

# Manual vFlash

Version 2.2  
English

## **Impressum**

Vector Informatik GmbH  
Ingersheimer Straße 24  
D-70499 Stuttgart

The information and data given in this user manual can be changed without prior notice. No part of this manual may be reproduced in any form or by any means without the written permission of the publisher, regardless of which method or which instruments, electronic or mechanical, are used. All technical information, drafts, etc. are liable to law of copyright protection.

© Copyright 2012, Vector Informatik GmbH  
All rights reserved.








## Table of Contents

<b>1</b>	<b>Manual Information</b>	<b>5</b>
1.1	About this user manual	5
1.1.1	Certification	6
1.1.2	Warranty	6
1.1.3	Registered trademarks	6
<b>2</b>	<b>Product Overview</b>	<b>7</b>
<b>3</b>	<b>Licensing and Installation</b>	<b>7</b>
3.1	Licensing	7
3.2	Installation	8
3.3	Communication Hardware Setup	8
3.3.1	Configuration of CAN network interface	8
3.3.2	Configuration of FlexRay network interface	9
3.3.3	Configuration of LIN network interface	10
3.3.4	Configuration of Ethernet connection	11
<b>4</b>	<b>Execute Reprogramming</b>	<b>13</b>
<b>5</b>	<b>Create Project</b>	<b>15</b>
<b>6</b>	<b>Configure Project</b>	<b>16</b>
6.1	Flashing based on different flash data containers	16
6.2	General	16
6.3	Data Configuration	17
6.3.1	Flashing based on Native Data	17
6.3.2	Flashing based on ODX-F Data	18
6.4	Communication Configuration	21
6.5	Miscellaneous Configuration	24
6.6	OEM Specific Configuration	26
6.6.1	Flashing based on Ford flash containers	26
6.6.2	Flashing based on GM flash containers	27
6.6.3	Flashing based on Fiat flash containers	29
<b>7</b>	<b>Save and Exchange Flash Project</b>	<b>32</b>
<b>8</b>	<b>Advanced Features</b>	<b>33</b>
8.1	Compression	33
8.2	Multi-Driver Support	35
8.3	Automation Interface for Batch Processing	36
<b>9</b>	<b>Additional Information</b>	<b>38</b>
9.1	Seed Extension	38
9.2	Force Boot Mode	38
9.3	vFlash Template	39

<b>10</b>	<b>FAQ</b>	<b>40</b>
<b>11</b>	<b>Examples</b>	<b>42</b>
11.1	Demos	42
11.2	Seed&Key DLL	42
11.3	vFlash Automation	42
<b>12</b>	<b>Vector UDS Reference Project</b>	<b>43</b>

# 1 Manual Information

## 1.1 About this user manual

Style	Utilization
<b>bold</b>	Blocks, surface elements, window- and dialog names of the software. Accentuation of warnings and advices. <b>[OK]</b> Push buttons in brackets <b>File Save</b> Notation for menus and menu entries
<b>Windows</b>	Legally protected proper names and side notes.
<b>Source Code</b>	File name and source code.
<b>Hyperlink</b>	Hyperlinks and references.
<b>&lt;CTRL&gt;+&lt;S&gt;</b>	Notation for shortcuts.
Symbol	Utilization
	Here you can obtain supplemental information.
	This symbol calls your attention to warnings.
	Here you can find additional information.
	Here is an example that has been prepared for you.
	Step-by-step instructions provide assistance at these points.
	Instructions on editing files are found at these points.
	This symbol warns you not to edit the specified file.

### 1.1.1 Certification

#### Certified Quality Management System

Vector Informatik GmbH has ISO 9001:2000 certification. The ISO standard is a globally recognized standard.

#### Spice Level 3

The Embedded Software Components business area at Vector Informatik GmbH achieved process maturity level 3 during a HIS-conformant assessment.

### 1.1.2 Warranty

#### Restriction of warranty

We reserve the right to change the contents of the documentation and the software without notice. Vector Informatik GmbH assumes no liability for correct contents or damages which are resulted from the usage of the documentation. We are grateful for references to mistakes or for suggestions for improvement to be able to offer you even more efficient products in the future.

### 1.1.3 Registered trademarks

#### Registered trademarks

All trademarks mentioned in this documentation and if necessary third party registered are absolutely subject to the conditions of each valid label right and the rights of particular registered proprietor. All trademarks, trade names or company names are or can be trademarks or registered trademarks of their particular proprietors. All rights which are not expressly allowed are reserved. If an explicit label of trademarks, which are used in this documentation, fails, should not mean that a name is free of third party rights.

> **Windows, Windows 7, Windows Vista, Windows XP, Visual Studio, .NET** are trademarks of the Microsoft Corporation.

## 2 Product Overview

### Overview

vFlash is an easy-to-use tool for programming one or more ECUs. Because of its flexible approach, it can support the different flash specifications of a wide variety of automotive OEMs without requiring modifications by the end user.

It supports ECU programming over CAN, FlexRay and over Ethernet (DoIP), either individually or in an ECU network group.

vFlash may be used either interactively or by remote control. In the interactive mode, the user loads a project and starts programming by the push of a button. In remote control mode, vFlash can be controlled via a C programming interface.

### Functions

vFlash utilizes flash projects to save and exchange the configuration required for programming individual ECUs. Flash projects contain all relevant settings and reference the data files to be programmed.

If the focus is on flexibility in changing the flash data, the referenced data files can simply be exchanged in the background, e.g. as part of an automated software build process. In this case, the flash project remains unchanged.

If the focus is on a process-safe exchanging of flash configurations, the flash projects may be exchanged in the Pack&Go format. In this case, all relevant settings together with the data files are packed in one archive file.

The user creates flash projects based on flash templates. A flash template defines the diagnostic services, the flash procedure and the pre-settings that are relevant to the flash process for a group of ECUs of a specific automotive OEM (e.g. for a vehicle platform). In practice, the flash specifications of automotive OEMs differ enormously, not least of all due to the differences in diagnostic protocols and in-house flash standards. The advantage of the flash templates concept is that the differences described above remain largely hidden from the user.

## 3 Licensing and Installation

### 3.1 Licensing

#### Requirements

1. For communication vFlash needs the Vector XL Interface Family communication hardware or an Ethernet adapter (DoIP).
2. vFlash needs a valid license bit set on a connected Vector CAN interface or a Vector USB dongle.



#### How to license vFlash:

If you ordered vFlash together with a Vector CAN/FlexRay hardware

1. vFlash is already licensed with the hardware you bought (the vFlash license bit is already set on the hardware).
2. Connect the hardware interface with the vFlash license bit to your PC before you run vFlash.

If you ordered vFlash for usage with an already available Vector CAN/FlexRay hardware

1. When ordering vFlash you have to provide the "License Status Report" of your Vector hardware. Use the "Vector Hardware Config" and select **File | Generate License Status**. Store the report and send it to Vector. If you have more than one hardware interface, choose the one which shall be the license carrier. This hardware interface has to be connected to your PC whenever you want to use vFlash. However, you can also communicate using another Vector hardware interface.
2. Vector will return the license file and instructions on how to activate your new Vector software.

If you ordered vFlash for usage with an already available Vector USB Dongle, you need to send the dongle to Vector to license vFlash.

## 3.2 Installation

### System requirements

Component	Windows XP (32-bit)	Windows Vista (32-bit)/ Windows 7 (32-bit / 64-bit)
Processor	1,6GHz	1,6GHz
Memory (RAM)	1GB	2GB
Hard disk	100MB	
Screen Resolution	1024 x 768	



#### How to install vFlash:

1. Ensure your Vector CAN/FlexRay hardware driver is already installed (if used on CAN/FlexRay, not required if only used on Ethernet).
2. Place the vFlash CD in the CD ROM drive of your computer. The vFlash setup program will appear.
3. If the setup program does not appear automatically, open the Windows Explorer, change to your CD ROM drive and double-click on the setup program Setup.exe to start the installation.
4. Follow the setup program instructions.

## 3.3 Communication Hardware Setup

### Supported communication network

vFlash is capable to reprogram ECUs using CAN, FlexRay or Ethernet– depending on the capabilities of the ECUs. Different configuration procedures are required.

### 3.3.1 Configuration of CAN network interface

#### Communication channels

Using a CAN network vFlash communicates via the Vector CAN hardware interface. Vector hardware interfaces have two or more channels and even multiple Vector hardware interfaces may be connected with your PC. Therefore you need to choose which hardware and which channel vFlash shall use. The "Vector Hardware Config" application is used to do the channel assignment.





### How to configure the communication channels:

1. Assign “vFlash - CAN 1” to a real hardware channel or a virtual channel: Right-click on the desired channel (e.g. CANcardXL – Channel 2) and select vFlash – CAN 1.

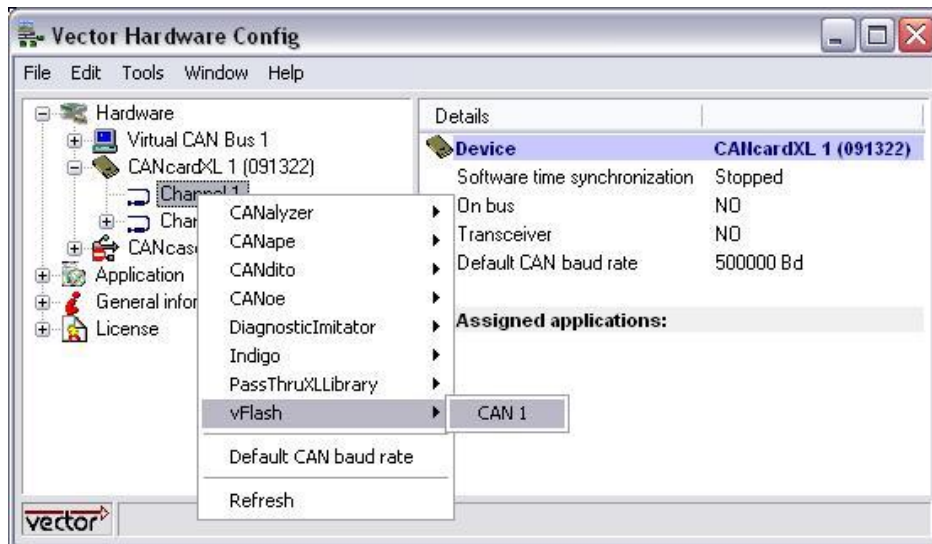


Figure 3-1: Vector Hardware Config – Assign channel

2. The result of these steps will be that your logical application channel “vFlash – CAN 1” is assigned to the desired real hardware channel or the virtual channel.

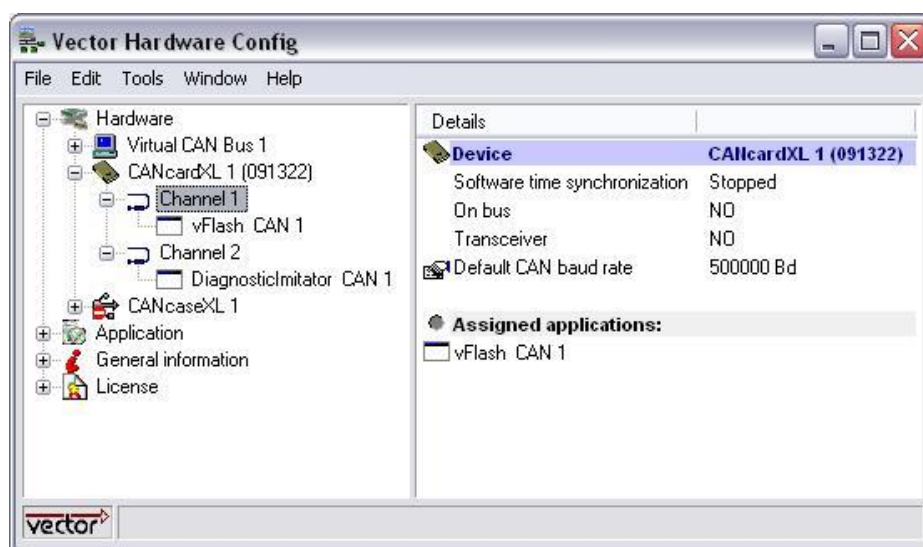


Figure 3-2: Vector Hardware Config – Configuration result

### 3.3.2 Configuration of FlexRay network interface

#### Communication channels

Using a FlexRay network vFlash communicates via the Vector FlexRay hardware interface. Vector hardware interfaces have two or more channels and even multiple Vector hardware interfaces may be connected with your PC. Therefore you need to choose which hardware and which channel vFlash shall use. The “Vector Hardware Config” application is used to do the channel assignment.



