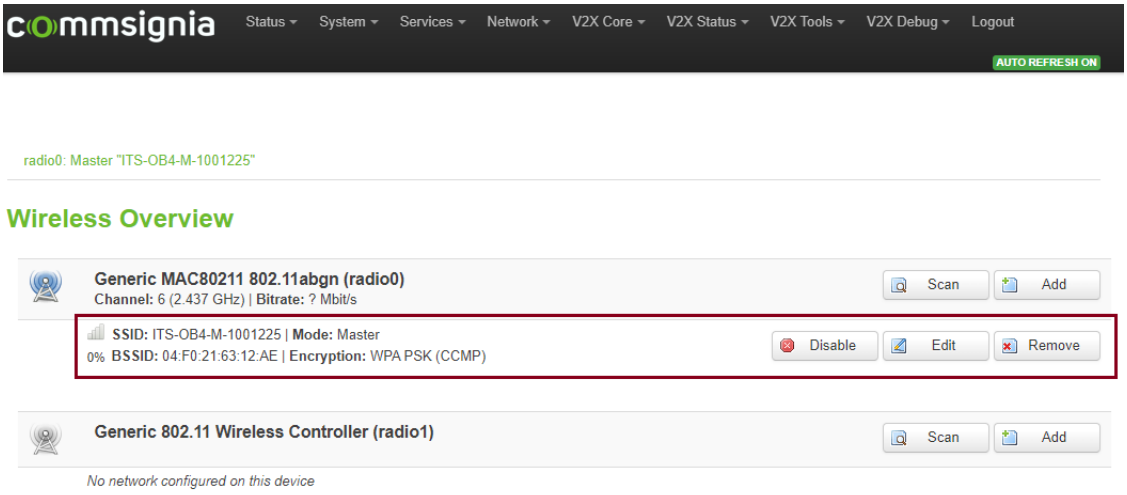

Figure 8. Wireless Overview group

Click on the Edit button.

3. In the Interface Configuration group, click on the Wireless Security tab as shown in Figure 9.

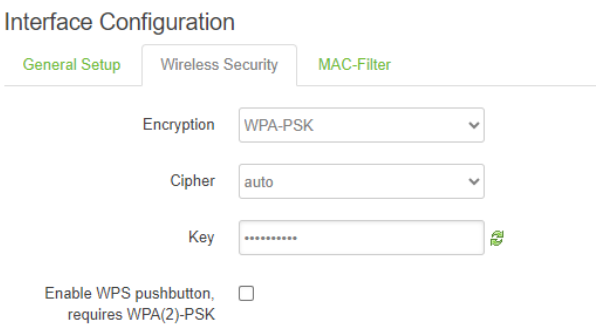

Figure 9. Interface configuration group

4. Specify the applicable encryption method and add a new password in the Key field.

5. Click on the Save & Apply button for the changes to take effect on the device.

4.1.1.2. Changing the WiFi password using the CLI

To change the WiFi password of the device using the CLI, proceed as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Use the following commands to set a new password:

```
uci set wireless.@wifi-iface[0].key='<Enter password here>'
```

3. Apply the changes as:

```
uci commit wireless
```

4. Reload the LuCI interface as:

```
reload_config
```

If an error message `uci: Entry not found` appears, please ignore it.

4.1.2. Changing the login password

Please note that the login password must contain at least 8 characters, including a capital letter, a special character, and at least 2 numbers.

4.1.2.1. Changing the login password using the GUI

To change the login password of the device using the GUI, proceed as follows:

1. Log into the GUI. For more information, refer to section "Connecting to the OBU" [2].

2. Open the **System** → **Administration** menu.

3. Under **Router Password** enter new password and confirm it as shown in Figure 10.

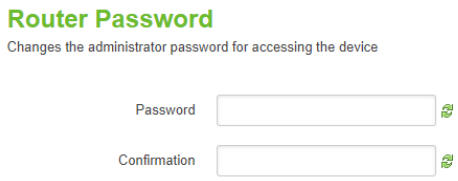

Figure 10. Creating new login password

4. Click on the Save & Apply button for the changes to take effect on the device.

4.1.2.2. Changing the login password using the CLI

To change the login password of the device using the CLI, proceed as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].
2. Use the following command to change the password:

```
passwd
```

3. Add your current password and press Enter.
4. Add your new password and press Enter.
5. Confirm your new password and press Enter.

4.2. Restoring the default configuration

If, for any reason, the default configuration of the device needs to be restored, proceed as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].
2. Open the **V2X Core** → **V2X profile preset** item, as shown in Figure 11.

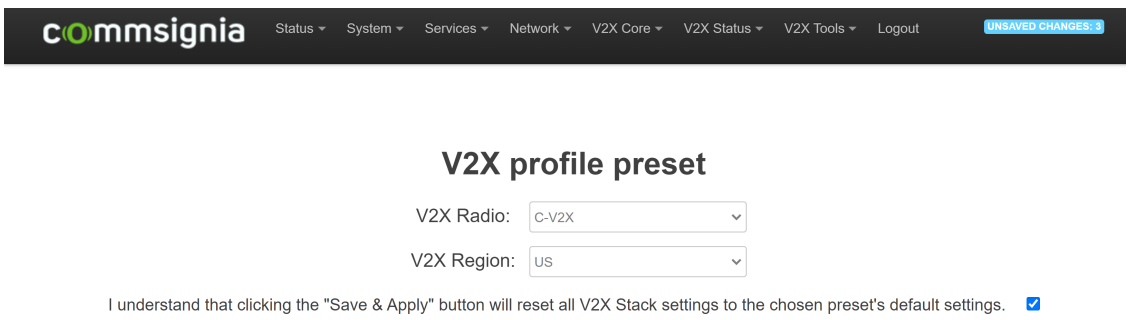

Figure 11. V2X profile preset

3. Set the appropriate V2X Radio and V2X Region, then check the box on the right of the disclaimer.
4. Click on the Save & Apply button for the changes to take effect on the device.

4.3. Configuring navigation settings

Three navigation modes are available for the device: "Real," "Gpsd," and "Manual." Te default setting is "Real;" however, if you require gpsd-based or manual navigation fix, it can be set using the GUI as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Open the **V2X Core** → **Core stack** menu item and expand the option **Navigation configuration**, as shown in Figure 12.

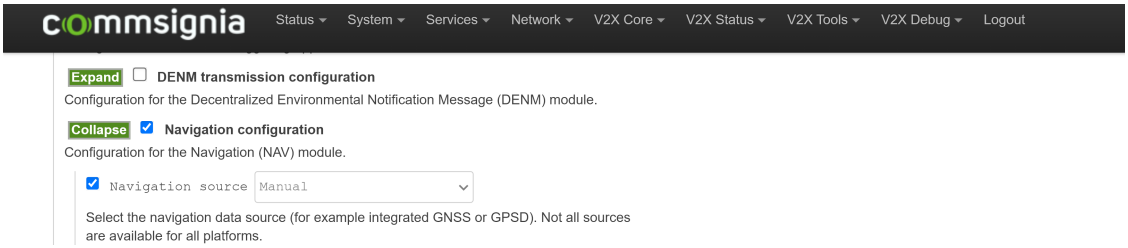

Figure 12. Navigation configuration

3. For the manual navigation source, set the `Navigation source` to `Manual`.

   a. To configure the manual navigation source, expand the **Manual navigation** option on the same page, as shown in Figure 13.

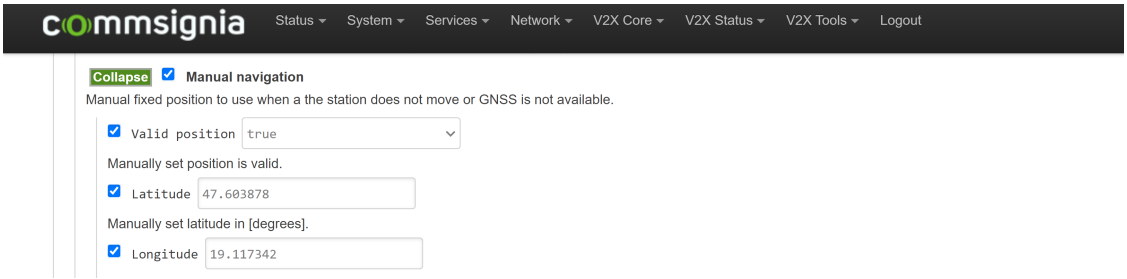

Figure 13. Manual navigation settings

   b. Expand and select the checkbox next to **Manual navigation** , check the box next to `Valid position` and set it to `true` , then enter the `Latitude` and `Longitude` information, as shown in Figure 13, and, if applicable, other values as well.

   c. Click on the Save & Apply button for the changes to take effect.

4. For the gpsd navigation source, set the`Navigation source` to `Gpsd`.

   a. To configure the gpsd navigation source, expand the **GPSD** option on the same page, as shown in Figure 13.

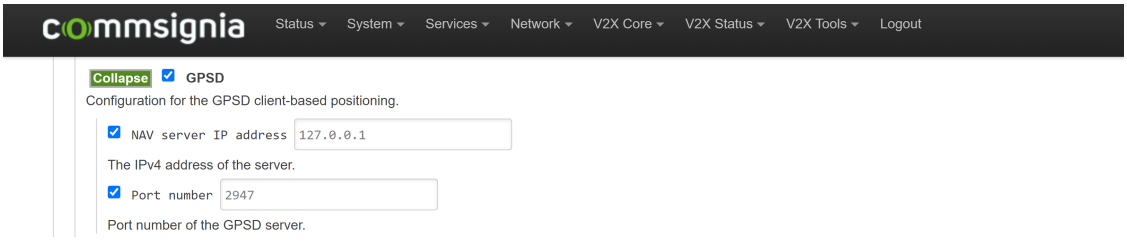

Figure 14. GPSD navigation settings

   b. Select the checkbox new **GPSD** and enter the gpsd server and port information, as shown in Figure 14.

c. Click on the Save & Apply button for the changes to take effect.

4.4. Configuring station parameters

Station parameters can be optionally set for the OBUas follows.

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Open the V2X Core → Core stack menu item and expand the option **Station information configuration**, as shown in Figure 15.

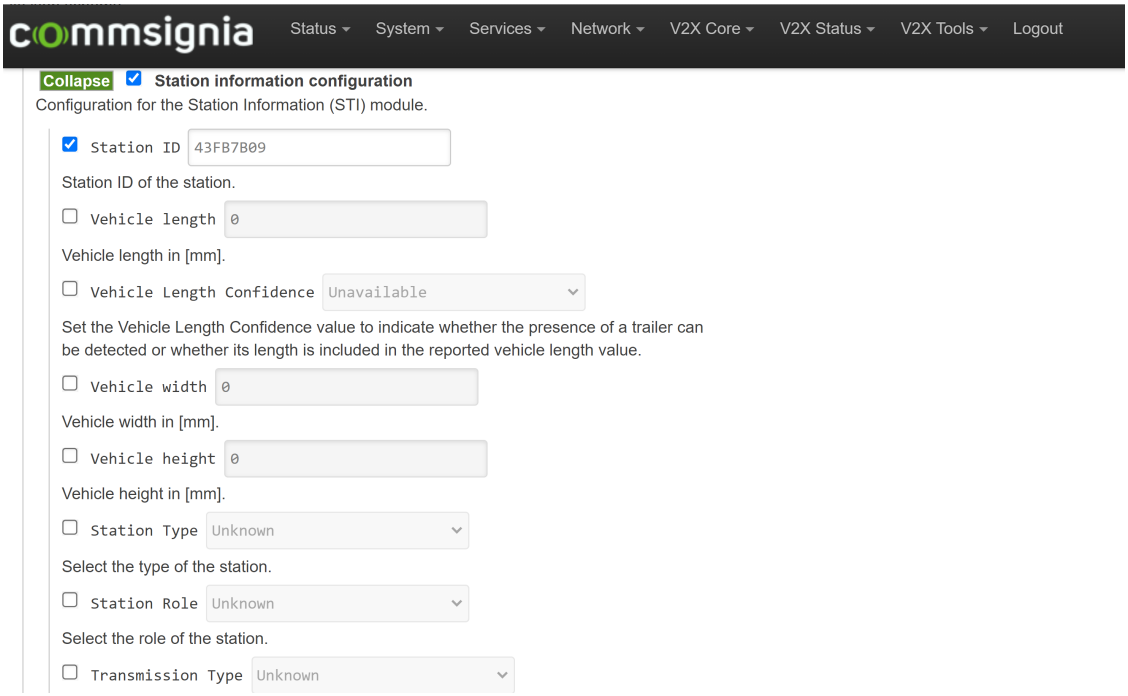

Figure 15. Station information settings

3. Select the checkbox next to **Station information configuration** and enter the required information.

4. Click on the Save & Apply button for the changes to take effect.

4.5. Configuring the radio interface

Radio interfaces can be set to be compliant with US or EU standards.

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. To enable the V2X communication, open the V2X Core → Core stack menu item, expand the option **Radio**. According to the radio hardware the following settings are available:

   a. For devices with Qualcomm radio (mainly US, C-V2X) the follow settings can be configured: Select the checkbox near **Qualcomm CV2X** as shown in Figure 16.

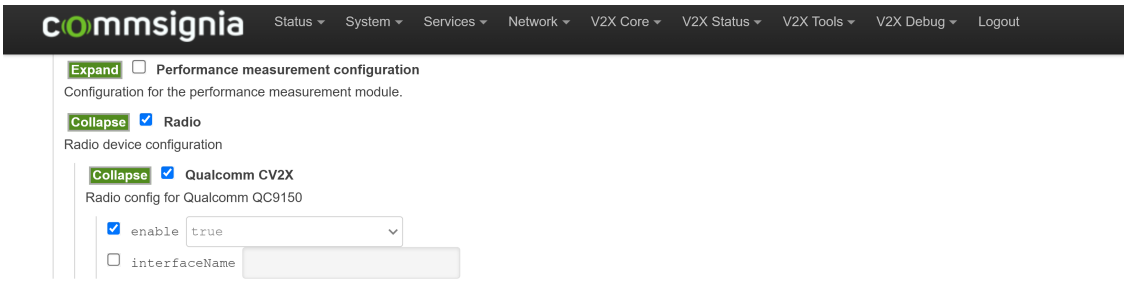

Figure 16. Qualcomm radio interface settings

As Qualcomm configures all parameters automatically upon startup based on GPS information, according to the region; this option cannot be configured further.

b. For devices with Autotalks CUT2 radios (mainly EU, DSRC) the follow settings can be configured:
Select the checkbox near **AT DSRC** as shown in Figure 17.

