GEUTEBRÜCK

Plugin Loader User Manual

Version: 1.3

05.05.2025

Table Of Contents

About This Documentation

Current software version: Geutebrück Plugin Loader 1.3.

Note that the illustrations in this documentation may not match those of your software version.

Legal Notice

This documentation may not be copied, translated or converted to a machine-readable form, whether in whole or in part, without prior permission.

GEUTEBRÜCK GmbH cannot guarantee the correctness of any information provided in this documentation, nor for the software or the information it contains. Any suggested guarantee, assurance of marketable quality or suitability for a specific purpose of the documentation, the software or other information is hereby explicitly rejected.

Under no circumstances is GEUTEBRÜCK GmbH liable for direct or indirect subsequent damage or for special subsequent damage resulting from or in association with this documentation, regardless of whether this arises as a result of illegitimate action, of a contract, or for other reasons in association with this documentation, the software or of the information contained or used within it.

GEUTEBRÜCK GmbH retains the right to change this documentation or the information contained within it at any time without warning. The software described in it is subject to the conditions of a special license contract.

Overview

With the Plugin Loader, you can integrate your camera devices into your G-Core system remotely and site-independent. You can connect several Plugin Loaders, which are distributed across different sites, to your G-Core system and control and manage them centrally (see **Installation**).

Feature Overview

Device Management:

You can add and configure your camera devices via Remote Plugins and connect them to your G-Core system (see **Add Device**).

Event Mapping:

You can receive and use metadata from Geutebrück H-Series, Hanwha or ONVIF Profile M cameras. The camera events are converted into the Geutebrück metadata format and mapped to predefined G-Core actions (see **Default Mappings**). You can also map the camera events to the G-Core actions manually (see **Add Custom Mapping**).

SDK Development:

You can independently develop camera integrations for your Geutebrück system. With the provided **SDK Backend** and **SDK Frontend** you can develop custom plugins and integrate them into your system.

Plugin Features

This table lists the features of the Remote ONVIF Plugin and compares them with the G-Core ONVIF Plugin.

= Feature is supported.
= Feature is not supported.

Feature	G-Core ONVIF Plugin	Remote ONVIF Plugin
General:		
PTZ		

Feature	G-Core ONVIF Plugin	Remote ONVIF Plugin
Area zoom and Click to move		
Multihead camera		
Video:		
Video streaming (H.264, H.265, JPEG)		
Dual streaming		
Dynamic live streaming (DLS)		
Dynamic stream selection (DSS)		
Audio:		
Audio streaming		
Audio backchannel (2-way audio)		
Event handling:		
Camera-side motion detection		
Event handling		
Event filter	per camera	per Plugin Loader
Event mapping to G-Core actions		
Additional:		
Digital inputs and outputs		
Edge recording		
Time synchronization		
Disable G-Core settings		

Installation

To use the Plugin Loader, you must install the gateway and the Plugin Loader.

Gateway:

- The gateway manages the connections of the Plugin Loader to the G-Core server and connects the Remote Plugins to the G-Core server. The network connection between the Plugin Loader and the gateway is made via gRPC.
- Install the gateway on the same server on which G-Core is installed (see **Server Installation**).

Plugin Loader:

- The Plugin Loader runs as a client-side microservice and manages the instances of the Remote Plugins, which are loaded dynamically as DLLs.
- There are two types of installation:
    - Plugin Loader Full:

      To use the Plugin Loader on the G-Core server. Install the Plugin Loader together with the gateway on the G-Core server (see **Server Installation**).
    - Plugin Loader Edge:

      To use the Plugin Loader on multiple sites. Install the Plugin Loader on a remote client with Windows (see **Remote Installation**) or Linux (see **Linux Installation**).

Plugin Loader Web Interface:

- In the Plugin Loader web interface, you can configure the Remote Plugins.
- You can open the web interface of the Plugin Loader on the G-Core server on which the gateway is installed or via remote access on a remote client (see **Open the Web Interface**).

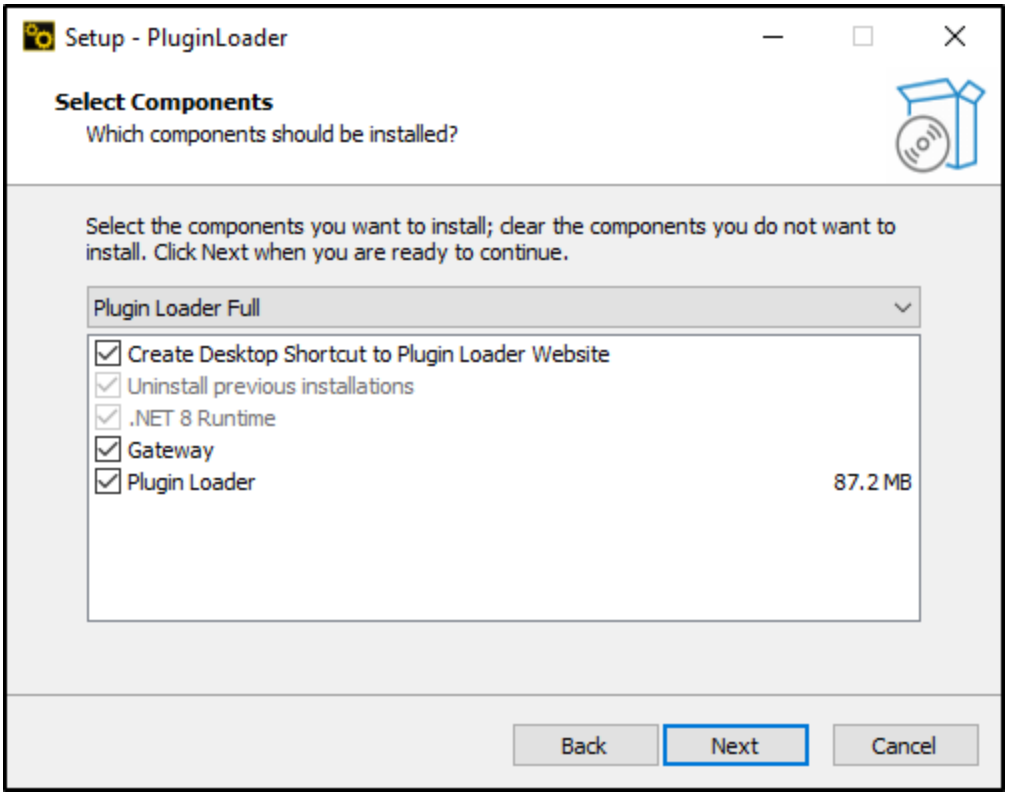
Server Installation

To use the Plugin Loader, install the gateway and the Plugin Loader on the G-Core server.

Requirement:
G-Core 8.3 or newer is required to use the Plugin Loader.

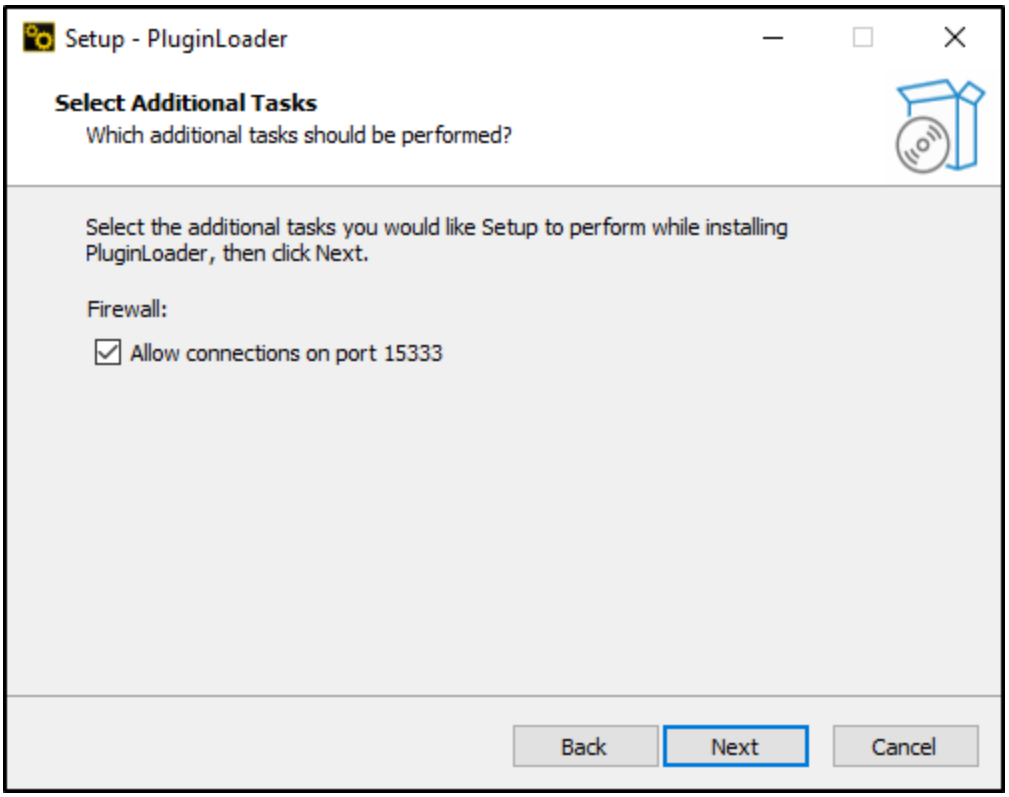
How to install the Plugin Loader:

1. Run the `PluginLoader_Installer_xxx.exe` file on the G-Core server.
2. In the **License Agreement** dialog window, select the option **I accept the agreement** and click **Next**.
3. In the **Select Components** dialog window, select the **Plugin Loader Full** installation from the drop-down menu and click **Next**.



4. In the **Select Additional Task** dialog window, select the **Allow connections on port 15333** option, to open the port automatically.

> **The Allow connections on port 15333 option is only required for the Remote Installation, as the port is needed for the connection of the gateway to a Plugin Loader on a remote client. If you need this port, select this option to open the port automatically or open it manually.**



5. In the **Ready to Install** dialog window, click **Install**.
6. When the installation is complete, click **Finish**.

Open the web interface:
Open the Plugin Loader web interface on the G-Core server with the URL `https://localhost:15333` or the desktop icon. For more information see Open the Web Interface.

Remote Installation

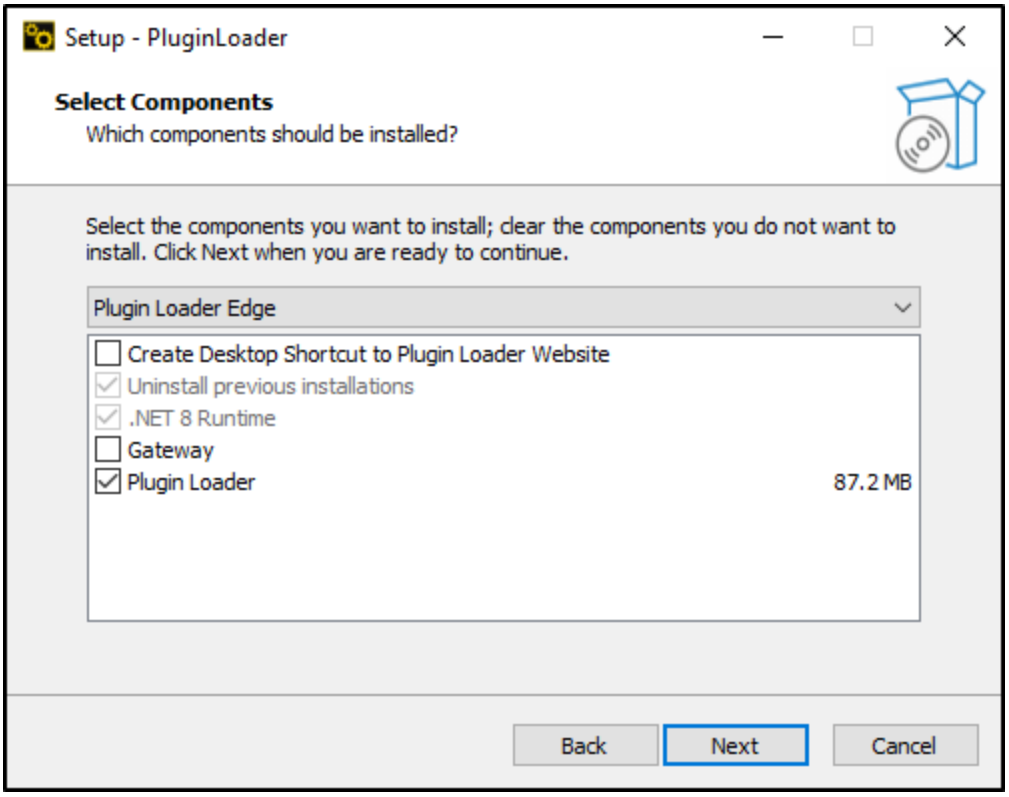
To use the Plugin Loader on multiple sites, install the Plugin Loader on the respective remote clients (see **Install Plugin Loader**) and connect the Plugin Loader to the gateway (see **Connect Plugin Loader to Gateway**).

Requirement:
The gateway must be installed on the G-Core server and port 15333 must be open (see Server Installation).

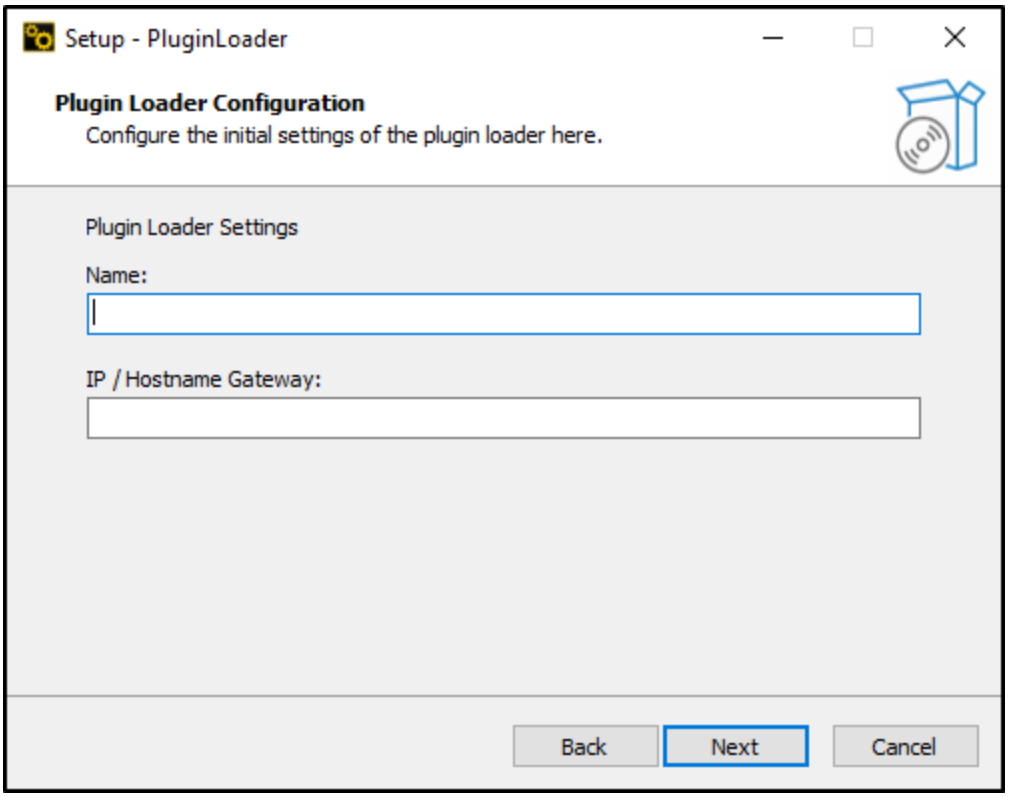
Install Plugin Loader

How to install the Plugin Loader on remote client:

1. Run the `PluginLoader_Installer_xxx.exe` file on the remote client.
2. In the **License Agreement** dialog window, select the option **I accept the agreement** and click **Next**.
3. In the **Select Components** dialog window, select the **Plugin Loader Edge** installation from the drop-down menu and click **Next**.

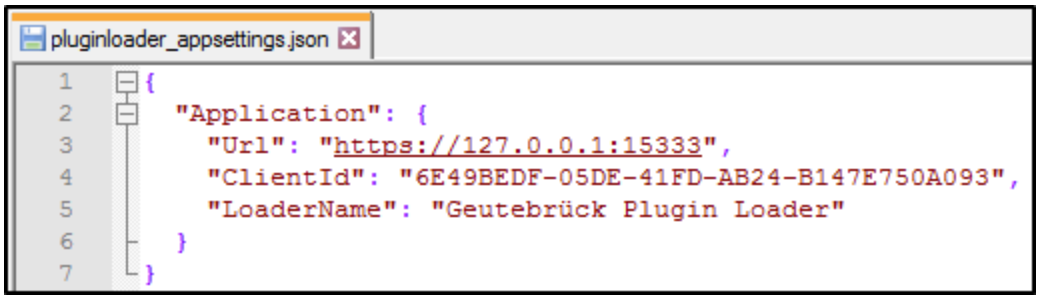


4. In the **Plugin Loader Configuration** dialog window, specify the Plugin Loader settings and click **Next**.
    - In the **Name** field, enter a name for your Plugin Loader.
    - In the **IP / Hostname Gateway** field, enter the IP address or hostname of the G-Core server on which the gateway is installed.



5. In the **Ready to Install** dialog window, click **Install**.
6. When the installation is complete, click **Finish**.

The Plugin Loader settings are saved in the `pluginloader_appsettings.json` file in the installation directory `C:\Program Files\Geutebrueck\PluginLoader`. You can subsequently change the settings in this file. If you have made changes, you must restart the **Geutebrueck.PluginLoader.Service** service.

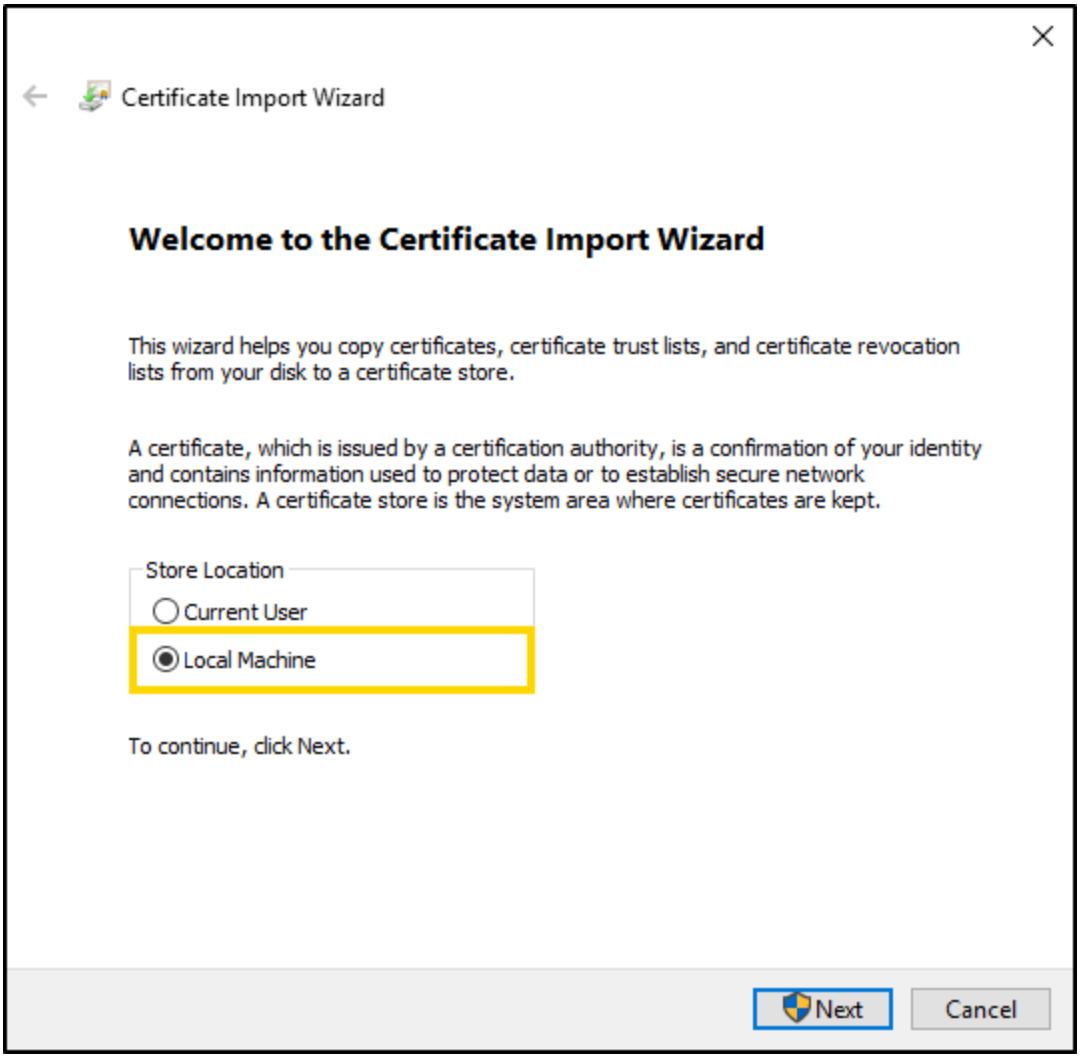


Connect Plugin Loader to Gateway

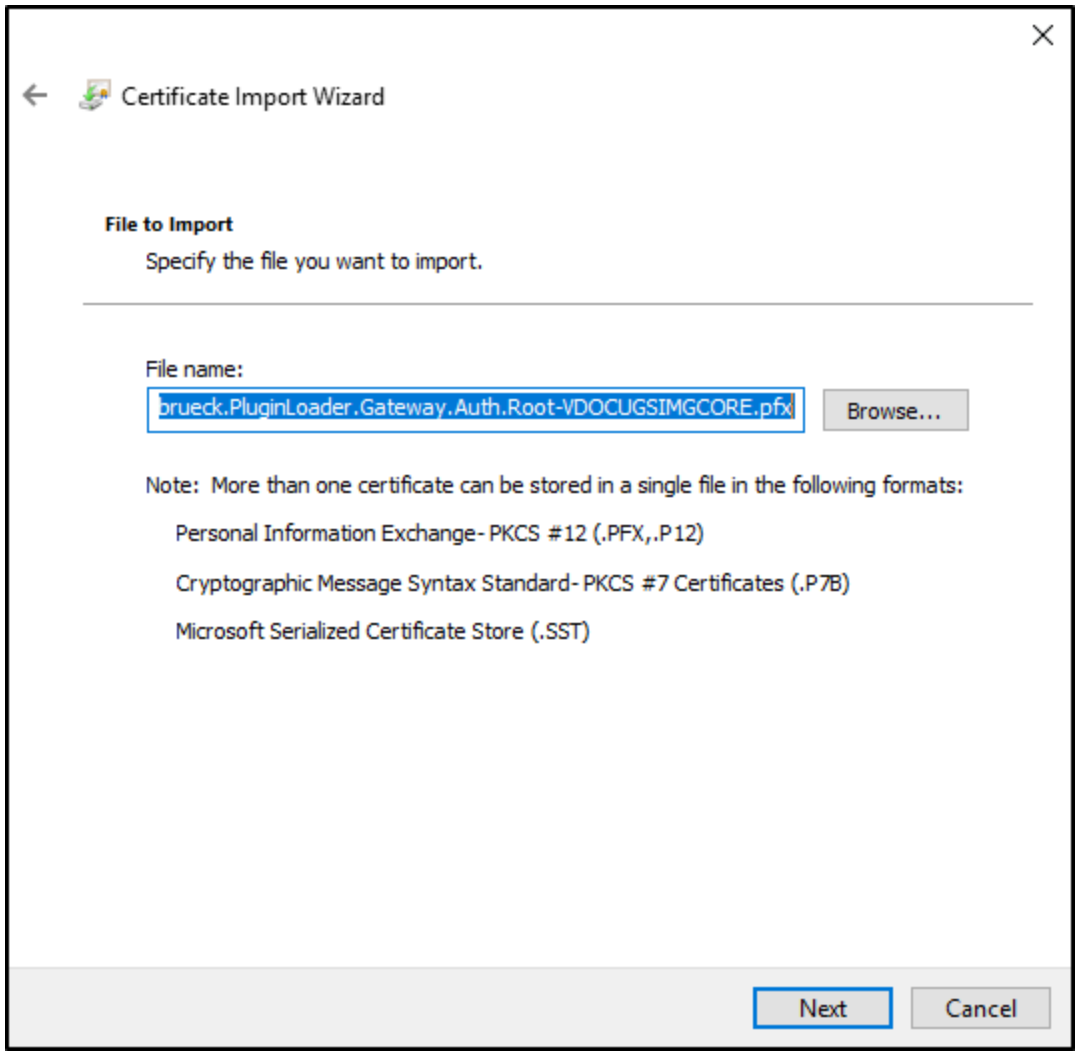
To connect the Plugin Loader on a remote client to the gateway on the G-Core server, install the authentication certificate on the remote client.

