

# Flash Bootloader OEM

## Technical Reference

CANfbl GM - Programmable Data File Creation

Version 1.2

<b>Authors:</b>	Dennis O'Donnell, Andreas Wenckebach, Jason Learst
<b>Status:</b>	Released

## 1 Document Information

### 1.1 History

Author	Date	Version	Remarks
Dennis O'Donnell	2014-01-31	1.0	Creation
Andreas Wenckebach	2014-10-20	1.1	Added Global B, update screenshots
Jason Learst, Andreas Wenckebach	2015-11-12	1.2	Added chapter Signer Info

### 1.2 Reference Documents

No.	Source	Title	Document No.	Version
[1]	GM	For Global A: Global-A Secure Bootloader Specification	For Global A: N/A	For Global A: 3.1, July 29, 2013
		For Global B: GB6002 Bootloader Specification	For Global B: GB6002	For Global B: 0.9-1.1
[2]	Vector	For Global A: Technical Reference – CANfbl GM SLP5	For Global A: N/A	For Global A: 1.0
		For Global B: Technical Reference – CANfbl GM SLP6	For Global B: N/A	For Global B: 1.0

Table 1-1 References Documents

## Contents

<b>1</b>	<b>Document Information .....</b>	<b>2</b>
1.1	History .....	2
1.2	Reference Documents.....	2
<b>2</b>	<b>Introduction.....</b>	<b>5</b>
<b>3</b>	<b>General Description of Programmable Data Files.....</b>	<b>6</b>
<b>4</b>	<b>Creating Programmable Data Files .....</b>	<b>7</b>
4.1	Determine logical configuration of the ECU .....	7
4.2	XML File Configuration .....	7
4.3	Script File Configuration .....	11
4.4	Generating and using Programmable Data Files. ....	13
4.4.1	Switching from development keys to production keys.....	13
4.4.2	File Generation Issues .....	14
<b>5</b>	<b>Signer Info .....</b>	<b>16</b>
5.1	Signer Info configuration options .....	16
5.1.1	Generate a Signer Info .....	17
5.1.1.1	Configuration of the Script.....	17
5.1.1.2	Using the Script.....	17
5.1.2	Extract Signer Info from SBAT .....	18
<b>6</b>	<b>Glossary .....</b>	<b>19</b>
<b>7</b>	<b>Contact.....</b>	<b>20</b>

## Illustrations

Figure 4-1	Example xml configuration. ....	8
Figure 4-2	Example configuration of Gen_All.bat .....	11
Figure 4-3	Example function calls to generate programmable data files for 1 application and n-calibrations (generic) .....	12
Figure 4-4	Example function calls to generate programmable data files for 1 application and 2 explicit calibrations .....	13
Figure 5-1	Signer Info elements per GB6002 .....	16
Figure 5-2	Configuration section of Generate_SignerInfo.bat.....	17
Figure 5-3	Format of SBA ticket, the SignerInfo can be extracted using bytes 28-565 (there is 2 "data type" bytes in front of Module ID to be additionally removed).....	18

## 2 Introduction

This document covers how to create Programmable Data Files as specified by GM (see [1]). Adding data structure containers with the help of the Vector tool HexView is specifically covered.

### 3 General Description of Programmable Data Files

A programmable data file is a container which includes content and information on how to process, program, and validate the content [1]. The content is either a single application or a single calibration module. Each programmable data file will contain one or more enhancements, which are applied recursively to the content. In general, each programmable data file will have at least two enhancements; plain and signed. More enhancements may be optionally used (e.g. compressed, encrypted, or difference).

For more information on programmable data files please refer to [1] Chapter 10 *Programmable Data Files* and [2].

## 4 Creating Programmable Data Files

Programmable data files can be created using the tool HexView along with an example script and xml file. HexView is able to create plain and signed data types and add them to actual application and calibration data, effectively creating programmable data files. The example script and xml file must be configured specifically for your ECU.

The result of the process will give two sets of files. One set of signed files and one set of plain files.

The signed files can be used during development as inputs to the programming tool (e.g. vFlash). In this case the ECU will be configured with the example security keys provided by Vector. After switching to production key that changes (compare 4.4.1)

The plain files can be used during production by passing them to GM to be signed. In this case the ECU will be configured with the production security keys provided by GM.