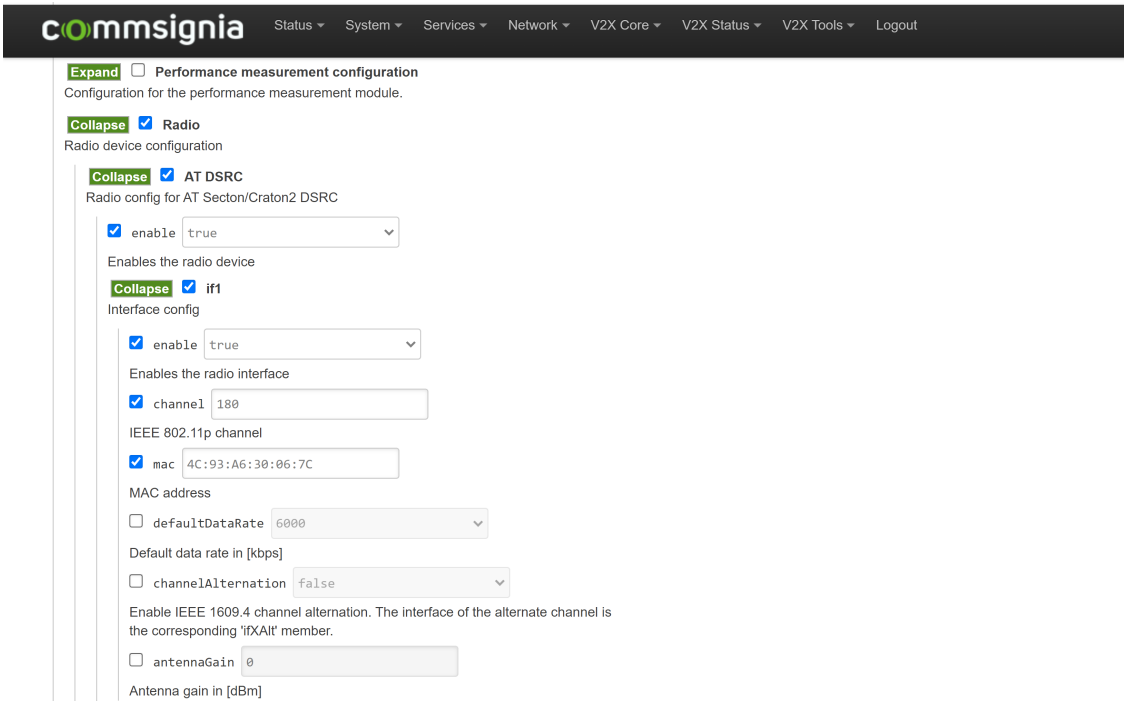

Figure 17. Autotalks CUT2 radio interface settings

For CUT2 radios the settings for the two interfaces can be configured separately. By default only the **if1** interface is enabled and all messages are transmitted and received through this interface. To enable the if2 interface expand the **if2** option and select the checkbox next to it and set the `enable` field to `true`. Channel, MAC address, data rate, antenna gain, and maximum transmission power can be set for both interfaces separately. Setting the `diversity` option to `true` set both interfaces to use the same, if1 channel to combine their signal for an improved signal reception.

c. For devices with Autotalks CUT3 radios (mainly EU, DSRC) the follow settings can be configured:
Select the checkbox near **AT DSRC CUT3** as shown in Figure 18.

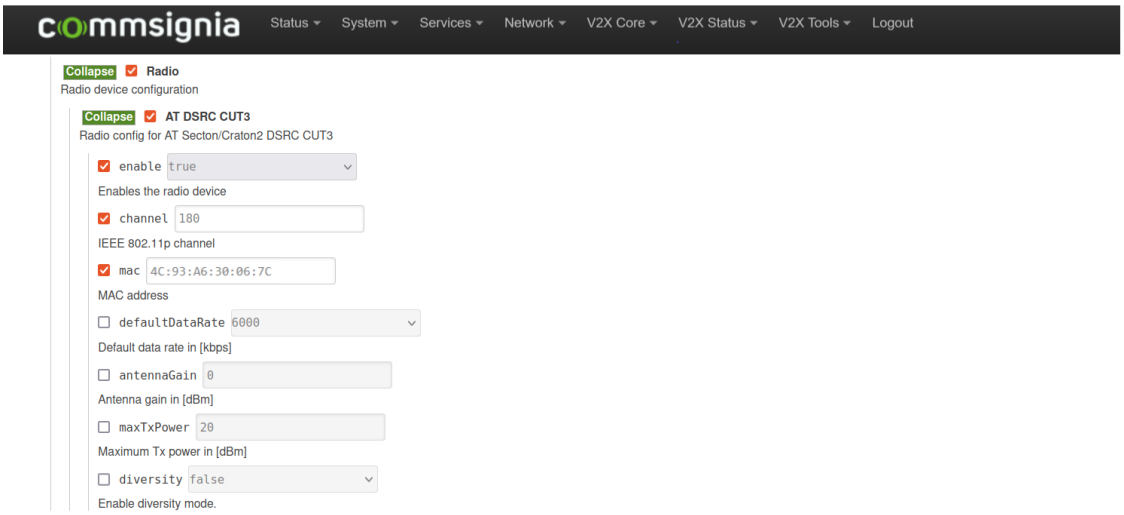

Figure 18. Autotalks CUT3 radio interface settings

Channel, MAC address, data rate, antenna gain and maximum transmission power can be set for only interface if1. Setting the `diversity` option to `true` enables the second interface to use the if1 channel to combine the signal of both interfaces for an improved signal reception. Channel alternation is not available for CUT3 radios.

3. Further configurations for US settings

   a. Turn on the WAVE Short Message Protocol (WSMP) module for US regional standard compliance, by expanding the option **WSMP** and selecting the checkbox next to **WSMP configuration**, as shown in Figure 19.

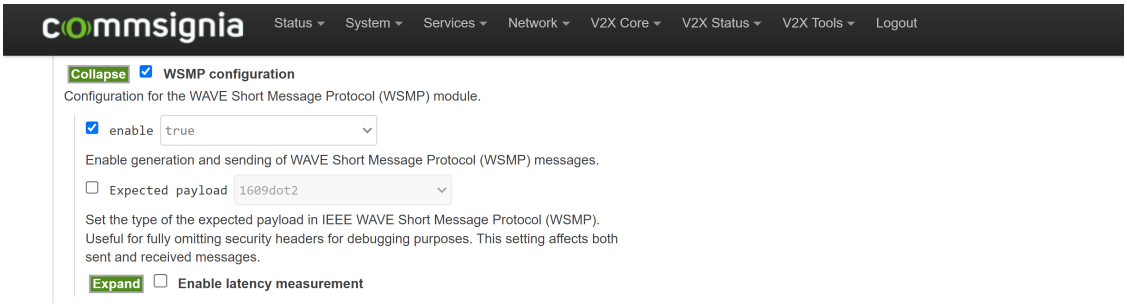

Figure 19. WSMP configuration

Select the checkbox next to `enable` and set it to `true`.

   b. Turn on the Basic Safety Message (BSM) module for US regional compliant basic V2X messages by expanding the option **BSM** and selecting the checkbox next to **BSM Transmission configuration**, as shown in Figure 20.

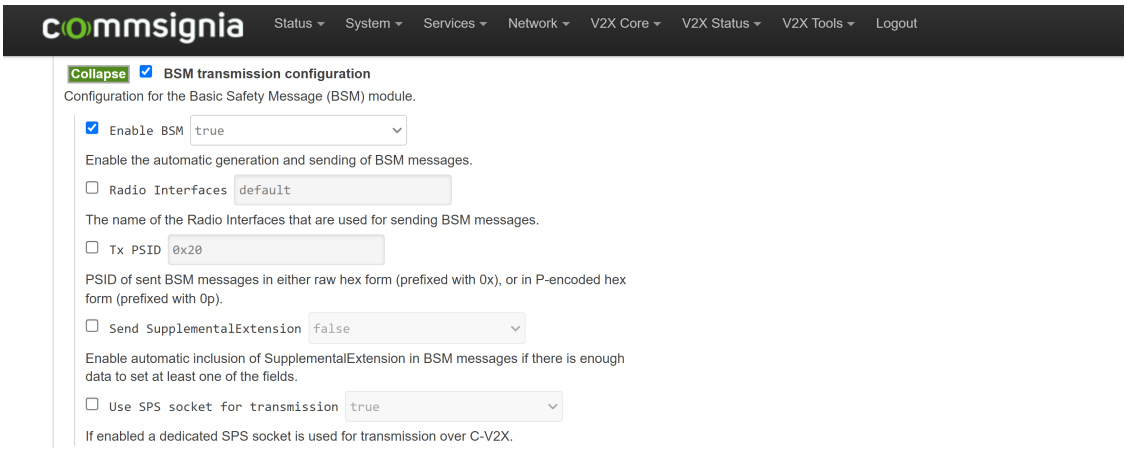

Figure 20. BSM module

Select the checkbox next to `Enable BSM` and set it to `true`.

Please note that for US regional standards Dedicated Short-Range Communication (DSRC) radio is not available.

4. Further configurations for EU settings

   a. Turn on the GeoNetworking module for EU regional standard compliance, by expanding and selecting the checkbox next to the option **GeoNetworking configuration**, as shown in Figure 21.

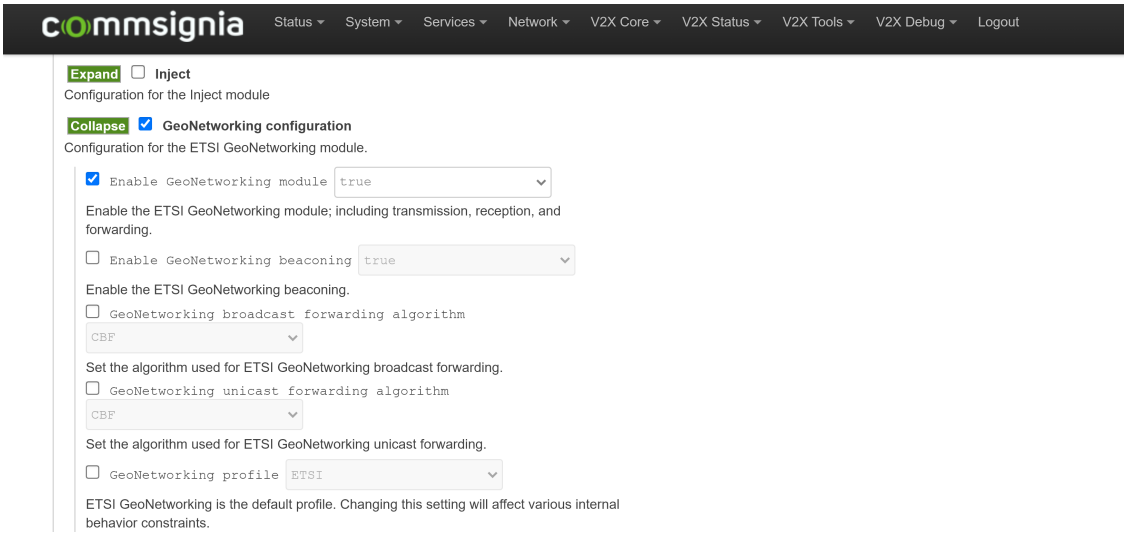

Figure 21. GeoNetworking configuration

Select the checkbox next to `Enable GeoNetworking module` and set it to `true`.

   b. Turn on the Cooperative Awareness Message (CAM) module for EU regional compliant basic V2X messages by by expanding and selecting the checkbox next to the option **CAM transmission configuration**, as shown in Figure 22.

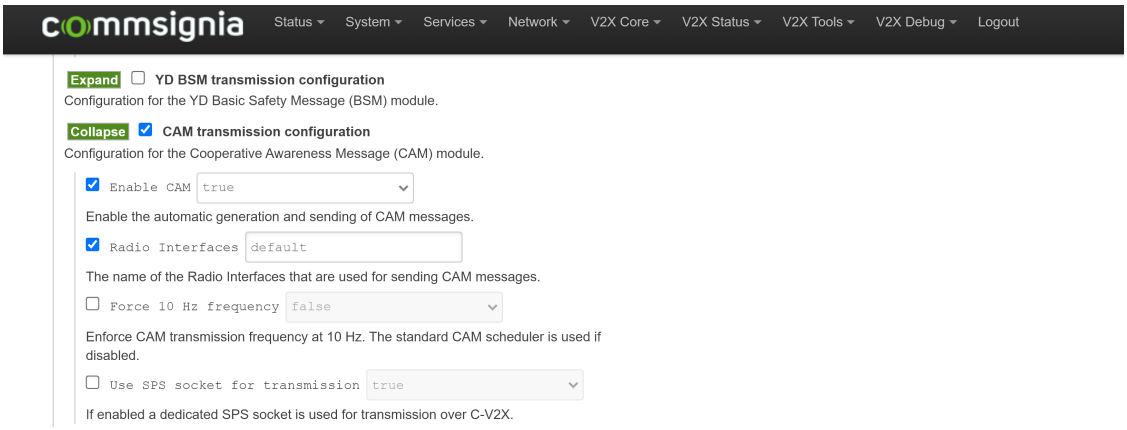

Figure 22. CAM transmission configuration

Select the checkbox next to `Enable CAM` and set it to `true`.

5. Click on the Save & Apply button for the changes to take effect.

5. Testing safety applications

Commsignia provides a set of V2X safety applications that cover the most common traffic safety related use cases.

To test the functionality of safety applications in real time, the Foresight HMI application can be used.

1. Open the Foresight application on your tablet and connect to the V2X device on which the applications are tested. For more information on connecting the device, please refer to section “Connecting to the OBU” [2].

2. After the device is connected, the Foresight application opens with the "Follow" view, as shown in Figure 23.

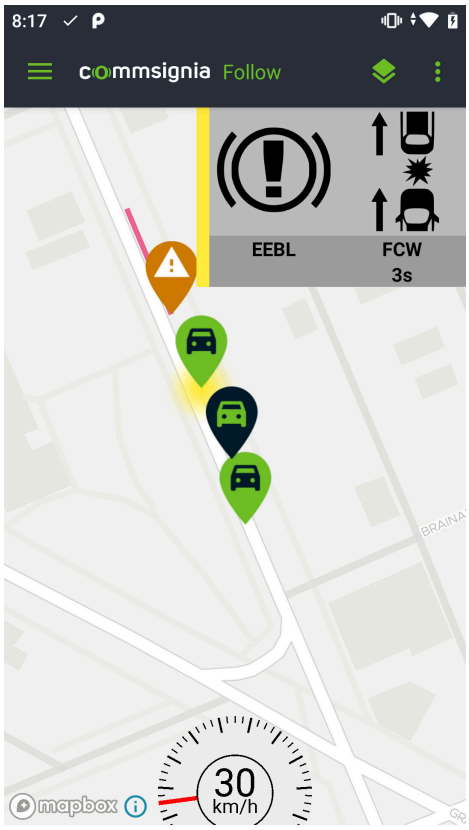

Figure 23. Follow view in the Foresight HMI application

3. Each V2X safety application has its own icon appearing at the upper right corner of the screen when the application is triggered.
   Table 1 shows the list of supported safety applications, their corresponding icons, and further information.