How to install the authentication certificate:

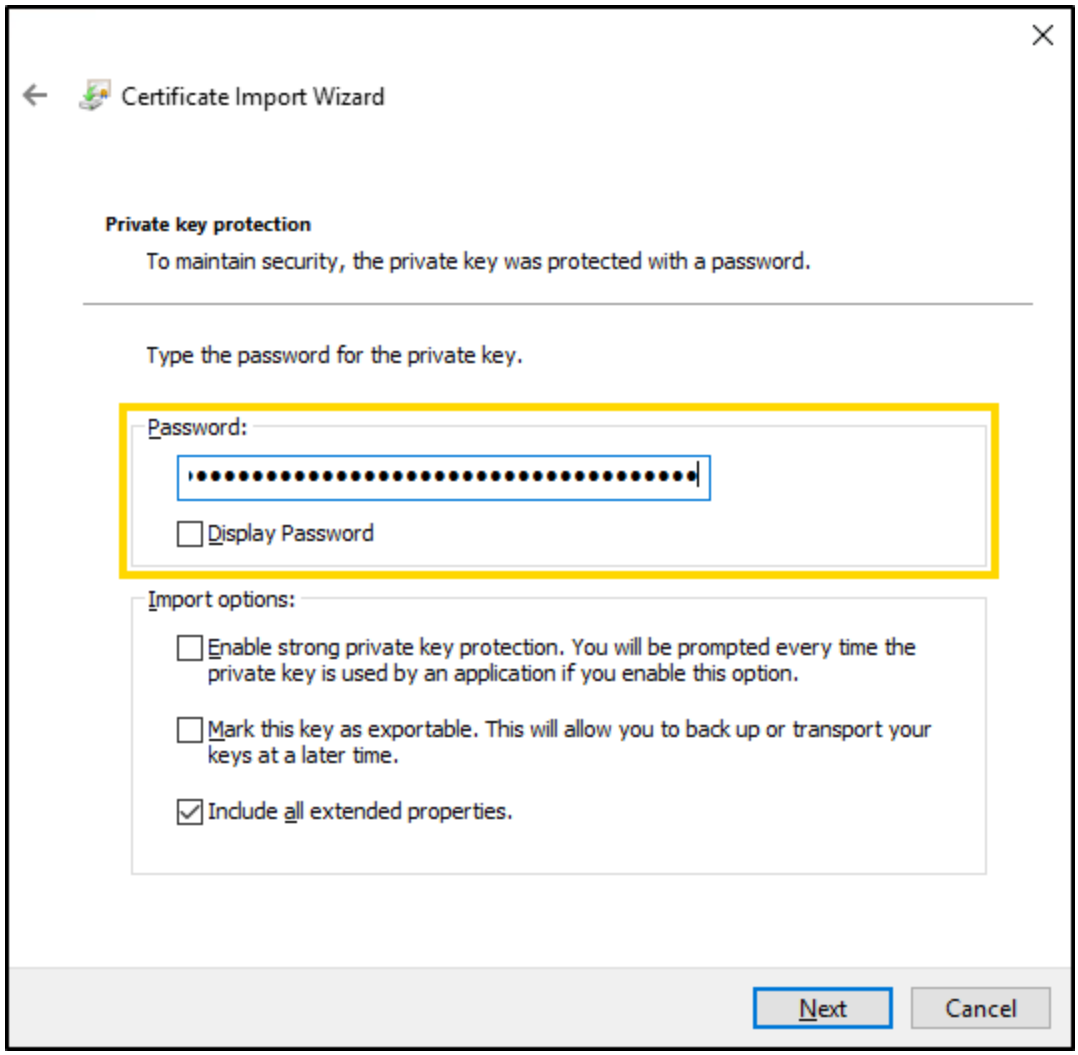
1. After installing the gateway on the G-Core server (see **Server Installation**), you will find the certificate files in the folder `C:\Program Files\Geutebrueck\GCore\RemotePlugins\child-root`. Copy these files to the remote client.
2. Run the certificate file `child-Geutebrueck.PluginLoader.Gateway.Auth.Root-<hostname>.pfx`.
3. In the **Certificate Import Wizard** dialog window, select **Local Machine** as **Store Location**. Click **Next**.



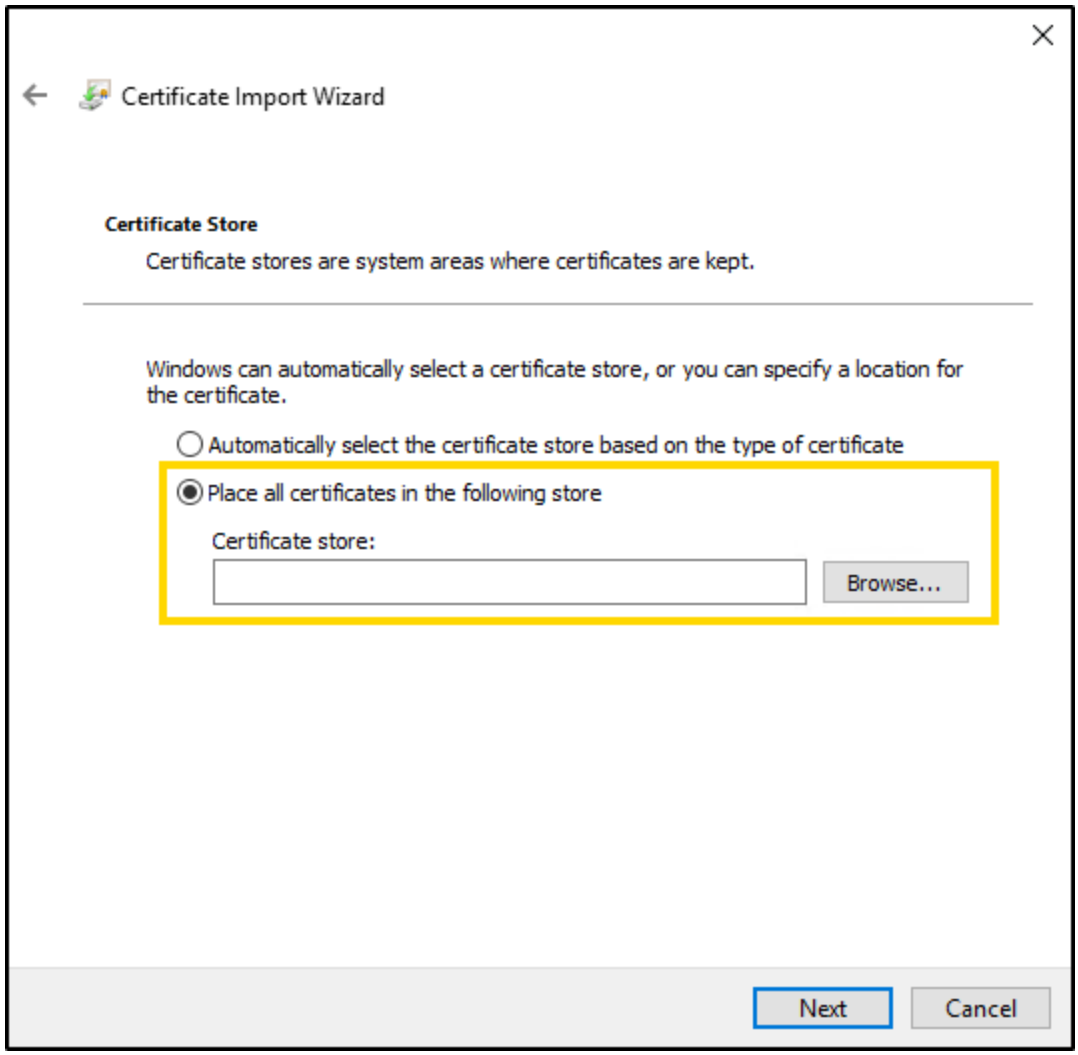
4. In the **File to Import** dialog window, the certificate file is already selected by default. Click **Next**.



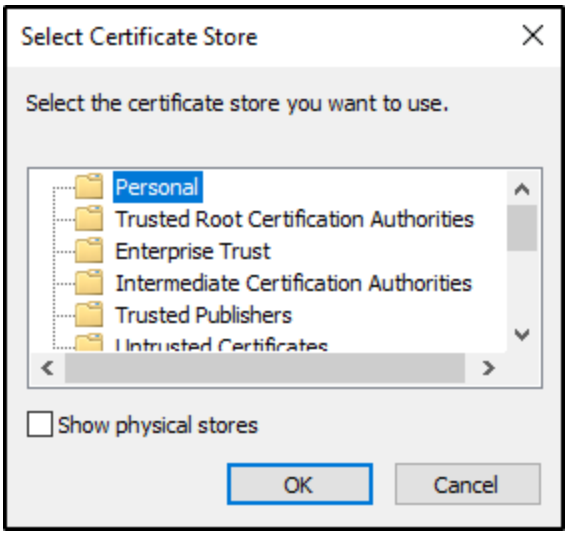
5. In the **Private key protection** dialog window, type the password for the private key. This password is noted in the certificate file `child-Geutebrueck.PluginLoader.Gateway.Auth.Root-<hostname>.pfx.$password.txt`. Click **Next**.



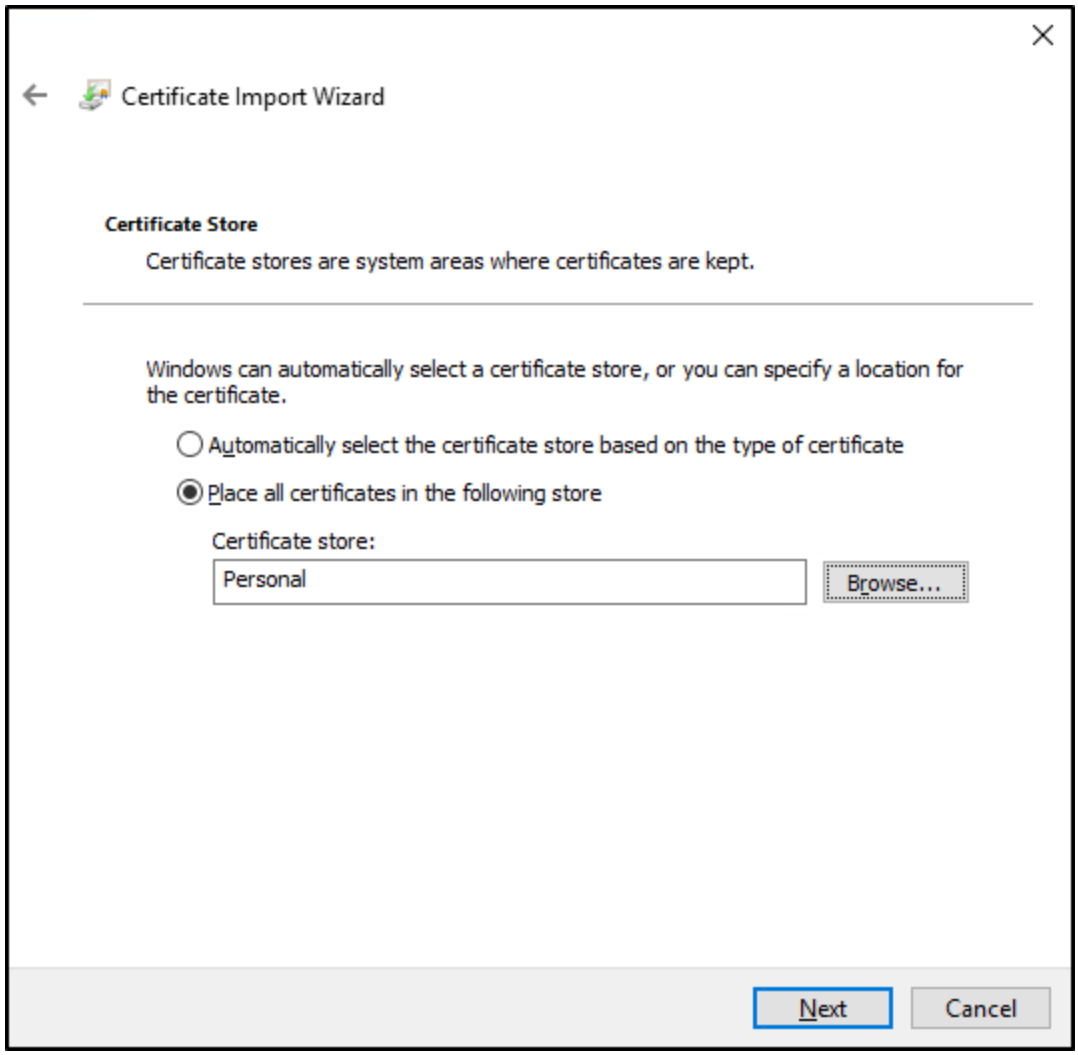
6. In the **Certificate Store** dialog window, select the option **Place all certificates in the following store** and click **Browse...**.



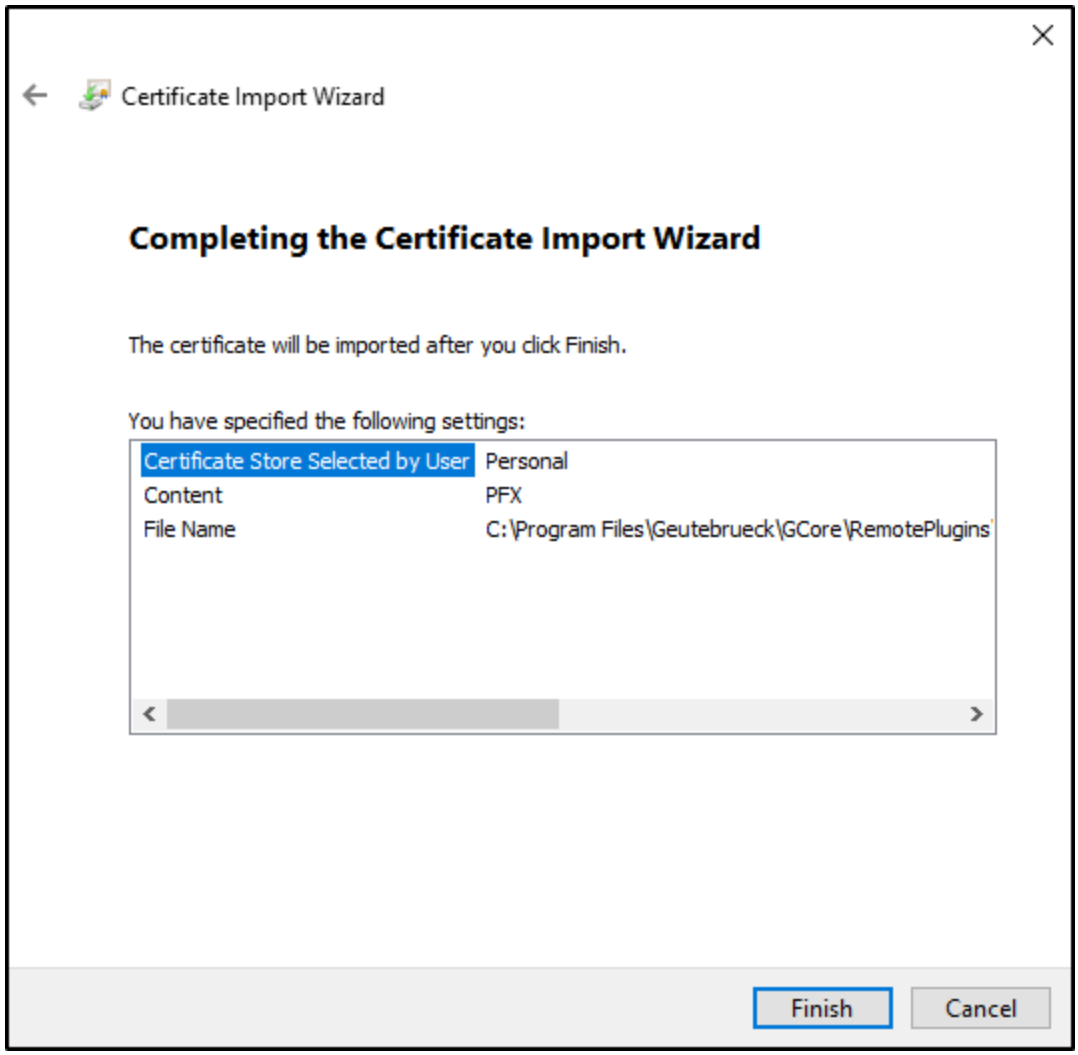
7. In the **Select Certificate Store** dialog window, select the **Personal** folder. Click **OK**.



8. In the **Certificate Store** dialog window, click **Next**.



9. In the **Completing the Certificate Import Wizard** dialog window, click **Finish**.



10. Restart the Geutebrueck.PluginLoader.Service service.

Open the web interface:
Open the Plugin Loader web interface on the G-Core server on which the gateway is installed with the URL `https://localhost:15333` **(see** Server Installation**) or install the authentication certificate on a remote client, to open the web interface via remote access with the URL** `https://<hostname or host-ip>:15333` **(see** Remote Access to Web Interface**). For more information see** Open the Web Interface**.**

Linux Installation

The installation file for Linux installation is only available on request.

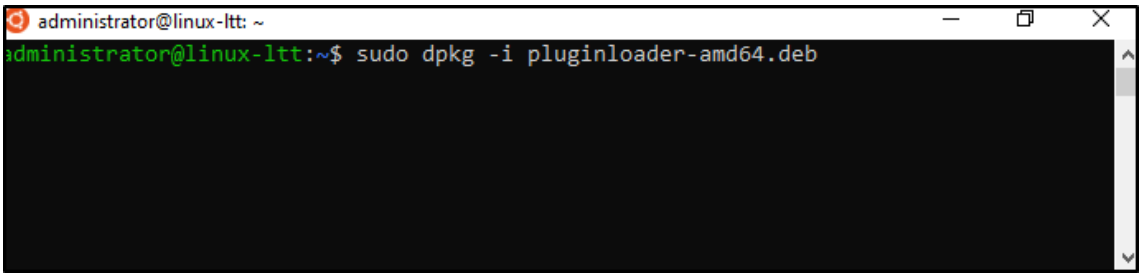
To use the Plugin Loader on multiple sites with Linux, install the Plugin Loader on the respective Linux remote clients (see **Install Plugin Loader**) and connect the Plugin Loader to the gateway (see **Connect Plugin Loader to Gateway**).

Requirements:
- The gateway must be installed on the G-Core server and port 15333 must be open (see Server Installation).
- The Linux system must have the Debian Package Manager (dpkg).

Install Plugin Loader

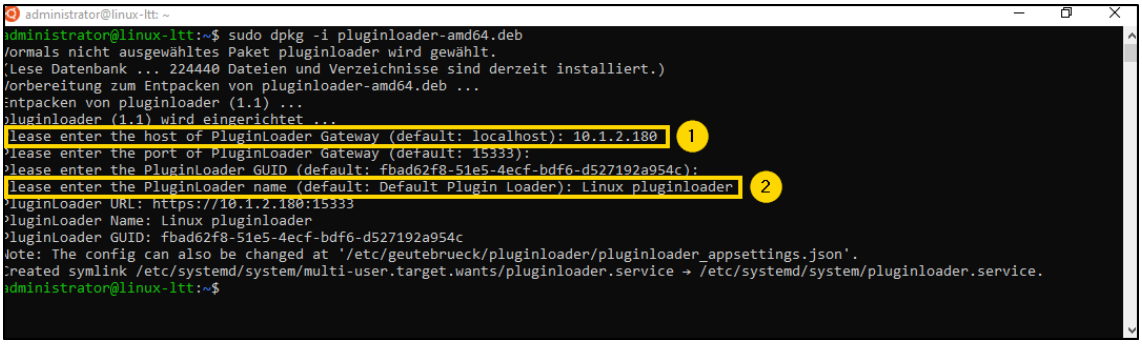
How to install the Plugin Loader on Linux client:

1. Copy the installation file `pluginloader-amd64.deb` to the Linux remote client.
2. Run the `pluginloader-amd64.deb` using the command `dpkg -i pluginloader-amd64.deb`.

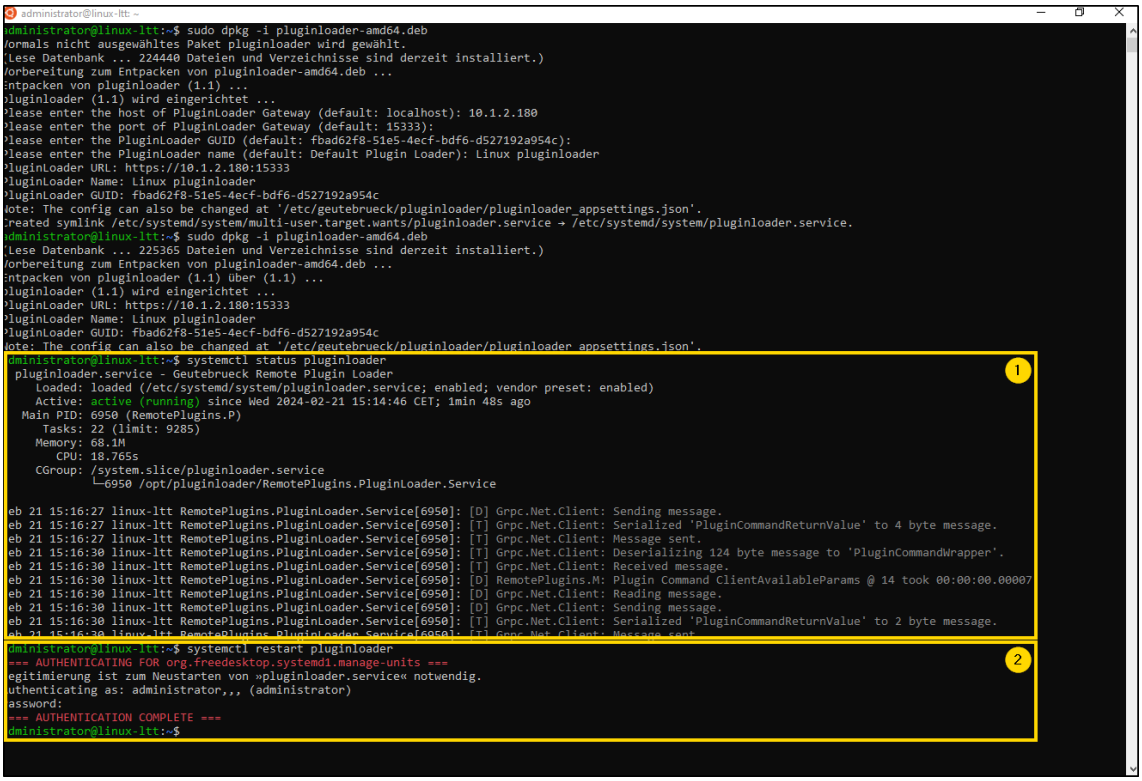


3. In the `Please enter the host of PluginLoader Gateway` prompt (1), enter the IP address of the G-Core server on which the gateway is installed (see **Server Installation**).
4. Take the default values for the port of the gateway and the Plugin Loader GUID by skipping the prompt with enter. If you change the GUID, you must specify a unique GUID.
5. In the `Please enter the PluginLoader name` prompt (2), enter a name for your Plugin Loader.

The settings are saved at `/etc/geutebrueck/pluginloader/pluginloader_appsettings.json` and can also be changed in this `appsettings.json` file. If you change the settings in this file, you must restart the Plugin Loader service using the `systemctl restart pluginloader` command.



6. You can query the status of the Plugin Loader using the `systemctl status pluginloader` command 1.
7. The Plugin Loader service is restarted after installation. You can restart the service manually using the `systemctl restart pluginloader` command 2.

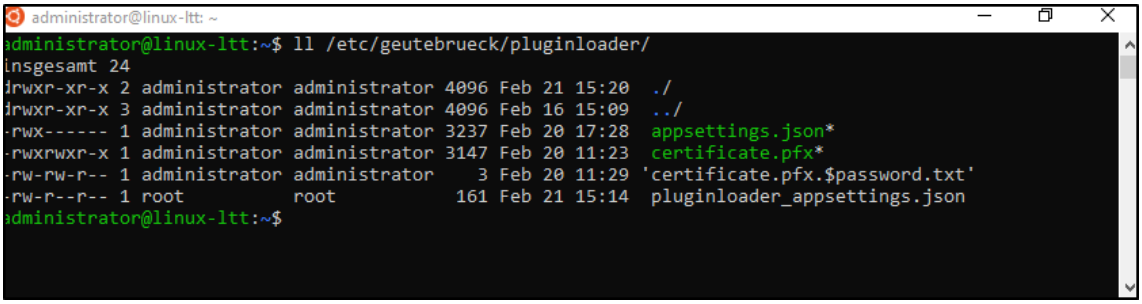


Connect Plugin Loader to Gateway

To connect the Plugin Loader on the Linux client to the gateway on the G-Core server, install the authentication certificate on the Linux remote client.