### 4.1 Determine logical configuration of the ECU

The first step is to determine the logical configuration of the ECU. Determine the memory map. Determine where the application will be located in memory. Decide how many calibrations are used and what partitions will be used to locate them. This information will be used to configure the XML and script files.



---

**Note**

When designing the logical configuration of the ECU it is necessary to reserve space at the lowest address of a module for the plain header. This is because the plain header must be sent before any other data that is programmed to flash.

---

### 4.2 XML File Configuration

An example XML file (\_ModGenBase.xml) is provided with Vector bootloader deliveries for GM Cyber Security ECUs. This file must be configured with information such as ECU ID, ECU Name, number of partitions, and number of calibration modules. The configuration must be set to meet your specific ECU requirements.



```



1  <?xml version="1.0"?>
2  <!-- edited with XMLSpy v2009 sp1 (http://www.altova.com) by Vector Employee
3  <GmProject Name="This is my GM Project" xmlns:xsi="http://www.w3.org/2001/X
4      <HeaderVersion>1.00</HeaderVersion>
5      <SignedHeaderInfo>
6          <image>SignerInfo.hex</image>
7          <EcuId>00000000000000000000000000000000</EcuId>
8          <EcuName>DemoFb11</EcuName>
9      </SignedHeaderInfo>
10     <AppHeaderInfo name="DemoAppl">
11         <HeaderAddress>0xa003FF00</HeaderAddress>
12         <ModuleId>01</ModuleId>
13         <BCID>0x8000</BCID>
14         <NBID>0000</NBID>
15         <DLS>AA</DLS>
16         <PartNumber>0x11111111</PartNumber>
17         <MessageDigestDataProcessing>32</MessageDigestDataProcessing>
18         <Partitions NumberOfPartitions="1">
19             <Partition NumberOfRegions="1">
20                 <Region start="0xa0030000" length="0x8000"/>
21                 <CalFiles NumberOfCalFiles="2">
22                     <Image BaseAddress="0xa0030000">cal1_plain.gbf</Image>
23                     <Image BaseAddress="0xa0034000">cal2_plain.gbf</Image>
24                 </CalFiles>
25             </Partition>
26         </Partitions>
27     </AppHeaderInfo>
28     <CalHeaderInfo Name="cal1">
29         <ModuleId>2</ModuleId>
30         <CCID>0x8000</CCID>
31         <DLS>AA</DLS>
32         <PartNumber>0x22222222</PartNumber>
33     </CalHeaderInfo>
34     <CalHeaderInfo Name="cal2">
35         <ModuleId>3</ModuleId>
36         <CCID>0x8000</CCID>
37         <DLS>AA</DLS>
38         <PartNumber>22334455</PartNumber>
39     </CalHeaderInfo>
40 </GmProject>

```

Figure 4-1 Example xml configuration.



XML Element	Description
GmProject	<p>The attribute "Name" can be any string to describe the name of the project.</p> <p>Required child elements:</p> <p>HeaderVersion</p> <p>SignedHeaderInfo</p> <p>ApplHeaderInfo</p> <p>CalHeaderInfo (if ApplHeaderInfo defines any calibrations)</p>
HeaderVersion	Version of the GM header format. Currently should always be 1.00.
SignedHeaderInfo	<p>Information for the signed header info. Required child elements:</p> <p>image</p> <p>Eculd</p> <p>EcuName</p>
image	<p>Filename of a hex file containing the Signer Info for the ECU.</p> <p>Compare chapter 5 for possible configuration options</p>
Eculd	ECU ID as specified by GM. Generally this is set to zero (as shown in figure 4-1), meaning that the file can be downloaded to any ECU.
EcuName	ECU Name as specified by GM.
ApplHeaderInfo	<p>Information for the application header.</p> <p>The attribute "name" must be equivalent to &lt;module name&gt; in the script file (e.g. Gen_All.bat) that will generate this application module.</p> <p>Required child elements:</p> <p>HeaderAddress</p> <p>ModuleId</p> <p>BCID</p> <p>NBID</p> <p>DLS</p> <p>PartNumber</p> <p>MessageDigestDataProcessing</p> <p>Partitions</p>
HeaderAddress	<p>Address of the application plain header.</p> <p>This determines where the plain header will be generated to.</p>
ModuleId	ID of the current module as specified by GM.
BCID	<p>Bootloader compatibility identifier.</p> <p>This must match what is stored in the bootloader</p>
NBID	Application not before identifier.
DLS	Design Level Suffix.
PartNumber	Part number of the module.
MessageDigestDataProcessing	<p>Data processing algorithm number to be used by HexView.</p> <p>Currently this should be 32.</p>