Table 1. Safety application notifications

Notification	Icon	Information
BSW - Blind Spot Warning		Direction (LEFT/RIGHT)

Notification	Icon	Information
EEBL - Electronic Emergency Brake Light		None
GLOSA - Green Light Optimal Speed Advise		Advised speed in km/h
IMA - Intersection Movement Assist		Time to collision
WWR - Wrong Way Remote		None
WWE - Wrong Way Entry		None
TTG - Time To Green		Time in seconds
CLW - Control Loss Warning		Time to collision
FCW - Forward Collision Warning		Time to collision
HLW - Hazardous Location Warning		None
LCA - Lane Change Assist		None
RLV - Red Light Violation		None
SW - Speeding Warning		Speed limit in km/h
LTA - Left Turn Assist		Time to collision
RTA - Right Turn Assist		Time to collision

Notification	Icon	Information
DNPW - Do Not Pass Warning		None
PCW - Pedestrian Collision Warning		Time to collision
GCW - Generic Collision Warning		Time to collision
AWW - Adverse Weather Warning		None
OHV - Overheight Vehicle		Height
REW/HLW - Hazardous Location Warning		None
RWW - Road Works Warning		None

6. Creating custom applications

In addition to the safety applications provided by Commsignia, custom safety applications can be created and added to your devices using the Commsignia software development kit (SDK). The SDK is a library to access low level functionality of the Commsignia V2X stack. It handles connecting to the stack through a proprietary TCP protocol and sends remote procedure call (RPC) commands. Commsignia provides an application programming interface (API) access using the SDK to certain functions of the software stack. The SDK also contains examples that can be built with the `make` command on your computer or the `CMake` command on the device. These examples can serve as templates to create your own applications.

To use the Commsignia SDK, the following items are required:

1. A computer running Ubuntu Linux 18.04 or later connected to a V2X device running the Commsignia software stack.
2. The Remote C SDK package provided by Commsignia, containing the headers, libraries, examples, and documentation required for development for the software stack.
3. A developer environment that can make executable files from `.c` source code and run them.

To start using the SDK, extract the Unplugged-RT Remote C SDK package in the Linuc terminal using the following command:

```
tar -xf Unplugged-RT-<Release-version>-linuxm_generic_F-remote_c_sdk.tar.xz
```

Use the applicable release version of the SDK package received from Commsignia.

The following example illustrates a basic API call that enables connection to the Commsignia software stack and gets and prints the version of the stack.

1. Add the base include file for the API:

   ```
   #include <cms_v2x/api.h>
   ```

2. Add the include file for the device status function:

   ```
   #include <cms_v2x/stat.h>
   ```

3. In the initialization phase (at the beginning of `main`) create an API session:

   ```
   cms_session_t session = cms_get_session();
   ```

4. Connect to the stack process:

   ```
   cms_api_connect_easy(&session, "127.0.0.1");
   ```

5. Get the current stack status by calling the following API function:

   ```
   cms_stat_get_device_status(&session, NULL, &data);
   ```

   In this example, the first parameter is the session and the second is an unused input parameter. This is necessary because all API calls follow the same format: session, input, output are always added even if there are no inputs for the call. This is implemented to ensure backwards compatibility. The third parameter is filled by the device status.

The call may return with an error flag. In this case an error message will be logged on the client side and possibly by the stack process as well.

6. The stack version can be printed using the `printf` function:

```
printf("The version of the running stack: %s\n",
data.device_data.version_info);
```

7. Close the connection and clean up the session:

```
cms_api_disconnect(&session);
```

8. Clean up the rest of the API resources, such as the background threads and global data structures:

```
cms_api_clean();
```

If there are subsequent `cms_api_connect` calls, do not call this function so that the global API structures remain intact and they do not need to be reinitialized.

Your application is compiled and ready to be executed. Its successful execution can be verified using the C2P interface of the stack to capture sent or received messages or by any other diagnostics tool of your choice.

7. Additional features

7.1. HDMI visualizer

The High-Definition Multimedia Interface (HDMI) visualizer function of the OBU enables the transmission of visual and audio warnings and messages from safety applications to a suitable car head-on display (HUD).

The HUD needs to meet the following specifications:

- It is equipped with a HDMI input.
- Supports a minimum resolution of 1024 × 768.
- It is equipped with an analogue stereo audio input.

7.1.1. Physical connections

Please note that the Commsignia OBU is equipped with a Micro HDMI output. Therefore, for HDMI connections, either a Micro HDMI to HDMI adapter or a Micro HDMI to HDMI cable is required.

Connect the OBU to the HUD as follows:

1. Using a HDMI cable, connect the HDMI output of the OBU to the HDMI input of the HUD.
2. Using an analogue stereo audio cable, connect the 3.5 mm stereo jack (minijack) output of the OBU to the stereo audio input of the HUD.
   Please note that the OBU provides line-level signal, and therefore appropriate volume control is required in the HUD.

7.1.2. Configuring the HDMI visualizer

The HDMI visualizer can send image and audio files to an appropriate visualizer device or HUD. The image files can be `.png`, `.jpeg`, or `.bmp`; however, for transparent background `.png` is required. Audio files can be `.wav` or `mp3`. To configure the HDMI visualizer on the OBU, proceed as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].
2. Open the **V2X Tools** → **Safety Applications HDMI visualizer** menu, expand the **config.schema.json** item, check the box next to `enable`, and set its value to `true`, as shown in Figure 24.

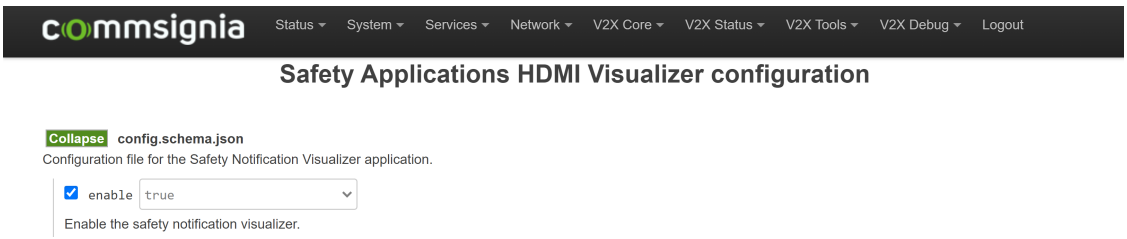

Figure 24. Enabling the HDMI visualizer

3. Configure the HDMI visualizer as follows:
   a. To configure the network settings for the safety applications, check the box next to **remoteConnection** and expand it as shown in Figure 25.

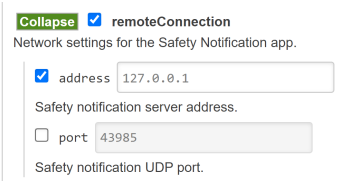

Figure 25. Network configuration for the safety applications

Here, the safety application server address and the corresponding port can be specified.

b. The location of the media files can be set by checking the box next to **mediaPath** and expanding it, as shown in Figure 26.

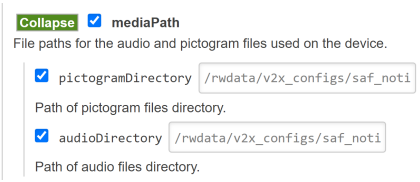

Figure 26. Path settings for the media files

i. Check the box next to `pictorgamDirectory` and specify the path for the image files.

ii. Check the box next to `audioDirectory` and specify the path for the audio files.

c. To set the minimum and maximum time for displaying the pictogram, check the box next to **pictogramDisplayDuration**, and expand it, as shown in Figure 27.

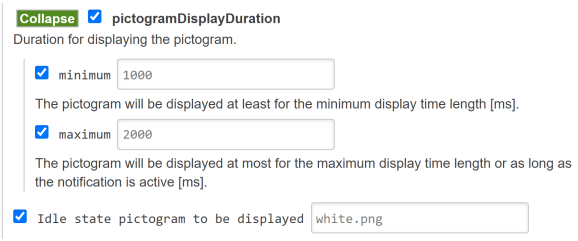

Figure 27. Setting the minimum and maximum displaying time of the pictogram

Check the box next to `minimum` and `maximum` and enter the minimum and maximum times in ms, respectively.

d. The background image for the idle visualizer can be set by checking the box next to `Idle state pictogram to be displayed`, expanding it, and entering the filename, as s shown in Figure 28.

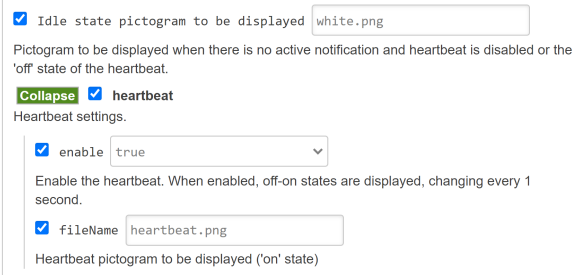

Figure 28. Settings for the idle visualizer

e. A blinking heartbeat can be set by checking the box next to **heartbeat** and expanding it, as shown in Figure 28. To enable the heartbeat, check the box next to `enable` and set its value to `true`. To set the image of the heartbeat, check the box next to `fileName` and enter the name of the image.

f. To configure the software handling the media files, check the box next to **systemUtilities** and expand it, as shown in Figure 29.

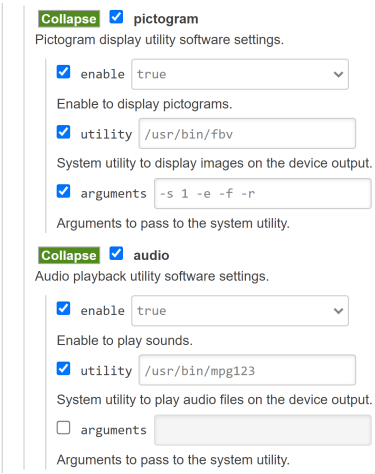

Figure 29. Configuring system utilities

i. To configure the utility software for the image files, check the box next to `enable` set its value to true, than set the path of the utility by checking the box next to `utility` and entering its path. The default **fvb** utility can be further configured by checking the box next to `arguments` and entering options. Typical options are:

-s <delay>:	Slideshow, <delay> is the slideshow delay in tenths of seconds.
-a:	Use the alpha channel; enable this option for transparent images.
-e:	Enlarge the image to fit the whole screen if necessary.
-f:	Stretch the image to fit onto screen if necessary.
-r:	Ignore the image aspect ratio while resizing.

To list all available options, log in the device using SSH (see section "Connecting to the OBU" [2]) and use to following command:

```
/usr/bin/fbv -h
```

ii. To configure the audio player, check the box next to `enable` set its value to true, than set the path of the utility by checking the box next to `utility` and entering its path. The default **mpg123** audio player can be further configured by checking the box next to `arguments` and entering options. For a full list of available options, log in the device using SSH (see section "Connecting to the OBU" [2]) and use to following command:

```
/usr/bin/mpg123 --longhelp
```

g. Configure the notifications for each safety application as follows:

i. Check the box next to **notifications** and expand it.

ii. Check the box next to the appropriate safety application and expand it. Figure 30 shows the configuration of **Forward Collision Warning** as an example.

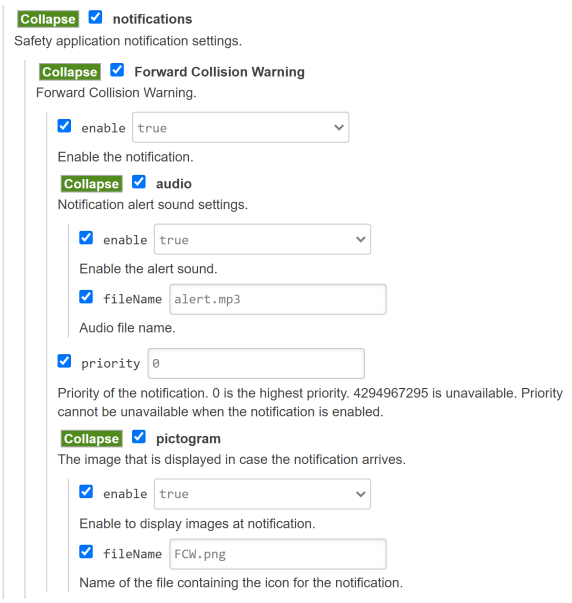

Figure 30. Notification settings for safety applications

iii. Enable the notification by checking the box next to `enable` and setiing its value to `true`.

iv. To set the audio settings, check the box next to **audio** and expand it. Check the box next to `enable` and set its value to `true`. Check the box next to `fileName` and enter the name of the audio file corresponding to the safety application.

v. To set the pictogram, check the box next to **pictogram** and expand it. Check the box next to `enable` and set its value to `true`. Check the box next to `fileName` and enter the name of the image file corresponding to the safety application.

vi. Define the priority for each safety application by checking the box next to `priority` and entering the number: the highest priority is 0.

4. Click on the Save & Apply button for the changes to take effect on the device.

7.2. Sirius XM V2X Receiver

The Sirius XM V2X Receiver is an external V2X Stack module, which connects a Sirius XM source (either real satellite receiver external device or a simulator environment) and a running V2X Stack.

Please note that this feature requires an optional external device for SIrius XM reception. For purchasing information please contact Commsignia Sales Team at sales@commsignia.com.

It connects to a Sirius XM source as an input, continuously downloads V2V BundleSets, decodes them, performs position-based filtering (if configured), and if the message has passed through the filter, it changes the validity time (if configured) and finally sends the message to V2X Stack as output.

Please note that the Commsignia OB4 firmwares contains Sirius XM V2X Receiver from version y20.34 and above; however, it is limited to ob4-generic variants only.

7.2.1. General description

The **Sirius XM V2X Receiver** module is built on the on Sirius XM SDK. First, it initializes the Sirius XM source (IP or UART) and starts receiving an V2V BundleSet. After receiving a V2V BundleSet, it extracts chunks from it and places successfully received chunks of V2V BundleSets in an `incomingData` container.

Next, the **V2V Decoder** obtains a chunk from the `incomingData` container and decodes it (ASN.1 UPER). It extract the V2V Message(s) from the chunk and places the successfully decoded V2V Message(s) into the `extractedV2VMsgs` container.

Finally, the V2V Sender obtains the decoded V2V Message(s) from the `extractedV2VMsgs` container and decodes the IEEE 1609.2 payload and the contained TIM from the V2V Message. Please note that only SignedData and UnsecuredData is supported. If enabled in the configuration, it also filters the TIM; if the filter is disabled, then the message will pass through the filter. Please note that filtering requires that the device has a valid GNSS fix/navigation data; otherwise, the message will be dropped. If enabled in the configuration, the `durationTime` field of the TIM message is overridden. If the message is passed through the filter, then it will be sent to the V2X Stack via the CMS Remote API (using `cms_rio_inject()` function calls).