How to install the authentication certificate:

1. After installing the gateway on the G-Core server (see **Server Installation**), you will find the certificate files in the folder `C:\Program Files\Geutebrueck\GCore\RemotePlugins\child-root`. Copy these .pfx and .txt files to the Linux remote client in the directory `/etc/geutebrueck/pluginloader/`.



2. If you install the certificate during the installation of the Plugin Loader, the Plugin Loader service is automatically restarted after the installation. Otherwise, you must restart the service manually with the `systemctl restart pluginloader` command.

Open the web interface:
Open the Plugin Loader web interface on the G-Core server on which the gateway is installed with the URL `https://localhost:15333` (see Server Installation) or install the authentication certificate on a remote client, to open the web interface via remote access with the URL `https://<hostname or host-ip>:15333` (see Remote Access to Web Interface). For more information see Open the Web Interface.

Failover Installation

To use the G-Core failover for the Plugin Loader, install the gateway and the Plugin Loader on the G-Core server and on the respective G-Core spare devices.

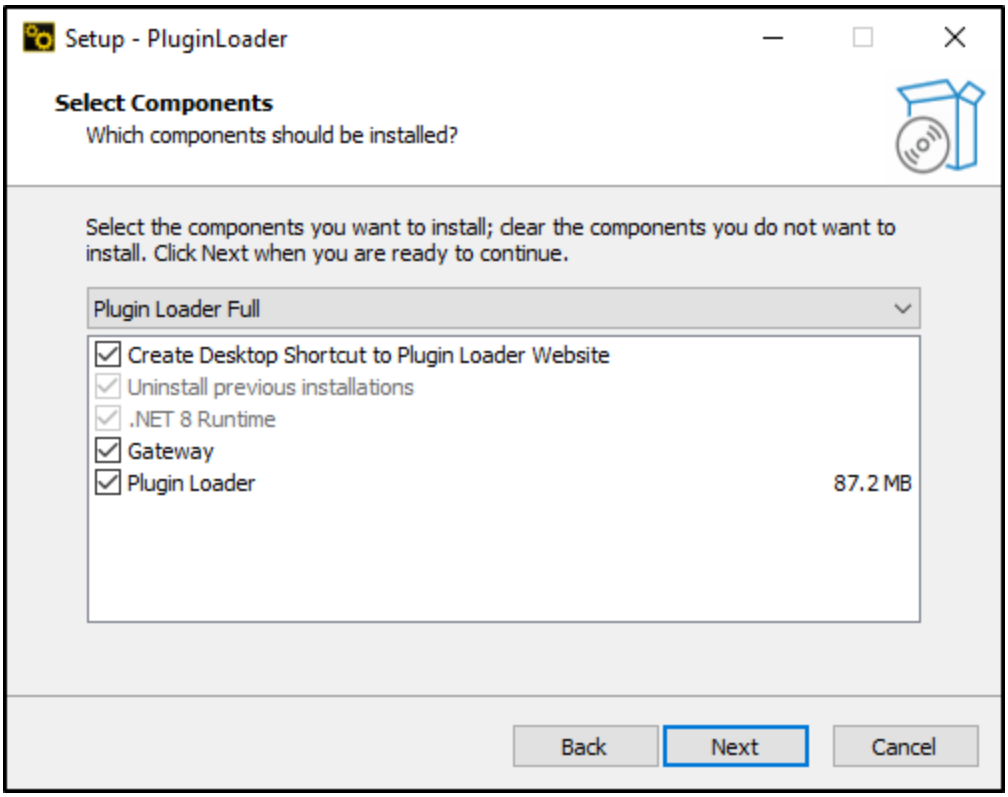
The gateway checks whether a failover case occurs on the primary G-Core server. If the failover case occurs, the gateway pushes all Remote Plugins to the connected Plugin Loader, regardless of the Plugin Loader ID, so that the G-Core spare device can use the plugins.

Requirements:
- G-Core 8.3 or higher.
- G-SIM system with a primary G-Core server and configured spare devices. For more information, see Failover in the G-SIM documentation.

How to use the G-Core failover for the Plugin Loader:

1. Install the **Plugin Loader Full** installation (see **Server Installation**) on the G-Core server.
2. Install the **Plugin Loader Full** installation (see **Server Installation**) on the G-Core spare device.

Install the Plugin Loader on all G-Core spare devices to avoid losing Plugin Loader channels on failover. If a failover occurs on a G-Core spare device without a Plugin Loader, the channels become unavailable.



3. In G-SIM, ensure that failover is configured. For more information, see **Failover** in the G-SIM documentation.

   Further configuration of the G-Core spare device is not necessary.

Remote Access to Web Interface

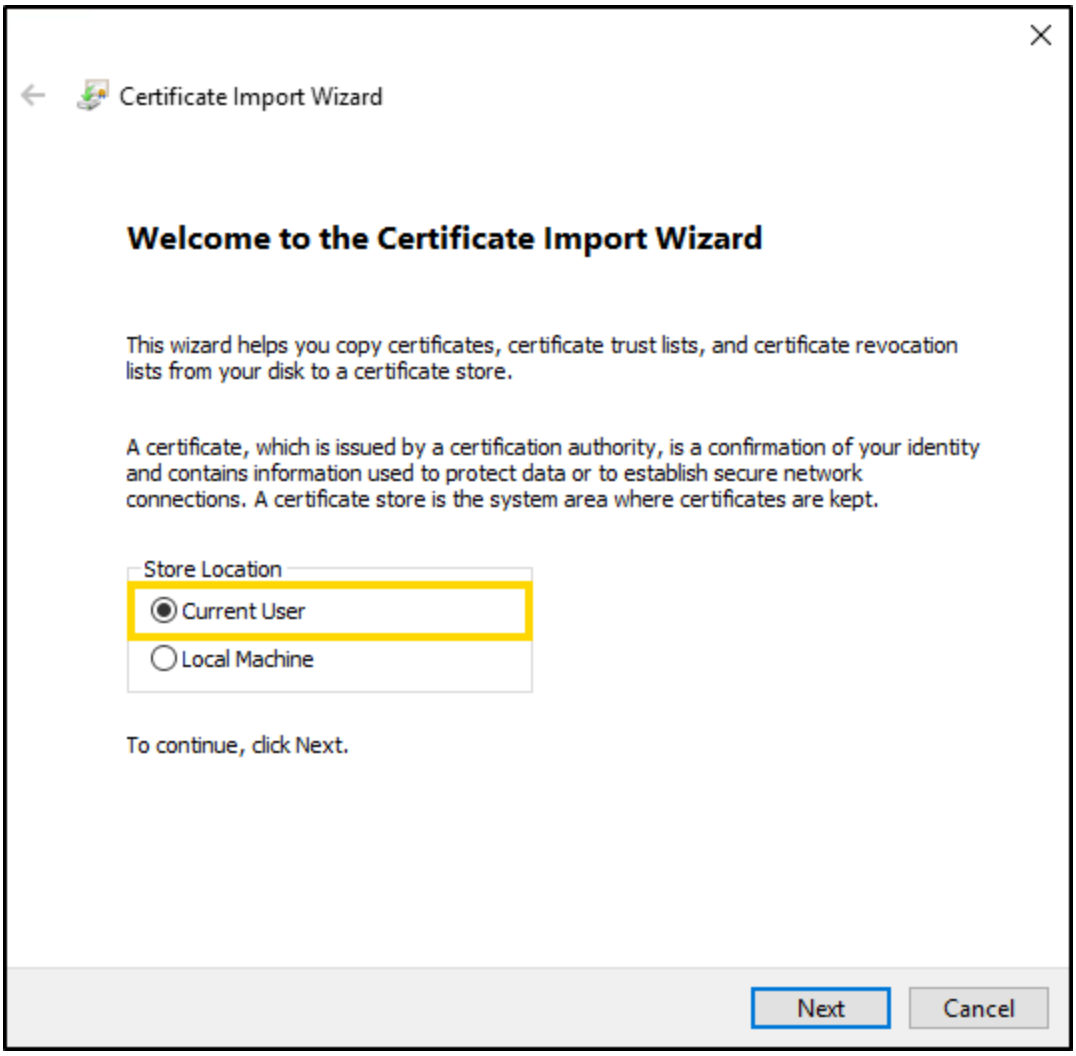
To access the Plugin Loader web interface via remote access on a remote client, install the authentication certificate on the respective remote clients.

Requirement:
The gateway must be installed on the G-Core server and port 15333 must be open (see Server Installation).

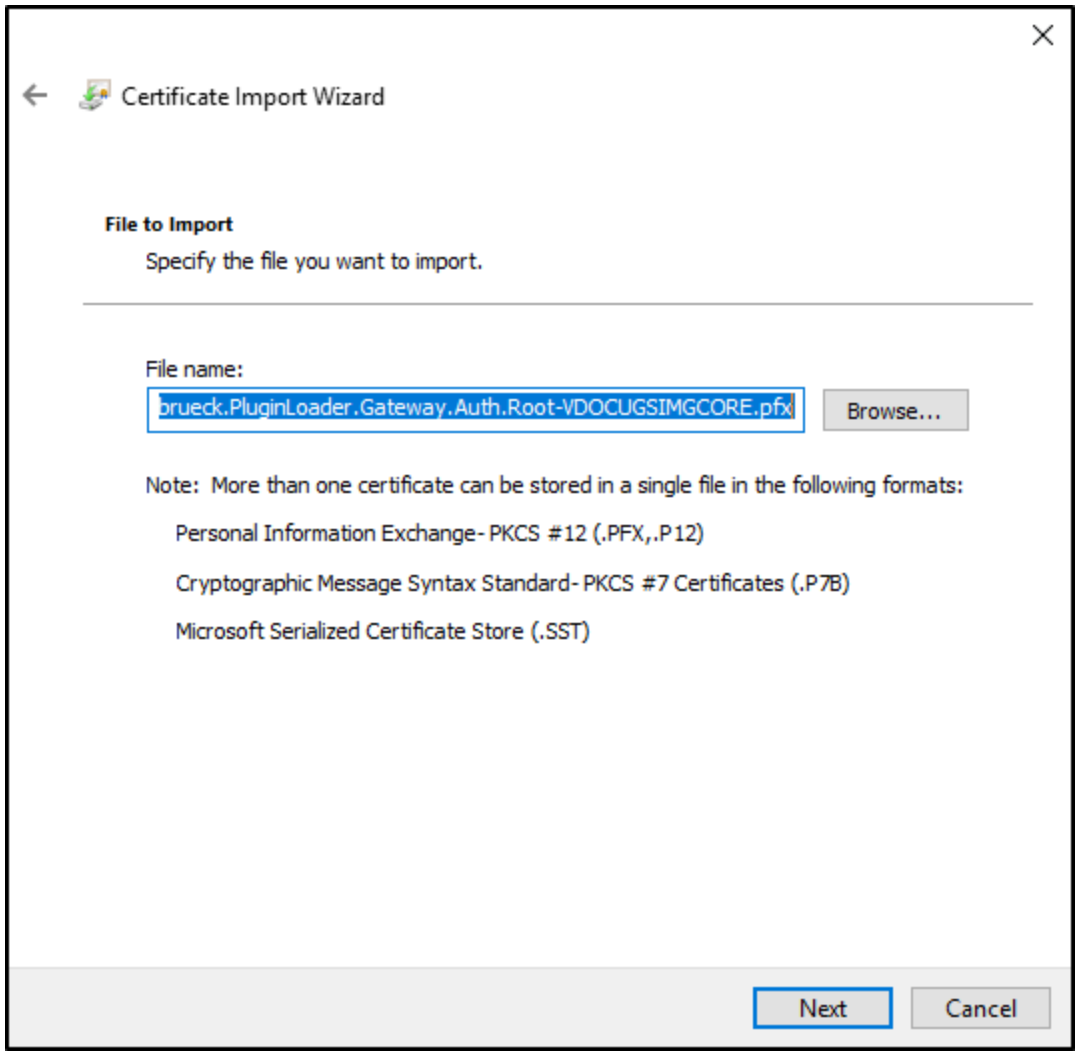
How to install the authentication certificate:

1. After installing the gateway on the G-Core server (see **Server Installation**), you will find the certificate files in the folder `C:\Program Files\Geutebrueck\GCore\RemotePlugins\child-root`. Copy these files to the remote client.

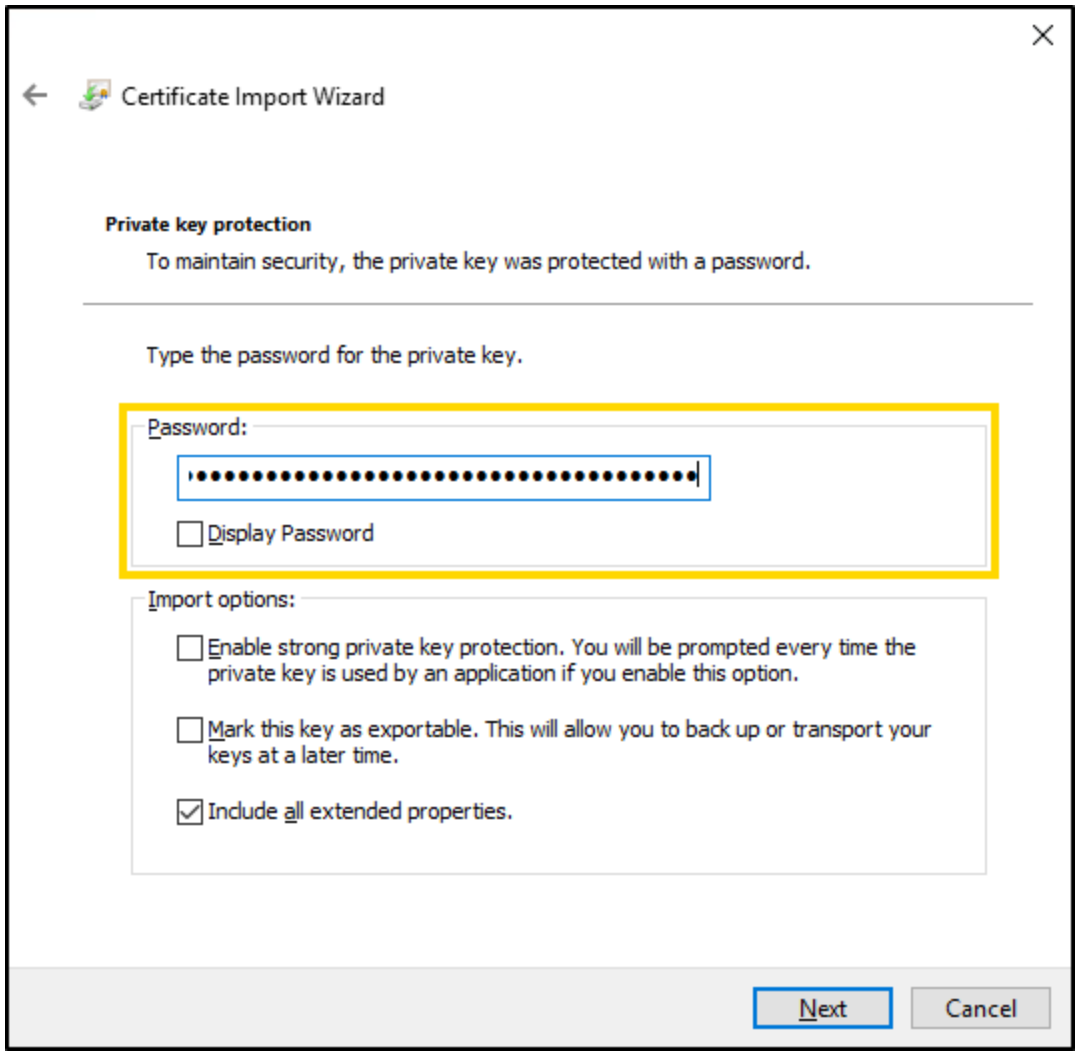
2. Run the certificate file `child-Geutebrueck.PluginLoader.Gateway.Auth.Root-<hostname>.pfx`.
3. In the **Certificate Import Wizard**, select **Current User** as **Store Location**. Click **Next**.



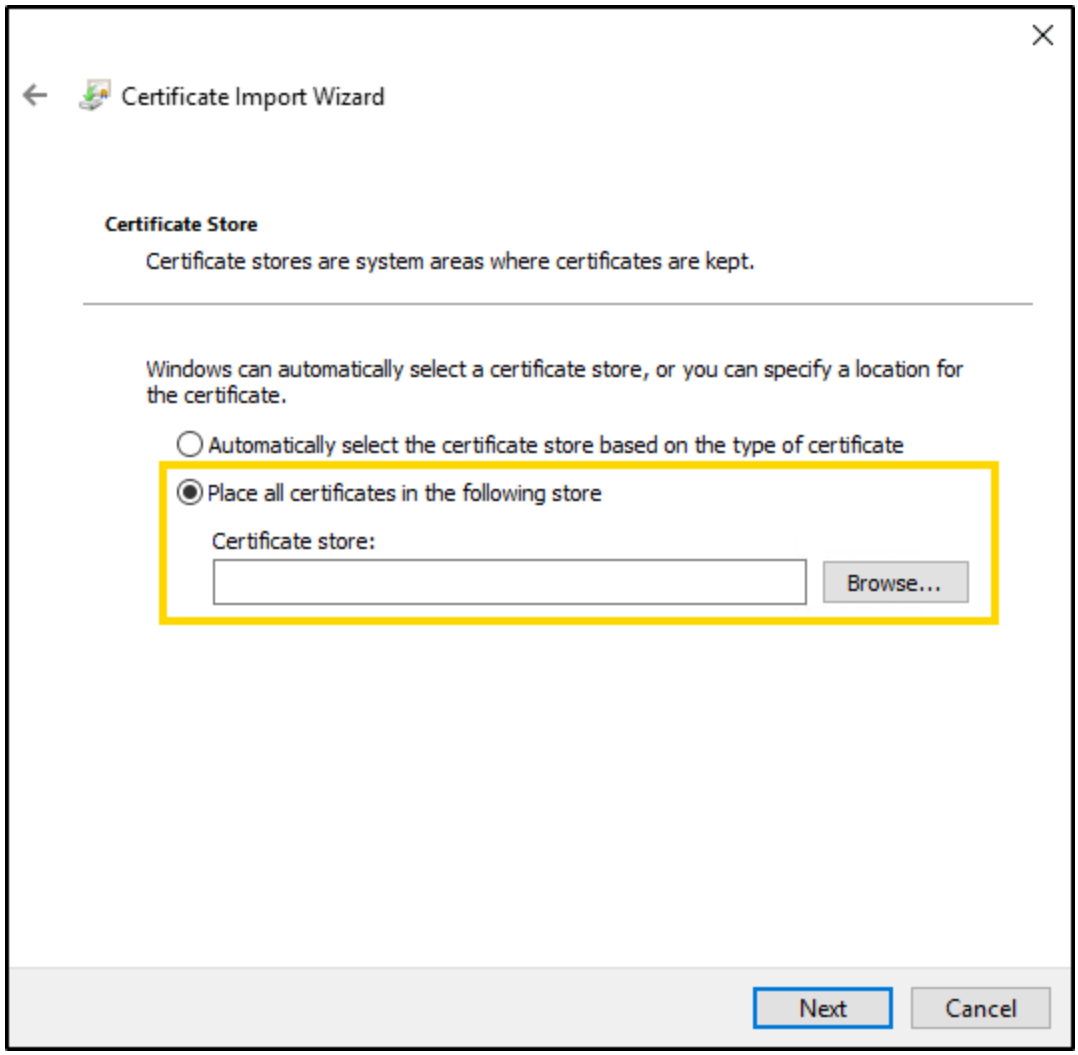
4. In the **File to Import** dialog window, the certificate file is already selected by default. Click **Next**.



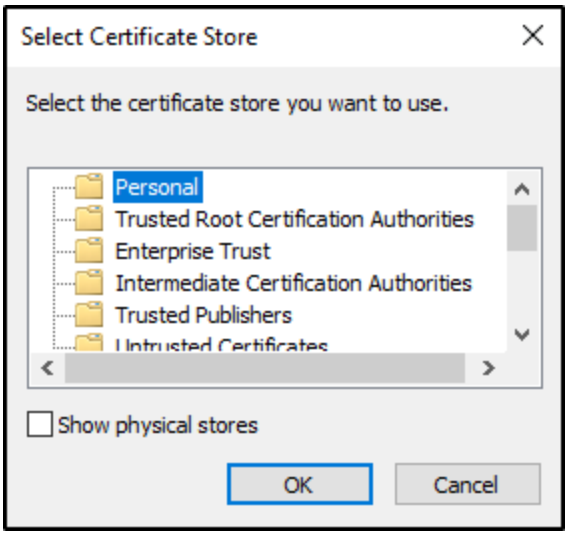
5. In the **Private key protection** dialog window, type the password for the private key. This password is noted in the file `child-Geutebrueck.RemotePlugins.Gateway.Auth.Root-<hostname>.pfx.$password.txt`. Click **Next**.



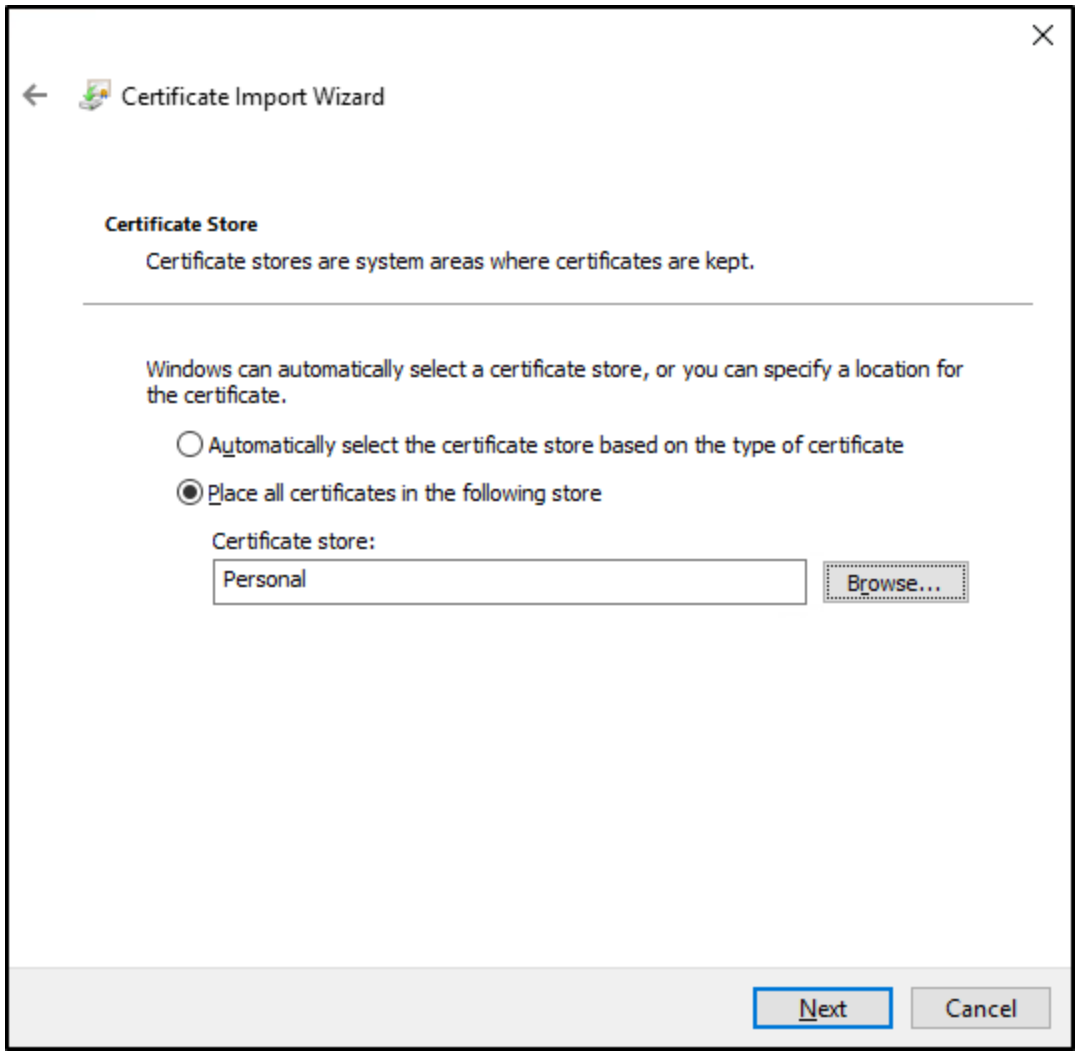
6. In the **Certificate Store** dialog window, select the option **Place all certificates in the following store** and click **Browse...**.