XML Element	Description
Partitions	<p>The attribute “NumberOfPartitions” must be set to the number of calibration partitions to be used.</p> <p>Required child elements (if “NumberOfPartitions” greater than 0):</p> <p>Partition</p>
Partition	<p>This element must be present the number of times specified by “NumberOfPartitions” in the element “Partitions”.</p> <p>The attribute “NumberOfRegions” must be set to the number of address regions used within this partition.</p> <p>Required child elements:</p> <p>Region</p> <p>CalFiles</p>
Region	<p>This element must be present the number of times specified by “NumberOfRegions” in the element “Partition”.</p> <p>The attribute “start” must be set to the start address of this region.</p> <p>The attribute “length” must be set to the length of this region (i.e. end address – start address + 1).</p> <div>  <p><b>Caution</b> Regions must be listed in ascending order by starting address.</p> </div>
CalFiles	<p>The attribute “NumberOfCalFiles” must be set to the number of calibration files within this partition.</p> <p>Required child elements:</p> <p>Image</p>
Image	<p>This element must be present the number of times specified by “NumberOfCalFiles” in the element “CalFiles”</p> <p>The attribute “BaseAddress” must be set to the address of the plain header in the calibration file.</p> <p>Filename of calibration file that has already been populated with the plain header.</p> <div>  <p><b>Caution</b> Calibration files must be listed in ascending order by “BaseAddress”.</p> </div>
CalHeaderInfo	<p>Information for the calibration header.</p> <p>This element must be present for each calibration file that is specified by the application.</p> <p>The attribute “name” must be equivalent to &lt;module name&gt; in the script file (e.g. Gen_All.bat) that will generate this calibration module.</p>

XML Element	Description
	Required child elements: ModuleId CCID DLS PartNumber
CCID	Calibration compatibility identifier.

Table 4-1 Description of XML elements in \_ModGenBase.xml.

### 4.3 Script File Configuration

An example script file (\_Gen\_All.bat) is provided with Vector bootloader deliveries for GM Cyber Security ECUs. This file must be configured to create the programmable data files required for your ECU. Several parameters must be set based on requirements and local machine.

```

6 .....
7 :: Configuration
8 .....
9
10 ::Path to HexView tool and external files
11 set HEXVIEW_EXE=..\..\..\..\Misc\Hexview\hexview.exe
12 set XML_MOD_DEF=ModGenBase.xml
13 set PRIVATE_KEY_SOURCE=..\..\..\..\Misc\DemoKey_2048\rsakeys_2048.txt
14 set CAL_FOLDER=DemoCal_withoutHeader
15
16 :: file format and file name extensions for generated files (added to module name)
17 set OUTFORMAT=gbf
18 set PEXT=_plain
19 set PLAIN_FOLDER=.\GeneratedAndToSignedByGm
20 set SEXT=_sign
21
22 :: Hexview configuration parameters (check hexview tool documentation):
23 :: - Fill all ( /FA )
24 :: - alignment (address /Ad<val> and length /Al)
25 :: - checksum param ("Cs5" for B. endian / "Cs6" for l. endian ECU)
26 :: - Call Hexview silently ("-s") or not (leave empty)
27 set ALIGN=/Ad:8 /Al /Af:0xCA
28 set CS=/Cs5
29 set FILL=
30 set SILENT=-s