#### How to configure the communication channels:

1. Assign “vFlash - FlexRay 1” to a real hardware channel: Right-click on the desired channel (e.g. VN3600 – Channel 2) and select vFlash – FlexRay1.

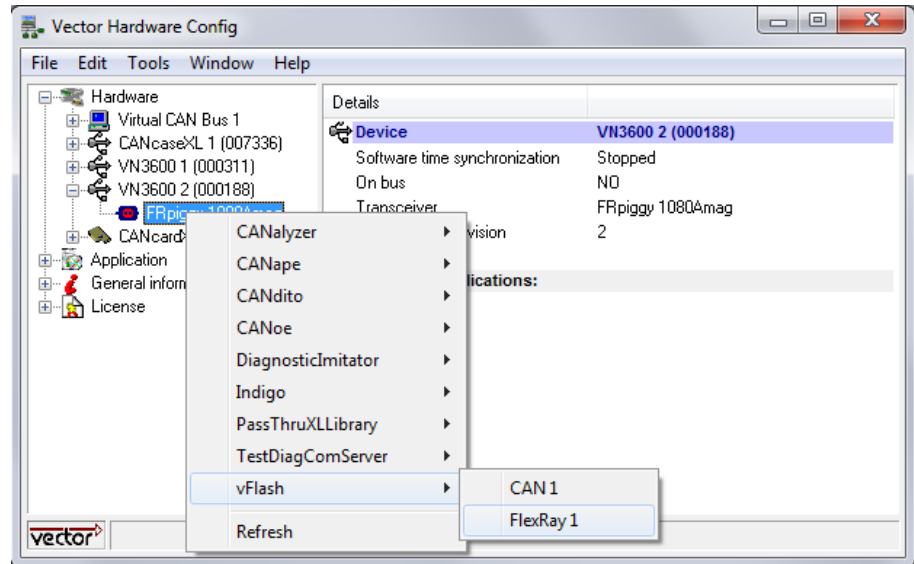


Figure 3-3: Vector Hardware Config – Assign channel

2. The result of these steps will be that your logical application channel “vFlash - FlexRay1” is assigned to the desired FlexRay hardware channel.

### 3.3.3 Configuration of LIN network interface

#### Communication channels

Using a LIN network vFlash communicates via the Vector LIN hardware interface. Vector hardware interfaces have two or more channels and even multiple Vector hardware interfaces may be connected with your PC. Therefore you need to choose which hardware and which channel vFlash shall use. The “Vector Hardware Config” application is used to do the channel assignment.



#### How to configure the communication channels:

1. Assign “vFlash - LIN 1” to a real hardware channel or a virtual channel: Right-click on the desired channel (e.g. VN1630 – Channel 1) and select vFlash – LIN 1.

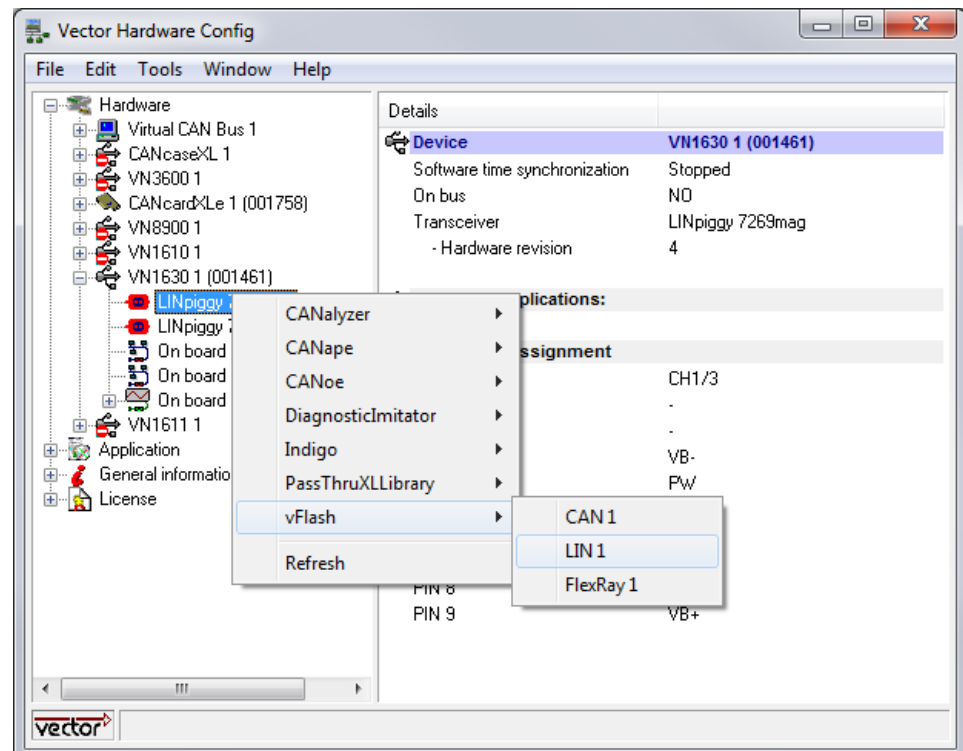


Figure 3-4: Vector Hardware Config – Assign channel

2. The result of these steps will be that your logical application channel “vFlash – LIN 1” is assigned to the desired real hardware channel or the virtual channel.

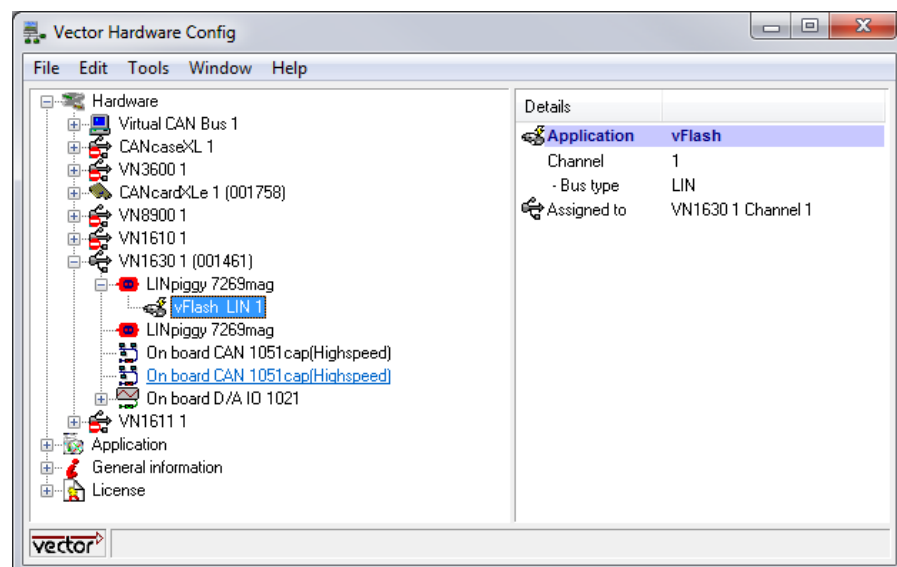


Figure 3-5: Vector Hardware Config – Configuration result

### 3.3.4 Configuration of Ethernet connection

#### Communication channels

Using Ethernet (DoIP) vFlash communicates via the standard Ethernet network adapter of the PC.

**How to configure the communication channels:**

1. Just connect your ECU – capable for Diagnostics over IP (DoIP) – to your PC. vFlash will automatically detect the ECU.
2. Note, since your ECU will most likely not contain a DHCP server you might need to re-configure your network adapter to use a fixed IP address.

## 4 Execute Reprogramming

### General Tool Structure

vFlash is basically organized into two separate sections – the “Flash” section and the “Configure” section.

To flash an ECU you only need the “Flash” page. To create a new flash project the “Configure” pages are used. You enter all ECU specific project settings on these pages. The configuration pages significantly differ between native reprogramming and ODX-F based reprogramming.

### Execute Reprogramming

To start reprogramming an ECU, you simply need to load an existing flash project. This can either be a project file (.vflash) or a Pack&Go file (.vflashpack).

If the ECU and vFlash are connected to the CAN-Bus, FlexRay bus or Ethernet you can start reprogramming immediately.

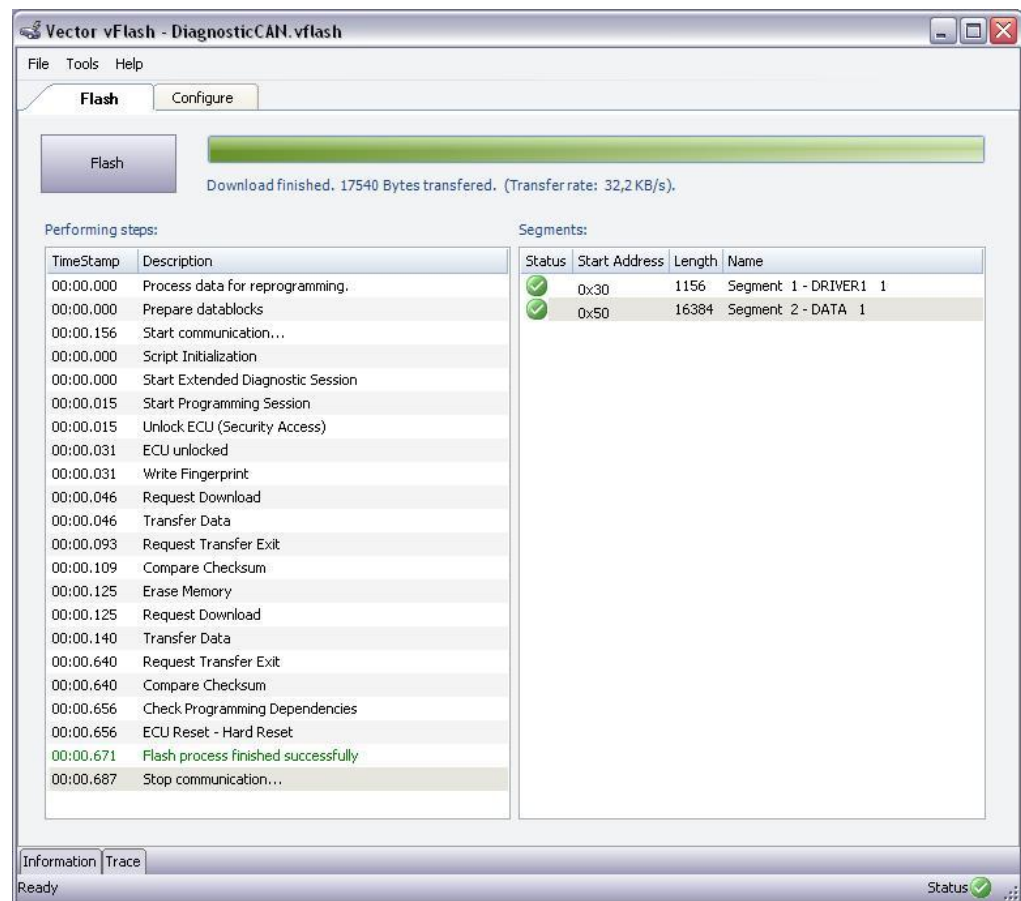


Figure 4-1: Flash Page

### Progress Information

vFlash provides the means to trace the progress of a currently running flash operation.

A progress bar with statistic information, such as transferred bytes, remaining time and average transfer rate, allows you to monitor the flash operation.

The “Progress Trace” (see left side of Figure 4-1: Flash Page) displays the current task and the already processed tasks – including timestamps.

And the “Segment Trace” (see right side of Figure 4-1: Flash Page) displays the segments scheduled for reprogramming – including size and status.

## Trace

Using the Trace window, located on the Flash Page, you can monitor the diagnostic bus traffic. To analyze the messages in detail, just double-click on a diagnostic transaction and a Details window will appear.

Request Time	Request T...	Request Data	Response Time	Response ...	Response Data	Communicati...
549482.67505s	VectorUds...	10 03	000000.001...	VectorUds...	50 03 00 64 01 F4	
549482.72579s	VectorUds...	10 02	000000.003...	VectorUds...	50 02 00 64 01 F4	
549482.73759s	VectorUds...	27 11	000000.001...	VectorUds...	67 11 0E 1E 91 3B	
549482.85072s	VectorUds...	27 12 F1 E1 6E C4	000000.002...	VectorUds...	67 12	
549482.86266s	VectorUds...	2E F1 5A DA 06 07 76 46 6C 61 73 68	000000.003...	VectorUds...	6E F1 5A	
549482.88407s	VectorUds...	34 00 41 30 00 00 04 84	000000.003...	VectorUds...	74 20 00 FE	
549482.89612s	VectorUds...	36 01 0D 00 02 12 02 CE 02 C5 03 4...	000000.003...	VectorUds...	76 01	
549482.90391s	VectorUds...	36 02 54 ED E8 20 68 E8 30 ED 88 E...	000000.003...	VectorUds...	76 02	
549482.91176s	VectorUds...	36 03 26 92 EC 82 26 8E 20 CB 3B 1B...	000000.003...	VectorUds...	76 03	
549482.91765s	VectorUds...	36 04 E8 20 E6 EA 01 00 E1 80 27 09...	000000.003...	VectorUds...	76 04	
549482.92327s	VectorUds...	36 05 80 E2 E8 29 A2 E8 28 6C 80 20...	000000.003...	VectorUds...	76 05	
549482.92890s	VectorUds...	37	000000.001...	VectorUds...	77	
549482.93485s	VectorUds...	31 01 02 02 90 54 87 0F	000000.003...	VectorUds...	71 01 02 02 00	
549482.94641s	VectorUds...	31 01 FF 00 01 50	000000.002...	VectorUds...	71 01 FF 00 00	

Figure 4-2: Communication Trace

## Information Window

The Information window, also located on the Flash Page, displays tool status information - informational messages but also detailed warning and error information. You may hide specific types of messages.