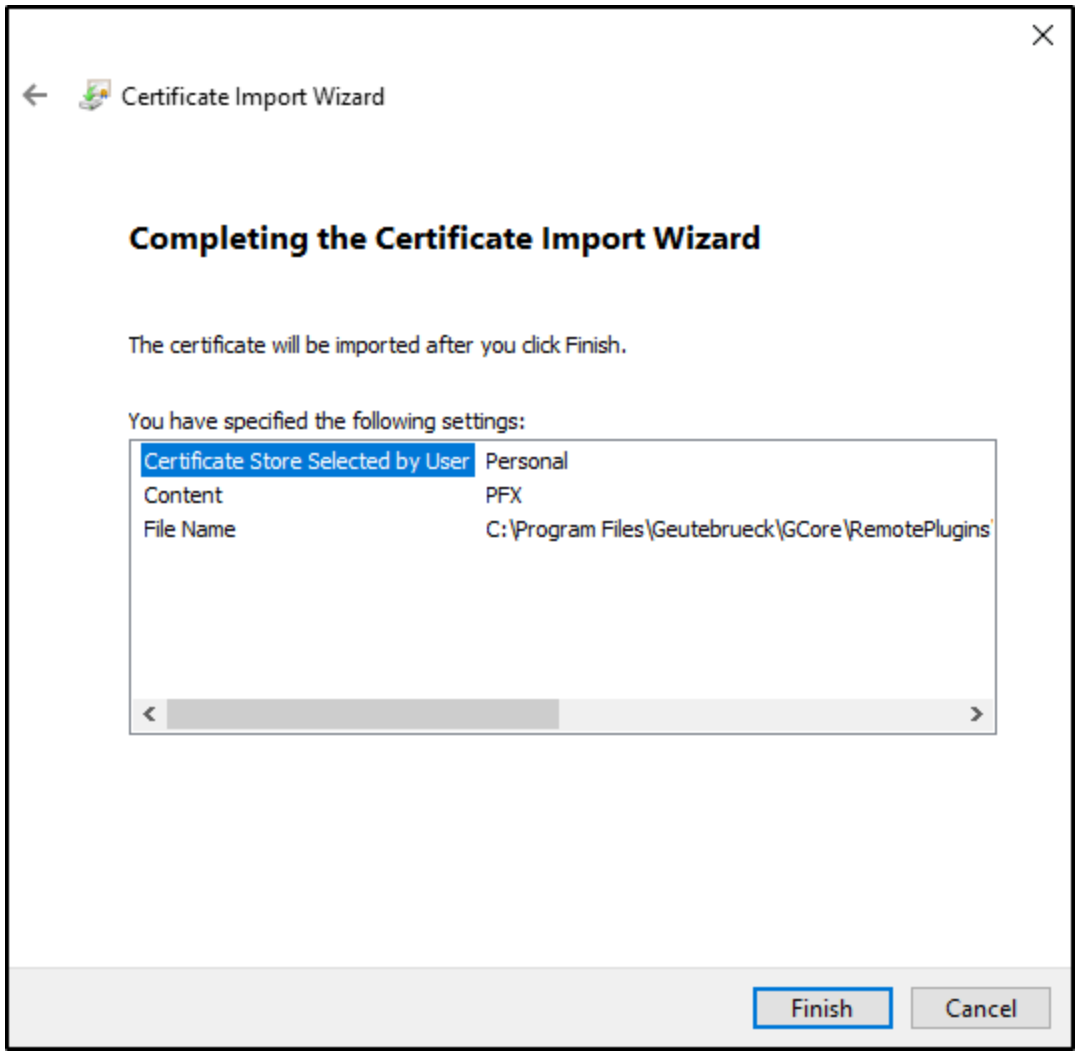
7. In the **Select Certificate Store** dialog window, select the **Personal** folder. Click **OK**.



8. In the **Certificate Store** dialog window, click **Next**.



9. In the **Completing the Certificate Import Wizard** dialog window, click **Finish**.



Open the web interface:
Open the Plugin Loader web interface on a remote client with the URL `https://<hostname or host-ip>:15333`**. For more information see** Open the Web Interface.

Upgrade

Upgrade 1.1 to 1.2

If you upgrade the Plugin Loader version 1.1 to 1.2, you must first uninstall version 1.1 and delete the installation directories. The Plugin Loader settings are saved in the `pluginloader_appesettings.json` file. Save this file to backup the settings.

Upgrade the Plugin Loader on the G-Core server (see **Server Installation**) and the Plugin Loader on a remote client (see **Remote Installation**).

Upgrade Plugin Loader on G-Core server:

1. Save a backup copy of the `pluginloader_appsettings.json` file, which is located in the installation directory `C:\Program Files\Geutebrueck\PluginLoader`.
2. Uninstall the G-Core Hub program via the Windows control panel.
3. Remove the installation directories `C:\Program Files\Geutebrueck\PluginLoader` and `C:\Program Files\Geutebrueck\GCore\RemotePlugins`.
4. Run the `PluginLoader_Installer_1.2.0.exe` on the G-Core server to install the Plugin Loader 1.2 (see **Server Installation**).
5. Replace the `pluginloader_appsettings.json` file in the installation directory `C:\Program Files\Geutebrueck\PluginLoader` with the backup copy.
6. Restart the GCoreServer and Geutebrueck.PluginLoader.Service service.

Upgrade Plugin Loader on remote client:

1. Save a backup copy of the `pluginloader_appsettings.json` file, which is located in the installation directory `C:\Program Files\Geutebrueck\RemotePluginLoader`.
2. Uninstall the Geutebrück PluginLoader program via the Windows control panel.
3. Remove the installation directory `C:\Program Files\Geutebrueck\RemotePluginLoader`.
4. Run the `PluginLoader_Installer_1.2.0.exe` on the remote client to install the Plugin Loader 1.2 (see **Remote Installation**).

5. Replace the `pluginloader_appsettings.json` file in the installation directory `C:\Program Files\Geutebrueck\PluginLoader` with the backup copy.
6. Restart the Geutebrueck.PluginLoader.Service service.

Upgrade 1.2 to 1.3

If you upgrade the Plugin Loader version 1.2 to 1.3, you must first install G-Core 8.3 before installing the Plugin Loader 1.3. The Plugin Loader settings are saved in the `pluginloader_appsettings.json` file. Save the this file to backup the settings.

Upgrade the Plugin Loader on the G-Core server (see **Server Installation**) and the Plugin Loader on a remote client (see **Remote Installation**).

How to upgrade the Plugin Loader:

1. Install G-Core 8.3 on the G-Core server.

   The problem may occur that outdated Plugin Loader files could not be completely removed and therefore G-Core cannot start. Proceed with the installation of Plugin Loader 1.3, which will update all relevant files and restart the G-Core server. After installation, check that G-Core has been restarted correctly with the new files.

2. Install the Plugin Loader 1.3 on the G-Core server (see **Server Installation**). The Plugin Loader installer restarts the G-Core service during installation.
3. Optional: Install the Plugin Loader 1.3 on the remote client (see **Remote Installation**).

Web Interface

Open the Web Interface

You can open the web interface of the Plugin Loader on the G-Core server on which the gateway is installed (see **Server Installation**) or via remote access on a remote client (see **Remote Access to Web Interface**).

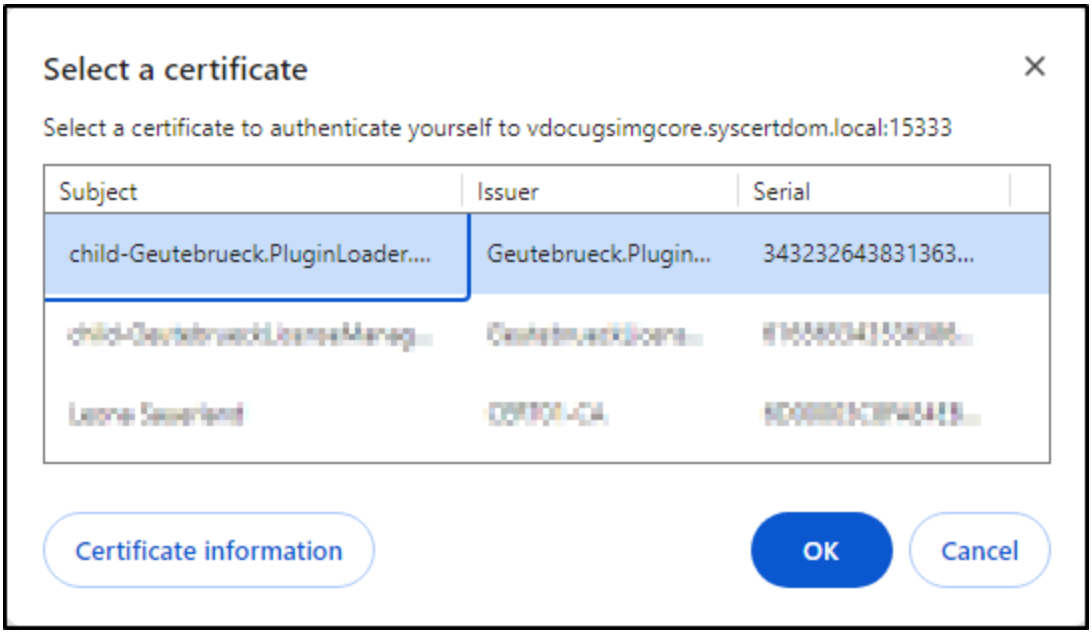
How to open the web interface:

1. Open the Plugin Loader web interface:
   - On the G-Core server: `https://localhost:15333` or desktop icon
   - On a remote client: `https://<hostname or host-ip>:15333`

     You can also find the URL in the `pluginloader_appsettings.json` file, in the installation directory of the Plugin Loader (`C:\Program Files\Geutebrueck\PluginLoader`).

     The G-Core server and the Geutebrueck.PluginLoader.Service service must be running to access the web interface.
2. The error message Your connection isn't private appears. Click the Advanced button and then Continue to localhost (unsafe).
3. The pop-up window Select a certificate appears, asking you to select a certificate for authentication. Select the `child-Geutebrueck.PluginLoader.Gateway.Auth.Root` certificate and confirm

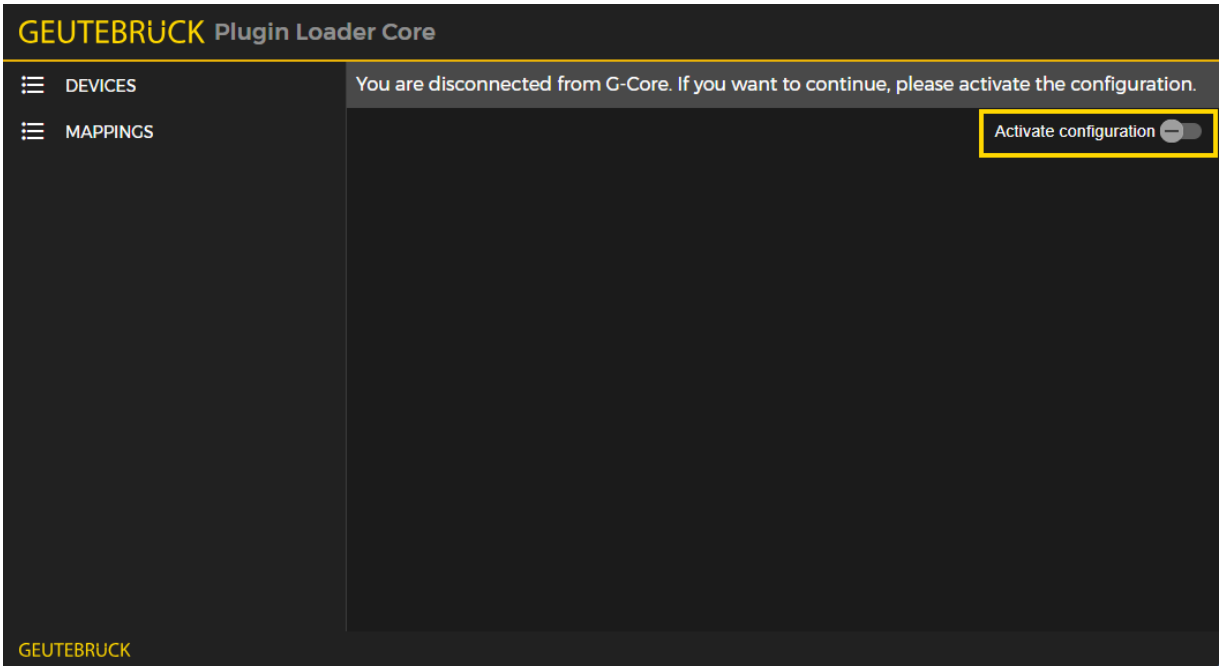
with **OK**.



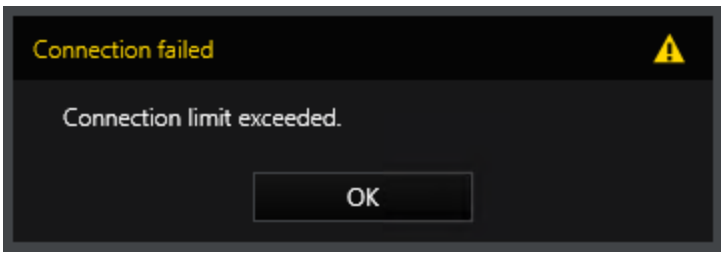
Activate Configuration

To access the configuration view of the Plugin Loader web interface, you must activate the configuration to connect the system to G-Core.

To do this, enable the **Activate configuration** slider.



When the configuration is activated, the system is connected to G-Core. G-Set is locked during this time.



If you changed the configuration, you must send your configuration to G-Core to update and unlock G-Set (see **Send Configuration to G-Core**). The configuration is deactivated and the system is disconnected from G-Core. The system is then locked and you cannot perform any configuration.

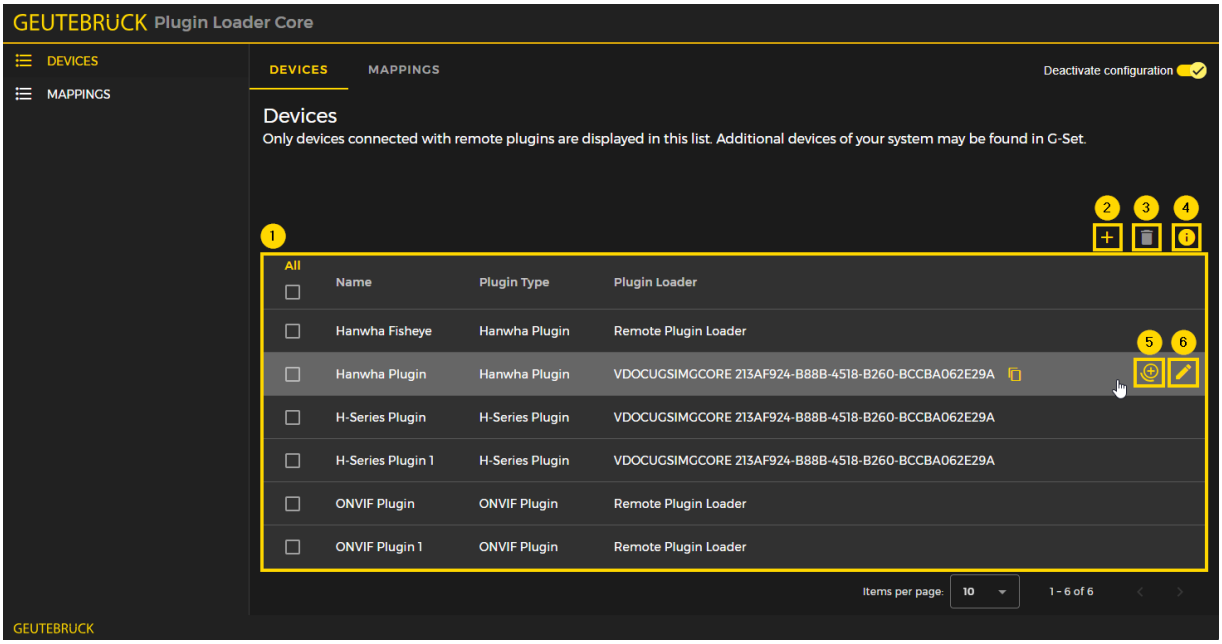
Devices

Device List

In the **Devices** view, you can configure and manage the devices connected with Remote Plugins.

> **To access the configuration view, you must activate the configuration (see Activate Configuration).**

At the first start or if no devices have been created, the device list is empty and the **Add your first device** + button is displayed. Click this button to add a device to the list (see **Add Device**).

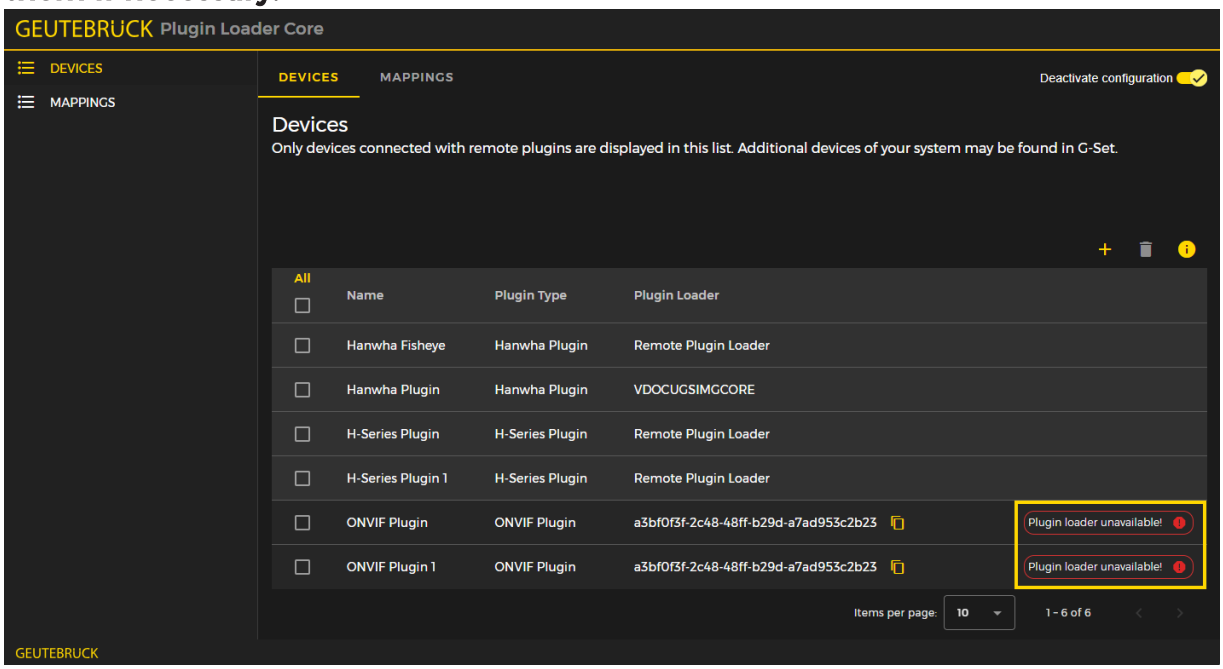


The view consists of the following elements:

	Element	Description
1	Device list	In the list all configured devices connected with Remote Plugins are listed. The following information is listed: • **Name**: Name of the Device. • **Plugin Type**: Type of the selected Remote Plugin. • **Plugin Loader**: Client ID of the selected Plugin Loader. Click the icon to copy it to the clipboard. **The client ID is also specified in the `pluginloader_appsettings.json` file in the installation directory `C:\Program Files\Geutebrueck\RemotePluginLoader` of the respective Plugin Loader.**
2	+	Click this icon to add a device (see **Add Device**).
3	🗑	Click this icon to delete a device (see **Delete Device**).

	Element	Description
4		Click this icon to display the version numbers of available plugins.
5		Click this icon to create multiple devices based on the settings of the selected device (see **Create Multiple Devices**). If multiple devices have already been prepared for the device but not yet created, an indicator appears next to the icon showing how many multiple devices are prepared for the selected device. Click the drop-down menu icon and select whether you want to  **Edit** or  **Delete** the prepared devices.
6		Click on the row of a device to edit it (see **Edit Device**).

If the selected Plugin Loader of a device is not available, the error message Plugin loader unavailable!  **appears and the device cannot be edited. Make sure that the G-Core server and the Geutebrueck.PluginLoader.Service service are running and restart them if necessary.**

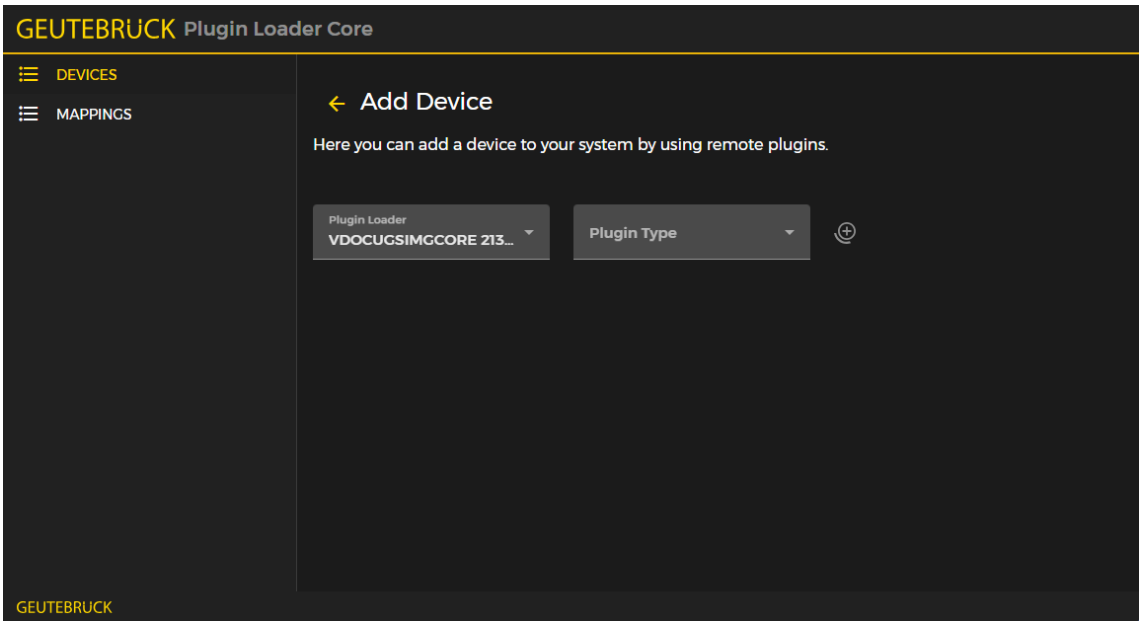


Add Device

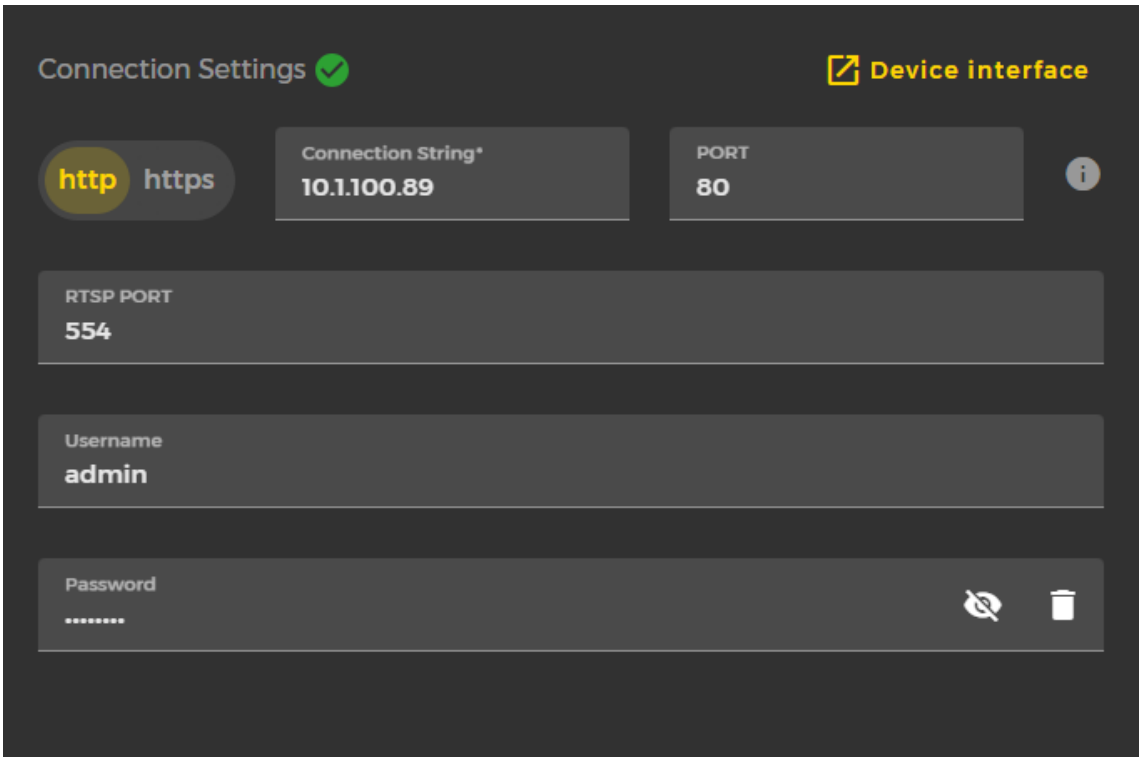
In the **Add Device** view, you can add a device to your system by using Remote Plugins. To open this view, click the + icon in the **Devices** view (see **Device List**).