7.2.2. Prerequisites

For receiving Sirius XM messages on the OBU, the following items are required:

- Commsignia OBU device with at least y20.34 firmware, ob4-generic variant
- Sirius XM source:
  - Satellite dish and clear view to the Sirius XM satellites for satellite use
  - or a simulator environment

7.2.3. Configuration

The Sirius XM V2X Receiver can be configured using the graphical user interface (GUI) or using a command line interface (CLI) either using the muci tool or by directly editing the JSON schema.

7.2.3.1. Connecting the external satellite receiver

To connect the external satellite receiver, proceed as follows:

1. Connect the external satellite receiver device using an USB cable to one of the USB external ports of the OBU.
2. Determine the device node of the external satellite receiver as follows:
   a. Access the kernel log using the GUI as:
      i. Log into the GUI. For more information, refer to section *Connecting to the OBU* in the OBU User Guide.
      ii. Open the Kernel log page under the menu **Status** → **Kernel log**
   b. Alternatively, access the kernel log using the CLI as:

i. Log into the OBU using SSH as described section *Connecting to the OBU* in the OBU User Guide.

ii. Use the command `dmesg | grep FTDI`

3. In the following example output, the device node of the external satellite receiver is `ttyUSB0`:

```
[10170.987708] USB Serial support registered for FTDI USB Serial Device
[10170.987915] ftdi_sio 9-1:1.0: FTDI USB Serial Device converter detected
[10170.991172] usb 9-1: FTDI USB Serial Device converter now attached to ►
ttyUSB0
[10170.991219] ftdi_sio: v1.6.0:USB FTDI Serial Converters Driver
```

This device node needs to be set in the configuration of the OBU, described in the next section.

7.2.3.2. Enabling the Sirius XM V2X Receiver using the GUI

To configure the Sirius XM module proceed as follows:

1. Log into the GUI. For more information, refer to section *Connecting to the OBU* in the OBU User Guide.

2. Open the configuration page under the menu **V2X Tools** → **Sirius XM V2X Receiver** as shown in Figure 31.

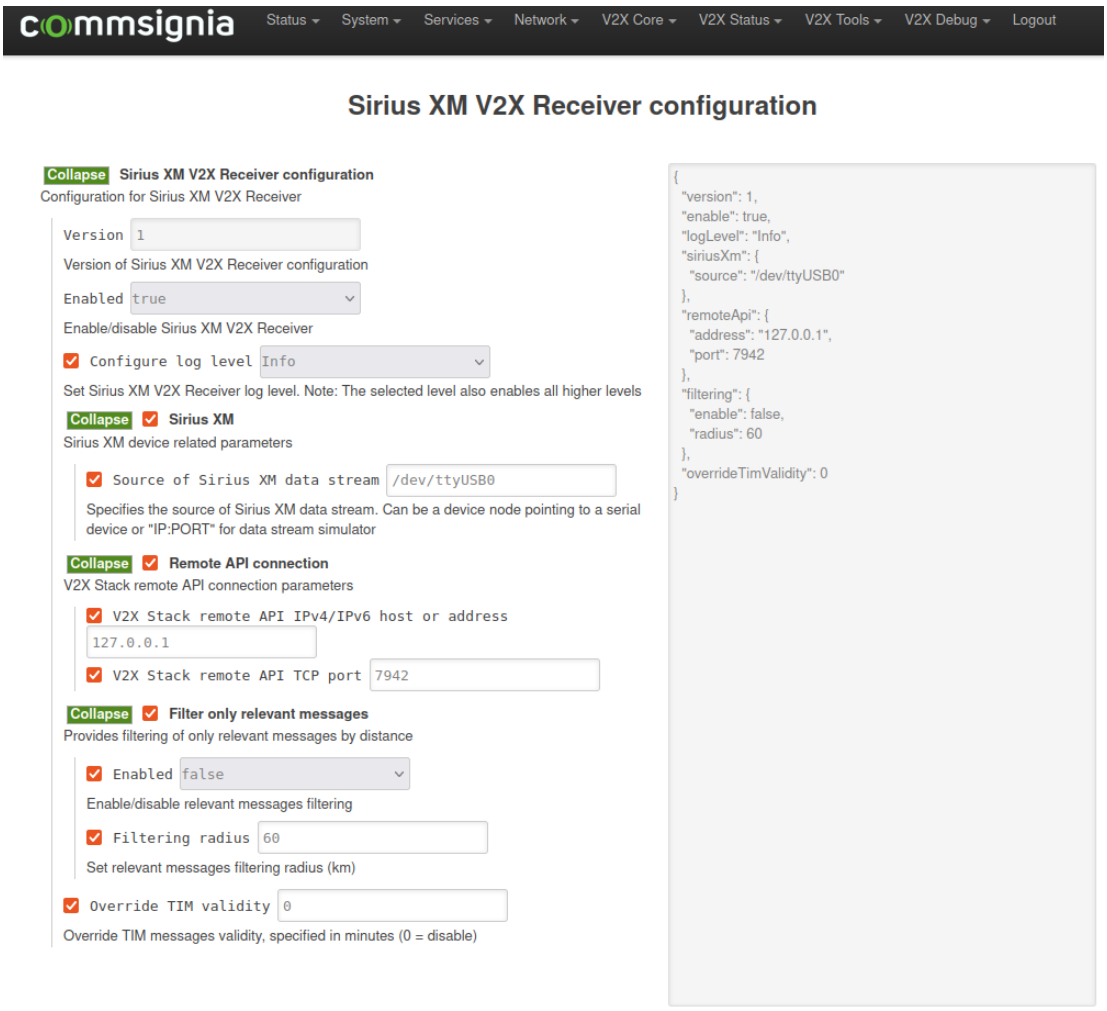

Figure 31. Configuration page of the Sirius XM V2X Receiver

3. To enable the Sirius XM V2X Receiver, set the field "Enabled" to `true`.

4. Set the device node of the external satellite receiver by checking the box next to the item "Source of Sirius XM data stream" and entering the value determined in step 3 in section "Connecting the external satellite receiver" [29] after the prefix `/dev/`.

5. Click on the Save & Apply button for the changes to take effect on the device.

7.2.3.3. Enabling the Sirius XM V2X Receiver using the CLI and the muci tool

Log into the OBU using SSH as described section *Connecting to the OBU* in the OBU User Guide.

The **muci** tool support supports the Sirius XM V2X Receiver configuration from version v2.0.4. For the configuration of the Sirius XM V2X Receiver, the `sxm` subcommand can be used with the following arguments:

```
muci sxm get <key>
muci sxm set <key> <value> [<key> <value>]
muci sxm del <key>
muci sxm find <filters>
```

To check the status of the Sirius XM module, use the following command:

```
muci sxm get
```

which prints all settings of the Sirius XM module as:

```
version = 1
enable = false
logLevel = "Info"
siriusXm.source = "/dev/ttyUSB0"
remoteApi.address = "127.0.0.1"
remoteApi.port = 7942
filtering.enable = false
filtering.radius = 60
overrideTimValidity = 0
```

To enable the Sirius XM V2X Receiver, use the **muci** command

```
muci sxm set enable true
```

Set the device node of the external satellite receiver using the following command:

```
muci sxm set esiriusXm.source "/dev/ttyUSB0"
```

where the `ttyUSB0` string needs to be replaced by the device node determined in step 3 in section "Connecting the external satellite receiver" [29].

For the changes to take effect on the device restart the stack as:

```
unplugged-rt-restart.sh
```

7.2.3.4. Enabling the Sirius XM V2X Receiver by directly editing the JSON schema

Log into the OBU using SSH as described section *Connecting to the OBU* in the OBU User Guide.

Using an editor, edit the JSON file `/rwdata/v2x_configs/sirius-xm-v2x-receiver.json`:

```
{
    "version": 1,
```

```
    "enable": false,
    "logLevel": "Info",
    "siriusXm": {
        "source": "/dev/ttyUSB0"
    },
    "remoteApi": {
        "address": "127.0.0.1",
        "port": 7942
    },
    "filtering": {
        "enable": false
    },
    "overrideTimValidity": 0
}
```

To enable the Sirius XM module change the value of `"enable":` to `true`.

Please note that the configuration has version, currently only version 1 is supported.

To set the device node of the external satellite receiver, replace the `ttyUSB0` string in the `"source"` line by the device node determined in step 3 in section "Connecting the external satellite receiver" [29].

For the changes to take effect on the device restart the stack as:

```
unplugged-rt-restart.sh
```

7.3. Configuring GPIO-triggered TSP/EVP applications

Using the GPIO interface of the OBU can add further functionalities to signal request applications as it can trigger or block them. Typical examples are as follows:

- To prevent an off-duty emergency vehicle from triggering an EVP in an intersection, the GPIO can provide an additional triggering source using the ON/OFF signal from the light bridge.
- To prevent a transport vehicle waiting in a bus bay before an intersection from triggering a TSP, the GPIO can can provide an additional source to block the triggering of the TSP by using the ON/OFF signal from the door opening circuitry.

7.3.1. Physical connections of the GPIO interface

7.3.1.1. General description of the GPIO connection

The OBU is equipped with a 16-pin Molex Micro-Fit 3.0 GPIO connector. Its position on the front plate of the OBU and its pin numbering scheme can be seen in Figure 32.

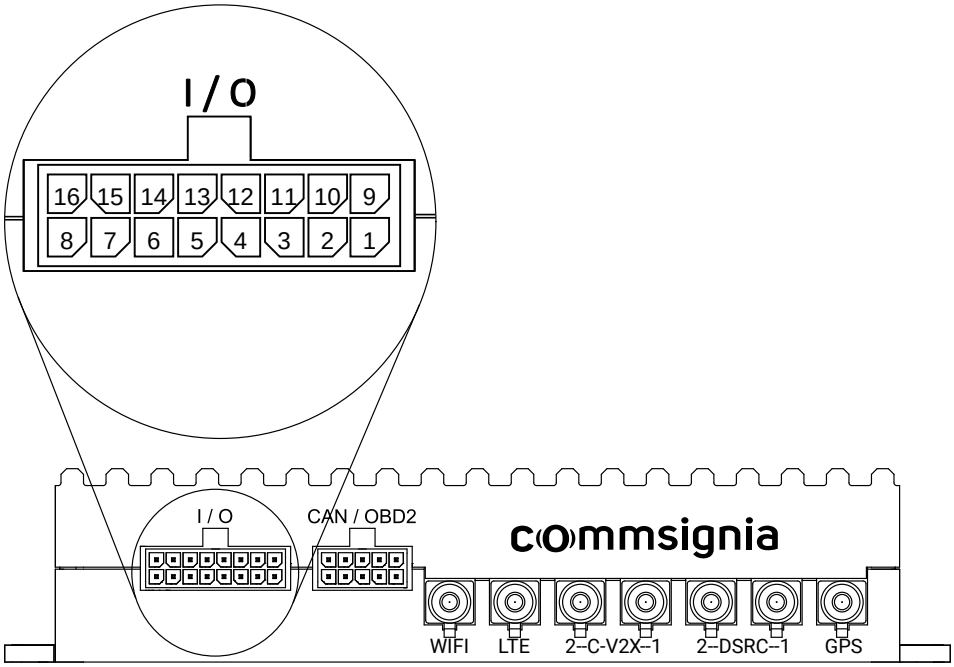

Figure 32. Position and pin numbering of the GPIO connector

In addition, a 16-conductor, 22AWG, 2 m long GPIO cable is provided with the OBU, with a 16-pin Molex Micro-Fit 3.0 plug fitted on one end, while the other and is cut without any connectors. The I/O assignment of the GPIO connector on the OBU can be seen in Figure 33 and the corresponding color coding of the GPIO cable and the GPIO numbers are summarized in Table 2.

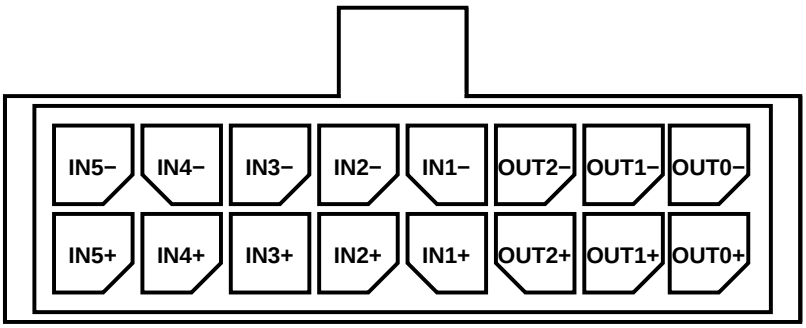

Figure 33. I/O assignent of the GPIO connector

Table 2. Pin and cable assignment of the GPIO connector

Pin No	Function	Cable color code
1	OUT0+	White
2	OUT1+	Brown
3	OUT2+	Green
4	IN1+	Yellow
5	IN2+	Gray
6	IN3+	Pink
7	IN4+	Blue
8	IN5+	Red

Pin No	Function	Cable color code
9	OUT0−	Black
10	OUT1−	Violet
11	OUT2−	Gray/pink
12	IN1−	Red/blue
13	IN2−	White/green
14	IN3−	Brown/green
15	IN4−	White/yellow
16	IN5−	Yellow/brown

7.3.1.2. GPIO connections for TSP/EVP applications

The triggering/blocking of signal requests for TSP/EVP applications can be selected in the range of IN1–IN5 and the default setting is IN1+, as given in this example an shown in Table 2. For the triggering/blocking of TSP/EVP applications the pin, input, and cable color assignment is shown in Table 3.

Table 3. Pin, input, and cable color assignment for TSP/EVP applications

Pin No	Function	Cable color code
4	IN1+	Yellow
12	IN1−	Red/blue

For the physical connection proceed as follows:

1. Connect a switchable voltage source in the range of +5–32 V DC to the yellow wire of the GPIO cable and connect the red/blue wire to a DC ground as shown in the schematics in Figure 34.

> Outputs and inputs can handle voltages up to 32 V DC. Do not connect voltages higher than 32 V DC to the wires!
>
> Please note that the logical low is represented by a disconnected state.