Timestamp	Category	Ecu	Description
15:42:51	Flash Project	---	Pack&Go Project loaded.

Figure 4-3: Information window

## 5 Create Project

### Template Concept

Flash projects are created based on Flash Templates. A Flash Template defines the flash procedure, the relevant diagnostic services and some pre-settings for a specific set of ECUs (e.g. the ECUs of one platform).

In general, the OEMs use significantly different flash specifications. vFlash hides these differences by its template concept. Each Flash Template represents an OEM specific flash specification.

Vector already has implemented several flash specifications for different OEMs. To get a Flash Template for a specific OEM, please contact Vector. In addition you can use the Vector UDS Reference Project to create your own Flash Template.

A plug-in mechanism is used to extend vFlash with new Flash Templates. You just need to execute a small installer to deploy a new template on your PC.



**Cross reference:** details see paragraph 12 Vector UDS Reference

### Create new Project

To create a new flash project for a new ECU you simply need to instantiate a Flash Template. Since the template already defines most settings and the flash procedure, you only need to configure a small set of ECU specific parameters.

All templates installed on your computer are offered for instantiation.

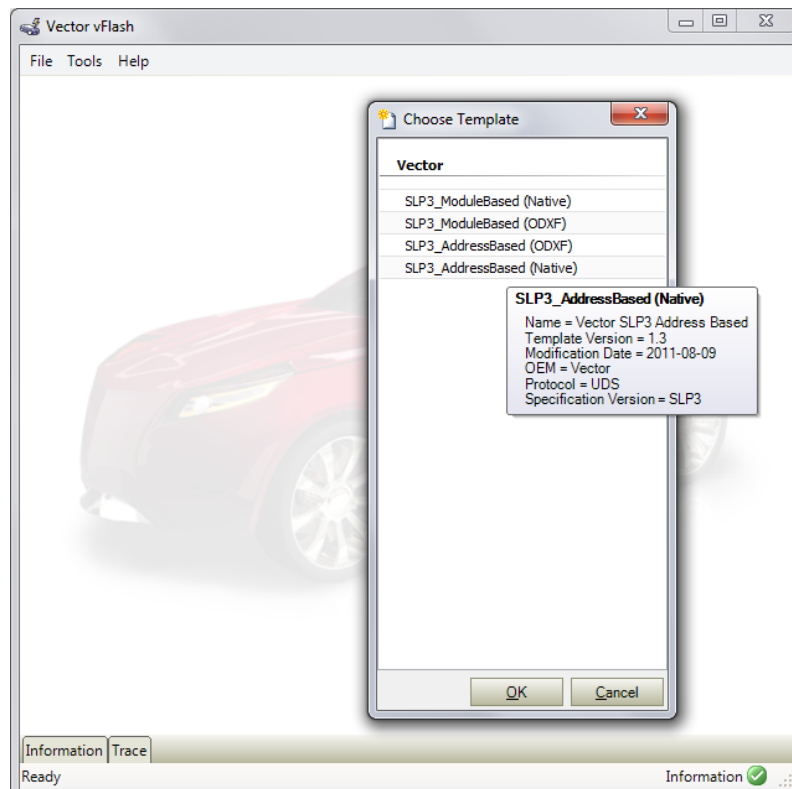


Figure 5-1: Instantiate Template



1. Choose **File | New**
2. Select Flash Template that is suitable for your ECU

## 6 Configure Project

### 6.1 Flashing based on different flash data containers

Flash use cases	<p>The flash data can be described natively (bin files, IntelHex files, Motorola-S files) or by specific container formats, such as ODX-F or Ford vbf. ODX-F also contains or references native files but additionally describes process information.</p> <p>vFlash is capable to flash ECUs based on native files as well as based on the flash containers ODX-F and Ford vbf. It provides different views for the different use cases.</p>
Native flashing	<p>To flash native data you have to choose the flashware files and you need to configure ECU specific data such as baudrate, address information, checksum files, and so on. Native flashing is ideally suitable for development use cases where frequently flash binaries are created. vFlash immediately uses the new flashware files and also the new checksum and signature files without any modification of the project file. With a Pack&amp;Go file you have the chance to easily transport the complete flash package to your partner – all required files are automatically packed together and can be executed on your partners PC.</p>
ODX based flashing	<p>To flash an ECU based on ODX-F you simply need to load the respective ODX files or the respective PDX file. After identifying which data can be flashed in the ECU (reading ExpectedIdents) you may select either one or several valid Sessions or a complete FlashClass for downloading.</p>
OEM specific flash containers	<p>Some OEMs have proprietary flash container format. The containers supported by vFlash and the configuration procedure is described in chapter 6.6.</p>

### 6.2 General

Project Configuration in general	<p>After creating a new project ECU specific configurations are necessary. The Configuration page offers three different tabs to guide you through the configuration procedure – “Data”, “Communication” and “Misc”.</p> <p>Note, depending on the Flash Template your new project is based on, the appearance of the tool might differ to the screenshots displayed in the subsequent sections.</p>
----------------------------------	--



## 6.3 Data Configuration

### 6.3.1 Flashing based on Native Data

#### Data Configuration for native flashing

With the <Configure | Data> tab you can select the flash data to be transferred to your ECU and the flash driver(s). Also the configurations related to the data packages are done on this tab.

#### Flash drivers:

Flash drivers have to be downloaded to an ECU if they do not permanently reside in the ECU (usually they don't). Note, a driver is a specific type of a datablock.

**Flash Driver 1** is the “erase driver”. If **Flash Driver2** is not used, it also contains the “write driver”.

**Flash Driver 2** is the “write driver”.

#### Datablock settings:

For each datablock specific settings are required. These settings can be modified within the Details section of each datablock. The number of parameters to modify depends on the type of flash data format (e.g. entering of address information is required for binary data but not for Intel Hex or Motorola-S data).

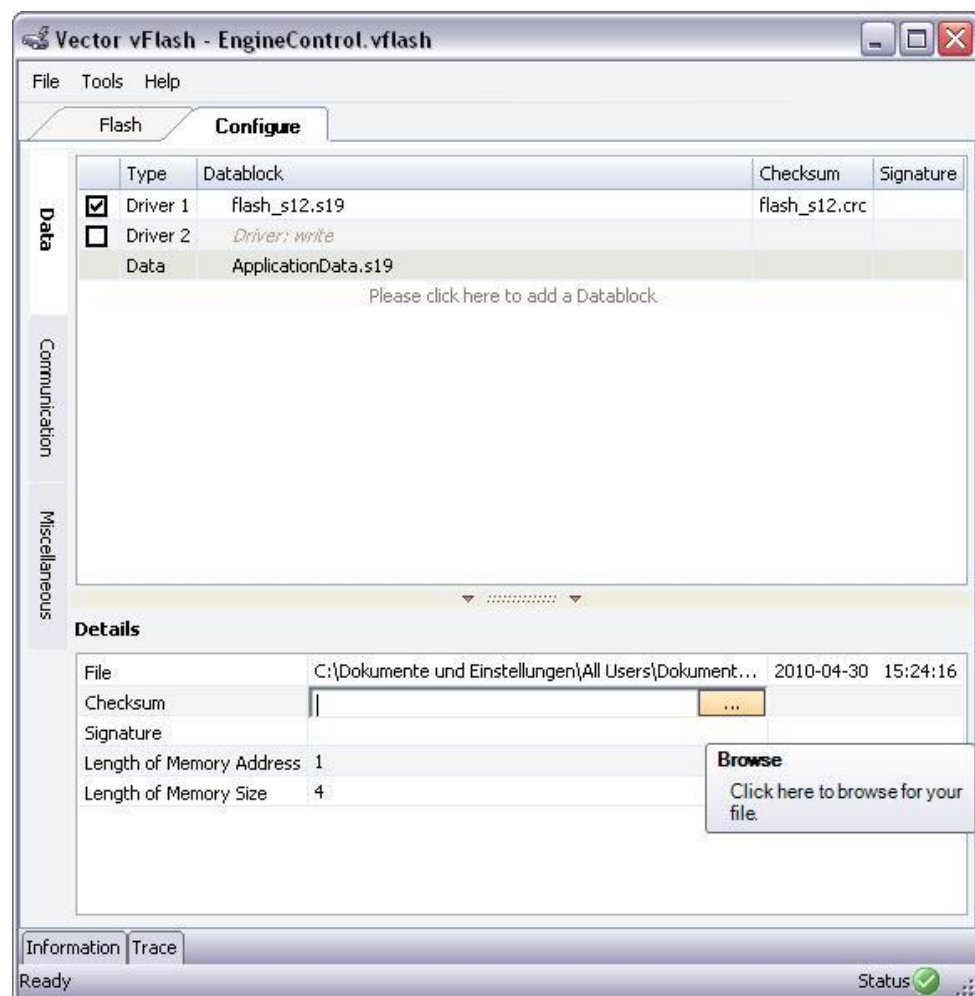


Figure 6-1: Native Data Configuration



1. Select Driver 1 if needed and follow the instructions to load the flash data file.
2. Go to the Details section ...
  - 2.1. Select the file that contains the Checksum
  - 2.2. Optionally select the file that contains the Signature
  - 2.3. Optionally modify the “Length of Memory Address”
  - 2.4. Optionally modify the “Length of Memory Size”
3. Select Driver 2 if needed and follow the instructions to load the flash data file.
4. Go to the Details section and repeat the steps described – see 2.
5. Select the data files that actually shall be flashed.
6. Go to the Details section and repeat the steps described – see 2.



---

**Info:** vFlash automatically detects the modification of the flash data files in the background and uses the latest ones. Thereby it is very easy to recompile the ECU software and immediately write the new software version to the ECU – without reloading the vFlash project and without any copy operation.

---

### 6.3.2 Flashing based on ODX-F Data

#### Data Configuration for ODX-F based flashing

After instantiating the ODX-F use case of your OEM specific vFlash template, you simply need to select the PDX container. It contains the ODX-F file and the flashware but also the required ComParams and Diagnostic Services. The ComParams are automatically extracted from the ODX data but you may also adapt the read ComParams.

Note: Instead of loading a PDX container you may also select the ODX files manually.

vFlash's <Configure | ODX> tab displays the ODX files and the flashware files contained in the PDX container.

If several ECU-MEMs (= several configurations) are contained in the ODX-F files you may select the ECU-MEM suitable for your ECU.