How to add a device:

1. In the **Plugin Loader** drop-down menu, select the Plugin Loader. If there is only one Plugin Loader, it is selected by default.
2. In the **Plugin Type** drop-down menu, select the Remote Plugin. Only plugins of the selected Plugin Loader are available.



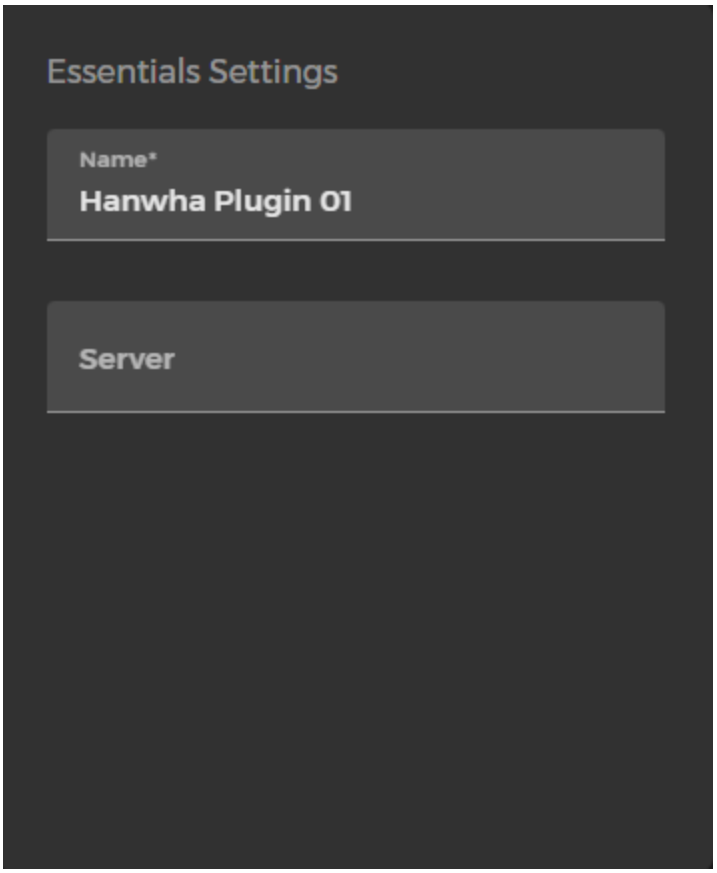
   The view changes to **Edit Device** and the  icon is activated (see **Create Multiple Devices**). The settings dialogs of the selected plugin appear.
3. Specify the **Connection Settings**:



Name	Description
Connection status	The system establishes the connection when you enter the connection settings and displays the status of the connection: **Connection successful** **No connection** Move the mouse over the status indicator to display the notification explaining why no connection could be established.
Device interface	Click this button to open the web interface of the connected camera. Only available if a camera is connected.
Protocol	Select HTTP or HTTPS. If you are using HTTPS communication, you must set up the camera to use HTTPS. To do this, you must install or

Name	Description
	create a certificate on the camera.
Connection String	Enter the IP address or DNS name of the camera. **If you insert an URL such as `<http/https>://<domain>:<port>` the Protocol and Port will be filled in automatically.**
Port	Enter the port of the camera. The port is determined by the selected protocol. If HTTP is selected, the default port automatically changes to 80 and if HTTPS is selected, the default port changes to 443. You can edit the port and enter the port of the camera.
RTSP Port	Enter the RTSP port of the camera. The default port is 554.
Username	Enter the username of the camera.
Password	Enter the password of the camera.

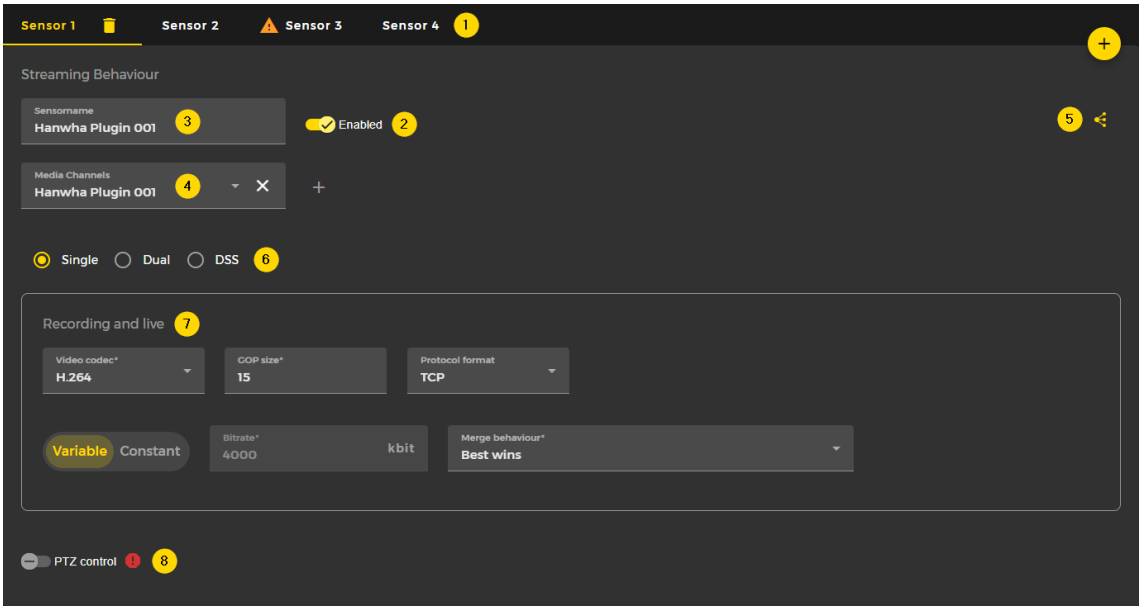
4. Specify the **Essentials Settings**:



Name	Description
Name	Specify a unique name for the device. If the name is not unique, an index is automatically appended. This name is also used for the sensor names and media channel name of the device, supplemented by an index. If you have specified a custom sensor name, the sensor name will not be changed.
Server	Specify the name of the server to which the plugin belongs (optional).

5. Specify the number of available camera sensors (1). The system automatically adds as many sensors as channels are available for the respective camera. A maximum of 5 sensors can be added.

Click the + icon to add a sensor or click the icon of the selected sensor to delete it.



6. Activate or deactivate the **Disabled**/ **Enabled** option 2, to enable or disable the selected sensor. Disabled sensors are marked with the symbol.

7. You can change the **Sensorname** 3 of the selected sensor.

8. Select a media channel for the selected sensor 4.

   By default, a new media channel is created for the respective sensor. You can also select a media channel from the **Media Channels** drop-down menu, which lists all available media channels from G-Core that are not already linked to a hardware.

   It is also possible to create a new media channel with the + icon (the media channel will be selected automatically) or to disconnect a selected media channel from the sensor with the ✖ icon.

   If no media channel is selected, the sensor is disabled and marked with the symbol.

9. Click the icon 5 to apply the streaming behaviour settings of the selected sensor to all sensors. The **Sync Sensors** dialog window opens. Confirm the **Do you want to sync all sensors?** dialog with **Confirm**.

10. Select the type of **Streaming Behaviour** 6 of the selected sensor:

Stream	Description
Single	Select this option to retrieve only one stream from the camera. The specified streaming settings apply to both the live stream and recordings.
Dual	Select this option to retrieve two streams from the camera, one for live stream and one for recordings. You can specify the streaming settings for both live stream and recordings to avoid live stream and recordings using the same resolution.
DSS (Dynamic Stream Selection)	Select this option to no longer permanently bind the live and recording streams of G-Core to streams of the camera. Instead, two to eleven streams with different resolutions are configured in the camera. Depending on the live and recording stream requirements in G-Core, one of them is selected. With this option, it is possible to switch between resolutions depending on the event without creating a gap in the stream.

11. Specify the settings for **Recording and live** stream 7 of the selected sensor:

Name	Description
Video codec	As video codec you can choose between H.264, H.265 and JPEG.
GOP size	With GOP (Group of Picture) size you can specify the ratio of IDR to P frames.
Protocol format	Select the transport protocol format: • **TCP**: Streaming communication with the camera takes place via port 554 by default. All communication (audio, video and control data) takes place only via this port. TCP is the preferred and best option of transport protocols.

Name	Description
	• **UDP**: With UDP, streaming also takes place via port 554, but video, audio and control data are transmitted via up to four additional ports. These ports are selected by the camera. Due to the higher number of ports and the associated firewall configuration issues, UDP is less suitable than TCP. • **Multicast**: With multicast, streaming also takes place via port 554. Audio, video and control data are streamed via an additional multicast address plus port. **Each camera in the network must be assigned to a unique multicast address. Multiple cameras must not be assigned to the same multicast address. It is possible for multiple cameras to use the same port, but they must not have the same multicast address because the primary separation in switches is performed using the multicast addresses. Using identical multicast addresses leads to network and device overload and incorrect display of the images.** • **HTTP**: With HTTP, an RTSP stream is established via HTTP port 80 (RTSP and RTP via HTTP). In this case, the streaming protocol is established in the same way as with TCP, with the difference that port 80 is used instead of port 554, as this is not blocked by most firewalls. • **HTTPS**: With HTTPS, an encrypted RTSP stream is established via HTTPS port 443 (RTSP and RTP via HTTP). **Some cameras may not be able to support encrypted streaming. In this case, you need to select another transport protocol.**
Constant	You choose between a variable bitrate and a constant

Name	Description
/ Variable	bitrate. With a variable bitrate, the quality settings are used for the profile that is assigned to the media channel. A variable bitrate is generally recommended to ensure good image quality. However, if the stream should not exceed a certain bandwidth, you should use a constant bitrate.
Bitrate	Enter the bitrate. You can choose between the unit kbit and mbit.
Merge behaviour	**Only for streaming behaviour Single.** Select how the possibly different settings for live and storage under media channels are combined: • **Best wins**: The highest value specified in the corresponding settings is used for all parameters. • **Worst wins**: The worst value specified in the corresponding settings is used for all parameters. • **Live wins**: The settings for the live stream are used. • **Storage wins**: The settings for the storage are used. • **Quality of storage wins but highest framerate**: Resolution and quality are selected via the settings for the storage channel. The higher frame rate of the two, live or storage, is used.
Resolution	**Only for streaming behaviour DSS.** Select which resolutions are to be set up in the camera. Select two resolutions.

12. Activate or deactivate the PTZ control option 8 for the respective sensor. The slider indicates whether PTZ is supported for the respective sensor or not.
    - Sensor supports PTZ
    - Sensor does not support PTZ

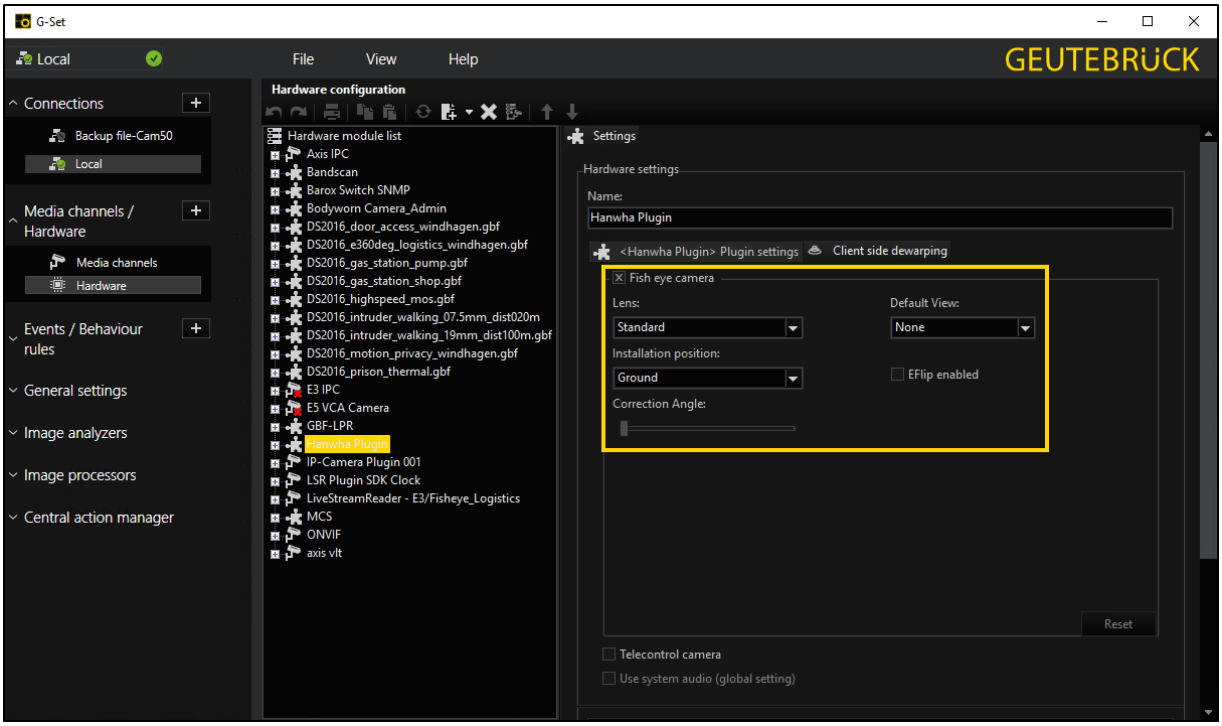
- ? It is not known whether the sensor supports PTZ

13. The device is added to the **Device List**and your configuration is buffered.

14. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**). The Remote Plugin and associated media channels are added in G-Set.

> **IMPORTANT:** Do not manually add a Remote Plugin in G-Set and do not make any changes to the plugin settings. The plugin should only be added and changed via the Plugin Loader web interface.

In G-Set, you can activate and configure the fisheye dewarping for the device. Select the corresponding Remote Plugin in the **Hardware module list** and activate the **Fish eye camera** option on the **Client side dewarping** tab.

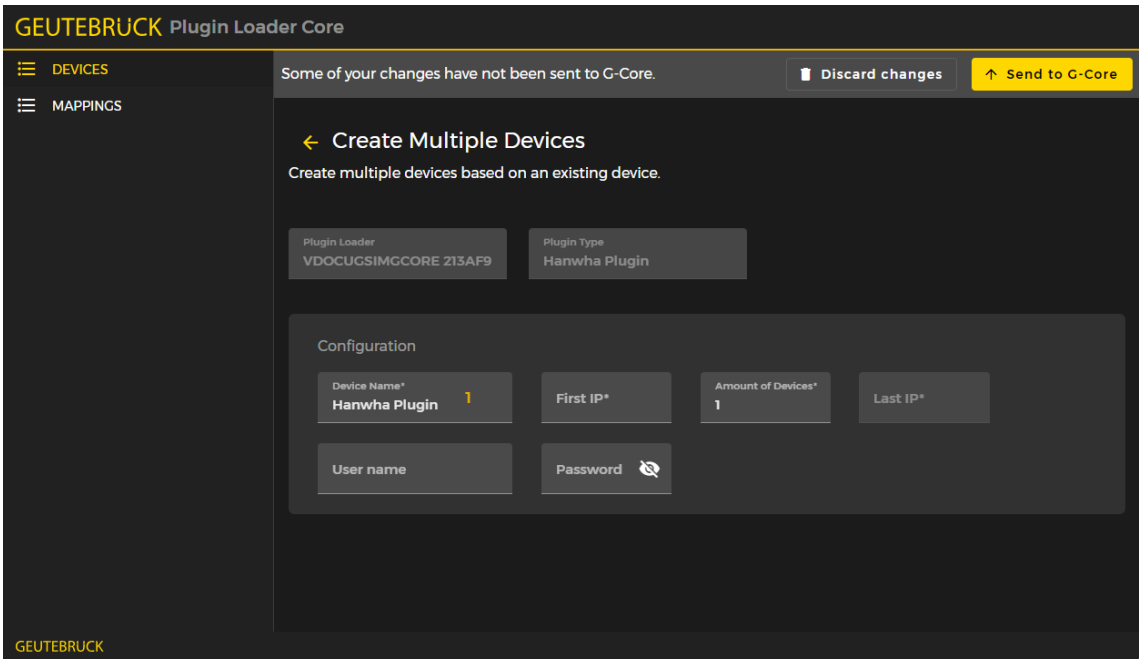


Create Multiple Devices

In the **Create Multiple Devices** view, you can create multiple devices based on the settings of a an existing device.

How to create multiple devices:

1. You can open the **Create Multiple Devices** view in the **Edit Device** view or the **Device List**.
   - In the **Edit Device** view, configure the device settings (see **Add Device**) and click the  icon.
   - In the **Device List**, hover your mouse over the row of the device you want to use as base for the multiple devices and click the  icon. The **Create Multiple Devices** view opens.

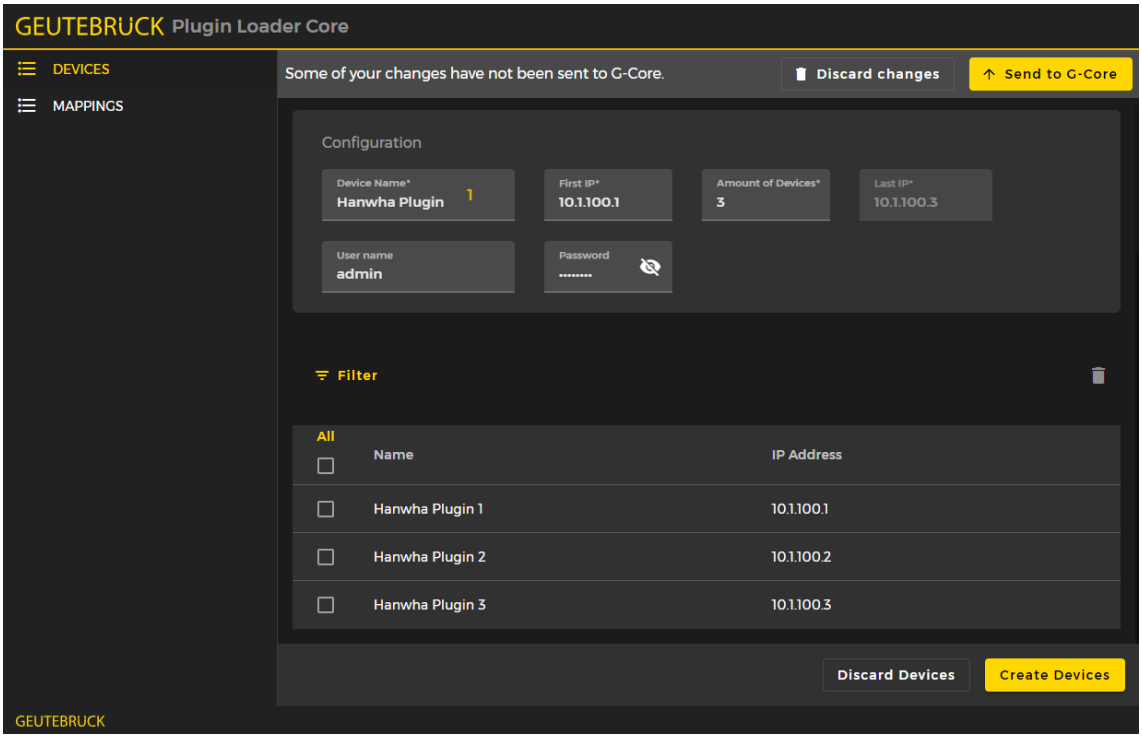


2. Specify the settings for the multiple devices:

Name	Description
Device Name	Enter the name of the device and the index with which the name of the first in the list of devices to be created should begin.
First IP	Enter the IP address with which the first device in the list of devices to be created should begin. The IP address is incremented for all subsequent devices. The list of devices to be created appears.
Amount of	Specify the number of devices you want to create. You can

Name	Description
Devices	create a maximum of 64 devices at the same time. **Note that if you change the number of devices, the multiple device list will be updated and any changes you have already made will be discarded.**
Last IP	The IP address of the last device in the list of devices to be created.
User name	Specify a username that is used for all devices to be created (optional).
Password	Specify a password that is used for all devices to be created (optional).

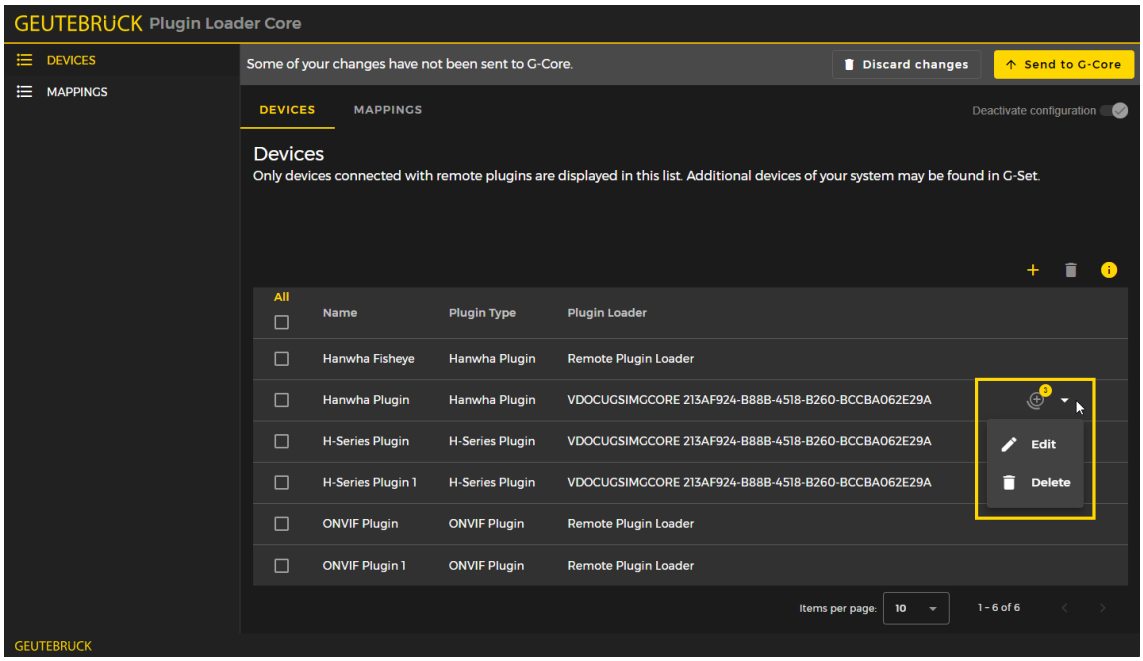
3. The list of devices to be created appears if you have specified the First IP.



4. To filter the list by the device name or IP address, click the icon.

5. To edit the settings of a device, click in the row of the respective device. The **Edit Device** dialog window opens. Edit the settings of the device (see **Add Device**) and click **Submit**.
6. To delete a device, select one or more devices by selecting the respective checkboxes. Click the icon. The **Delete Device Templates** dialog window opens. Confirm the dialog **Do you want to delete the selected templates?** with **Delete**.
7. Click **Create Devices**. The devices are added to the **Device List** and your configuration is buffered.

If you do not click Create Devices, the devices are prepared but not created. In the Device List and the Add Device view, an indicator appears next to the icon showing how many multiple devices are prepared for the selected device. Click the drop-down menu icon and select whether you want to Edit or Delete the prepared devices.