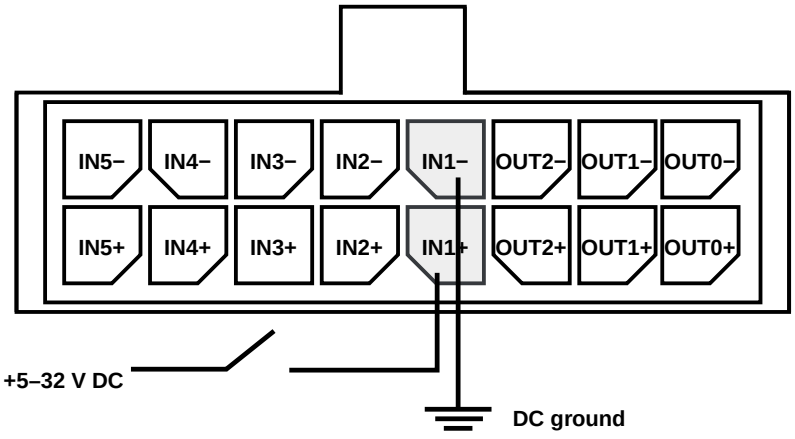

Figure 34. Schematics of the physical GPIO connection for TSP/EVP

2. Connect the Molex connector of the GPIO cable to the connector on the OBU.

7.3.2. Configuring GPIO settings

To configure the GPIO settings on the GUI of the OBU proceed as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].
2. The GPIO settings can be configured under the **V2X Tools** → **GPIO-triggered TSP** as shown in Figure35.

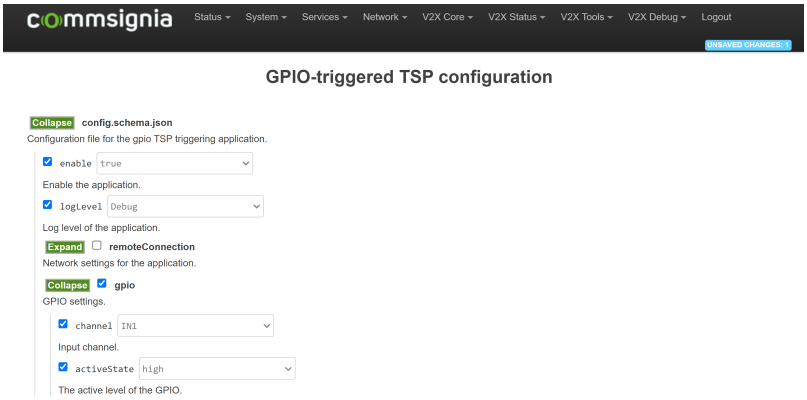

Figure 35. GPIO settings

3. Check the box next to `enable` and set its value to `true`.
4. The default GPIO settings are channel IN1 and "high" as active state. These can be changed by expanding and checking the box next to the item **gpio** as follows:
   a. Check the box next to `channel` and set its value in the range of `IN1`–`IN5`.
   b. Check the box next to `activeState` and set as `high` or `low` according to your application.

7.4. Configuring immediate forwarding

The OBU is capable of relaying appropriately formatted messages fed through the User Datagram Protocol (UDP) port of the device. These Immediate Forward Messages (IFMs) are directly forwarded to the radio and only signed if it requested.

7.4.1. Configuring the listening port

The default listen port for IFMs is 1516. To change the listen port proceed as follows:

1. Log into the GUI. For more information, refer to section "Connecting to the OBU" [2].
2. Open the **V2X Tools** → **SRM/IFM** menu item as shown in Figure 36.

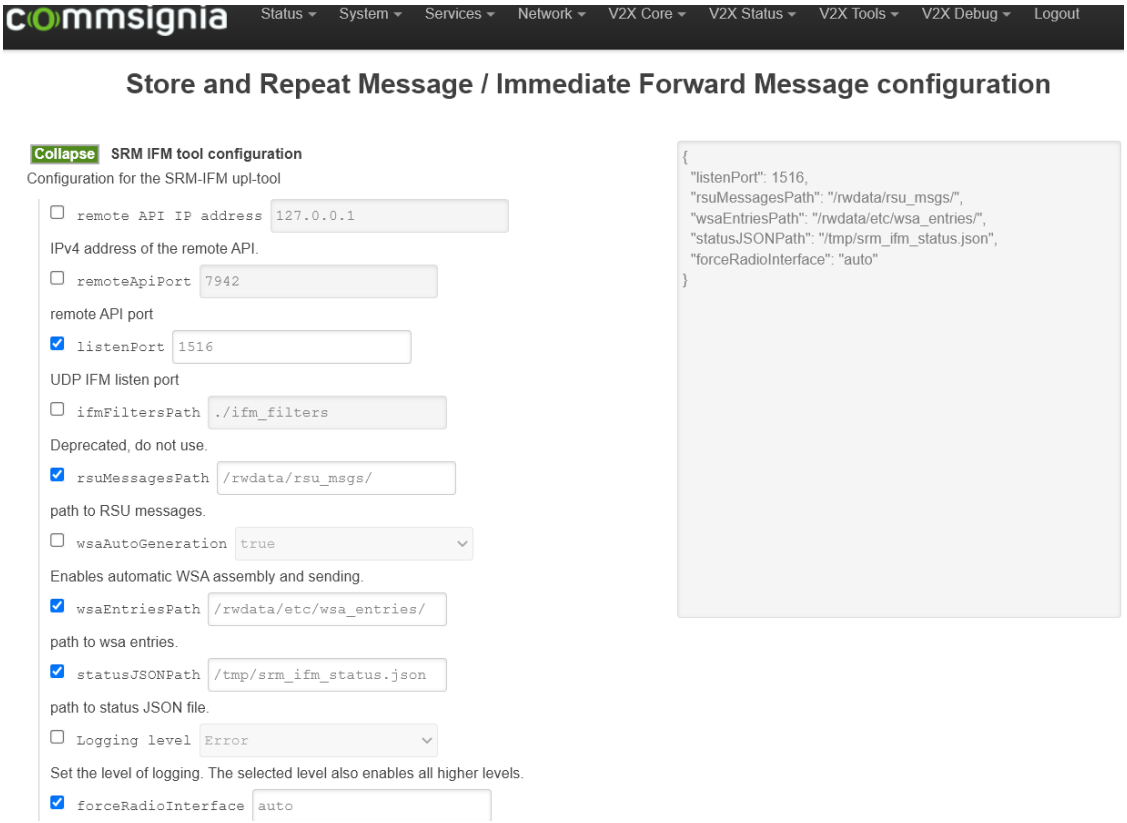

Figure 36. SRM/IFM configuration menu

3. Select the checkbox next to `listenPort` and enter a new value.
4. Click on the Save & Apply button for the changes to take effect on the device.

Alternatively, the muci tool can be used as follows:

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].
2. Set a new port number by running the following command:

```
muci set ifm.listenPort <new port number>
```

3. Restart the stack using the command

```
unplugged-rt-restart.sh
```

7.4.2. Formatting the message

Please note that all IFMs need to be in RSU 4.1 format. The message format needs to match the regional standard.

The following example shows a US-standard IFM:

```
Version=0.7
Type=BSM
```



```
PSID=20
Priority=0
TxMode=CONT
TxChannel=SCH
TxInterval=0
DeliveryStart=
DeliveryStop=
Signature=False
Encryption=False
Payload=0012...
```

The following example shows an IFM adapted for ETSI standard messages:

```
Version=0.7
Proto=GNP
GnMethod=GBC
DestinationAreaType=circle
DestinationAreaLatitude=456612674
DestinationAreaLongitude=81757023
DestinationAreaDistanceA=1000
DestinationAreaDistanceB=0
DestinationAreaAngle=0
Type=DENM
PSID=25
Priority=1
TxMode=CONT
TxChannel=180
TxInterval=0
DeliveryStart=
DeliveryStop=
Signature=False
Encryption=False
Payload=0201...
```

As can be seen in both examples, the transmission interval (`TxInterval`) needs to be set to 0, while the `DeliveryStart` and `DeliveryStop` fields needs to be left empty.

Please ensure that the signing of the messages are compatible with the security settings of the device:

- If the system is enrolled and the security is turned on, use `Signature=True` to transfer signed IFMs. Setting the signature to `False` is typically not recommended on an enrolled device.
- If the system is not enrolled or security is turned off, use `Signature=False` to trasmit unsigned IFMs. **Do not** set the signature to `True` on such devices, as the messages cannot be signed, and fail to be transmitted.

7.5. OBU V2X logger tool

7.5.1. Introduction

The OBU V2X Logger is a Commsignia tool that logs user-defined events (e.g. incoming V2X messages, Safety Application Alerts) together with the BSMs transmitted by the OBU. It continuously collects the transmitted BSMs transmitted by the OBU in a configurable-sized ring-buffer, and in case of an event the log will contain previously transmitted BSMs.

The V2X Logger can additionally upload these logs to a user-defined remote server.

Using this tool information can be gathered about the status of the vehicle during a certain event (e.g. spot weather warning, stopped vehicle warning, etc.) or about the reaction of the driver to certain events.

The OBU Logging feature consists of the following major parts:

- a **logger** part that collects information (messages, driver alerts) and creates the required log files based on a (configured) filtering criteria and logic
- an **uploader**, that is responsible to upload the log files to a configured remote destination
- a **connection-handler**, that ensures that the OBU will connect to a remote Wifi AP if a given trigger is received (a WSA message is received)
- a **storage manager**, that ensures the storage usage of the logger application can be kept within a limit, e.g. deletes old log files

7.5.2. General description

7.5.2.1. Logging of events

The following events trigger the logging mechanism:

- Received BSMs from other vehicles
- Received V2X messages (currently supported messages: WSA and MAP messages and TIMs (see the SAE J2735 standard))
- Driver Alerts (i.e. notifications from Commsignia Safety Appllications):
  - Forward Collision Warning (FCW)
  - I2V Situational Awareness (IA)
  - Workzone Warning (WZW)
  - Spot Weather Impact Warning (SWIW).

V2X messages received from sources other than the V2X radio will be logged as well (e.g. from the Sirius XM V2X Receiver). Please note, that additional configuration may be required for these tools.

When an event occurs on the OBU, the V2X Logger logs

- The transmitted BSM history of the OBU before the event has started (the time interval can be configured, by default it is 10 s)
- The complete BSM history during the event
- Transmitted BSM history after the event finished (the time interval can be configured, by default it is 10 s).

When configuring the buffer size of the transmitted BSMs, please take into consideration that BSMs are sent with a frequency of 10 Hz.

If there are no events to be logged, the V2X Logger logs a BSM transmitted by the OBU into a separate log file (see “txBSM” log file type). The time interval of this periodic logging can be configured.

7.5.2.2. Log files

Logs are collected in separate log files by type. The directory of log collection is configurable, the default value is `/tmp/log_collector`. Closed log files are gzip compressed and moved to the upload directory (the value is configurable, the default is `/rwdata/log_upload`). The log file is closed when

- the file size reaches configured limit (default value is 100 KB)
- a UsWsa message is received (containing the WiFi mode switch service with PSID 0x4020).

The following types of log files are saved:

- **txBSM**: BSMs transmitted by the OBU. Not all BSMs are logged. The elapsed time between two logged BSMs is configurable (default value is 30 s). Logged only when no events occur, when the V2X logger is in idle state.
- **rxMsg**: Received configured V2X messages, configurable message types: UsTim, UsMap, UsWsa. Besides these, all received UsBsm messages logged.
- **bsmLogDuringEvent**: BSMs related to an event. These contain:
  - all received BSMs
  - all BSMs transmitted by the OBU in the configured timeframe of the event (default value of the timeframe is 10 s before and after the event)
- **DriverAlert**: Configured safety application notifications, configurable notifications: FCW, I2VSA, WZW, SWIW.
- **System logs**

Log files are created in a custom binary format (except for system logs, which are stored in text format). Each record in the log file has a header and a payload part. Every record is separated by an additional newline character (ASCII code 0x0A). The header part contains the length of the payload. Tables 4–6 summarize the details of the different formats.

*Table 4. **txBSM**and **bsmLogDuringEvent** parameters*

Name	Size in bytes	Description
direction	1	Direction of the BSM message. • 0: transmitted b the OBU • 1: received by the OBU
latitude	4	Latitude part of GPS position according to SAE J2735
longitude	4	Longitude part of GPS position according to SAE J2735

Name	Size in bytes	Description
elevation	4	Elevation part of GPS position according to SAE J2735
speed	2	Speed according to SAE J2735
heading	2	Heading according to SAE J2735
time in sec	4	Timestamp based on Unix epoch UTC.
millisec	2	Millisecond part of timestamp
sign status	1	Security validation result. • 0: verified • 1: unknown
payload length	2	Length of UPER encoded payload.
payload	payload length	UPER encoded messages with WSMP headers and 1609.2 frame
newline	1	Unix new line character

*Table 5. **rxMsg** parameters*

Name	Size in bytes	Description
received from	1	Source of the received message. 0: RSU 1: Satellite
latitude	4	Latitude part of GPS position according to SAE J2735
longitude	4	Longitude part of GPS position according to SAE J2735
elevation	4	Elevation part of GPS position according to SAE J2735
speed	2	Speed according to SAE J2735
heading	2	Heading according to SAE J2735
time in sec	4	Timestamp based on Unix epoch UTC.
millisec	2	Millisecond part of timestamp
verification status	1	Security validation result. • 0: verified • 1: unknown
payload length	2	Length of UPER encoded payload.
payload	payload length	UPER encoded messages with WSMP headers and 1609.2 frame
newline	1	Unix new line character

*Table 6. **DriverAlert** parameters*

Name	Size in bytes	Description
latitude	4	Latitude part of GPS position according to SAE J2735
longitude	4	Longitude part of GPS position according to SAE J2735
elevation	4	Elevation part of GPS position according to SAE J2735
speed	2	Speed according to SAE J2735
heading	2	Heading according to SAE J2735
time in sec	4	Timestamp based on Unix epoch UTC.
millisec	2	Millisecond part of timestamp
payload length	2	Length of UPER encoded payload.
payload	payload length	UPER encoded messages with WSMP headers and 1609.2 frame
newline	1	Unix new line character

System logs are stored in the usual text-based Syslog format (RFC3164). The following example shows a system log:

```
Sat  1 Apr 13:15:00 UTC 2023 local7.info ITS-OB4-M-1001718 its: Unplugged-
RT y20.34.3-b187909 on linuxm ob4_secton platform
```

File names contain the following:

- type of the log file (e.g. rxMsg, bsmLogDuringEvent, txBsm, driverAlert)
- unique identifier of the OBU (the serial number, e.g. 2229401003605)
- timestamp of log file creation (format is ISO 8601 timestamp, e.g. 2017-06-30T19:53:00.000)

The following example shows a filename:

```
rxV2xMessage_2023-04-01T13:15:22.405_2229401003605
```

7.5.2.3. Uploading

In addition, the V2X Logger tool can upload the compress log files to a pre-configued remote server.

The uploading is performed by the Rsync program via an SSH connection. For the SSH connection, publickey authentication method is used.

The V2X Logger tool is primarily designed to upload files via the WiFi interface of the OBU. The Wifi interface runs in access point mode by default and can be switched to station/client mode upon receiving WSA messages (that contain the PSID of a given service) by the WSA WiFi mode switcher tool.