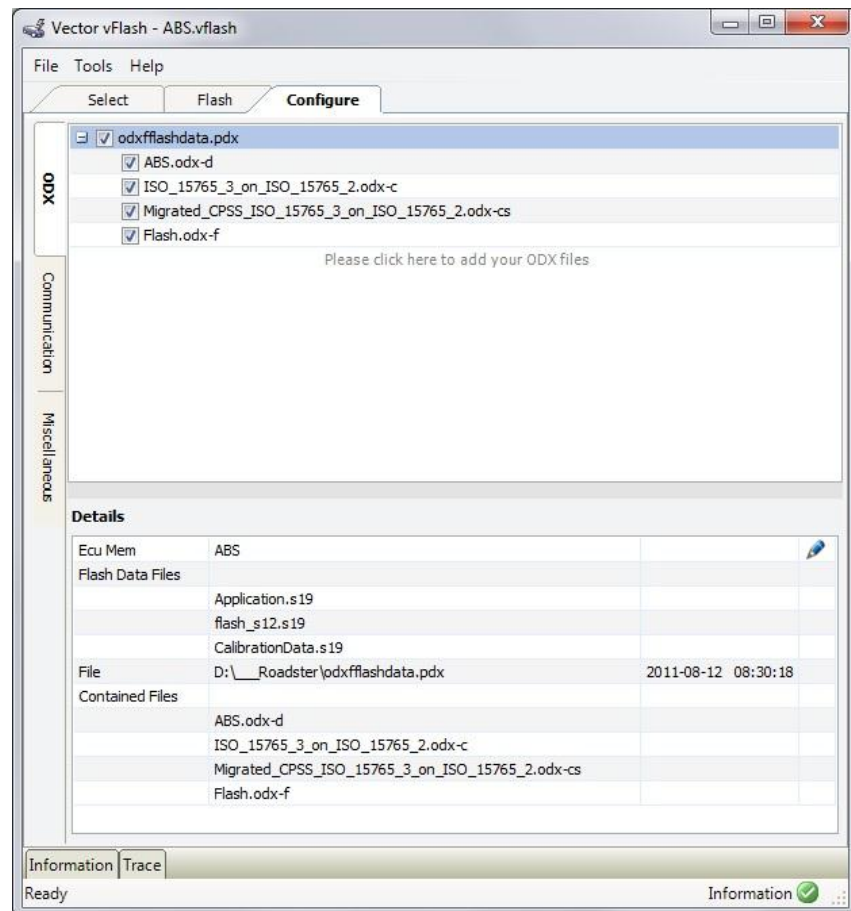


Figure 6-2: ODX Data Configuration

### Sessions and FlashClasses

ODX-F basically provides two approaches for reprogramming an ECU. You may either select one or several ODX Sessions for flashing or one FlashClass that references several Sessions.

By default only the identified Sessions are flashed to the ECU – the identified Session marked with a matching check sign. However you may force to write also the not identified Sessions to the ECU. Just check the respective checkbox on the lower left corner of the <Select> tab.

You may start the identification manually by just clicking “Identify”. All specified identification services are executed and the results are displayed in the “Identification Data” table. Anyhow, before actually flashing the ECU the identification services are executed automatically to ensure that the ECU is reprogrammed only with the suitable data packages.



**Note:** To compare the read identification values and the expected values of a specific Session you may double-click on the exclamation mark or check symbol

Select by Session

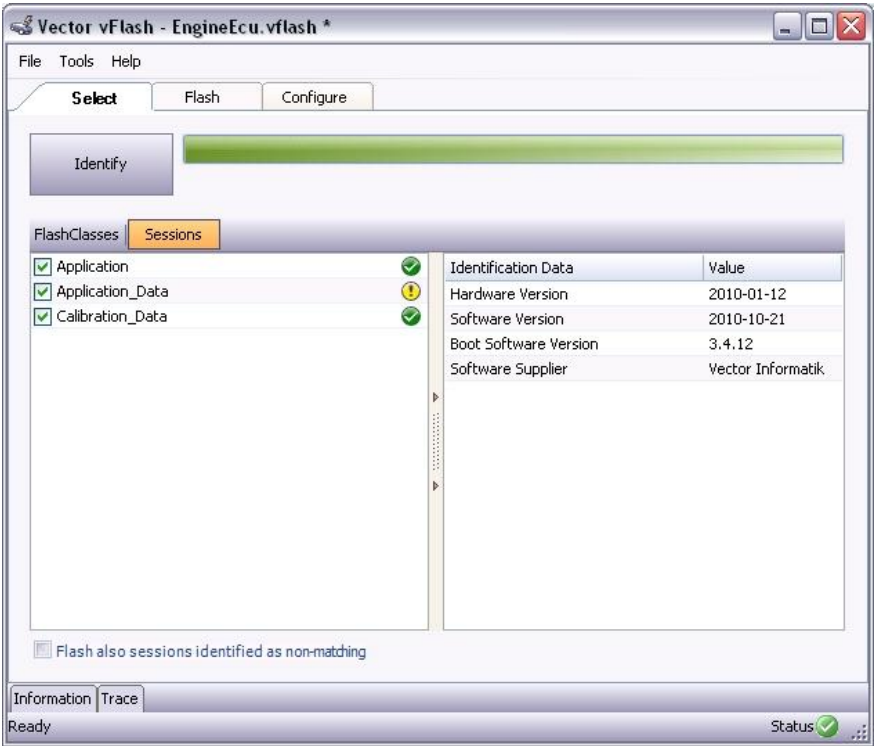


Figure 6-3: Select Flashware by Session

Select by FlashClass

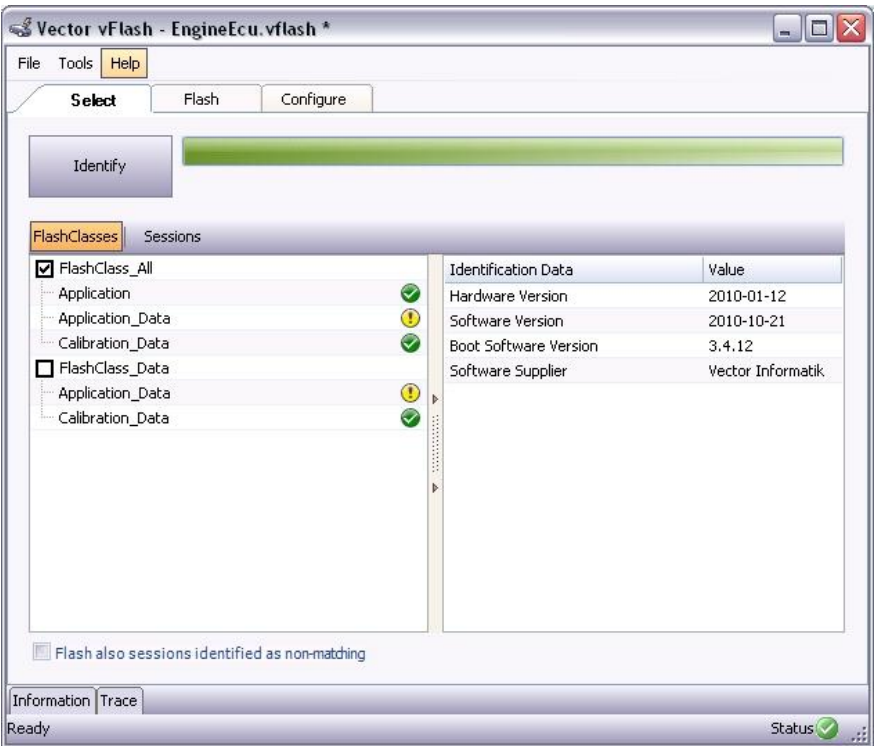


Figure 6-4: Select Flashware by FlashClass

## 6.4 Communication Configuration

### General Communication Configuration

With the <Configure – Communication> tab you can choose how to access the ECU (communication interface / logical link) and enter the address information required to communicate with the ECU.

Depending on the type of communication interface (CAN, FlexRay or Ethernet via DoIP) and addressing schemes (for CAN) different communication parameters are displayed. You just need to modify the provided parameters as suitable for the ECU.

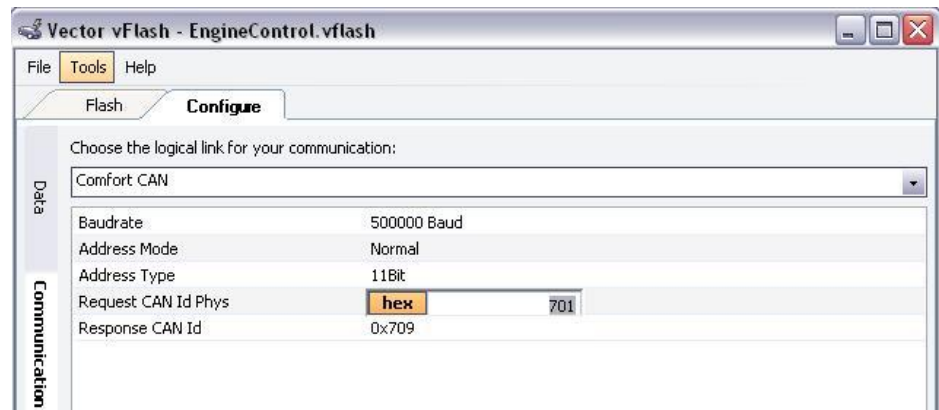


Figure 6-5: Configuration - Communication

### FlexRay specific settings

Since a FlexRay network needs to be started-up, it is not always enough to connect vFlash to the FlexRay network, but vFlash may have to initiate the start-up and may have to keep the network awake.

This is necessary if there is only one or even none “Coldstart Node” available in the network or if no node sends Network Management messages.

So if the connected network only has one “Coldstart Node”, activate “Coldstart Node 1”. If the connected network has none “Coldstart Node”, activate both “Coldstart Nodes” in vFlash. And if required also select the Network Management PDU to keep the Network awake.

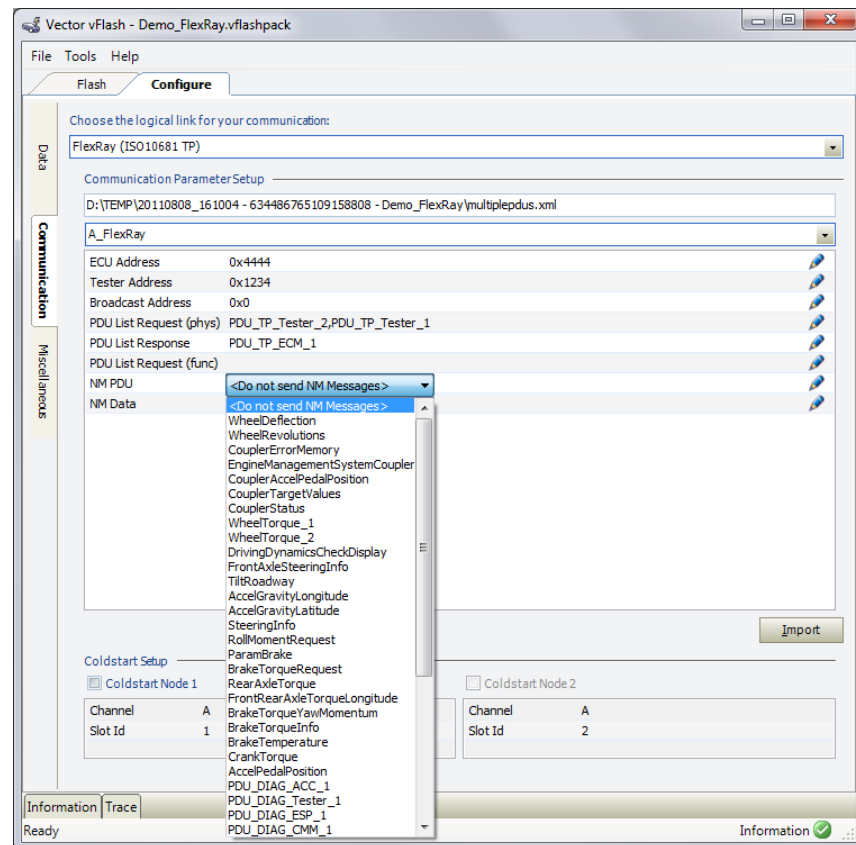


Figure 6-6: Configuration – FlexRay Communication (NM PDU)

### FlexRay - Network activation

If at least one Coldstart Node is activated, vFlash has to participate in starting-up the network. Therefore initializing communication is separated in two steps – the network activation and set-up of the communication channels.

You just need to activate the network with the “Activate” button (the “Flash” button is now called “Activate”), then switch on your ECU and afterwards start flashing with the “Flash” button (the “Flash” button is now called “Flash” again).

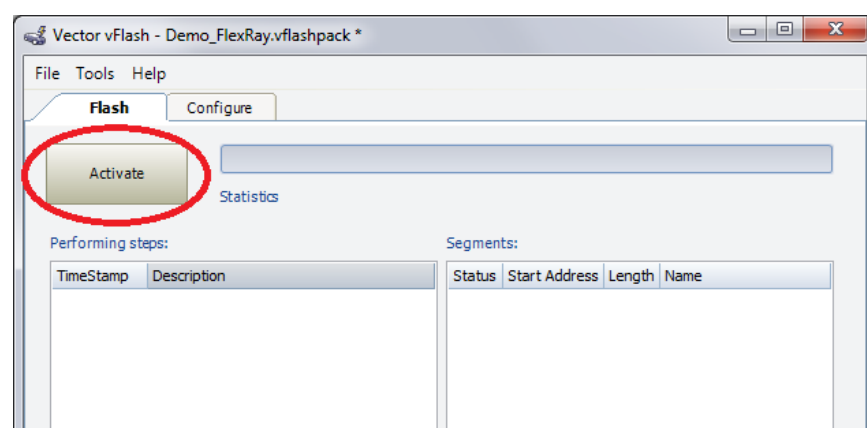


Figure 6-7: FlexRay Communication - Network activation



**Note:** If several PDUs are used for one direction, the user has to ensure that the PDUs are chosen in ascending order!