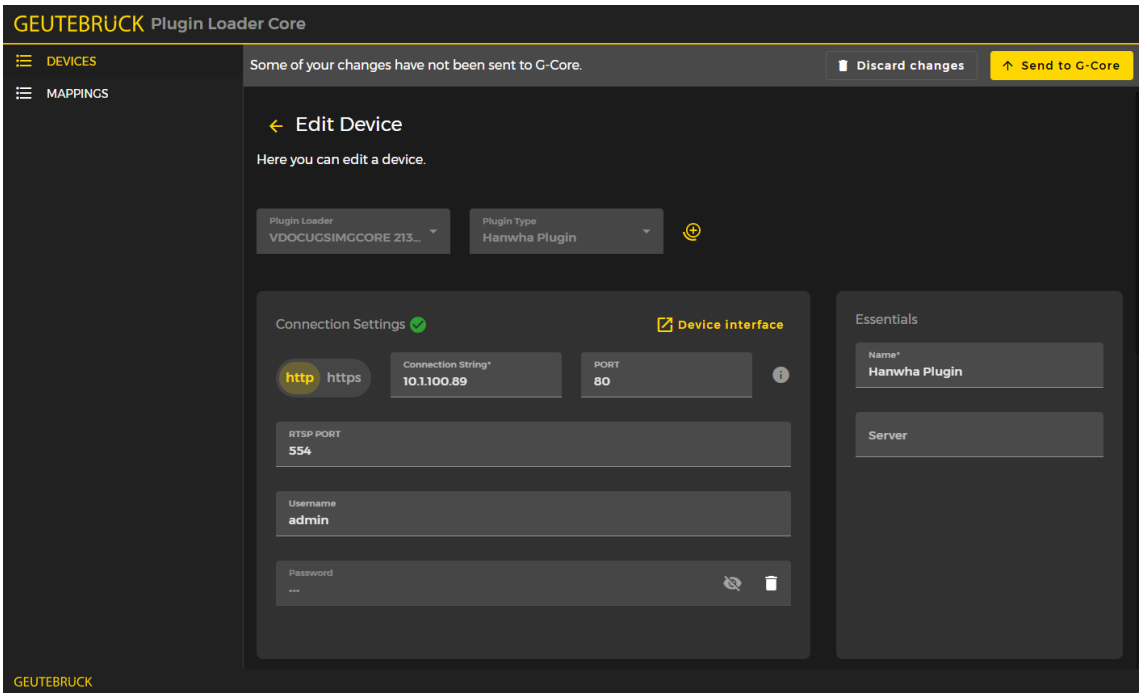
8. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**).

Edit Device

In the **Edit Device** view, you can edit a selected device.

How to edit a device:

1. In the **Device List**, hover your mouse over the row of the device you want to edit and click the ✎ icon. The **Edit Device** view opens.



2. Edit the device settings (see **Add Device**). Your configuration is buffered.

3. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**). The Remote Plugin and associated media channels are updated in G-Set.

> ⚠ **IMPORTANT:** Do not manually add a Remote Plugin in G-Set and do not make any changes to the plugin settings. The plugin should only be added and changed via the Plugin Loader web interface.

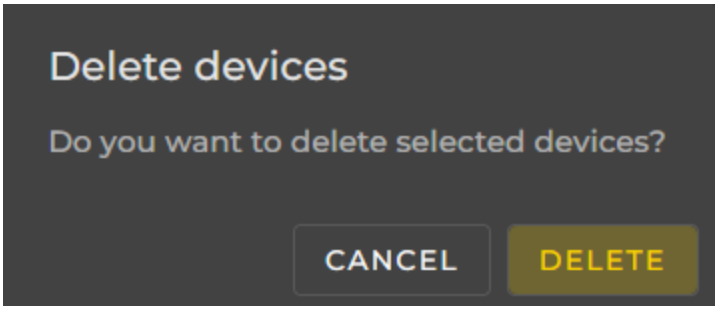
Delete Device

How to delete a device:

1. In the **Device List**, select the device you want to delete by selecting the checkbox.

> ⓘ **You can select multiple devices. To select all devices at once, click the checkbox in the list header.**

2. Click the icon. The **Delete devices** dialog window opens.



3. Confirm the dialog **Do you want to delete selected devices?** with **Delete**. Your configuration is buffered.
4. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**). The Remote Plugin is deleted in G-Set.

Use CameraRAWOutput Action

The `CameraRAWOutput` action enables you to send HTTP requests to a camera, allowing you to interact with the settings or functions of the camera via raw HTTP requests.

It is supported by the Remote Plugins **ONVIF**, **Hanwha** and **H-Series**.

Parameters

The action uses a combination of parameters to define the request details.

Parameter	Description
Camera (required)	Specify the camera to which the request should be sent. You can use a unique identifier or a name. Example: `Camera: {2, 1, "ONVIF Plugin 001"}`
Output (required)	Specify the relative URL path to which the HTTP request should be sent. This is the path component of the base URL of the camera for the HTTP request. Example: `Output: "/cgi-bin/admin/param.cgi?action=update&Image.I0.Text.String=Example"`
HTTPRequestMethod (optional)	Specify the HTTP method to be used for the request. The following methods are possible:

Parameter	Description
	• `GET`: Retrieves data from the camera (default). • `POST`: Sends data to the camera. Example: `HTTPRequestMethod:"POST"`
HTTPMessageBody (optional)	Specify the message body content to be sent with the HTTP request. This is used with `POST` requests. Specify the parameters or data to be included in the message body of the request. Example: `HTTPMessageBody:"param=value"`

Behavior

If the **Output** parameter is set, the Plugin Loader constructs the URL to send the request. The protocol (`http` or `https`), host/IP address and port are derived from the plugin configuration for the selected camera.

If only **Output** is specified, a `GET` request is sent to `<protocol>://<host>:<port><Output>`.

- `<protocol>` is determined from the plugin configuration (either `http` or `https`).
- `<host>` and `<port>` are taken from the plugin configuration.
- `<Output>` is the specified path component.

If **HTTPRequestMethod** is set to `POST`, then **HTTPMessageBody** is used to define the content of the request messagebody.

Example

GET request

```
CameraRAWOutput (Camera: { 2, 1, "ONVIF Plugin 001" },
Output: "/cgi-
bin/admin/param.cgi?action=update&Image.I0.Text.String=Hallo
Review")
```

This sends a `GET` request to:

```
<protocol>://<host>:<port>/cgi-
bin/admin/param.cgi?action=update&Image.I0.Text.String=Exam
ple
```

Example

POST request with message body

```
CameraRAWOutput (Camera: { 2, 1, "ONVIF Plugin 001" },
Output: "/cgi-bin/path?param=value",
HTTPRequestMethod:"POST", HTTPMessageBody:"param=value")
```

This sends a `POST` request to:

```
<protocol>://<host>:<port>/cgi-bin/path?param=value
```

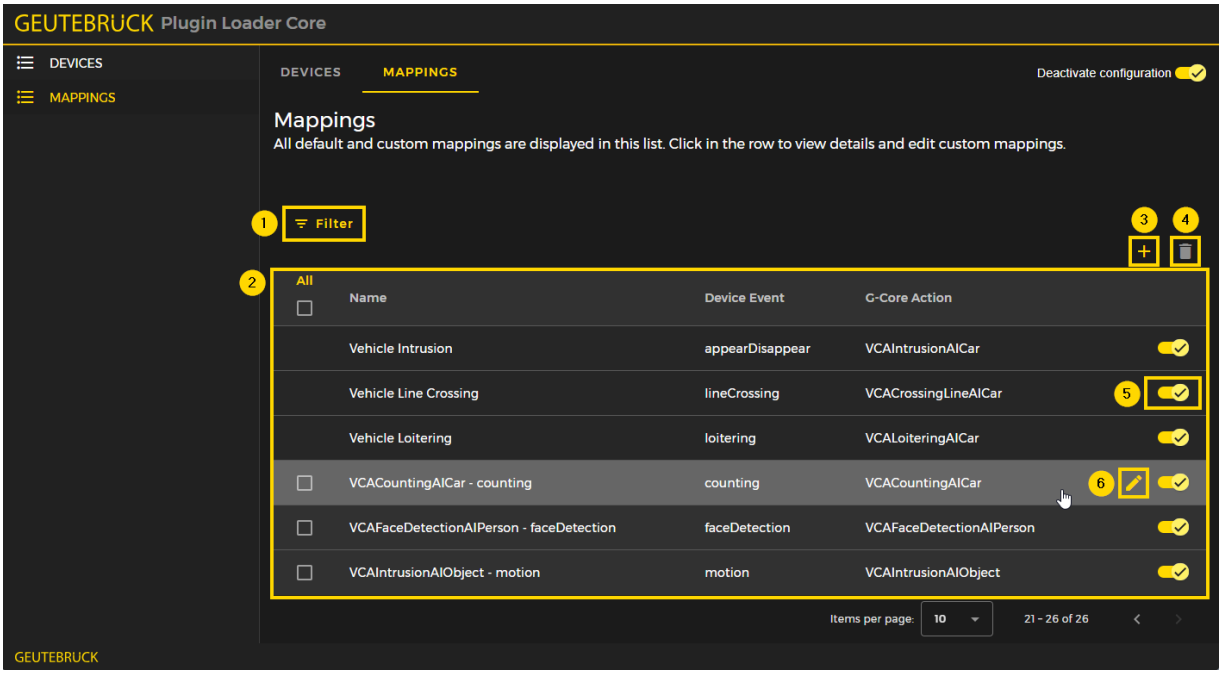
The request message body contains `param=value`.

Mapping

Mapping List

In the **Mappings** view, you can configure and manage the mappings of your system.

To access the configuration view, you must activate the configuration (see Activate Configuration).



The view consists of the following elements:

	Element	Description
1		Click this icon to enter a search term and filter the mapping list.
2	Mapping list	In the list all default and custom mappings are listed. The following information is listed: • **Name**: Name of the mapping. • **Device Event**: Name of the mapped device event. • **G-Core Action Name**: Name of the mapped G-Core action.
3		Click this icon to add a custom mapping (see **Add Custom Mapping**).
4		Click this icon to delete a custom mapping (see **Delete Custom Mapping**).
5		Activate or deactivate a mapping. Activated mappings are applied to all available

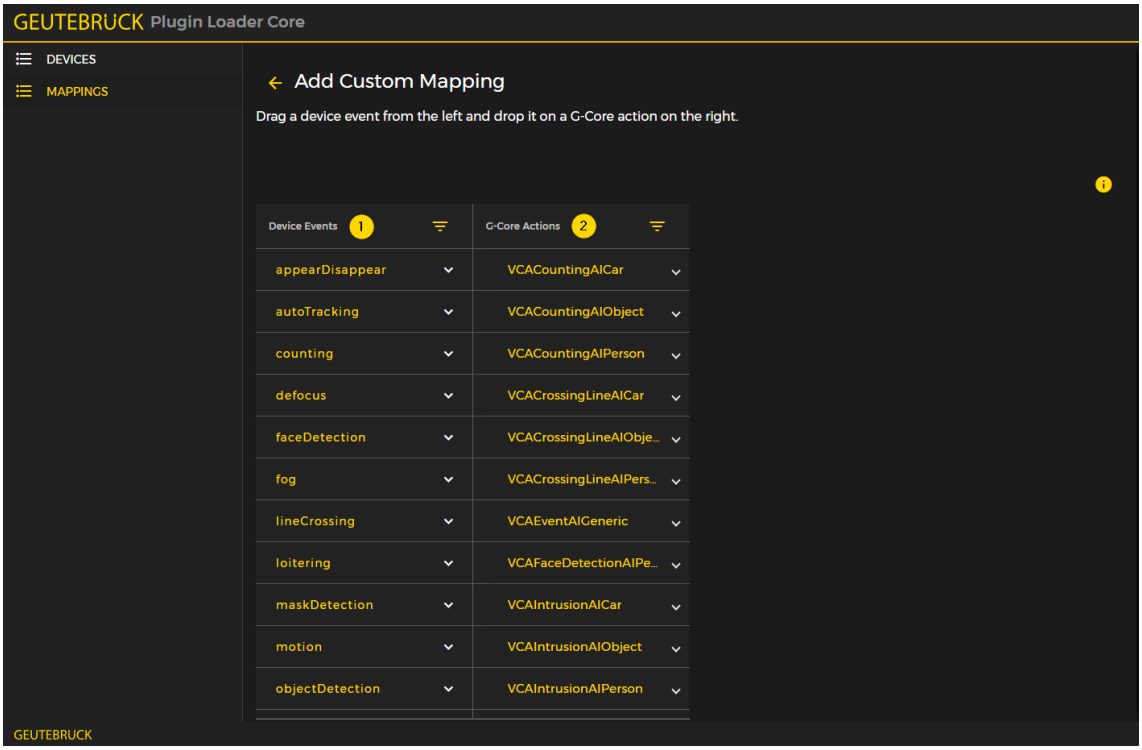
	Element	Description
		cameras connected with remote plugins (see **Device List**).
6	/	: This is a default mapping. Click on the row to display the predefined settings (see **Default Mappings**). : This is a custom mapping. Click on the row to edit it (see **Edit Custom Mapping**).

Add Custom Mapping

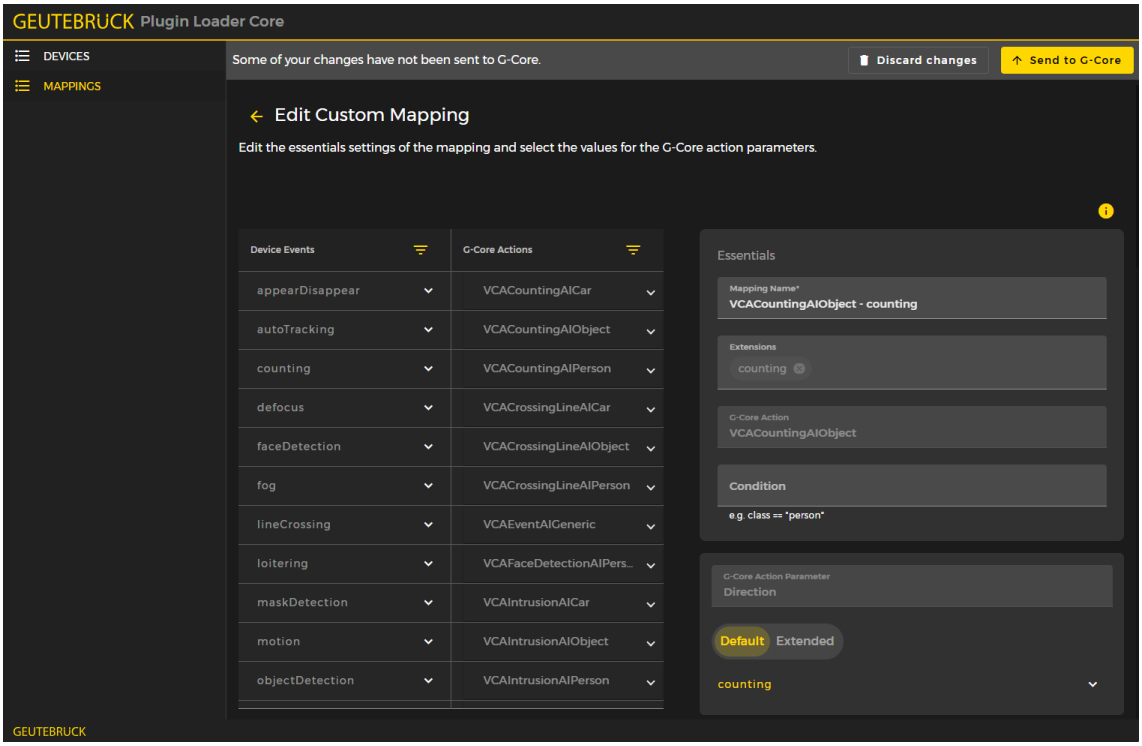
In the Add Custom Mapping view, you can add a custom mapping to your system. To open this view, click the + icon in the Mappings view (see **Mapping List**).

How to add a custom mapping:

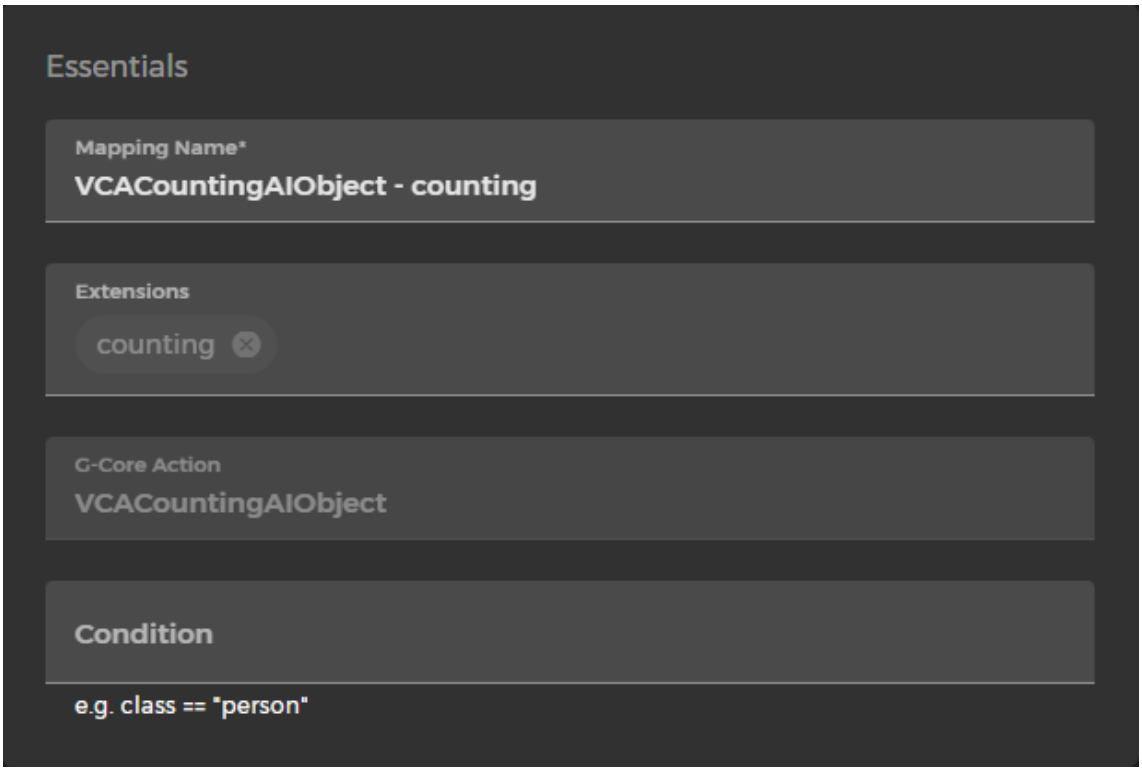
1. Map a device event from the Device Events list 1 to a G-Core action in the G-Core Actions list 2 using drag and drop. For descriptions of the values of the device events and G-Core actions, see **Mapping Values**.



The view changes to **Edit Custom Mapping** and the settings dialogs of the selected mapping appear.

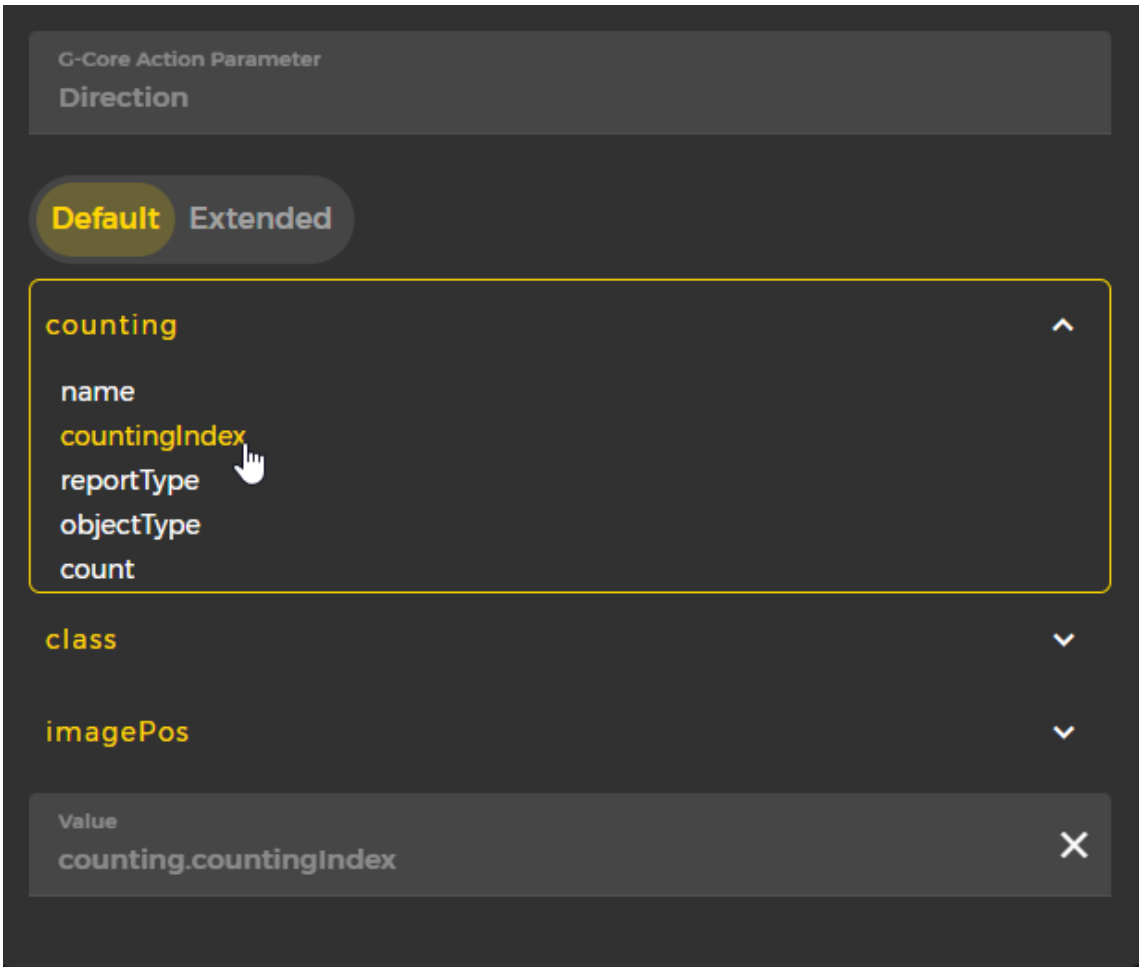


2. Specify the **Essentials** settings:

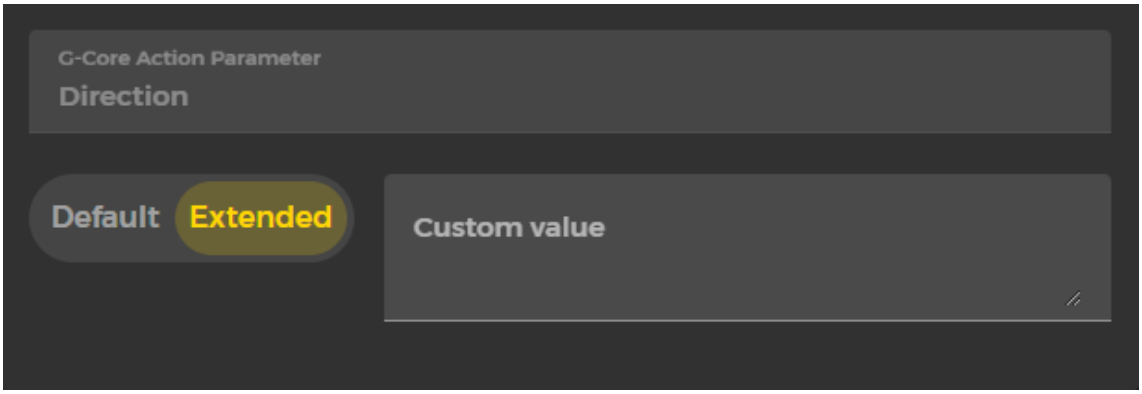


Name	Description
Mapping Name	Specify the name of the mapping. The default name is composed of `<G-Core Action> - <Device Event>`.
Extensions	Click in this field and select the extensions you want to add to your mapping from the drop-down menu. For descriptions of the values of the extensions, see **Mapping Values**.
G-Core Action	Name of the G-Core action (not editable).
Condition	Specify which extension data should be displayed for an event.

3. You can select an extension value for each G-Core Action Parameter to display only this value for the action parameter. The value for the parameter Channel is set by default and cannot be selected. For descriptions of the values of the extensions, see **Mapping Values**.



If you want to select multiple values for a G-Core action parameter, click on the **Extended** slider. Enter the values in the **Custom value** input field. To enter a custom value, use the Scriban scripting language. Click on the ℹ icon at the top right to display examples of custom values. For detailed information on Scriban, see **here**.