For this, the WSA WiFi mode switcher tool has to be configured besides the V2X Logger tool.

The upload is triggered by a WSA message containing the WiFi mode switch service (the default PSID is 0x4020); however, during the startup of the V2X Logger, it checks the network connection to the remote server and uploads the log files if a connection has already been established.

Upon receiving a WSA message from an RSU with a WiFi mode switch service, the OBU changes the mode of its WiFi interface from access point mode to client mode and expects its network interface to be configured by the DHCP server on the RSU (e.g. network address, default gateway).

Upon a successful upload, the log files are removed from the OBU.

To limit storage usage, log files are removed after seven days by default; however, this duration can be configured.

There is an upload priority for different log file types. The order of log upload (starting with the highest priority) is as follows:

- DriverAlert
- txBSM
- bsmLogDuringEvent
- rxMsg
- System logs,

Multiple log files from the same type are uploaded starting from the latest log file to the oldest.

7.5.3. Configuration

The additional log upload feature of the V2X Logger is performed by two Commsignia tools: the V2X Logger tool and the WSA WiFi mode switcher. The uploading of the log is performed by the V2X Logger tool; however, the triggering of the log file uploads upon receiving the "WiFi-switch" WSA message is performed by the WSA WiFi mode switcher tool. Therefore, these two Commsignia tools need to be configured for a complete logging solution.

For the complete logging solution, the following configuration steps are required:

1. Configuration of the V2X Logger - WSA Wifi mode switcher integration with the additional integration script `v2x_logger_wsa_integration.sh`. This script sets certain specific configuration options for these tools (e.g. to trigger file closing and file upload on the WiFi mode switch WSA message). For the configuration proceed as follows:

   a. Copy the received the shell script onto the OBU (e.g. via SCP) from the folder `Firmware/OBU`

   b. Log into the OBU using an SSH connection

   c. Execute the shell script

   ```
   /tmp/v2x_logger_wsa_integration.sh
   ```

   which gives the following output:

   ```
   Configuring V2X Logger - WSA Wifi Mode Switcher integration ...
   V2X Logger - WSA Wifi Mode Switcher integration successfully done
   ```


2. Configuration of the WSA message to be sent (RSU-side configuration, optional)
3. Configuration of the V2X Logger tool (including setting up the SSH public key authentication)
4. Configuration of the WSA WiFi mode switcher tool

7.5.3.1. Generation of the WSA message

Starting from the Commsignia RS4 firmware, version y20.34.3-b187909, the generation of the WSA message can be performed on the RSU. The format of this message is accepted by the SRM-IFM tool on the RSU.

Please note that this guide only discusses the WSA message generation and the configuration of the SRM-IFM tool is not included.

For the generation of the WSA message, use the following command with the the SSID that will be used for the log upload (such as the broadcasted SSID of the RSU, "ITS-RS4-M-0001234" in this example):

```
generate_wifi_wsa.sh -s ITS-RS4-M-0001234 -o /etc/rsu_msgs/WIFI_WSA.msg
```

7.5.3.2. Configuring the V2X Logger

To configure the V2X Logger, proceed as follows:

1. Log into the GUI of the OBU. For more information, refer to section *Connecting to the OBU* in the OBU User Guide.
2. Open the V2X Logger configuration page under menu item **V2X Tools** → **V2X Logger** as shown in Figure 37.

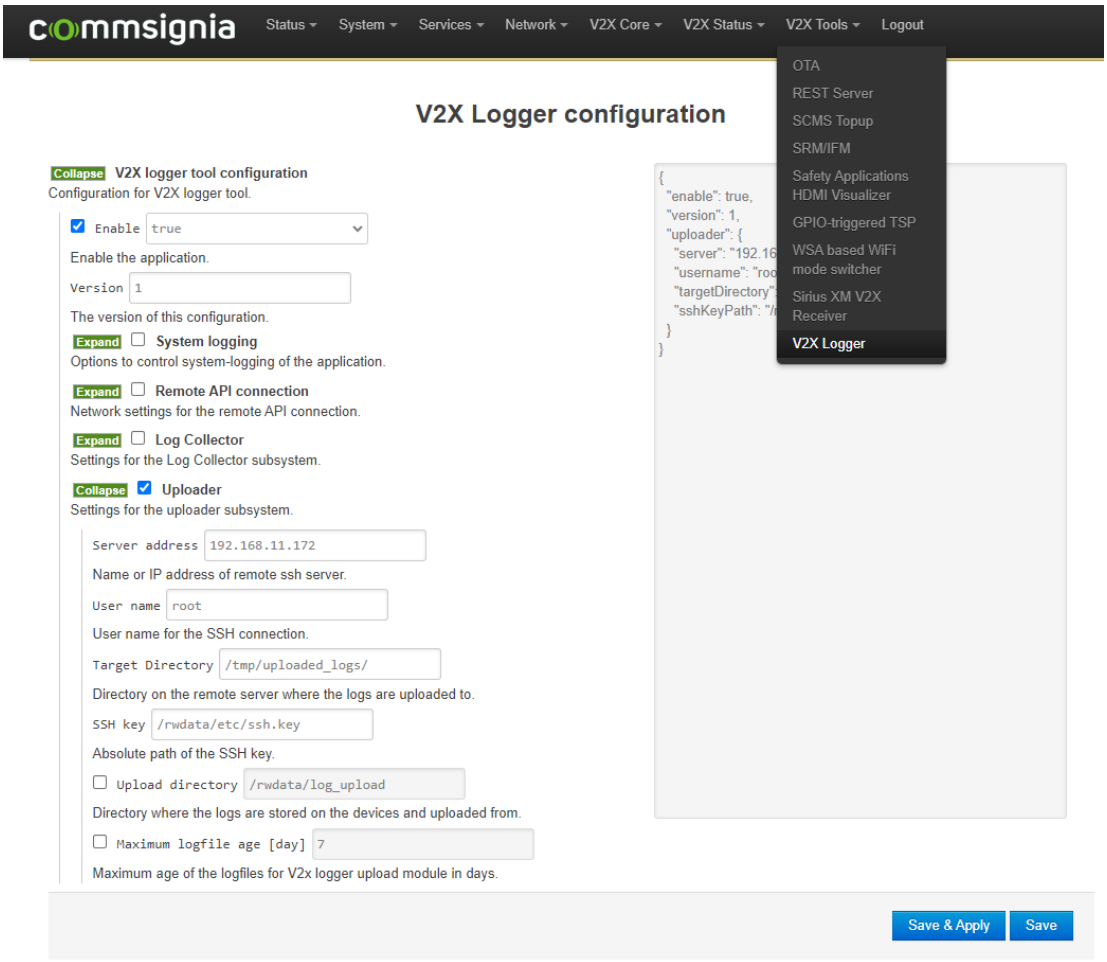

Figure 37. V2X Logger configuration page

3. Enable the V2X Logger by checking the box next to "Enable" and setting its value to `true`.

4. Configure further options according to Table 7.

Table 7. V2X Logger configuration options

Group	Parameter	Necessity	Note
	Enable	Mandatory	Set this to true to enable the module. By default it is turned off.
System logging	Logging level	Optional	Default value = Info The selected level also enables all higher levels. Set this to Debug if you experience any malfunction.
Remote API connection		Optional	Default values can be applied.
Log Collector	Working directory	Optional	By default log files are stored on tmpfs (`/tmp/log_collector`). Every file will be gzipped and moved into "Upload directory" (see later), if:

Group	Parameter	Necessity	Note
			• the file size limit is exceeded, or • the V2X Logger is reset externally (e.g. WiFi mode is switched triggered by WSA), or • the V2X Logger is stopped/restarted (e.g. OBU is turned off)
	Maximum file size [kB]	Optional	Default value = 100 kB (should meet the requirement).
	Interaction log-duration [s]	Optional	Default value = 10 s (should meet the requirements). New interactions received within the defined interval will reset the timer.
Transmitted BSM messages	Log Interval [s]	Optional	Default value = 30 s (should meet the requirement).
	Log prefix	Optional	Default value = "txBsm" (should meet the requirement).
BSM messages during events/ interactions	Buffering duration [s]	Optional	Default value = 10 s (should meet the requirement). Assuming 10 Hz of BSM generation frequency this translates into buffering the last 100 (duration * 10) transmitted BSMs.
	Log prefix	Optional	Default value = "bsmDuringEvent" (should meet the requirement).
Received V2X messages	Facility message types	Optional	Default value = [UsTim, UsMap, UsWsa] (should meet the requirement). Turn off any facility type to limit the amount of entries in the log.
	Log prefix	Optional	Default value = "rxMsg" (should meet the requirement).
Driver Alerts	Alert types	Optional	Default value = [FCW, I2VSA, WZW, SWIW] (should meet the requirement). Turn off any alert type to limit the amount of entries in the log.
	Log prefix	Optional	Default value = "driverAlert" (should meet the requirement).
Syslog	Log prefix	Optional	Default value = "syslog" (should meet the requirement).
Uploader	Server address	Mandatory	Name or IP address of the remote ssh/rsync server.
	User name	Mandatory	User name for the SSH connection.

Group	Parameter	Necessity	Note
	Target Directory	Mandatory	Directory on the remote server where the logs are uploaded.
	SSH key	Mandatory	Absolute path of the SSH private key. It is recommended to use ssh-keygen on the device or locally to generate the key-pair. The public key should be added on the server side.
	Upload directory	Optional	Default value is on persistent storage (`/rwdata/log_upload`). It is recommended to set this to a directory referring to persistent storage to ensure no loss of important data. To avoid flash degradation, an SD card formatted to ext4 file-system is recommended. The automatically mounted SD card is available under "`/tmp/sdcard` file path. It is recommended to insert/remove the SD card when the device is turned off
	Maximum logfile age [day]	Optional	Default value = 7 (should meet the requirement).

5. Click on the Save & Apply button for the changes to take effect on the device.

7.5.3.3. Configuring the WSA-based WiFi mode switcher

To configure the WSA-based WiFi mode switcher, proceed as follows:

1. Log into the GUI of the OBU. For more information, refer to section *Connecting to the OBU* in the OBU User Guide.
2. Open the WSA-based WiFi mode switcher configuration page under menu item **V2X Tools** → **WSA based WiFi mode switcher** as shown in Figure 38.

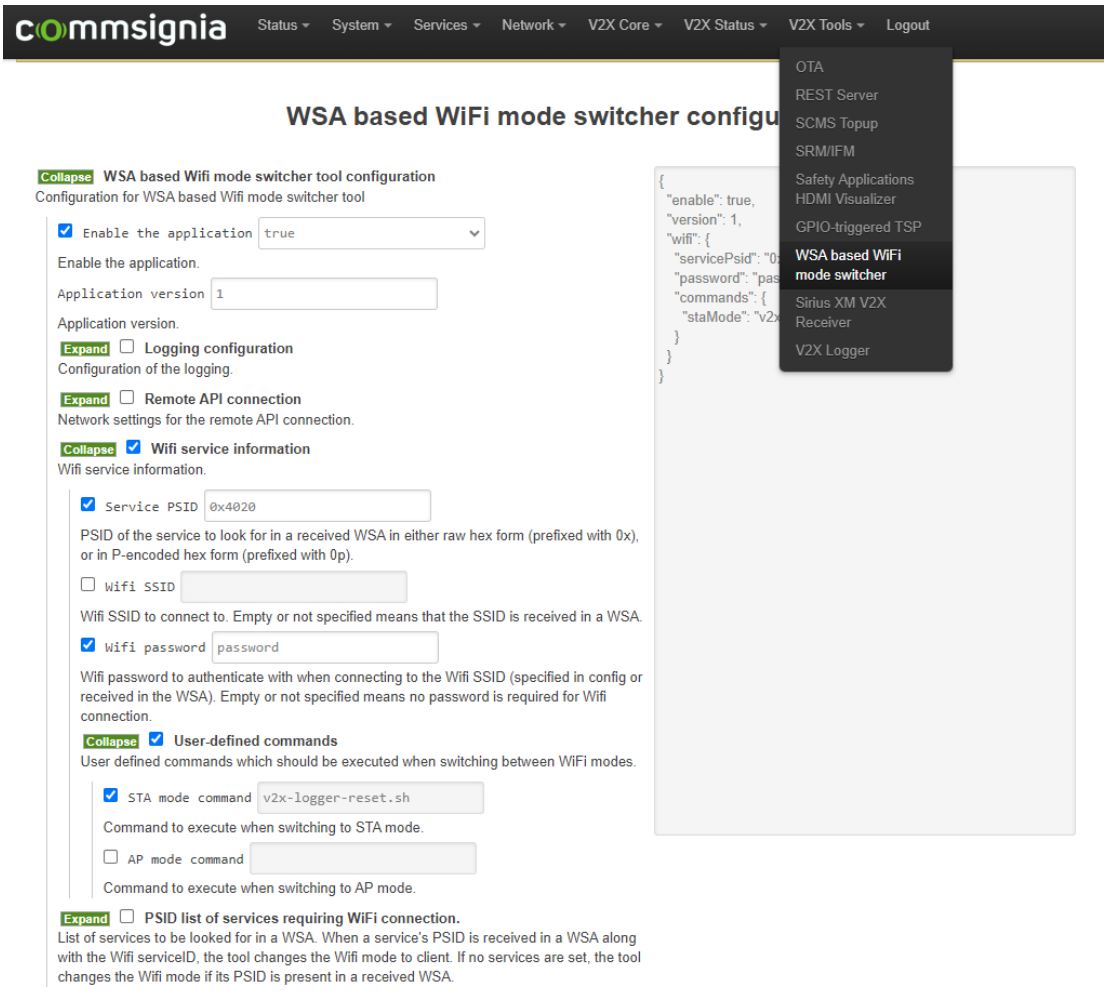

Figure 38. WSA based WiFi mode switcher configuration page

3. Enable the WSA-based WiFi mode switcher by checking the box next to "Enable the application" and setting its value to `true`.

4. Configure further options according to Table 8.

Table 8. WSA-based WiFi mode switcher configuration options

Group	Parameter	Necessity	Note
	Enable the application	Mandatory	Set this to true to enable the module. By default it is turned off.
Logging configuration	Logging level	Optional	Default value = Info The selected level also enables all higher levels. Set this to Debug if you experience any malfunction.
Remote API connection		Optional	Default values can be applied.
Wifi service information	Service PSID	Optional	PSID of the WiFi service.WSA should be generated using the provided generator script and this value should be set according to the parameters of that.

Group	Parameter	Necessity	Note
	Wifi SSID	Optional	SSID to connect. It is recommended to leave this unset and include this information in the generated WSA instead.
	Wifi password	Mandatory	Wifi password to authenticate when connecting to the SSID. Every RSU providing WiFi access should use the same password.
User-defined commands	STA mode command	Read-only	
	AP mode command	Read-only	
PSID list of services requiring WiFi connection.duringevents/interactions		Optional	Can be omitted (only relevant when using OTA feature).