**LIN specific settings**

The selection of a LIN ECU is based on the NAD (Network Address). This NAD is extracted from the LDF (LIN Description File). Therefore a LDF has to be selected. Usually each vFlash Template that allows flashing via LIN comes with a default LDF file. It is automatically copied into the project folder, when the project is saved.

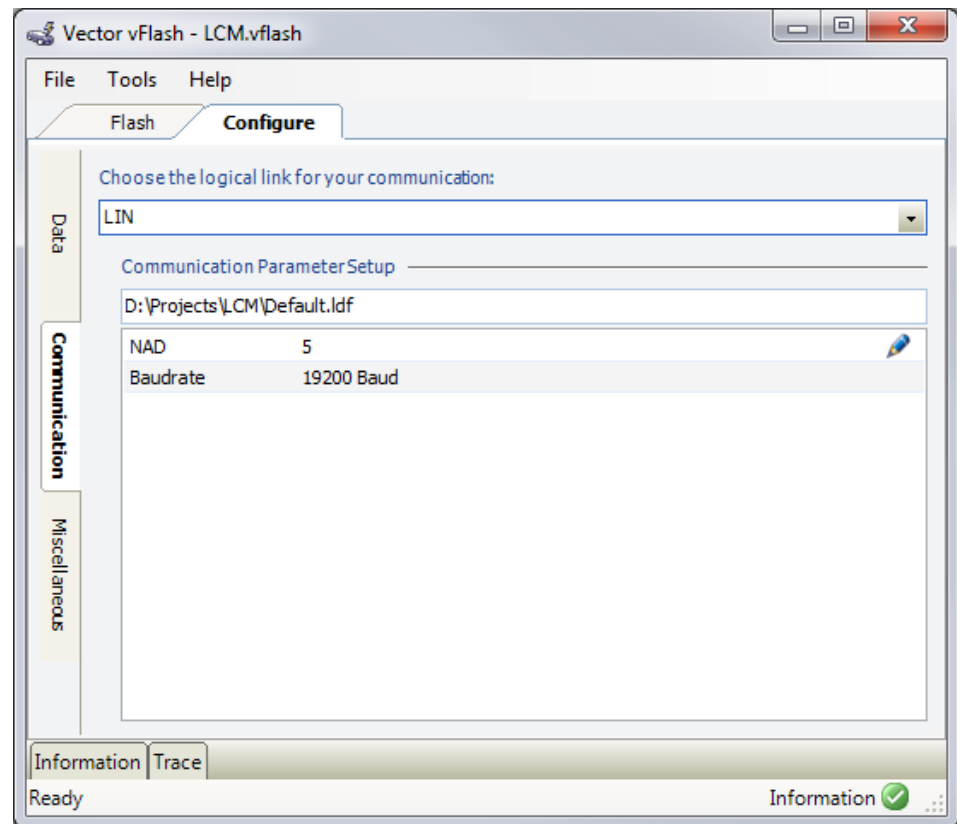


Figure 6-8: Configuration – LIN LDF selection

## 6.5 Miscellaneous Configuration

### Configure Miscellaneous Settings

With the <Configure – Miscellaneous> tab you can enter certification information. The certification information consists of the tester serial number and the Security Access settings – if applicable.

To unlock the ECU usually a Seed&Key DLL is required. Just select the your Seed&Key DLL. If you need to create your own Seed&Key DLL, vFlash comes with an example to get you started. Some vFlash Templates already come with a Seed&Key DLL – in this case the DLL is already selected when you instantiate your Template.

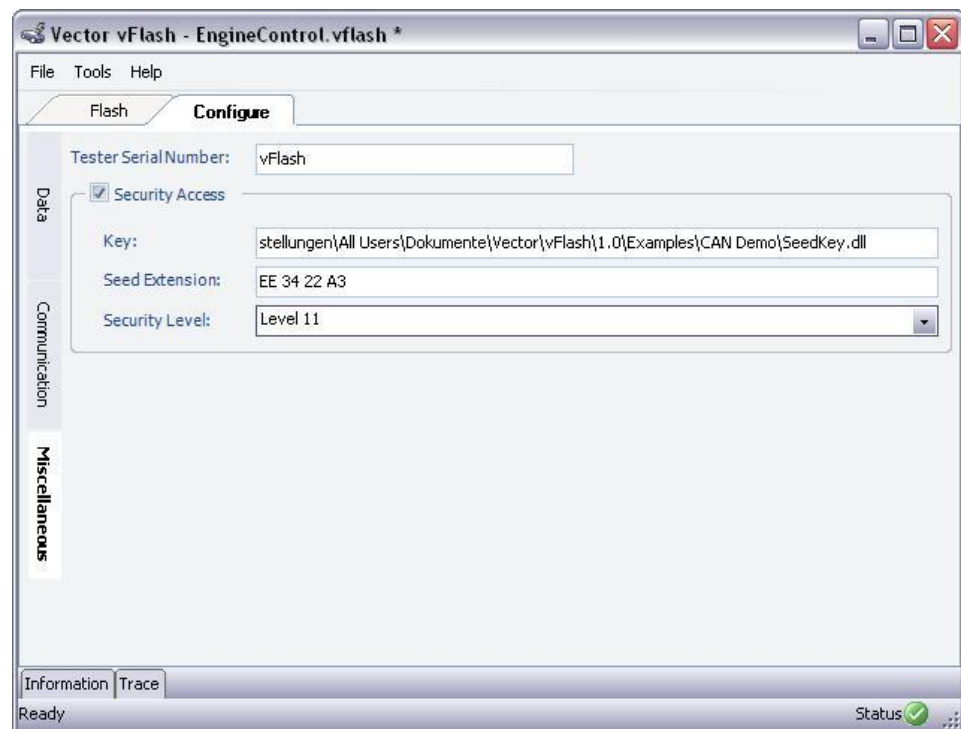


Figure 6-9: Configuration - Miscellaneous



**Info:** The Seed Extension is only available if activated in the Flash Template. Note, since the Seed Extension is a security relevant setting its value is stored encrypted.



### Flash Attributes

A vFlash Template may define additional parameters – Flash Attributes - that are used to configure the flash procedure. The user may enter values for these Attributes in the vFlash user interface.

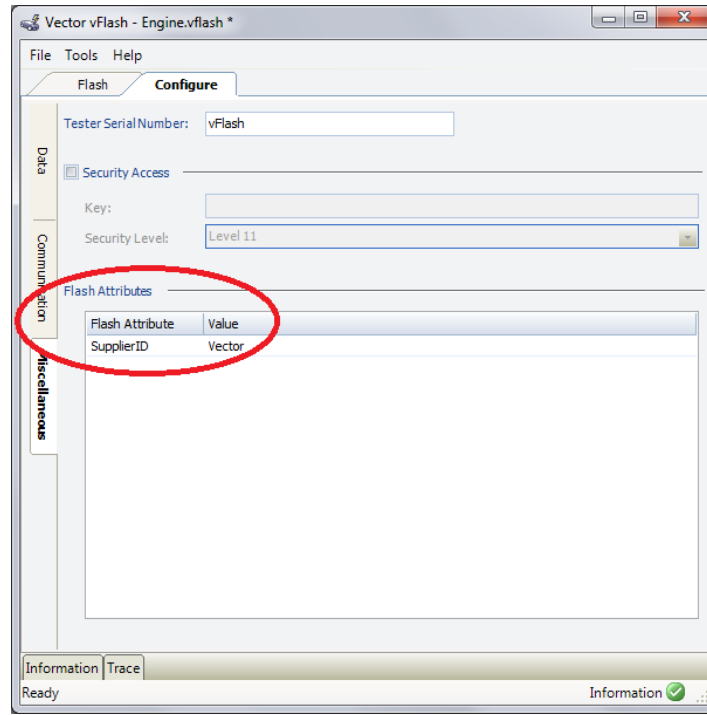


Figure 6-10: Flash Attributes



**Example:** In the example above the Supplier ID can be entered. It may be transferred to the ECU after a successful flash process.

## 6.6 OEM Specific Configuration

### 6.6.1 Flashing based on Ford flash containers

#### Data Configuration for Ford Vbf based flashing

With the <Configure | Data> tab you can select the Ford flash containers (vbf) containing the data to be transferred to your ECU and the flash driver(s). Also the configurations related to the data packages are done on this tab.

Similar to the native data configuration you need to select the driver files and the data files. However, the Ford vbf containers already contain all information relevant for flashing. So there is no need to enter any additional data. Even the ComParams are defined within the container – but can be overwritten on the <Data> tab.

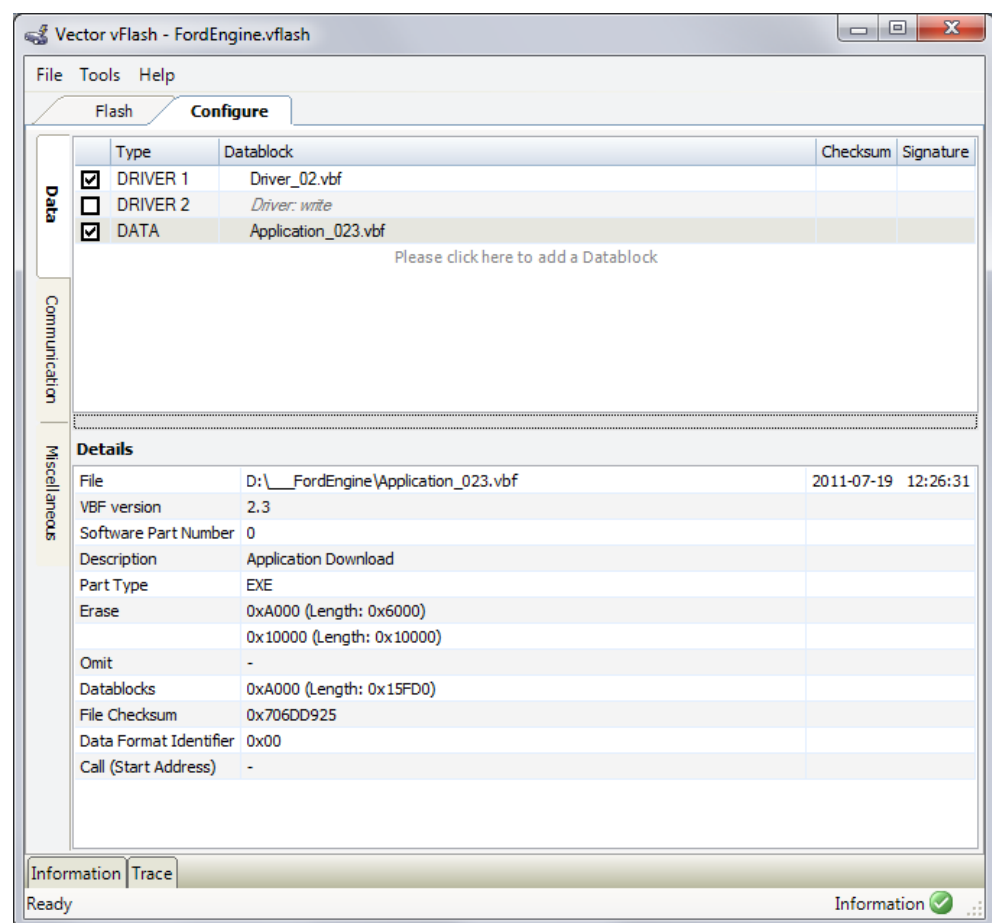


Figure 6-11: Configuration - Data

## 6.6.2 Flashing based on GM flash containers

### Data Configuration for GM GBF files

GM ECUs are flashed, based on the GM specific GBF container file.

On the <Configure | Data> tab you can select the GBF files representing the driver, the application and the data files. Since the checksum and datablock address information are contained in the GBF files you don't need to enter any data in the datablock's "Details" section.