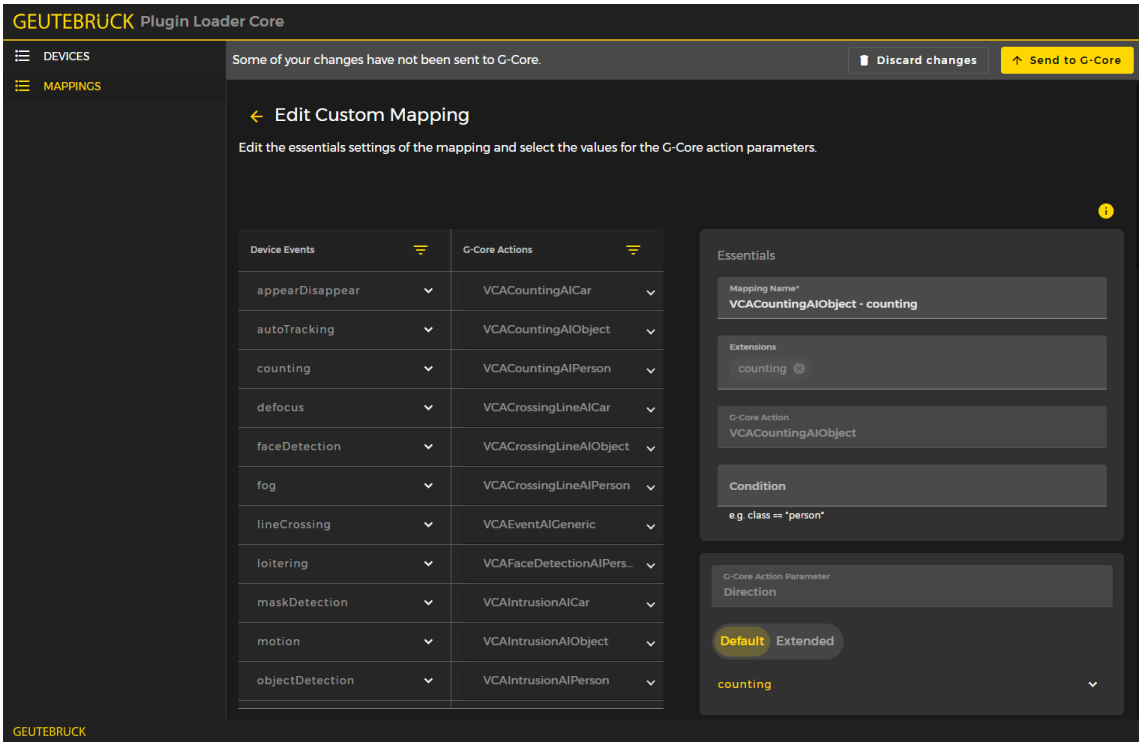
4. The mapping is added to the **Mapping List** and your configuration is buffered.
5. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**).

Edit Custom Mapping

In the Edit Custom Mapping view, you can edit a selected custom mapping.

How to edit a custom mapping:

1. In the **Mapping List**, hover your mouse over the row of the custom mapping you want to edit and click the ✎ icon. The Edit Custom Mapping view opens.



2. Edit the mapping settings (see **Add Custom Mapping**). Your configuration is buffered.
3. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**).

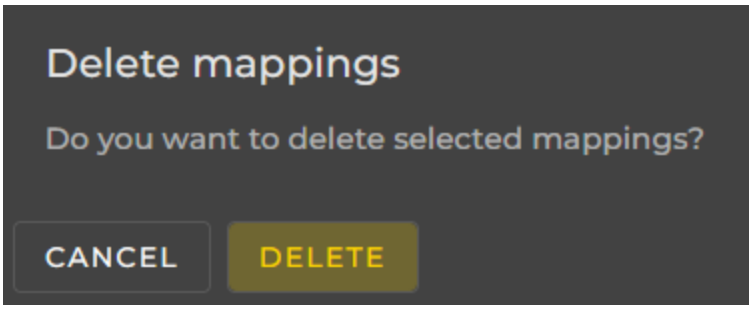
Delete Custom Mapping

How to delete a custom mapping:

1. In the **Mapping List**, select the custom mapping you want to delete by selecting the checkbox.

   You can select multiple mappings. To select all custom mappings at once, click the checkbox in the list header.

2. Click the icon. The Delete mappings dialog window opens.

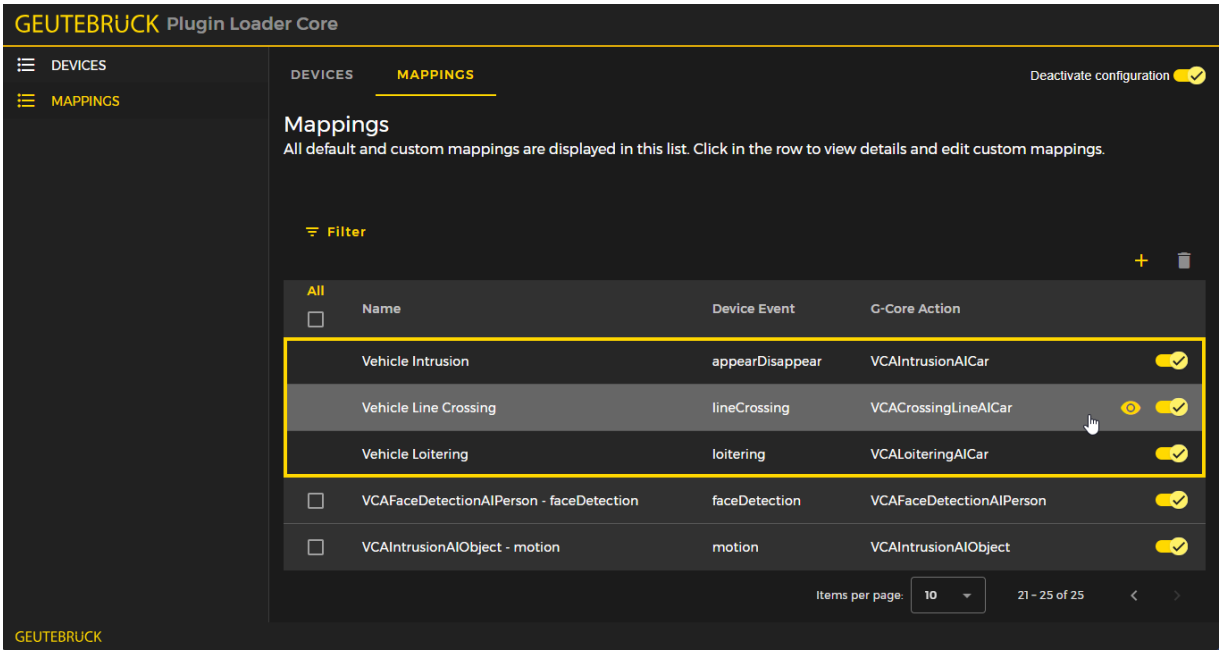


3. Confirm the dialog Do you want to delete selected mappings? with Delete. Your configuration is buffered.
4. To complete your configuration and update G-Set, you must send your configuration to G-Core (see **Send Configuration to G-Core**).

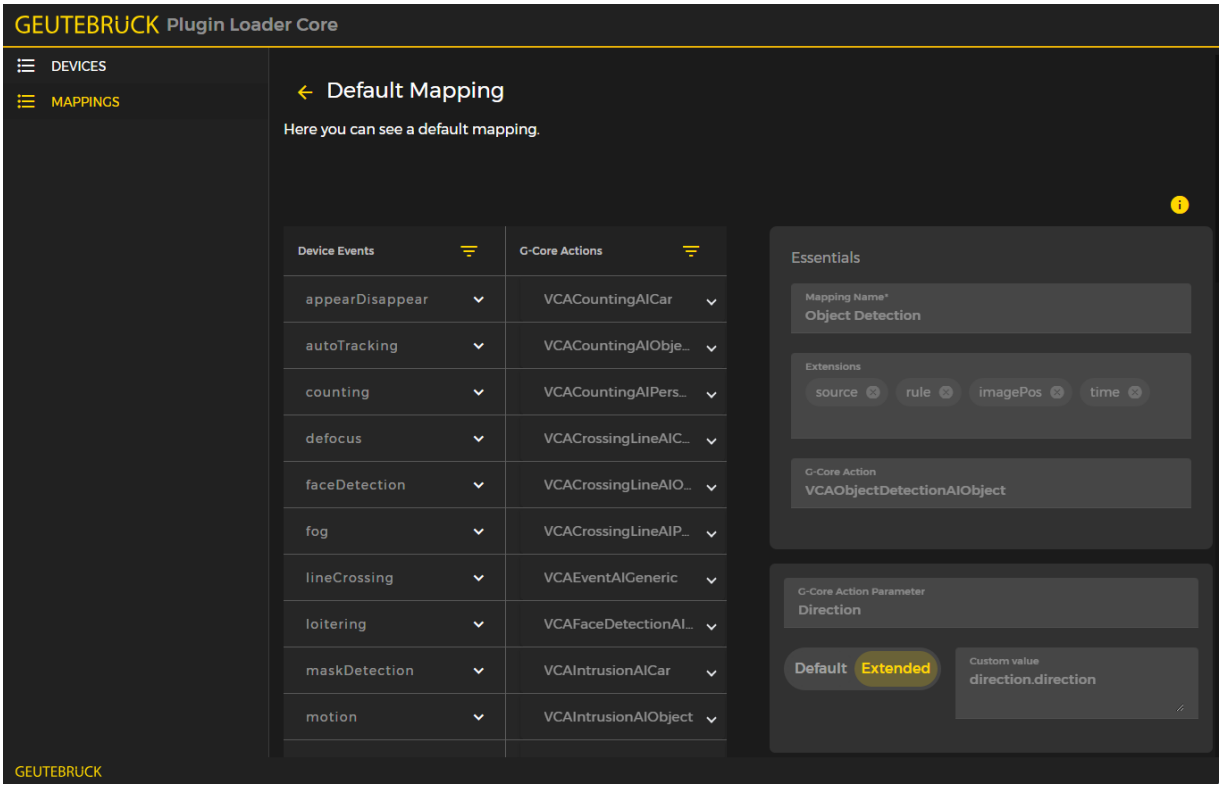
Default Mappings

The mapping list contains some predefined default mappings. These are marked with the icon and cannot be edited or deleted from the list.

The default mappings are activated by default. You can deactivate a mapping using the slider.



Click on the row of a default mapping to display the predefined settings.



For descriptions of the values of the device events and G-Core actions, see **Mapping Values**.

The following default mappings are available:

Defocus Detection

Device Event: defocus

G-Core Action: VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__Defocus
TimeStamp	time.iso

Face Detection

Device Event: faceDetection

G-Core Action: VCAFaceDetectionAIPerson

G-Core Action Parameter	Value
Gender	gender.gender
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Fog Detection

Device Event: fog

G-Core Action: VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__Fog
TimeStamp	time.iso

Motion Detection

Device Event: motion

G-Core Action: VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__Motion
TimeStamp	time.iso

Object Counting

Device Event: counting

G-Core Action: VCACountingAIObject

G-Core Action Parameter	Value
Direction	direction.direction
TotalCount	counting.count
ObjectClass	class.labels[0].label
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Object Detection

Device Event: TriggerEvent

G-Core Action: VCAObjectDetectionAIObject

G-Core Action Parameter	Value
Direction	direction.direction
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Object Intrusion

Device Event: appearDisappear

G-Core Action: VCAIntrusionAIObject

G-Core Action Parameter	Value
Direction	direction.direction
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Object Line Crossing

Device Event: lineCrossing

G-Core Action: VCACrossingLineAIObject

G-Core Action Parameter	Value
Direction	direction.direction
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Object Loitering

Device Event: loitering

G-Core Action: VCALoiteringAIObject

G-Core Action Parameter	Value
ObjectClass	class.labels[0].label
RuleName	rule.name
PositionX	imagePos.center.x
PositonY	imagePos.center.y
TimeStamp	time.iso

Person Counting

Device Event: counting

G-Core Action: VCACountingAIPerson

G-Core Action Parameter	Value
Direction	direction.direction
TotalCount	counting.count
Mask	if maskDetection.detected __yes else __no end
Gender	gender.gender
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Person Detection

Device Event: maskDetection

G-Core Action: VCAObjectDetectionAIPerson

G-Core Action Parameter	Value
Direction	direction.direction
Mask	if maskDetection.detected __yes else __no end
Gender	gender.gender
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Person Intrusion

Device Event: appearDisappear

G-Core Action: VCAIntrusionAIPerson

G-Core Action Parameter	Value
Direction	direction.direction
Mask	if maskDetection.detected __yes else __no end

G-Core Action Parameter	Value
Gender	gender.gender
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Person Line Crossing

Device Event: lineCrossing

G-Core Action: VCACrossingLineAIPerson

G-Core Action Parameter	Value
Direction	direction.direction
Mask	if maskDetection.detected __yes else __no end
Gender	gender.gender
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Person Loitering

Device Event: loitering

G-Core Action: VCALoiteringAIPerson

G-Core Action Parameter	Value
Mask	if maskDetection.detected __yes else __no end
Gender	gender.gender
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Shock Detection

Device Event: shock

G-Core Action: VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__Shock
TimeStamp	time.iso

Tampering Detection

Device Event: tampering

G-Core Action: VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__Tampering
TimeStamp	time.iso

Temperature Detection

Device Event: temperature

G-Core Action:VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__Temperature
TimeStamp	time.iso

Too Bright Detection

Device Event: tooBright

G-Core Action: VCAEventAIGeneric

G-Core Action Parameter	Value
EventName	__TooBright
TimeStamp	time.iso

Vehicle Counting

Device Event: counting

G-Core Action: VCACountingAICar

G-Core Action Parameter	Value
Direction	direction.direction
TotalCount	counting.count
VehicleClass	vehicle.type.label
Make	vehicle.brand.label
RuleName	rule.name

G-Core Action Parameter	Value
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Vehicle Detection

G-Core Action: TriggerEvent

Device Event: VCAObjectDetectionAICar

G-Core Action Parameter	Value
VehicleClass	vehicle.type.label
Direction	direction.direction
Make	vehicle.brand.label
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Vehicle Intrusion

G-Core Action: appearDisappear

Device Event: VCAIntrusionAICar

G-Core Action Parameter	Value
VehicleClass	vehicle.type.label

G-Core Action Parameter	Value
Direction	direction.direction
Make	vehicle.brand.label
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Vehicle Line Crossing

G-Core Action: lineCrossing

Device Event: VCACrossingLineAICar

G-Core Action Parameter	Value
VehicleClass	vehicle.type.label
Direction	direction.direction
Make	vehicle.brand.label
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Vehicle Loitering

G-Core Action: loitering

Device Event: VCALoiteringAICar

G-Core Action Parameter	Value
VehicleClass	vehicle.type.label
Make	vehicle.brand.label
RuleName	rule.name
PositionX	imagePos.center.x
PositionY	imagePos.center.y
TimeStamp	time.iso

Mapping Values

AI VCA Actions

Description of the G-Core actions.

VCAObjectDetectionAIObject

Generic object detection action for unspecified objects.

Parameter	Description
Channel	Channel.
Direction	Direction of movement.
Color	Color of detected object.
ObjectClass	Class of the detected object.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAObjectDetectionAIPerson

Generic object detection action for persons.

Parameter	Description
Channel	Channel.
TopColor	Top color of the detected Person.
BottomColor	Bottom color of the detected Person.
Direction	Direction of movement.
HairColor	Hair color of the detected Person.
Mask	True if mask was detected.
Bag	True if bag was detected.
Gender	Gender of the detected Person.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAObjectDetectionAICar

Generic object detection action for vehicles.

Parameter	Description
Channel	Channel.
VehicleClass	Vehicle class of the detected vehicle.
Direction	Direction of movement.

Parameter	Description
Color	Color of the detected vehicle.
Make	Make or brand of the detected vehicle.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCACrossingLineAIObject

Generic line crossing detection action for unspecified objects.

Parameter	Description
Channel	Channel.
Direction	Direction of movement.
Color	Color of detected object.
ObjectClass	Class of the detected object.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Time stamp.
PositionX	X Coordinate of the detected object.
Position Y	Y Coordinate of the detected object.

VCACrossingLineAIPerson

Generic line crossing detection action for persons.

Parameter	Description
Channel	Channel.
TopColor	Top color of the detected Person.
BottomColor	Bottom color of the detected Person.
Direction	Direction of movement.
HairColor	Hair color of the detected Person.
Mask	True if mask was detected.
Bag	True if bag was detected.
Gender	Gender of the detected Person.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Time stamp.
position x	PositionX
position y	PositionY

VCACrossingLineAICar

Generic line crossing detection action for vehicles.

Parameter	Description
Channel	Channel.
VehicleClass	Vehicle class of the detected vehicle.
Direction	Direction of movement.
Color	Color of the detected vehicle.
Make	Make or brand of the detected vehicle.

Parameter	Description
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAIntrusionAIObject

Generic intrusion detection action for unspecified objects.

Parameter	Description
Channel	Channel.
Direction	Direction of movement.
Color	Color of detected object.
ObjectClass	Class of the detected object.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAIntrusionAIPerson

Generic intrusion detection action for persons.

Parameter	Description
Channel	Channel.
TopColor	Top color of the detected Person.
BottomColor	Bottom color of the detected Person.
Direction	Direction of movement.
HairColor	Hair color of the detected Person.
Mask	True if mask was detected.
Bag	True if bag was detected.
Gender	Gender of the detected Person.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAIntrusionAICar

Generic intrusion detection action for vehicles.

Parameter	Description
Channel	Channel.
VehicleClass	Vehicle class of the detected vehicle.
Direction	Direction of movement.
Color	Color of the vehicle.
Make	Make or brand of the detected vehicle.

Parameter	Description
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAFaceDetectionAIPerson

Generic face detection action for persons.

Parameter	Description
Channel	Channel.
EyeColor	Eye color of the detected Person.
Gender	Gender of the detected Person.
Mask	True if mask was detected.
PersonName	Name of the detected Person.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCALoiteringAIObject

Generic loitering detection action for unspecified objects.

Parameter	Description
Channel	Channel.
Color	Color of the detected object.
ObjectClass	Object class name.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
Since	Since timestamp.
TimeStamp	Timestamp.
PositionX	X Coordinate
PositionY	Y Coordinate

VCALoiteringAIPerson

Generic loitering detection action for persons.

Parameter	Description
Channel	Channel.
TopColor	Top color of the detected Person.
BottomColor	Bottom color of the detected Person.
HairColor	Hair color of the detected Person.
Mask	True if mask was detected.
Bag	True if bag was detected.
Gender	Gender of the detected Person.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.

Parameter	Description
Since	Since timestamp.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCALoiteringAICar

Generic loitering detection action for vehicles.

Parameter	Description
Channel	Channel.
VehicleClass	Vehicle class of the detected vehicle.
Color	Color of the detected vehicle.
Make	Make or brand of the detected vehicle.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
Since	Since timestamp.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCACountingAIObject

Generic counting action for counting objects.

Parameter	Description
Channel	Channel.
Direction	Direction of movement.
TotalCount	Total object count.
Color	Color of detected object.
ObjectClass	Object class name.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
Since	Since timestamp.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCACountingAIPerson

Generic counting action for persons.

Parameter	Description
Channel	Channel.
TopColor	Top color of the detected Person.
BottomColor	Bottom color of the detected Person.
Direction	Direction of movement.
TotalCount	Total object count.
Mask	True if mask was detected.
Bag	True if bag was detected.

Parameter	Description
Gender	Gender of the detected Person.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCACountingAICar

Generic counting action for vehicles.

Parameter	Description
Channel	Channel.
VehicleClass	Vehicle class of the detected vehicle.
Direction	Direction of movement.
Total count	Total object count.
Color	Color of the detected vehicle.
Make	Make or brand of the detected vehicle.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

VCAEventAIGeneric

Generic action for VCA AI events.

Parameter	Description
Channel	Channel.
EventName	Event category name.
RuleName	Name of the rule that triggered this event.
ObjectData	Raw object data.
TimeStamp	Timestamp.
PositionX	X Coordinate of the detected object.
PositionY	Y Coordinate of the detected object.

Device Events

Description of the device events.

appearDisappear

Appear Disappear event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

---

autoTracking

Auto tracking event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

---

counting

Counting event.

Value	Value Schema	Description	Example
name	String	""	""
countingIndex	Int	""	1
reportType	String	""	""
objectType	String	""	""
count	Int	""	100

defocus

Defocus event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

faceDetection

Face detection event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true
face	Bool	is detected?	true

fog

Fog event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

lineCrossing

Line crossing event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true
since	String	Time (iso time stamp) from which the object is loitering	2024-09-23T09:57:56.662Z

loitering

Loitering detection event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

maskDetection

Mask detection event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

motion

Motion detection event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

objectDetection

Object detection event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

shock

Shock event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

tampering

Tampering event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

temperature

Temperature event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

tooBright

Image too bright event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

wrongDirection

Wrong direction event.

Value	Value Schema	Description	Example
detected	Bool	is detected?	true

Extensions

Description of the extensions.

age

Label age.

Value	Value Schema	Description	Example
age	String	age range of person detected	25,34

class

Object classification.

labels

Label candidates.

Value	Value Schema	Description	Example
type	StringEnum	classification label	Human [Vehicle, Human]
likelihood	Float	label likelihood	0.74f

gender

Label gender.

Value	Value Schema	Description	Example
gender	StringEnum	gender of person detected	male [male, female]

imagePos

Image position information in normalized frame coordinates.

bounding

Bounding rectangle.

Value	Value Schema	Description	Example
x0	Float	min x	0.1f
y0	Float	min y	0.2f
x1	Float	max x	0.3f
y1	Float	max y	0.6f

center

Center of gravity.

Value	Value Schema	Description	Example
x	Float	x	0.35f
y	Float	y	0.4f

licensePlate

License plate information.

Value	Value Schema	Description	Example
number	String	""	SUJW1666
likelihood	Float	Likelihood of detection	0
type	String	Plate type	""
countryCode	String	Country Code of the plate	D
issuingEntity	String	Issuing entity	""
plateColor	String	color of the plate	blue

movement

Movement information.

Value	Value Schema	Description	Example
imageDir	StringEnum	Direction of movement in the image plane	left [up, upLeft, left, downleft, down, downright, right, upright]

object

Object tracking reference.

Value	Value Schema	Description	Example
id	Uuid	object id	C50F22C8-D91E-4C81-B46F-4A088BC7EAA5

rule

Rule engine parameters.

Value	Value Schema	Description	Example
name	String	Name of the Rule	Rule1

source

Emitter reference information.

Value	Value Schema	Description	Example
module	Uuid	Origin module id	C50F22C8-D91E-4C81-B46F-4A088BC7EAA0
channel	Uuid	Mapped media channel id	C50F22C8-D91E-4C81-B46F-4A088BC7EAA1

time

Detection time.

Value	Value Schema	Description	Example
utc	String	iso time string	2024-05-17T13:09:49.502Z

vehicle

Vehicle classification.

type

Vehicle class / type.

Value	Value Schema	Description	Example
label	StringEnum	Classification label	Car [Car, Truck, Truck Trailer, Bus, Bike, Motorbike, Scooter]
likelihood	Float	Label likelihood	0.74f

brand

Vehicle brand / make.

Value	Value Schema	Description	Example
label	String	Classification label	Audi
likelihood	Float	Label likelihood	0.74f

model

Vehicle model.

Value	Value Schema	Description	Example
label	String	Classification label	A3
likelihood	Float	Label likelihood	0.74f

color

Vehicle color.

Value	Value Schema	Description	Example
name	String	Name of the color	red
likelihood	Float	Color likelihood	0.8f

Send Configuration to G-Core

During configuration in the Plugin Loader, G-Set is locked and you cannot make any changes. To complete your configuration and update G-Set, you must send your configuration to G-Core.

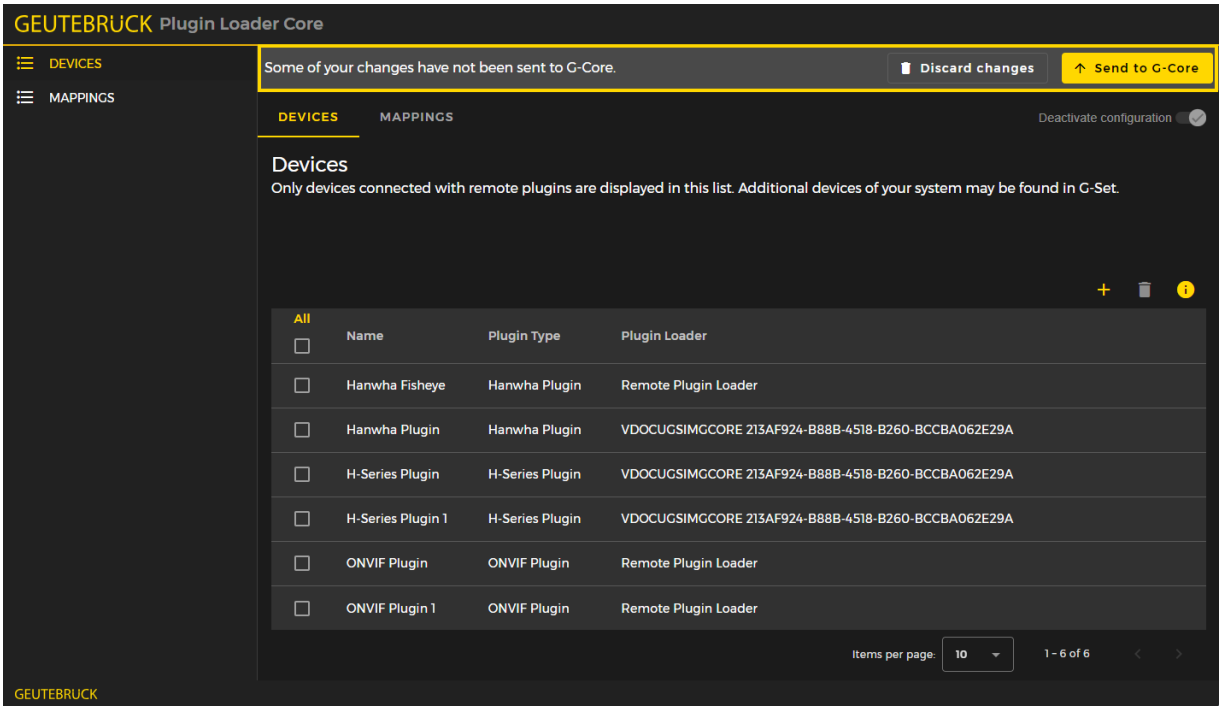
If you have made changes in the Plugin Loader, the banner **Some of your changes have not been sent to G-Core** appears. Your configuration is buffered until you send it to G-Core or discard it. You are not able to deactivate the configuration, while changes exist.