```

Figure 4-2 Example configuration of Gen\_All.bat

Script file parameter	Description
HEXVIEW_EXE	Path and filename of HexView executable file.
XML_MOD_DEF	Path and filename of the xml file that contains information for your ECU (e.g. ModGenBase.xml).
PRIVATE_KEY_SOURCE	Text file containing 2048 bit RSA key (private key). The delivery includes a valid demo key (rsakeys_2048.txt) that can be used during development to create development suited signed header containers.
CAL_FOLDER	Folder containing all calibration files. The script will add the enhancements to all files found in this folder that are referenced by the XML configuration also.
OUTFORMAT	File extension of output files (e.g. gbf or bin).
PEXT	Sub-name of output files that contain only the plain header.
PLAIN_FOLDER	Path to place output files that contain only the plain header. These files can be passed to GM to be signed.
SEXT	Sub-name of output files that contain the signed header.
ALIGN	Alignment options for HexView. This is used to pad data before adding plain and signed headers.
CS	Checksum algorithm to be used to calculate the checksum of the data that is programmed. The checksum is placed in the first two bytes of the plain header. Cs5 should be used for big endian ECUs and Cs6 should be used for little endian ECUs.
FILL	Fill options for HexView. This is used to fill gaps in between data before adding plain and signed headers.
SILENT	This can be used to call HexView silently (use -s) or not (leave blank).

Table 4-2 Description of Gen\_All.bat configurations

The header generation function must be called for each module used by the ECU. For application modules, the GenAppHeaders function should be called. For calibration modules, the GenCalHeaders function should be called. All required modules should be handled in Gen\_All.bat. The Demo will use a “for loop” to call GenCalHeaders for all modules (files) found in %CAL\_FOLDER%. Use the same <module name> as used in the XML configuration file. Specify the path and filename of the hex file that will be used for each module.

```

47 ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
48 :: Required Function Parameters for GenAppHeaders (Application files), GenCalHeaders (calibration files):
49 :: - <module name> : "Module Name" - Needs to match configured module name (XML_MOD_DEF file)
50 :: - <file name>   : "File name" - File name of raw input file
51 ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
52 :: call :<func name>    "<module name>" "<file name>"
53
54 :: For all cal files in CAL_FOLDER call function GenCalHeaders (Assumption: module name == file name without extension )
55 for /f %f in ('dir /b .\%CAL_FOLDER%') do (
56     call :GenCalHeaders "%~nf" "%CAL_FOLDER%\%f"
57 )
58 call :GenAppHeaders "DemoAppl"      "%..\DemoAppl.mot"

```

Figure 4-3 Example function calls to generate programmable data files for 1 application and n-calibrations (generic)

```
37 :: call :<func name>    "<module name>" "<file name>"
38 call    :GenCalHeaders  "cal1"          "DemoCal_withoutHeader\cal1.hex"
39 call    :GenCalHeaders  "cal2"          "DemoCal_withoutHeader\cal2.hex"
40 call    :GenAppHeaders  "DemoApp1"      "DemoApp1.hex"
```

Figure 4-4 Example function calls to generate programmable data files for 1 application and 2 explicit calibrations

**Caution**

Calibration files should be generated before application files because the application file generation depends on the calibration files.

**Caution**

The functions GenCalHeaders and GenAppHeaders used in Gen\_All.bat are designed to generate programmable data files that are sufficient in most cases. Please contact Vector for further assistance in case the generated files are not suitable for your use case.

## 4.4 Generating and using Programmable Data Files.

Once the XLM file and script file are configured properly the programmable data files can be generated. The files are generated by running the script file (Gen\_All.bat). The script file will call HexView to generate the files. The file generation will produce a set of files that contain only the plain header and a set of files that contain the plain and signed headers.

The signed files can be downloaded to an ECU that uses the example keys from Vector. Specifically, the bootloader is configured to use the example public key that matches the example private key specified in the script file and through the signer info hex file.

The plain files can be passed to GM for signing. In this case, the private key configured in the script file is not used. The files signed by GM can be used with a bootloader configured with the public key that is given by GM.

### 4.4.1 Switching from development keys to production keys

When switching from development keys (provided by Vector) to production keys (provided by GM) it is not always necessary to change the XML or script files described in this document. The keys themselves only affect the signed header and do not affect the plain header. This means that changing the keys in the XML or script files does not affect the generation of the plain files. Therefore, without making any changes, the plain files still can be provided to GM for signing.