5. Click on the Save & Apply button for the changes to take effect on the device.

7.5.4. Limitations

The tool has the following known limitations:

- Security validation results will not be added to the log files, its place is filled with the value “unknown” (value 1)
- Upgrade related logs are not stored in the system logs and thus, will not be uploaded
- No log rotation is done on the system log files, i.e. in case of and error that generates many log messages, this can result in a lot of system log files

8. Advanced configuration of the software stack

8.1. Upgrading the firmware

It is generally recommended to use the latest firmware version on the V2X device. The latest firmware releases are regularly announced by Commsignia.

8.1.1. Prerequisites

To update the firmware of the device, the following items are required:

- An operational OBU.
- The latest firmware obtained directly from Commsignia.
- Computer with internet connection.
- On Windows computers, SCP and SSH clients.

8.1.1.1. Checking the firmware variant

According to the filesystem and booting of the OBU, there are three firmware variants. These variants can be identified by the firmware filenames, as shown in Table 9.

Table 9. Firmware variants

Filesystem	Secureboot	Firmware filename
Read-only	Enabled	ob4-<...>-**ro-secureboot**-y<...>.tar.sig
Read-write	Enable	ob4-<...>-**rw-secureboot**-y<...>.tar.sig
Read-write	Disabled	ob4-<...>-**rw**-y<...>.tar.sig

To check the filesystem and booting of the OBU, proceed as follows:

1. Log into the OBU using SSH. For more information, refer to section "Connecting to the OBU" [2].
2. To print out the device information, use the following command:

   ```
   commsignia-device-info
   ```

   The command lists all device information; the filesystem and boot type can be found in the first section. This can be

   ```
   Secure boot: secureboot enabled
   Root filesystem: read-only
   ```

   or

   ```
   Secure boot: secureboot enabled
   Root filesystem: read-write
   ```

   or

   ```
   Secure boot: secureboot disabled
   Root filesystem: read-write
   ```

   In any other case or if these information cannot be determined, please contact Commsignia Support before commencing the firmware upgrade.

8.1.2. Upgrade process

If all prerequisites have been satisfied, upgrade the firmware as follows:

1. Obtain a firmware file from Commsignia Support.
2. Copy the firmware file to the `/tmp` directory of the OBU as follows:
   a. On a Linux computer, open a terminal and use the following command:

   ```
   scp ob4-<...>-{variant}-y<...>.tar.sig root@<IP_address_of_the_OBU>:/tmp/
   ```

   b. On a Windows computer, use an SCP client, such as WinSCP to copy the file.
3. Log into the OBU using SSH. For more information, refer to section "Connecting to the OBU" [2]. Run the `signedUpgrade.sh` command as follows:

   ```
   signedUpgrade.sh /tmp/ob4-<...>-{variant}-y<...>.tar.sig
   ```

   The script provides the following information about its progress:

   ```
   Starting signedUpgrade with /tmp/rs4-generic-rw-y20.34.5-
   b190705.tar.sig firmware
   Verified OK
   /dev/updateaux
   bin/imx6-glibc/openwrt-v2.2.3-imx6-development-initramfs-signed
   CP437: Success
   fsck.fat 3.0.28 (2015-05-16)
   /dev/bootfs: 6 files, 34498/516216 clusters
   VALIDATING RESULT
   /dev/bootdir/zImage-signed-b: OK
   /dev/bootdir/dtb-signed-b: OK
   /dev/bootdir/initramfs-signed-b: OK
   bin/imx6-glibc/uboot-imx6-apalis_imx6_it-development//u-boot-SRK_fuse.bin: OK
   bin/imx6-glibc/uboot-imx6-apalis_imx6_it-development//u-boot-
   SRK_table.bin: OK
   bin/imx6-glibc/uboot-imx6-apalis_imx6_it-development//u-boot-signed-
   mmc.imx: OK
   bin/imx6-glibc/uboot-imx6-apalis_imx6_it-development//u-boot-signed-
   usb.imx: OK
   bin/imx6-glibc/uboot-imx6-apalis_imx6_it-development//u-boot.imx: OK
   Current U-Boot version: 1
   New U-Boot version: 1
   U-Boot is up to date.
   ```

   After running, the program confirms the successful firmware upgrade as follows:

   ```
   ALL OK
   Firmware upgrade from "ob4-generic-rw-y20.23.3-b168981" to "rs4-generic-rw-
   y20.34.5-b190705": OK
   ```

4. Reboot the device with the `reboot` command as:

   ```
   reboot
   ```

8.2. Enabling security for V2X messages using the GUI

Automatic signing of all outgoing facility messages (such as BSMs and CAMs) using a trusted certificate pack can be enabled using the GUI. Please note that a verified certification pack, including a

root certificate authority (CA) certificate, is required. Commsignia provides a certification pack for testing purposes; however, in a live enrollment scenario a certificate pack is required that is acquired from a verified source. Subscribing to a certificate provider service is not in the scope of this guide. Test certificate packs provided by Commsignia need to be copied to `/rwdata/etc/security_us` or `/rwdata/etc/security_eu`.

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Open the **V2X Core** → **Core stack** menu and expand and check the box next to the **Security configuration** item, as shown in Figure 39.

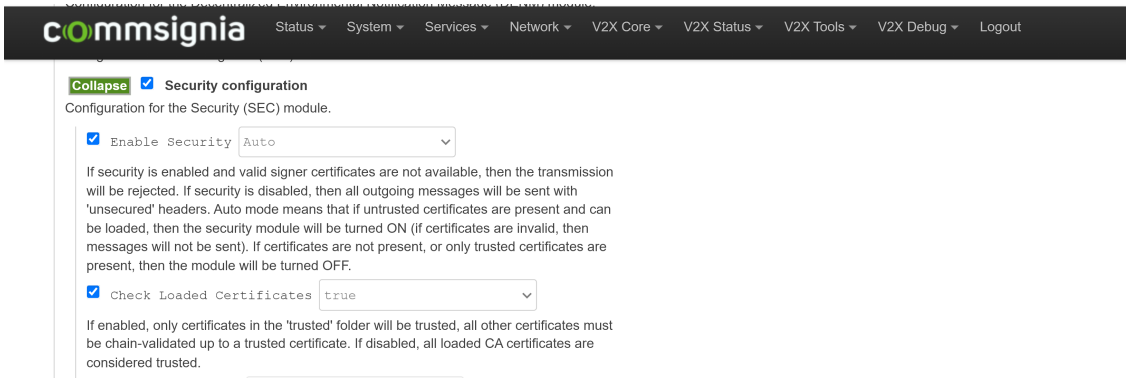

Figure 39. Security configuration

3. Check the box next to `Enable security` and set its values as follows:

   - if the value is `Auto` or `Yes` and valid certificates are present on the device, then the messages are sent signed and received messages are verified. Messages with invalid signature or without security header are dropped according to the settings of the Facility receive module.
   - if the value is `Auto` and no certificates are present on the device, then the messages are sent with unsecured headers and received messages are processed without security verification.
   - if the value is `Yes` and no certificates are present on the device, then the messages are not sent and received messages are dropped due to verification failure.
   - if the value is `Auto` or `Yes` and invalid certificates are present on the device, then the messages are not sent.
   - if the value is `No` then the messages are sent with unsecured header in any case and received messages are processed without security verification.

   If the box next to `Check Loaded Certificates` is checked and its value is set to `true`, then the stack validates the certificates against the trusted root certificates located either at `/etc/security_us` or `/etc/security_eu`. If this value is set to `true` and a test certificate pack is used, then the root certificate from the test certificate pack needs to be copied to `/etc/security_us` or `/etc/security_eu`. If this value if set to `false` than all CA certificates are considered trusted.

4. To enable the secure sending of V2X messages, the pseudonymity module needs to be enabled as well. This handle the periodic changing of MAC addresses, identifiers, certificates on the device. Expand and check the box next to the **Pseudonymity module configuration** item, as shown in Figure 40.

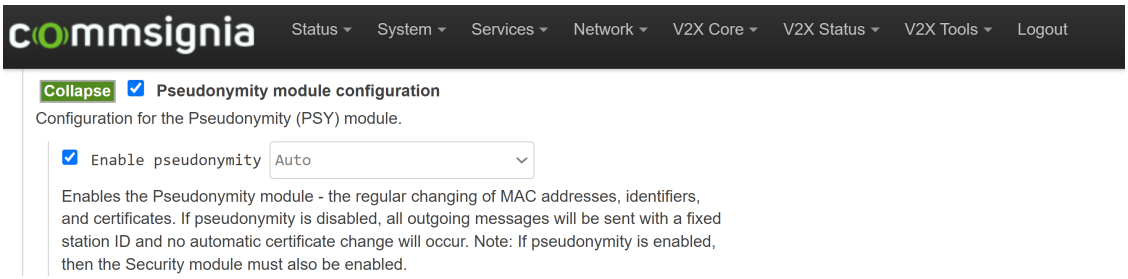

Figure 40. Pseudonymity module configuration

Check the box next to `Enable pseudonymity` and set its value as follows:

- if the value is `Auto` or `Yes` and valid pseudonym certificates are present on the device, then the messages are sent with a pseudonym method
- if the value is `Auto` and no pseudonym certificates are present on the device, then the messages are sent with previously set security configuration
- if the value is `Yes` and no pseudonym certificates are present on the device, then the messages are not sent
- if the value is `Auto` or `Yes` and invalid pseudonym certificates are present on the device, then the messages are not sent
- if the value is `No` then the messages are sent with the previously set security configuration

5. Optional security settings for transmitting messages are available by expanding the **Facility Rx configuration** item and selecting the checkbox next to it as shown in Figure 41.

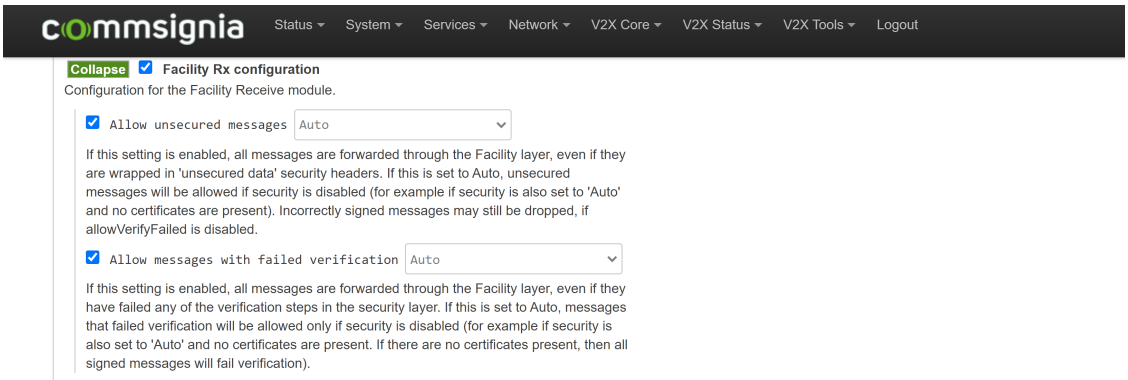

Figure 41. Facility receive module configuration

Select the checkbox next to `Allow unsecured messages` and set its value as follows:

- If the value is set to `Auto`, then unsecured messages are allowed through the Facility layer only if security is disabled (for example if security is also set to `Auto` and no certificates are present).
- If the value is set to `Yes`, then all unsecured messages are allowed through the Facility layer.
- If the value is set to `No`, then no unsecured messages are allowed through the Facility layer.

Select the checkbox next to `Allow messages with failed verification` and set its value as follows:

- If the value is set to `Auto`, then messages that failed verification in the Security layer are allowed through the Facility layer only if security is disabled (for example if security is also set to `Auto` and no certificates are present).
- If the value is set to `Yes`, then all messages that failed verification in the Security layer are allowed through the Facility layer.
- If the value is set to `No`, then no messages that failed verification in the Security layer are allowed through the Facility layer.

6. Click on the Save & Apply button for the changes to take effect on the device.

7. To verify that the outgoing V2X messages are automatically signed before they are sent, open the **V2X Core** → **Core status** menu item, as shown in Figure 42, expand **statistics**, expand **security**, and expand **1609.2**. To access the counters, expand the appropriate region (**eu/us**) under this item. The transmission of secured messages can be verified by the increase of the value of the txSignedPacket counter.

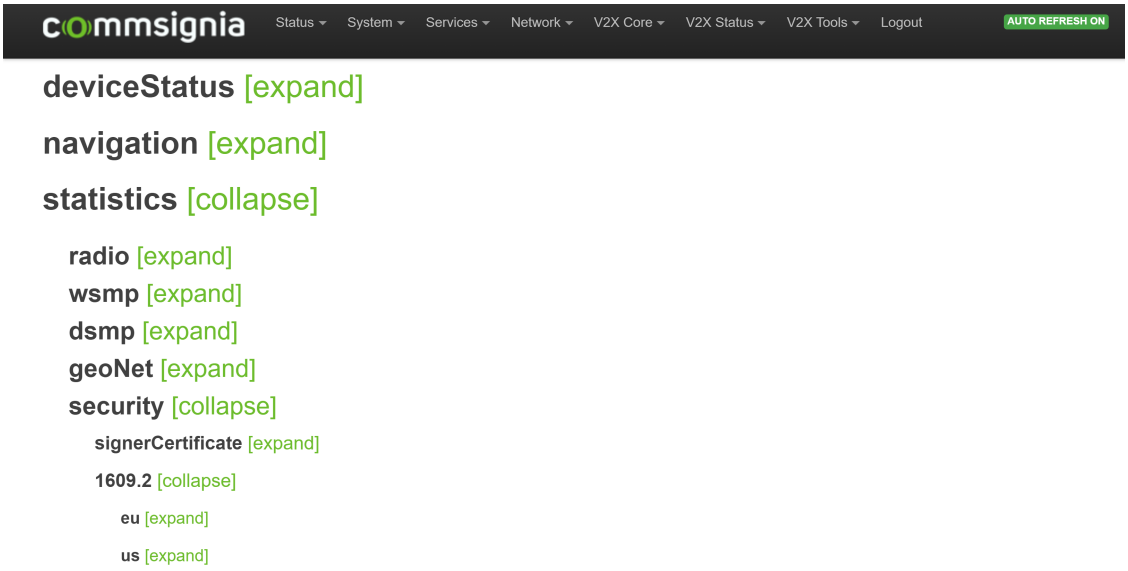

Figure 42. Statistics for secured messages

8.3. Relicensing the device

In certain cases, such as problems due to a new firmware verson or to enable new features on an older device, the OBU might require to be relicensed. For these cases Commsignia provides new license packs.