-  Send to G-Core:

  Click this button to send your configuration to G-Core. G-Set is updated.

-  Discard changes:

  Click this button to discard your changes. The **Discard changes** dialog window appears. Type the security word to confirm the deletion and click **Discard**. All changes you have made since the last time you sent a configuration to G-Core are discarded.


GEUTEBRÜCK Plugin Loader Core
DEVICES
MAPPINGS
Some of your changes have not been sent to G-Core.
Discard changes
Send to G-Core
DEVICES
MAPPINGS
Deactivate configuration
Devices
Only devices connected with remote plugins are displayed in this list. Additional devices of your system may be found in G-Set.
All
Name
Plugin Type
Plugin Loader
Hanwha Fisheye
Hanwha Plugin
Remote Plugin Loader
Hanwha Plugin
Hanwha Plugin
VDOCUGSIMGCORE 213AF924-B88B-4518-B260-BCCBA062E29A
H-Series Plugin
H-Series Plugin
VDOCUGSIMGCORE 213AF924-B88B-4518-B260-BCCBA062E29A
H-Series Plugin 1
H-Series Plugin
VDOCUGSIMGCORE 213AF924-B88B-4518-B260-BCCBA062E29A
ONVIF Plugin
ONVIF Plugin
Remote Plugin Loader
ONVIF Plugin 1
ONVIF Plugin
Remote Plugin Loader
Items per page:
10
1 - 6 of 6
GEUTEBRUCK

Development

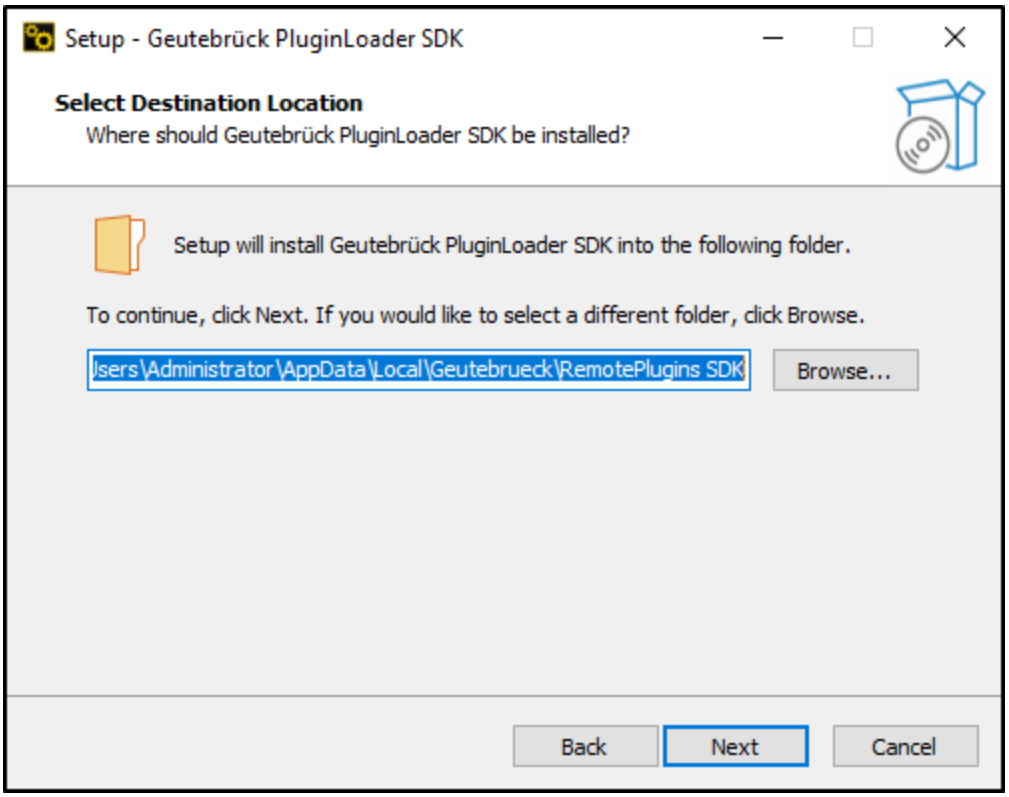
SDK Installation

The installer for SDK installation is only available at sdk.geutebrueck.com.

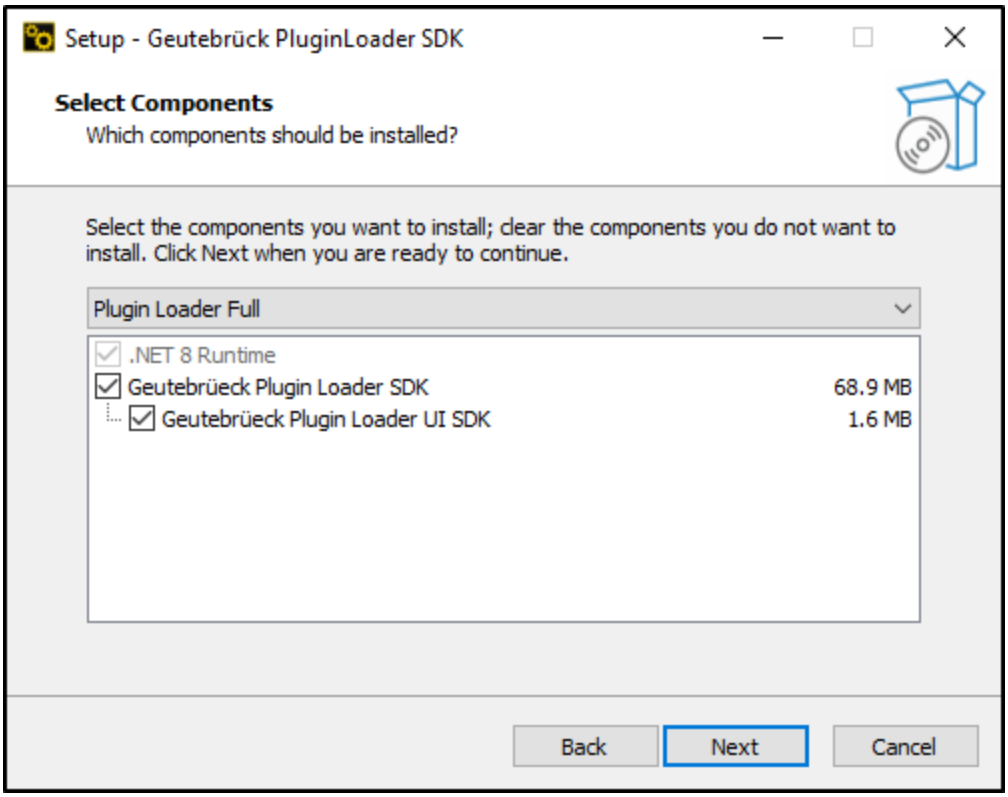
To develop your own Remote Plugins, install the Plugin Loader SDK. The SDK installer contains several example projects to explain different workflows (ImageModules, PTZ and I/Os) of the Remote Plugin architecture. It also contains a Plugin Loader Simulator that can be used to test newly developed plugins directly without the need of an entire G-Core system.

How to install the Remote Plugins SDK:

1. Run the `Geutebrück PluginLoader SDK_xxx.exe` file.
2. In the **License Agreement** dialog window, select the option **I accept the agreement** and click **Next**.
3. In the **Select Destination Location** dialog window, you can change the installation path. Click **Next**.



4. In the **Select Components** dialog window, select the components you want to install. Click **Next**.



5. In the **Ready to Install** dialog window, click **Install**.
6. When the installation is complete, click **Finish**.

SDK Backend

You can also find this technical documentation for the SDK backend in the `README.md` file in the installation directory of the SDK.

Components

A Remote Plugin consists of the following components:

Component	Description
Plugin	The main instance of the Remote Plugin. Manages the entire life cycle of an instance of the Remote Plugin and instantiates modules.

Component	Description
Plugin provider	Provides information about the plugin. Aggregates metadata and default settings.
Module	Processes events and commands of an interaction category. • Image modules stream image data from a camera to the server (`IImageModule`). • IO modules handle input/output capabilites of a camera (`IIOModule`). • PTZ modules forward pan, tilt, and zoom capabilites to a camera as commanded by an operator (`IPtzModule`).
Individual settings	Represents the individual configuration of the plugin.

Interfaces

The following interfaces must be considered during SDK development:

Interface	Description
IPluginProvider	Provides information about the plugin. Aggregates metadata and default settings.
RemotePluginBase IImagePlugin	These interfaces are crucial for creating a plugin. They contain basic methods about the ID, information, and modules that a plugin can contain, as well as methods for creating and retrieving modules.
ImageModuleBase	In order to add modules for plugins, this interface must be inherited so that it is possible to work with the streams.
IImageStreamer	This interface is used by plugins to push the images.

Write a Remote Plugin

These imports are used in the following example:

```
using Google.Protobuf.WellKnownTypes;
```

```
using Google.Protobuf;
using Microsoft.Extensions.DependencyInjection;
using RemotePlugins.Api.Interfaces;
using RemotePlugins.Api.Structures.Enums;
using RemotePlugins.Sdk;
using RemotePlugins.Sdk.Modules;
using SkiaSharp;
using RemotePlugins.Api.Structures;
```

How to write a Remote Plugin:

1. Create a C# class library project (.NET 8.0).

2. Use `Geutebrueck.RemotePlugins.Sdk` as SDK and specify the version to use.

   To do this, replace `Microsoft.NET.Sdk` with `Geutebrueck.RemotePlugins.Sdk/<version>`. The `.csproj` file should now look like this:

   ```
   <Project Sdk="Geutebrueck.RemotePlugins.Sdk/1.0.0">

       <PropertyGroup>
       <TargetFramework>net8.0</TargetFramework>
       <ImplicitUsings>enable</ImplicitUsings>
       <Nullable>enable</Nullable>
       </PropertyGroup>

   </Project>
   ```

3. Write a plugin class derived from the base class `PluginBase`.

   The virtual method `OnSettingsChanged()` can be overridden and is called when the JSON configuration of the instance has changed. Services are automatically injected into the constructor (if any).

   ```
   public class SamplePlugin : PluginBase
   {
       //protected override void OnSettingsChanged() { }

       protected override void OnDispose() { }
   }
   ```

4. Write a provider class that extends `PluginProviderBase<T>`.

```
public class SamplePluginProvider : Plu-
ginProviderBase<SamplePlugin>
{
    private readonly int _maxImageModules = 1;

    protected override IBasePluginSettings GetPlu-
ginDefaultSettings()
    {
        return PluginSettings.Create(_maxImageModules);
    }

    protected override PluginDefinition GetPluginDefinition()
    {
        return new("Sample Plugin")
        {
            MaxInstances = 16,
            MaxImageModules = _maxImageModules,
            InputChannels = 0,
            OutputChannels = 0,
        };
    }

    /* Optionally configure your services */
    public void ConfigureServices(IServiceCollection services)
    {
        // Register your plugin services here
    }

    /* Optionally configure the app */
    public void Configure(IPluginLoaderApplication app)
    {
        // configure your plugin here
        // app.Services.GetRequiredServcie<SampleService>
().DoAtStartup();
    }
}
```

5. You can define individual settings for the plugin in addition to the default settings.
    a. Define the individual settings as public properties in an additional class derived from `IndividualPluginSettingsBase`.

```
public class SamplePluginIndividualSettings : Indi-
vidualPluginSettingsBase
{
    public int Test1 { get; set; } = 10;
    public string Test2 { get; set; } = "test";
}
```

b. Pass the type of the settings class in `GetPluginDefaultSettings()`.

```
protected override IBasePluginSettings GetPlu-
ginDefaultSettings()
{
    return Plu-
ginSettings.Create<SamplePluginIndividualSettings>(_
maxImageModules);
}
```

Write a Plugin Module

There are many functionalities that plugins can support, but not every plugin has to support all functionalities. For this reason, there are the modules that a plugin can provide to support certain functionalities. To do this, the module must implement the corresponding module interface. A plugin can have several modules, even several instances of the same module.

In this example, an ImageModule is implemented that allows streaming of images.

The currently available interface types for modules which are recognized by the framework are:

- `IImageModule` for image modules
- `IIOModule` for IO modules
- `IPtzModule` for PTZ modules

How to write a plugin module:

1. Write a module class derived from the base class `PluginModuleBase` and implementing the interface `IImageModule`.

Call the `StopStream()` method from the `OnDispose()` method in order to stop the image stream if a module instance is removed. The private fields and properties are specific to this module and may be different for other modules.

- `_streamActive` controls starting and stopping the image stream.
- `_imageBase` holds the static image to stream.
- `IndividualSettings` gives access to the individual settings of our `SamplePlugin`.

In addition, a small set of virtual methods can be overridden if necessary:

- `RegisterSubmodules(IList<>)` informs the module about other modules of the same plugin instance. Module instances shared in this way must be added to the `CreateModuleParams.SubmoduleReferences` property in the plugin before calling `CreatePluginModule()`.
- `SetPluginSettings()` is called when new settings have been deserialized. The base implementation can handle this and calls `OnSettingsChanged()`.
- `OnSettingsChanged()` is called once all settings have been set.

Additional base classes are provided (`ImageModuleBase`, `IOModuleBase`, `PTZModuleBase`) that contain frequently used code for their respective module type and encapsulate events into abstract methods that can be overridden.

Services are automatically injected into the constructor.

```
public class SamplePluginImageModule : ImageModuleBase
{
    private readonly SamplePluginImageStreamer _streamer;

    public SamplePluginIndividualSettings IndividualSettings =>
        PluginSettings.DeviceSettings.IndividualSettings as
SamplePluginIndividualSettings ?? new();

    public SamplePluginImageModule()
    {
        _streamer = new(this);
    }

    protected override void StartStream(Guid streamId)
    {
```

```
            _streamer.StartStream();
        }

        protected override void StopStream(Guid streamId)
        {
            _streamer.StopStream();
        }

        protected override IEnumerable<IImageStreamer>
GetImageStreamers()
        {
            yield return _streamer;
        }

        protected override IImageStreamer SelectStream(string tag,
ImageResolution resolution, StreamSelectionBehaviour
streamingBehaviour)
        {
            return _streamer;
        }

        //protected override void Initialize() { }

        //protected override void RegisterSubmodules(IList<IBaseMod-
ule> modules) { }

        //protected override void SetPluginSettings
(IBasePluginSettings settings) => base(settings);

        //protected override void OnSettingsChanged() { }

        protected override void OnDispose()
        {
            _streamer.StopStream();
        }
}
```

2. Register the module as module capability to your plugin inside the plugin constructor.

   See the example plugins for reference on how to use the other overloads of `AddModuleCapability`.

```
public class SamplePlugin : PluginBase
```

```
{
    [...]

    public SamplePlugin()
    {
        // register the image module using the default factory
        AddModuleCapability<IImageModule, SamplePlu-
ginImageModule>();
    }

    [...]
}
```

3. Implement an image streamer defined by `IImageStreamer`.

   Start with the `RunStream` function. The task of this function is to stream images continuously. For this purpose, the static image is initialized at the beginning. Then the function CreateAndPublishCommand is called in an infinite loop to stream the image along with a current time stamp. The gateway receives the stream and forwards it to the G-Core server.

```
public class SamplePluginImageStreamer : IImageStreamer
{
    public Guid Id {get;} = Guid.NewGuid();
    private readonly SamplePluginImageModule _imageModule;

    private bool _streamActive = false;

    public SamplePluginImageStreamer(SamplePluginImageModule
imageModule)
    {
        _imageModule = imageModule;
    }

    private async Task RunStream()
    {
        _streamActive = true;
        var imageBase = new SKBitmap(800, 600);
        using (SKCanvas canvas = new(imageBase))
            canvas.DrawRect(new SKRect(0, 0, 800, 600),
new SKPaint() { Style = SKPaintStyle.Fill, Color = SKColors.Blue
});

        while (_streamActive)
```

```
        {
            var img = imageBase.Encode(SKEn-
codedImageFormat.Jpeg, 100).ToArray();
            var picTime = Timestamp.FromDateTimeOffset
(DateTimeOffset.UtcNow);

            _imageModule.PushImage(
                ByteString.CopyFrom(img),
                ImageCodecType.Jpg,
                Id
            );

            await Task.Delay(1000);
            Console.WriteLine($"{_imageMod-
ule.IndividualSettings?.Test1 ?? -1} {_imageMod-
ule.IndividualSettings?.Test2 ?? "null"}");
        }
    }

    public void StartStream()
    {
        if (!_streamActive)
            _ = RunStream();
    }

    public void StopStream()
    {
        _streamActive = false;
    }

    public ImageStreamInfo GetStreamInfo()
    {
        return new()
        {
            Id = Id.Wrap()
        };
    }

    public void HandleStreamTask(StreamTask streamTask, ImageRes-
olution resolution) { }

    public void UpdateImageResolution(ImageResolution
resolution) { }
}
```

Multihead Support

Some cameras have several heads, each with its own sensor and usually looking in different directions. Each camera head corresponds to an `IImageModule` in the plugin. The number of camera heads supported by the plugin can be set in the `PluginProviderBase<T>.GetPluginDefinition` via the parameter `MaxImageModules`. Individual settings for the different heads (derived from the base class `IndividualChannelSettingsBase`) can be registered in the method `GetPluginDefaultSettings`, as well as the global individual settings for all channels (derived from the base class `IndividualPluginSettingsBase`).

```
public class ExampleMultiheadPluginProvider : Plu-
ginProviderBase<ExampleMultiheadPlugin>
{
    private readonly int _maxImageModules = 5;

    protected override IBasePluginSettings GetPluginDefaultSettings()
    {
        return Plu-
ginSettings.Create<ExampleMultiheadIndividualSettings, ExampleMulti-
headIndividualChannelSettings>(_maxImageModules);
    }


    protected override PluginDefinition GetPluginDefinition()
    {
        return new("Example Multihead Plugin")
        {
            MaxInstances = 16,
            MaxImageModules = _maxImageModules,
            InputChannels = 0,
            OutputChannels = 0,
        };
    }
}
```

A useful default channel setting is the `HeadId` which is predefined as a sequential number starting by zero. This index can be used to access the corresponding camera head in the streaming URL, and can be adjusted in the setup. It can be easily requested by `PluginSettings?.GetChannelSettings (ModuleId)?.HeadId` from the image modules.

Include Your Own Settings UI

1. Create a new UI project and an initial component for the settings UI using the UI SDK. You will find this in the installation directory of the SDK.
2. Write the frontend SDK. See **SDK Frontend** for detailed information.
3. You need to define two values:
   - `UiModule`: The name of the entry module, usually `./<your-project-name>`. You can find this name inside the `module-federation.config.js` as key for `exposed`.
   - `UiComponent`: The component of the entry module, usually `<your-project-name>Component`.
4. Provide the defined values to the framework via the `PluginDefinition` object from the `PluginProvider` (see **Write a provider class**).

```
public class ExampleMultiheadPluginProvider : Plu-
ginProviderBase<ExampleMultiheadPlugin>
{
    [...]

    protected override PluginDefinition GetPluginDefinition()
    {
        return new()
        {
            [...]

            UiApp = "<public-dist-dir>",
            UiModule = "./<entry-module>",
            UiComponent = "<entry-component>"
        };
    }
}
```

5. Build the UI files and copy the built UI files, which are located in `dist` after building, to the plugin directory that can be accessed by the Plugin Loader. The default directory structure of a plugin is:

```
YourPlugin/
├─ net8.0/
├─ public/
```

```
│   └─ <public-dist-dir>/
│      └─ ** PLACE THE UI DIST FILES HERE **
└─ plugin.json
```

Do not place your files directly in the `public` directory. The value for `<public-dist-dir>` must be unique for your plugin.

The directory `public` is specified in the `plugin.json` as the root directory for public files that the browser can access. While it is recommended to keep the default value, you can change the value for `PublicFiles > Root` in `plugin.json` to any directory relative to the directory containing `plugin.json` or to an absolute path. The path provided via `RemotePluginInfo.UiApp` is always considered relative to this path. This value can also be changed at design time within the `.csproj` of your plugin. Set the property `PublicFilesDir` to the desired path. It will automatically be placed in the `plugin.json` during the build.

For example, if you want your assets to be loaded from `C:\MyPublicAssets`, the `.csproj` file should look like this:

```
<Project Sdk="Geutebrueck.RemotePlugins.Sdk/1.0.0">

    <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <PublicFilesDir>C:\MyPublicAssets</PublicFilesDir>
    </PropertyGroup>

</Project>
```

Add a Plugin to a Plugin Loader

Every plugin has a `plugin.json` file which defines the path to load the plugin assembly. All relative paths in this file are relative to the directory containing it. Absolute paths can also be specified.

The default installation path of the Plugin Loader is `C:\Program Files\Geutebrueck\PluginLoader`. To add a plugin to a plugin loader, copy the plugin files to a new `plugins/<your-plugin-name>` directory. The actual name of the directory inside `plugins` is not important. Then restart the Plugin Loader.

```
RemotePluginLoader/
├─ plugins/
│  └─ <your-plugin-name>/
│     ├─ net8.0/
│     │  └─ ...
│     └─ plugin.json
...
```

Example Plugins

The examples of the plugin implementation source code are located in the **Exampels** folder in installation directory of the SDK.

Plugin	Description
ExamplePlugin	A blank, loadable plugin without any functionality.
ExampleJpegPlugin	Streams continuously generated JPEG frames to the server.
ExampleH264Plugin	Takes H.264 frames from a file and streams them to the server in a loop. Uses individual settings to configure the file location.
ExamplePtzPlugin	Visualizes pan, tilt and zoom commands with generated JPEG frames.
ExampleMultiheadPlugin	Supports up to five image modules per instance, each simulating a camera head.
SDKPlugin	A comprehensive example that combines and demonstrates most of the possible functions.

Plugin Loader Simulator

The Remote Plugin Simulator is a tool to test your plugin and view its images. The plugin is loaded into a simulated environment, no G-Core system is required.

1. To make your plugin available to the simulator, copy it to the `Simulator/plugins` directory as described for the Plugin Loader (see **Add a Plugin to a Plugin Loader**):

```
Simulator/
├─ plugins/
│  └─ <your-plugin-name>/
│     ├─ net8.0/
│     │  └─ ...
│     └─ plugin.json
...
```

2. Run the `PluginLoaderSimulator.exe` file located in the **Simulator** folder in the installation directory of the SDK.

SDK Frontend

The frontend SDK is installed with the Plugin Loader SDK Installer. After the installation the frontend SDK is located in the **RemotePluginsUISDK** folder in the installation directory of the SDK.

Documentation:

You can find the technical documentation for the SDK frontend in the file `documentation.md` in the subdirectory `RemotePluginsUISDK\docs\plugin-loader-doc`. Follow the steps in this documentation to write the frontend SDK.

> **This documentation describes the technologies used as well as the setup of the development environment. The Compodoc environment described there gives an overview and describes the available source code.**

Frontend Plugin:

The documentation file `documentation.md` describes how a plugin is created and integrated in the Plugin Loader service. This is necessary to load the created plugin in the Geutebrück system or in the appropriate web user interface.

For more information on creating a user interface for a plugin and integrating it into the Geutebrück system, refer to the technical documentation in the project.

Standalone Plugin:

In addition to the provided frontend SDK, which provides a number of useful libraries for connecting Angular specific applications, there is also the possibility to use another frontend framework or even only HTML, CSS and JavaScript for the development of the user interfaces.

Integrating other frameworks may lead to unexpected problems. Implementation using only HTML, CSS and JavaScript requires hosting the web application independently and must be used as a standalone web application.

The Angular libraries provided allow for easy use, implementation, and integration. Using other frontend frameworks or only HTML, CSS, and JavaScript require standalone implementation by using the provided API.

Technical alterations reserved.

GEUTEBRÜCK GmbH
Im Nassen 7-9 | D-53578 Windhagen
Tel. +49 (0)2645 137-0 | Fax-999
info@geutebrueck.com

www.geutebrueck.com