The plain files only need to be regenerated for the following reasons:

- There is a general ECU change that affects the plain header (e.g. application/calibration update/rebuild, partition/memory map change, etc.)
- The App-NBID needs to be updated (e.g. this may be requested by GM even if the application/calibration data has not changed in any way).

The signed files generally are not needed any more after switching to production keys. However, with loaded SBA ticket (this way by-passing the Security mechanism) you can use the script again to generate a signed header correctly formatted. The signature can be wrong in this use case because it is not validated by the FBL. Note that you will have to configure a GM SignerInfo (XML element “image”) file in this case to our environment, as the signature within the SignerInfo has still to be valid. See 5.1.2 for more information.

#### 4.4.2 File Generation Issues

In case the XML or script file is not configured properly, HexView will not report an error while attempting to generate the files. Because of this, it is important to check that the intended files were actually generated and that they generated correctly. This can be done, for example, by checking the timestamps of the files and testing them by programming them with the bootloader.



##### Note

In some instances HexView will generate an error if something is wrong with the XML configuration. Check the folder where Gen\_All.bat is located for any .err files. Because HexView does not generate an error in all failure cases this method cannot be solely relied upon. It is still necessary to check if the files generated correctly.

In case the files did not generate correctly, it is recommended to briefly check the files and then contact Vector if the issue cannot be quickly resolved. When contacting Vector for support with programmable data file generation please provide the following.

- Description of the observed issue (e.g. the files do not generate).
- The XML file used.
- The script file used.
- The application and calibration hex/s19 files that are used as inputs.



##### Caution

HexView version 1.08.03 or greater is required. Please check your HexView version in case of error.



## 5 Signer Info

The Signer Info structure contains the following elements:

Signer Info	Length (Bytes)	Cvt	Description
Subject Name	16	M	Subject Name is an ECU family name which is embedded in the subject name field of the signing certificate. It is a 16 byte hexadecimal value that identifies the group of ECUs for which the Signer Info is applicable. It is used during the programming process to determine if the Signer Info is compatible with the target. Example: Powertrain, Infotainment, Telematics, etc.
Certificate ID	8	M	Certificate ID is the unique identifier for the entire Signer Info. It is an 8 byte hexadecimal value used by the release system to select the appropriate signing certificate and key to sign the software or calibration. This parameter is not used by the bootloader.
Key-NBID	2	M	Key-Not Before Identifier. This parameter is intended to prevent roll back to a previous key. The bootloader shall compare the Key-NBID in this field to the current Key-NBID as known to the bootloader. The bootloader shall reject the Signer Info if the Key-NBID in this field is less than the current value of the Key-NBID stored in the bootloader.  Once the application software is successfully programmed, the value of this parameter is used as a reference point to prevent roll back to a previous key with lower Key-NBID. The ECU shall allow a minimum of 16 updates to the Key-NBID.
Signer Public Key	256	M	The Signer Public Key is an RSA 2048 bit public key that shall be used to verify the signature on the signed header.
Root Signature	256	M	This field holds the RSA 2048 bit signature of the Signer Info using GM root key.

Figure 5-1 Signer Info elements per GB6002

The Root Signature will be verified using the bootloader configured Root Public Key. This verification is mandated in any case, also if security is bypassed using SBAT.

The Signer Info is static data content, that is reused unchanged for any signed container data for a given ECU as long as the Signer Key is considered secure. If the currently used Signer Key is no more considered secure, Signer Info is to be updated by GM.

### 5.1 Signer Info configuration options

In order to create download containers, suitable Signer Info is to be configured to ModGenBase.xml. In case you are already configuring Gm production key, you can download your own generated containers only if SBAT is present on ECU (4.4.1).

Options while using Vector Development Key:

- Option1: In your delivery you will find SignerInfoDummyKey.hex that can be used while using Vector Development keys.
- Option2: SignerInfoDummyKey.hex file is fixed static elements (e.g. Subject Name "Engine"), if you want to change Signer Info Content, check 5.1.1



Options while using GM Production Key:

- Option3: Use Gm provided Signer Info, check 5.1.2

### 5.1.1 Generate a Signer Info

Included with the delivery is a script to generate a Signer Info with the provided Vector Dummy Key. This script generates a new Dummy Signer Info with updated Subject Name, Certificate ID, and Key-NBID.