8.3.1. Prerequisites

To relicense the device, the following items are required:

- An operational OBU with GUI.
- A license pack (*.pack extension) obtained directly from Commsignia.
- Computer with web browser and internet connection.

8.3.2. Relicensing process

If all prerequisites have been satisfied, relicense the device as follows:

1. Obtain a license pack file from Commsignia Support.

2. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

3. Open the **V2X Core** → **License** menu item and click on the link "Commsignia License Activation page" as shown in Figure 43. This opens the License Activation page.

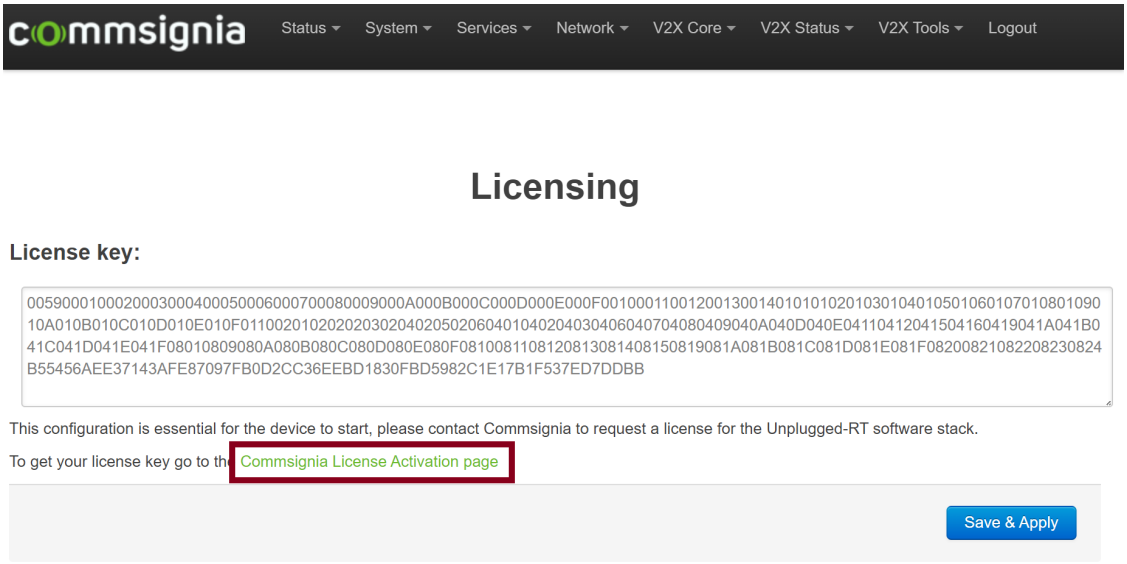

Figure 43. Licensing page on the GUI

4. On the License Activation page, drag and drop the license file (*.pack) onto the gray area or click on the area and select the file in the browser as shown in Figure 44.

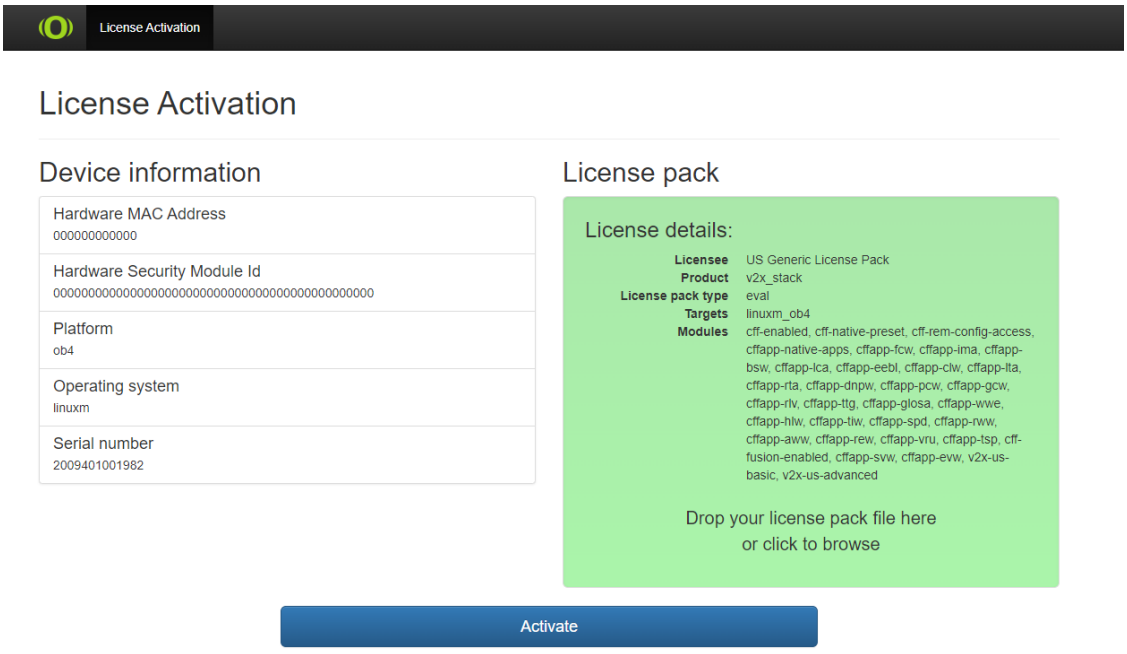

Figure 44. License Activation page

Click on the Activate button.

5. After the validation of the license pack, a new license string will be displayed, as shown in Figure 45.

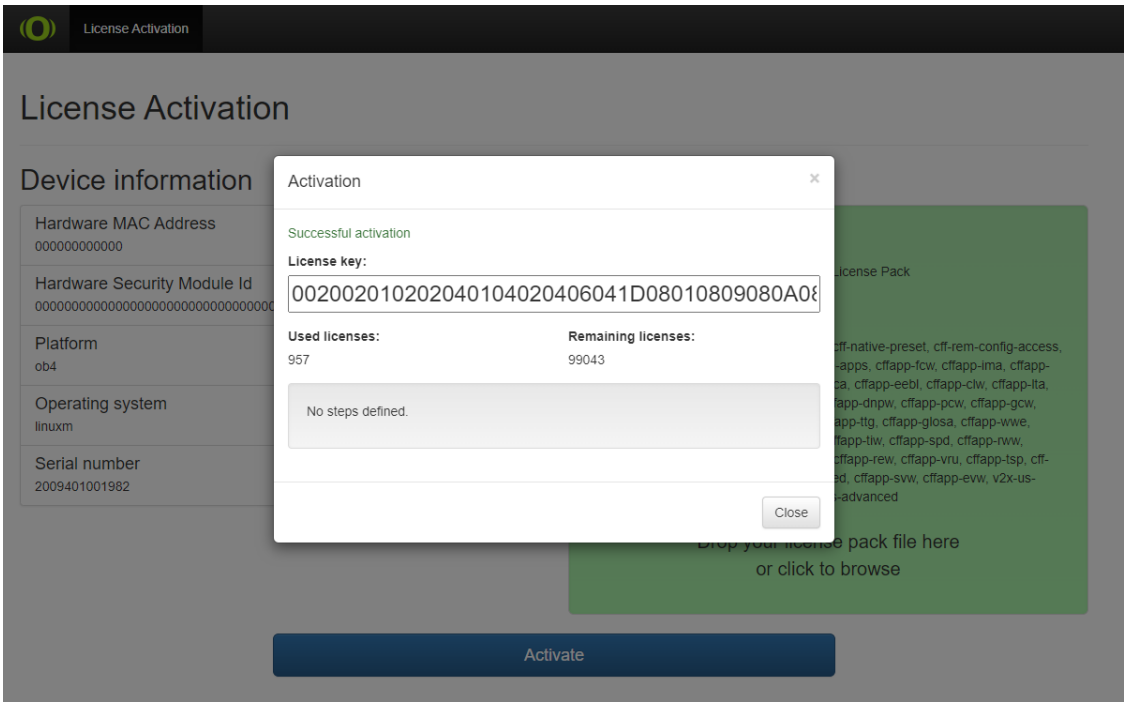

Figure 45. Generation of a new license key

Copy this license key, open the **V2X Core** → **License** menu and paste this key into the License key field, overwriting the old key.

6. Click on the Save & Apply button for the changes to take effect on the device.

8.4. Enabling IPv6 tunneling on OBUs

To use the IPv6 tunneling feature at least two OBUs are required, that are correctly configured and can communicate with each other. The IPv6 communication between OBUs is compliant with both EU and US regional standards. A RSU can be optionally included in the testing environment with IPv6 tunneling and WAVE Service Advertisement (WSA) sending enabled (see the RSU user guide for more information). Communication over IPv6 through an RSU is only compliant with US regional V2X standards.

IPv6 tunneling enables the communication of compatible V2X devices with each other using the IPv6 protocol. Using this feature, properly configured OBU devices that can successfully communicate with a compatible RSU using standard V2X messages can also access the Internet.

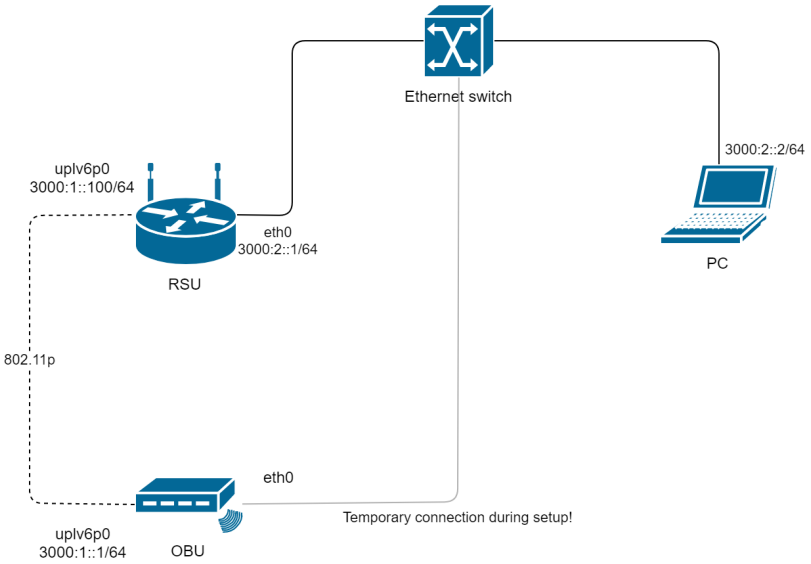

Figure 46. IPv6 tunneling setup

1. Log into the device using SSH. For more information, refer to section "Connecting to the OBU" [2].

2. Under the **V2X Core** → **Core stack** menu item expand and check the box next to **IPv6 module configuration**, as shown in Figure 47.

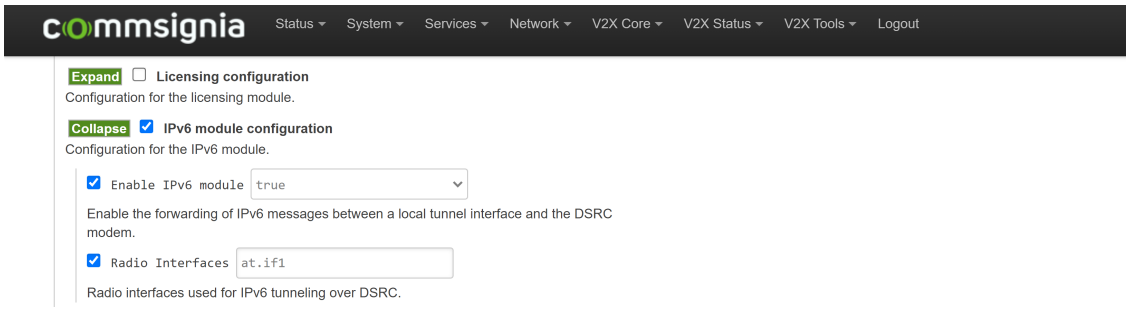

Figure 47. IPv6 module configuration

3. Check the box next to `Enable IPv6 module` and set its value to `true`.

4. Click on the Save & Apply button for the changes to take effect on the device.

The RSU is configured to communicate over IPv6 either with the central router or with nearby OBUs.

9. Troubleshooting V2X communication

This chapter details the most common problematic conditions of V2X communications using the Commsignia software stack and the possible solutions for them.

9.1. General validation steps

Go through these steps first if you experience any problems with your V2X device:

- Make sure that all antennas are properly connected according to the device's hardware description.
- Make sure the device is powered on.
- Make sure that the device and the software stack is licensed with Commsignia.
- Make sure you are connected to the same network as the device.
- Check if your navigation settings are set to "Manual" or if the device has a proper GNSS fix.

9.2. V2X messages are not secured or not transmitted

Go through the general validation steps detailed in this chapter, then check the following:

- SEC module is enabled
- PSY module is enabled
- A valid certificate pack (including a CA root certificate) is used on the device
- The same CA root certificate is used on both the transmitting and receiving test devices

9.3. The HMI is not displaying SPaT on the map

Go through the general validation steps detailed in this chapter, then check the following:

- Make sure that the HMI is connected to the Internet or it has offline maps downloaded for the location you are testing for.
- Make sure that the Intersection ID contains the same location for both the MAP and the SPaT.

9.4. The HMI displays a SPaT that is different from the actual traffic signal

Go through the general validation steps detailed in this chapter, then check the following:

- Make sure that the MAP message contains the correct lane-to-lane connection signal group ID
- Make sure that a compatible data format is used, for example Batelle or UDP
- Make sure that the signal group ID, phase, and overlap values are all correct

9.5. The HMI is not displaying the local vehicle

Go through the general validation steps detailed in this chapter, then check the following:

- Make sure that the vehicle has valid navigation data - either set in "Manual mode" or it has a proper GNSS fix
- Make sure that the SEC module and the PSY module are enabled in the configuration and there is a valid certificate pack (including a CA root certificate) on the device

- Make sure that you are connected to the correct station in the HMI

Appendix A. Glossary of terms

API	Application programming interface
ASN.1	Abstract Syntax Notation One
BSM	Basic Safety Message
C2P	Commsignia Capture Protocol
CA	Certificate authority
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CLI	Command line interface
DNS	Domain Name System
DSRC	Dedicated Short-Range Communication
GUI	Graphical user interface
HMI	Human–machine interface
IP	Internet Protocol
JDK	Java Development Kit
OBU	Onboard unit
RPC	Remote procedure call
RSU	Roadside unit
SCP	Secure Copy Protocol
SDK	Software development kit
SNMP	Simple Network Management Protocol
SSH	Secure Shell Protocol
SSID	Service set identifier
TLC	Traffic Light Controller
UDP	User Datagram Protocol
UPER	Unaligned Packed Encoding Rules
WAVE	Wireless Access for Vehicular Environment
WSA	WAVE Service Advertisement
WSMP	WAVE Short Message Protocol