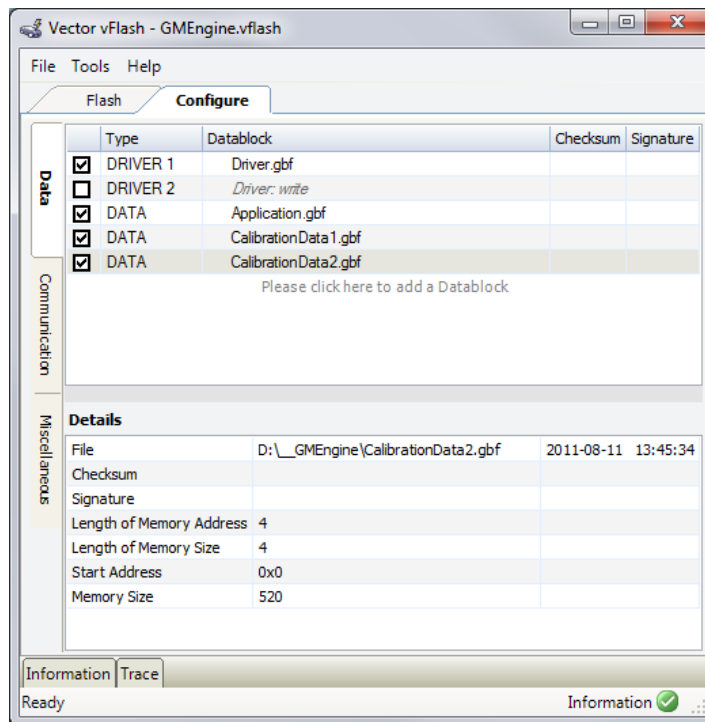


Figure 6-12: Configuration – Data

### Communication Configuration: Addressing

The GM flash process basically distinguishes two different ways on how to address an ECU – either with the Node Address or with the standard request and response CAN-IDs.

vFlash automatically detects which addressing has to be used. However, to be able to reprogram your ECU independently of the addressing mechanism, you need to enter all address information – “Physical Request CAN ID”, “Response CAN ID”, “Functional Request CAN ID”, “Broadcast Address” and “Node Address”.

### Communication Configuration: Bus Speed

Some GM ECUs support reprogramming with an increased speed. In this case the reprogramming procedure is started with low speed and the bus speed is increased after enabling the programming mode.

If a baudrate switch is required you need to check the respective checkbox on the <Configure | Communication> tab and enter the desired baudrate.

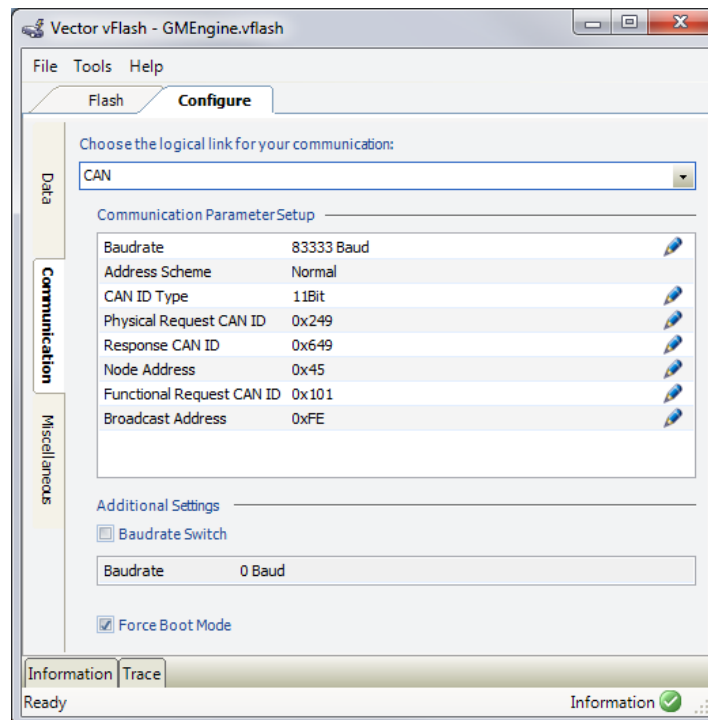


Figure 6-13: Configuration - Communication

### Miscellaneous Configuration

GM ECU do not respond on Request Download request with the maximal number of bytes to be transferred with the Request Download request. Therefore the user has to enter the "Diagnostic Buffer Size". vFlash provides a default size, however you might need to modify this default value.

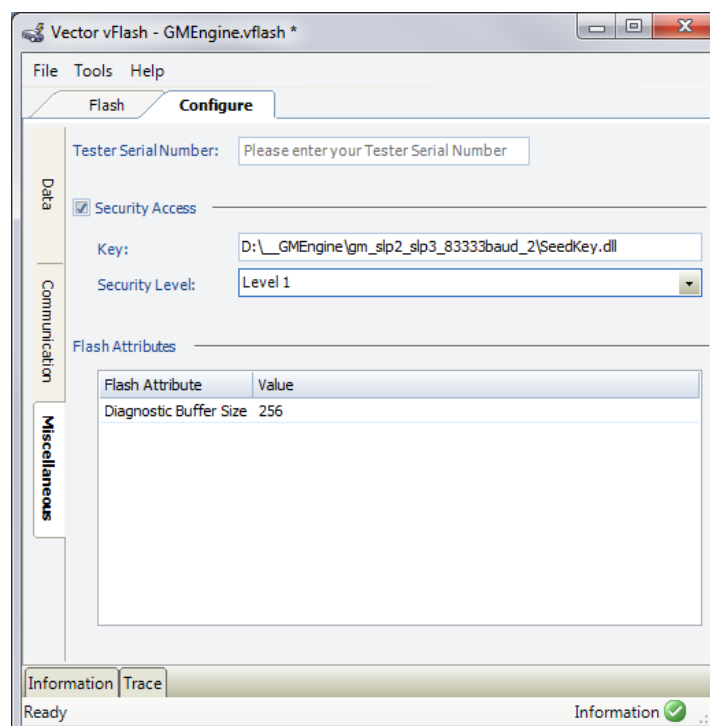


Figure 6-14: Configuration - Miscellaneous

### 6.6.3 Flashing based on Fiat flash containers

#### Data Configuration for Fiat idx, prm and bin files

Fiat ECUs are flashed based on a Fiat specific container format consisting of the three file types <.idx>, <.prm> and <.bin>. With selecting the index file <.idx> also the other files are fixed.

On the <Configure | Data> tab you can select the index file. The file references the .bin file. It also contains additional information displayed in the Details section. For Fiat it is not required to manually select a driver that is downloaded before flashing the application/data to the ECU. Therefore you may directly select a datablock. Also manually selecting a checksum or a signature file is not required, since this information is already contained in the <.prm> file.

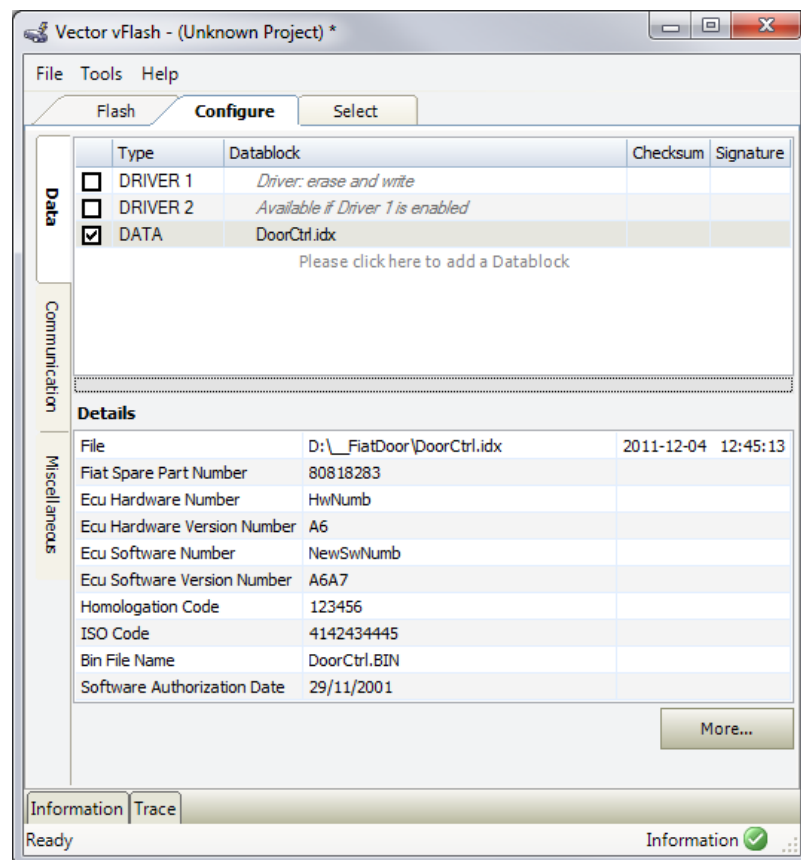


Figure 6-15: Configuration - Data

### Additional information

The <.prm> file contains a large number of data relevant for flashing and controlling the flash sequence. These information can be displayed using the <More> button on the <Configure | Data> tab.

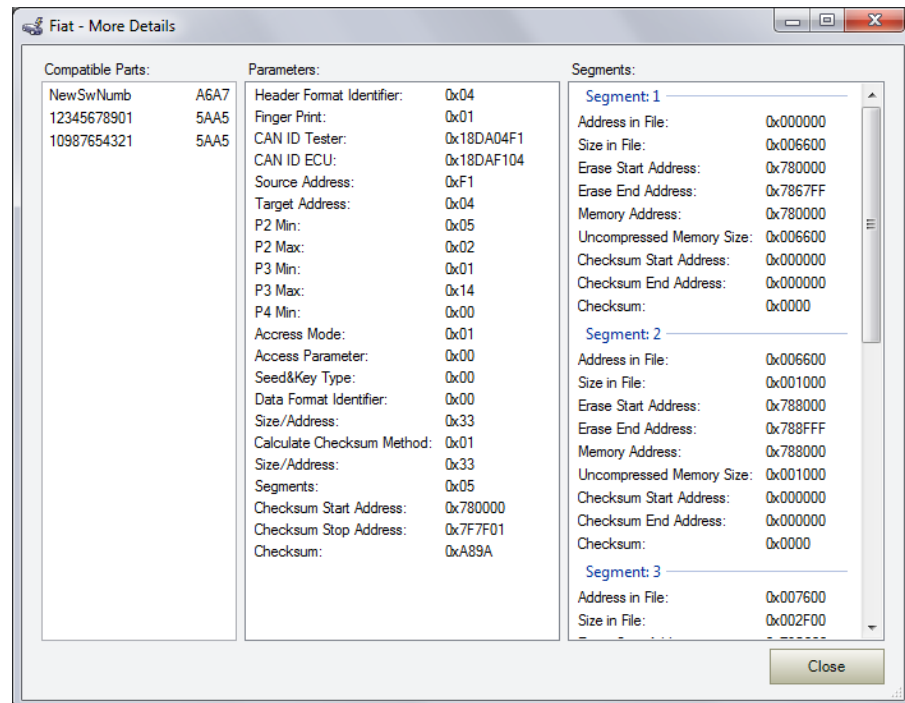


Figure 6-16: Additional information in .prm file

**Compatibility Check**

The Fiat flash procedure provides a compatibility check mechanism – identification information are read from the ECU and are compared with expected data stored in the .idx file.

On the <Select> tab you may manually start the identification. The read values and the expected values are also displayed on this tab.

If the identification data do not match with the expected values, the ECU cannot be flashed with these container files. However, you may force vFlash to ignore the identification result and flash the ECU even if the data are incompatible – just check the checkbox on the <Select> tab.

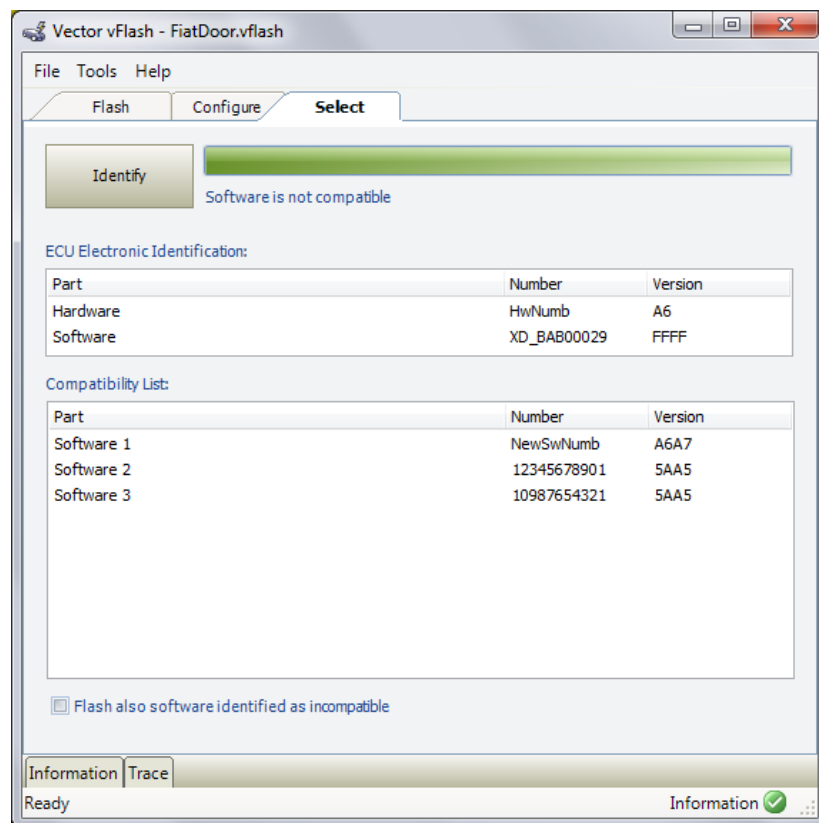


Figure 6-17: Compatibility check



**Note:** If you do not manually execute the identification it is executed automatically when you start flashing your ECU.