The script is located here: .\Misc\HdrGen\Generate\_SignerInfo.bat.

#### 5.1.1.1 Configuration of the Script

```

11  ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
12  :: Configuration
13  ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
14
15  ::Set new SignerInfo parameters
16  set SUBJECT_NAME=Engine
17  set CERT_ID=0xAAAAAAAAAAAAAAAA
18  set KEY_NBID=0x0000
19
20  ::Set output file name
21  set OUTPUT_FILE=%SUBJECT_NAME%_SignerInfo.hex
22
23  ::Path to Hexview tool and external files
24  set HEXVIEW_EXE=..\Hexview\hexview.exe
25  set PRIVATE_KEY_SOURCE=..\DemoKey_2048\rsakeys_2048.txt
26  set SILENT=/s
27  set SIGNER_INFO=SignerInfoDummyKey.hex

```

Figure 5-2 Configuration section of Generate\_SignerInfo.bat

Edit the parameters to reflect the options that you want set for the Signer Info. The Certificate ID is not used in the bootloader and usually there is no need to change this in the Dummy Signer Info. Edit the Subject Name with the ASCII Subject Name that your ECU has been assigned to. Key-NBID value can be updated with some value different to 0x0000 if required.

The Output File name can be adjusted. Verify that the paths to Hexview and the other external files are correct. If this script is executed from the default package installation folder these paths should not need to be adjusted.

#### 5.1.1.2 Using the Script

Double click the Generate\_SignerInfo.bat file to create a new Signer Info with the parameters that you specified in the file. Be default this will create a new file beginning with the Subject Name that was configured. For example: Engine\_SignerInfo.hex.

Use this file when generating your download containers by adjusting your ModGenBase.xml script that is being used for the generation of the containers.

If the bootloader is properly configured with a matching Subject Name and its current Key NBID is less than or equal to the Key NBID set in the script, the Signed Header should be successfully validated.

### 5.1.2 Extract Signer Info from SBAT

A Signer Info signed with GM production key is provided to you by GM only indirectly, included in signed download containers and SBAT. Included with the delivery is a script to extract Signer Info from SBAT, .\Misc\HdrGen\Extract\_SignerInfo.bat.

Use the GM container extracted Signer Info when generating your download containers by adjusting your ModGenBase.xml in case you configure GM production key and want to bypass Security using loaded SBAT (compare 4.4.1.).

Signature Bypass Authorization Header	Length (Bytes)	Cvt	Description
Module ID	2	M	Module ID. See Module ID table for more information.
ECU Name	8	M	This field contains an ASCII representation of the ECU Name.
ECU ID	16	M	ECU Identity. This field must be equal to the ECU identity as known to the bootloader.
Signer Info	538	M	This parameter contains information about the key used to validate the signature.
Signature	256	M	The Signature is a PKCS#1 V1.5 as defined in section "Signature-Bypass Authorization Signature".

Figure 5-3 Format of SBA ticket, the SignerInfo can be extracted using bytes 28-565 (there is 2 "data type" bytes in front of Module ID to be additionally removed).

The corresponding command used in the Script is:

**hexview.exe SBAT.bin -cr:0-27:566-821 -XB -o SignerInfo.bin**

*(cuts out 28 bytes at the beginning and 256 bytes at the end)*

## 6 Glossary

Abbreviations and Expressions	Explanation
HexView	Vector provided PC tool that is used to manipulate hex and s19 files.
vFlash	Programming test tool available from Vector.
SBAT	<p>Signature Bypass Authorization Ticket</p> <p>Allows Bypassing security options on a single ECU. The ticket itself is protected by signature valid for one specific ECU_ID/ECU only. /ECU only.</p> <p>The ECU will check for presence of such a ticket during startup. If a valid ticket is present, the bootloader will no more check digest and Digital Signature of download containers.</p> <p>This allows for modifying and downloading containers without having GM to sign the corresponding containers even though production keys are configured to the ECU (use cases: development, debugging of production ECU).</p>

## 7 Contact

[FbiSupport@vector-informatik.de](mailto:FbiSupport@vector-informatik.de)

Visit our website for more information on

- > News
- > Products
- > Demo software
- > Support
- > Training data
- > Addresses

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