## 7 Save and Exchange Flash Project

### Exchange Flash Projects

A vFlash configuration is stored in a project file. Two different project file types can be distinguished – the standard vFlash project and the Pack&Go project.

The standard vFlash project contains configuration information but only references resources such as the flashware, the Seed&Key-dll, and so on. This allows you to simply exchange resources without modifying the project file itself – vFlash automatically uses the new files if available.

The vFlash Pack&Go project file (.vflashpack) has the configuration information and all resources packed in one file. To create such a package, just export your project to a Pack&Go file via the <File> menu.



---

**Info:** If the focus is on flexibility in changing the flash data, the referenced data files can simply be exchanged in background, e.g. as part of an automated software build process. In this case, the flash project remains unchanged.

If the focus is on a process-safe exchanging of flash configurations, the flash projects may be exchanged in the Pack&Go format. In this case, all relevant settings together with the data files are packed in an archive file.

---



## 8 Advanced Features

### 8.1 Compression

#### Compression

The diagnostic protocols define a feature for transferring flash data to the ECU in a compressed format. Thereby the bandwidth can be increased implicitly and the number of bytes transferred from the flash tool to the ECU can be increased significantly.

The data can either be compressed before handed over to vFlash (pre-compression) or vFlash automatically does the compression before transferring the data to the ECU (live-compression).

To be able to use compression, “Pre-Compression” and/or “Live Compression” the features have to be enabled in the template. This is required since the flash tool can only use compression if the bootloader is capable to receive compressed data and decompress this data.



---

**Note:** Compression is only activated if you change the value of the “Compression Method” item in the “Configuration – Data” page’s “Details” table. To enable compression the value has to be none-zero.

---

#### Pre-Compression

Pre-Compression means that the binary or hex data are already compressed when handed over to vFlash.

If “Pre-Compression” is activated in the vFlash Template the “Pre-Compression” feature can be selected on the page “Configuration – Data”.

For programming the ECU vFlash needs to know the uncompressed memory size and the compressed memory size. The compressed memory size is automatically extracted from the flash data. However, the uncompressed memory size needs to be entered: You may either enter the uncompressed memory size manually or you may reference the respective uncompressed file.

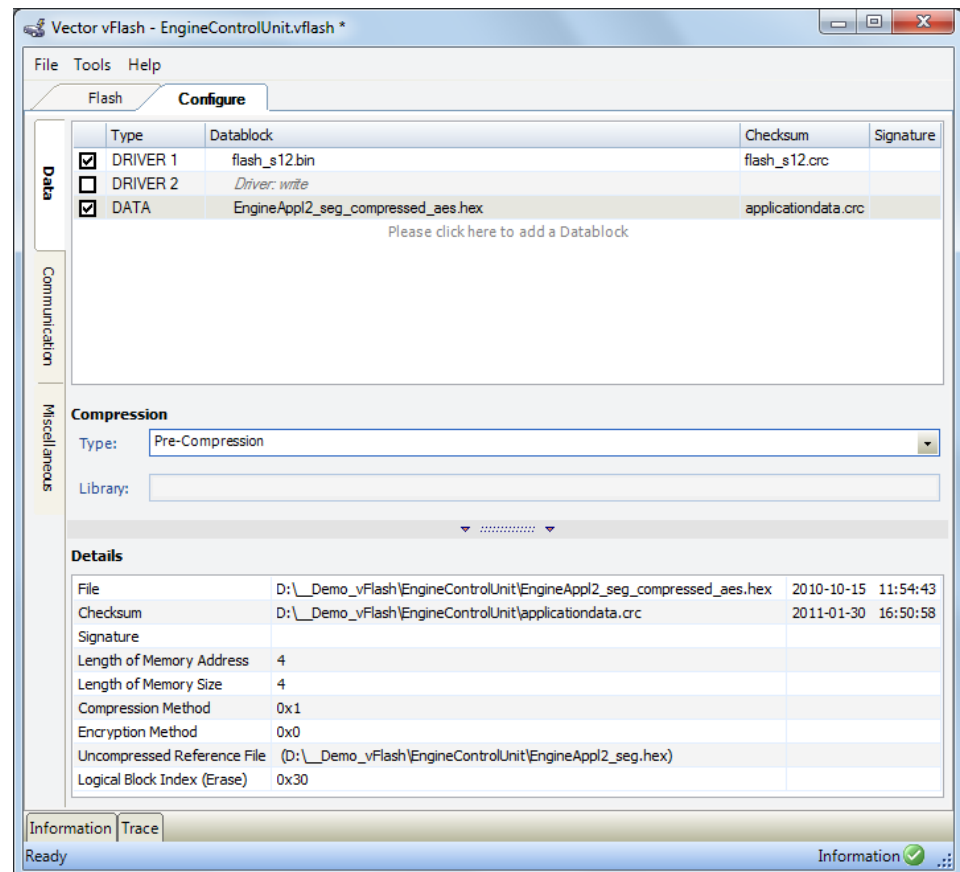


Figure 8-1: Pre-Compression Configuration

### Live Compression

Live Compression means that the binary or hex data are compressed by vFlash, just before they are transferred to the ECU.

To activate Live Compression you need to select the Compression Type "Live Compression" and you need to enter a compression method that is none-zero.

For Live Compression a compression library is required. You need to select a compression library that implements the identical compression algorithm as used in the bootloader. Additionally the API has to fit to the definition of the respective vFlash Template.

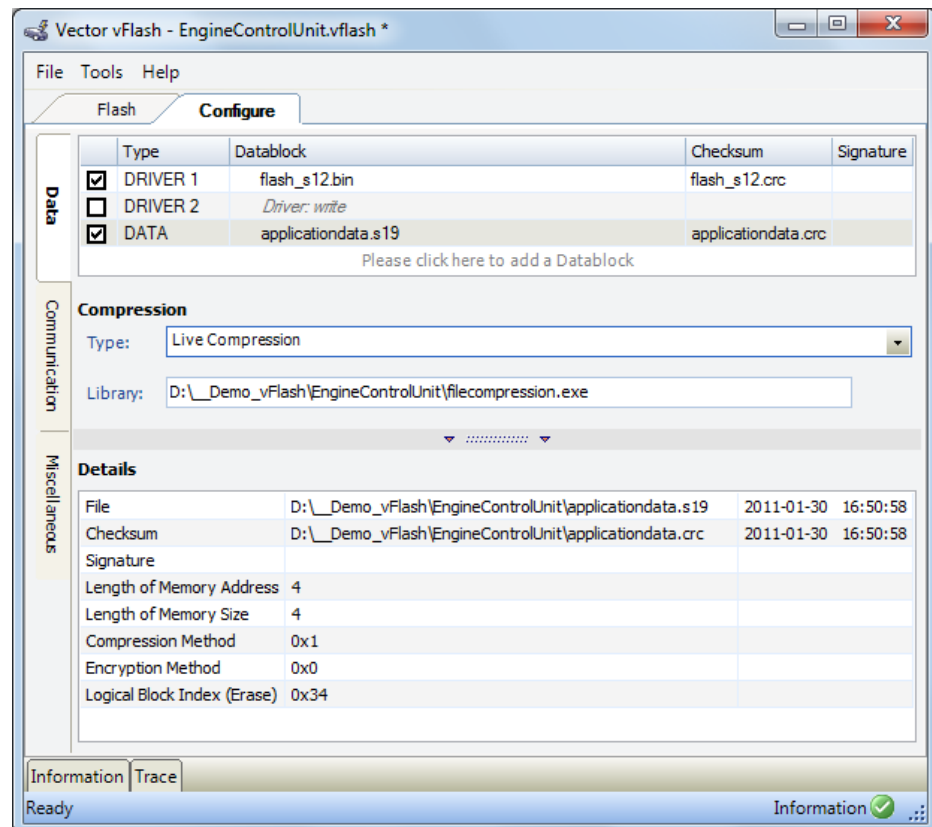


Figure 8-2: Live Compression Configuration

## 8.2 Multi-Driver Support

**Multi-Driver Support** Several Bootloaders are capable to use different drivers for different Datablocks. With vFlash you can now define which Bootloader is used for which Datablock(s).



1. Choose the Datablock to be downloaded
2. Select the type of the Datablock (Driver 1, Driver 2 or Data).

Note: You may simply reorder the Datablocks in the Selection table by moving them up and down. However the position of the first two drivers is fixed – though do not need to be filled with data.

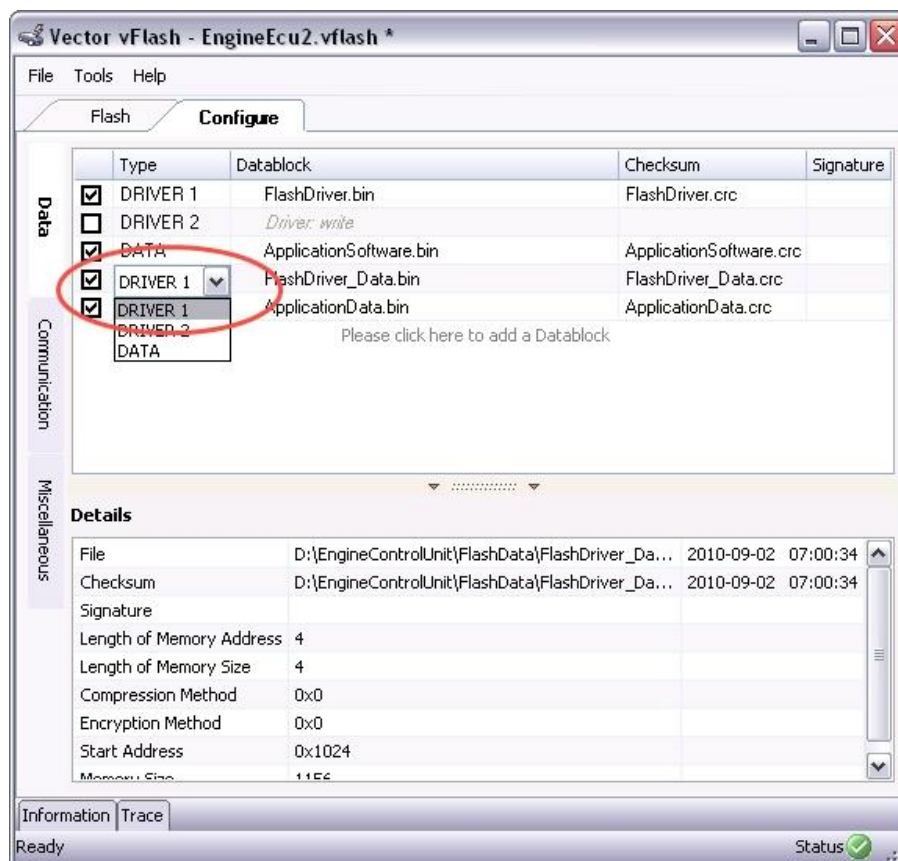


Figure 8-3: Use several Drivers

## 8.3 Automation Interface for Batch Processing

**Automation Interface** vFlash can not only be controlled via its graphical user interface but also with an **Automation Interface**. Using the **Automation Interface** a client application can load an existing flash project, start and stop the flash process and get progress and status information.

The Automation Interface is implemented as a C Library.

**API** The C header file, that describes the API of the Automation Interface, is located in the installation folder: <installation directory>\vFlashAutomation\VFlashAutomation.h.

**Usage** The usage of the automation interface is quite simple. Only a set of 7 functions is required to execute reprogramming remotely. Just follow the steps below and consult the example (see also below).



1. Create a new Visual Studio C++ project.
2. Include the header files “VFlashAutomation.h” and “VFlashAutomationTypes.h”.
3. Implement your application based on the VFlashAutomation API. The usage of the library is quite simple – you just have to stick to the following command order:

```
> vFlashInitialize ()
> vFlashLoadProject()
> vFlashStart()
> vFlashUnloadProject()
> vFlashDeinitialize()
```

**vFlashStart()** can be called several times within this sequence. Note, this function executes asynchronously – so you have to wait until the function completes. Using **vFlashStop()** you can interrupt the flash procedure. If required, **vFlashGetLastError()** reports detailed error information.

Your application basically has to call **vFlashInitialize()** to initialize the VFlashAutomation library. Afterwards it can start reprogramming the ECU via **vFlashStart()**.

Since the **vFlashStart()** function is called asynchronously, callback functions are used to signal the progress and the status of the running flash operation.

4. Copy the “vFlashAutomation.dll” from the vFlash’s “bin” folder into your application’s binaries folder.




---

**Info:** To use the Automation Interface a previously created vFlash project is required.

---




---

**Example:** An example application including source code is available in the installation folder <installation folder>\Examples\VFlashAutomationExample. To adapt the example we recommend to copy the VFlashAutomationExample folder to a local folder. Afterwards you may have to adapt the paths in the project settings and the source code.

**Source Code:**

- > Project load path: Path to your vFlash project (e.g. “D:/MyProject/Engine.vFlash”)
- > Include path: Path to the vFlash Automation headers “VFlashAutomation.h” and “VFlashAutomationTypes.h” (e.g. in “C:\Program Files (x86)\Vector vFlash 2.0\VFlashAutomation\”)

## 9 Additional Information

### 9.1 Seed Extension

#### SeedExtension

The **SeedExtension** is an optional parameter that is used as a kind of password for generating the Security Access key. It is only available if the feature is activated in the vFlash Template (if required by the OEM).

The SeedExtension value is readable during entering the value but is hidden as soon as you confirm the new value. The entered value is stored encrypted in the project file and is available again after loading the project.

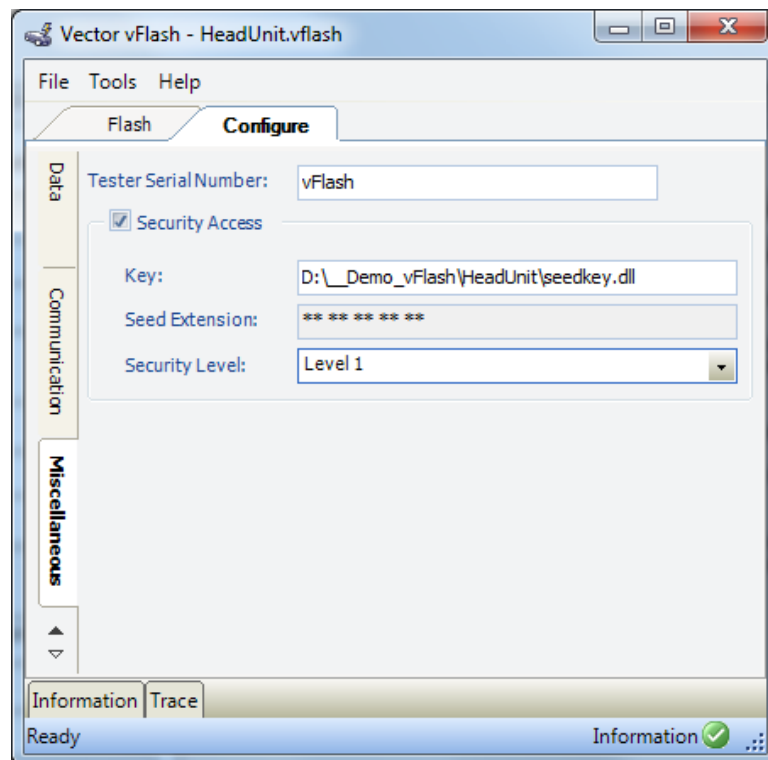


Figure 9-1: Seed Extension

### 9.2 Force Boot Mode

#### Force Boot Mode

The **“Force Boot Mode”** feature is an optional function that is only available if activated in the vFlash Template (available if supported by the OEM / Bootloader).

If an invalid application is loaded into an ECU it is usually not possible to return the ECU into the bootloader, since the Diagnostic Session requests are not processed anymore. With the “Force Boot Mode” feature the Bootloader can be detained from loading the invalid application at all and keep the ECU in Boot Mode.

If the function is available and is activated via the respective checkbox on the <Configure | Miscellaneous> tab, vFlash periodically sends “ping” messages to the ECU to force the ECU to stay in boot. If the ECU is booted while the ping messages are sent, the bootloader detects these messages and does not load the application.

As soon as the ECU has successfully received the message, it responds positively to inform vFlash that the ECU is ready for programming.

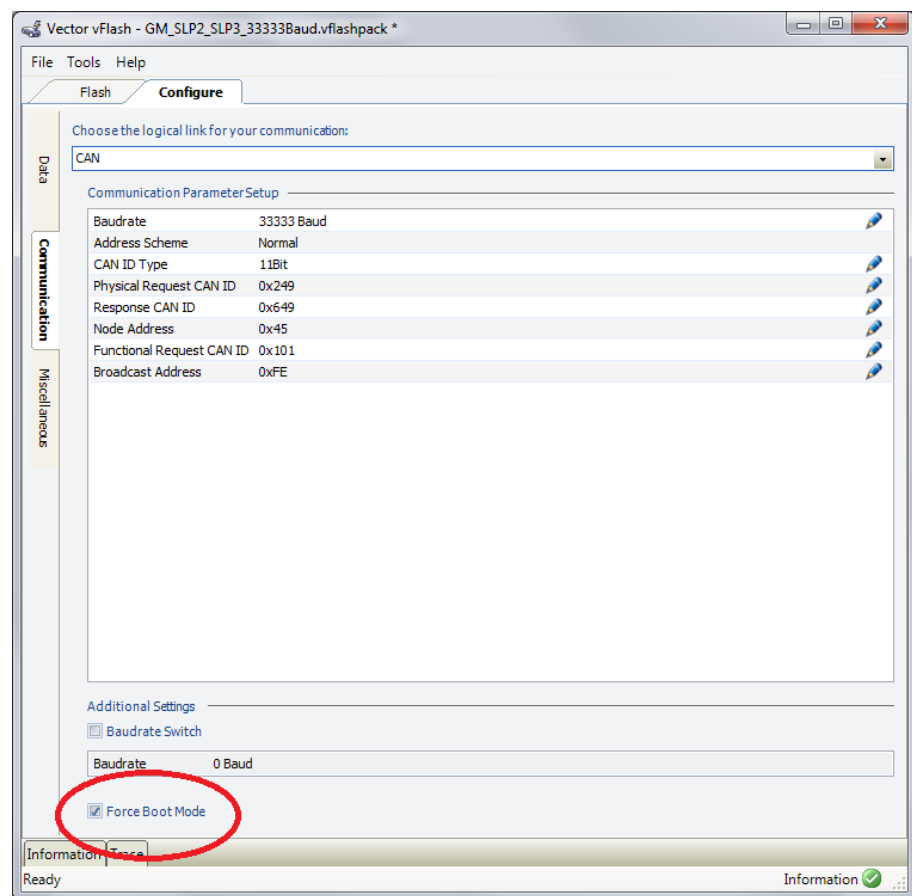


Figure 9-2: Force Boot Mode

## 9.3 vFlash Template

### vFlash Template

A vFlash Template serves as base for every flash project. It describes the OEM bootloader specific behavior and therefore the flash procedure. It basically contains the bootloader specific settings, the services required for flashing and the flash procedure itself.

### vFlash Template Upgrade

As described a new vFlash project is created based on a vFlash Template. New templates can be simply installed via template installers (already available for many OEMs and bootloaders).

**Note:** If a specific template is already installed on the PC and you execute the installer of a new version of this template, the new version overwrites the existing template.

Already existing projects have to be re-created again in order to user the new template behavior.

## 10 FAQ

Q

### What do I need to flash an ECU?

A

- > vFlash is an off-the shelf flashing tool.
- > The supported network hardware is:
  - > Flashing via CAN: Vector CAN family hardware (CANcardXL, CANcardXLc, CANcaseXL, ...)
  - > Flashing via FlexRay: Vector FlexRay hardware family (VN3600, VN7600)
  - > Flashing via Ethernet: Standard Ethernet network adapter
- > Use a prepared flash project (.vflash or .vflashpack) provided by another engineer.
- > Or create your own "Flash Project" based on a "Flash Template".

Q

### Where can I get those Flash Templates?

A

- > Each OEM has its own flash specification with individual flash sequence (some are quite similar, some are quite different) and a dedicated format for flash data (proprietary or ODX-F). If the flash sequence of your OEM is compliant to the general "Vector Flash Sequence", you can use the Flash Template shipped with vFlash.
- > Please contact Vector to check whether a specific Flash Template is already available for your OEM.
- > The Vector reference flash sequence is shipped in source code. If you like, you can use this and perform any adaptations yourself.

Q

### What exactly is a Flash Project?

A

- > A flash project contains all files needed to flash an ECU
  - > .vflash + folder + files
  - > .vflashpack (one self-contained file)
- > Content in detail: Flash sequence script, security access DLL, reference to data and security (e.g. checksum) files, dedicated configuration (e.g. address width)

Q

### What exactly is a Flash Template?

A

- > A basis for a flash project ("master project").
  - > .vflashtemplate + folder + files
- > Content in detail: Predefined configuration for a dedicated OEM variant.
- > When creating a new project, vFlash offers the installed Flash Templates as starting point.



Q

**May I view and/or modify the flash sequence?**

A

- > You can view and modify a flash sequence script, if you have the source code of it.
- > It is an OEM decision whether source code or just binaries are distributed.
- > The source code of the flash sequence script for "OEM Vector" is shipped with the distribution of vFlash.
- > The script is written in C# and linked versus dedicated flash assemblies as part of vFlash.
- > Microsoft Visual Studio 2005/2008 is required. If not available, you can download and install Microsoft Visual C# Express for free.
- > You can run, edit and debug flash scripts easily. However, a high-performing flash sequence is not a trivial task.
- > Vector provides coaching and customer projects to support you in tailoring flash scripts.

Q

**What flashware format (Intel-Hex, Motorola-S or Binary) do I need for my OEM?**

A

- > vFlash supports flash data in Intel-Hex, Motorola-S and Binary format.
- > You can use any of these formats for each OEM - the actual flash sequence is independent from the flash data format.

Q

**My OEM requires flash containers in ODX-F format. Does this mean that I cannot use vFlash?**

A

- > You can use vFlash with the native flash data internally (Intel-Hex, Motorola-S or Binary format).
- > The requirement to use ODX-F format primarily refers to the data exchange process between you and your OEM.
- > Explanation: ODX-F is intended to drive processes, especially in vehicle context (including ECU and software variant identification plus selection).

Q

**I need a way to flash out of my software build process. Can I use vFlash for this use-case?**

A

- > vFlash comes with a simple automation interface.
- > You may find details in paragraph: "8.3 Automation Interface for Batch Processing"

## 11 Examples

### 11.1 Demos

Ecu flash simulations and vflash demo projects

The vFlash installation comes with three different Example projects:

- > CAN Demo
- > FlexRay Demo
- > DoIP Demo

The demos are available via <Start Menu – vFlash – Demos>. All three demos contain a guide with configuration instructions, a vFlash demo project and a simulation. Note, to use the DoIP and the FlexRay simulation you need at least CANoe 7.2 installed on your PC.



---

**Cross reference:** You find detailed instructions in the Demo Guides located in the vFlash Start Menu.

---

### 11.2 Seed&Key DLL

Example project to create Seed&Key DLL

vFlash uses the Seed&Key DLL, provided during configuration, to generate a key from a received seed.

To be able to use a Seed&Key DLL, vFlash needs a library that satisfies a specific DLL interface. The definition of this interface can be found in the folder <installation directory>\Examples\SeedKeyExample. In the same folder there is also an example project, you may adapt to create your own Seed&Key DLL.

### 11.3 vFlash Automation

Example project to use the vFlash Automation Interface

vFlash provides an Automation Interface, implemented as C-DLL. An example is provided in the folder <installation directory>\Examples\. To work with the project you need to copy the file to a local folder and adapt the paths in the project settings and in the source code.

## 12 Vector UDS Reference Project

### Vector UDS Reference Project

Usually Vector provides the Flash Templates for the different OEM specific flash specifications.

However, if you want to define your own flash sequence, the vFlash installation comes with the **Vector UDS Reference Project**, located in the vFlash Example folder. The contained reference flash sequence from Vector is available as source code (C#). All that is needed for modifying is Microsoft Visual Studio for C#. The Express Edition can be obtained from Microsoft free-of-charge.



---

**Cross reference:** An application note is available that describes how to adapt the flash procedure of the **Vector UDS Reference Project**. The application note is located in your installation folder (<installation folder>\Docs).

---

## **Get more Information!**

### **Visit our Website for:**

- > News
- > Products
- > Demo Software
- > Support
- > Training Classes
- > Addresses

**[www.vector.com](http://www.vector.com)**