
DPS User Guide

Revision 1.9



March 14, 2022

Contents

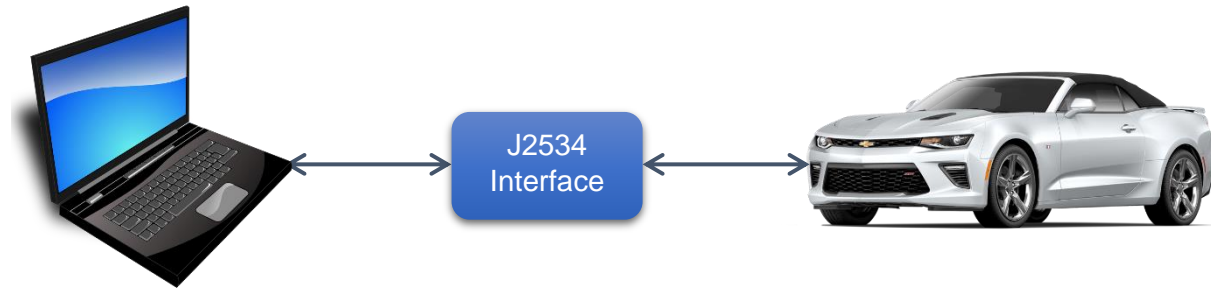
Overview.....	4
DPS.....	4
DPS Software License.....	4
J2534 Interface.....	4
Programming	5
Opening the Programming Dialog.....	5
Programming Dialog.....	5
Protocol/Pin Selection Comments.....	6
Selecting a J2534 Device	6
Program an ECU.....	8
Clearing DTCs.....	8
Get Controller Info (Reading ECU Data).....	9
Testing Type 4 Applications.....	11
ECU Configuration (XML).....	13
Create Build Data Record File.....	15
ECU Bypass Ticket.....	17
Selecting a VIN for use during Programming Event.....	19
Service Programming Archive Tool.....	20
View Contents of an Archive.....	21
View Utility File	22
Convert Files	23
Create New Archive.....	24
Appendix A: Other Specialized DPS Functions	27
Supersession Programming (SST = Supersession Table).....	27
BATCH Supersession Programming	27
Sequential Programming	27
Vehicle Key Provisioning (Global B/GEM only).....	27
Appendix B: Other DPS “Option” Functionality	28
Disable Class 2 ID Filtering	28
Disable SST VinLog.Dat Writing.....	28
Disable GMLAN Automatic P/N VIT Setting.....	28
Enable Prog Event Logging	28
Disable Auto File Conversion	29
Enable GMLAN (\$28, \$A5) Error Toleration	29

Appendix C: What is "Populate VIT"30
 Populate VIT (Vehicle Information Table)30
Appendix D: What is a Utility File31
Appendix E: Program an ECU that requires a Security Code32

Overview

DPS

The Development Programming System (DPS) application is a 32-bit Windows-based application used for the flash programming of General Motors electronic control units (ECU) and verification of *Service Programming*. DPS is a development tool intended for the GM engineering community and ECU suppliers. It is designed to interface with SAE J2534-compliant hardware interfaces during the communication process.



The communications subsystem in DPS is the same as the Service Programming System (SPS), which is used at dealerships. Therefore, proof of functionality within DPS ensures success within the dealership system from a communication perspective.

DPS is used by all groups in GM as well as other groups supporting a GM product or ECU, such as GM suppliers, joint venture groups, educational institutions, and over 10,000 other users.

DPS Software License

DPS requires a unique per-PC software license. DPS License requests are submitted through the [DPS User Portal](#).

J2534 Interface

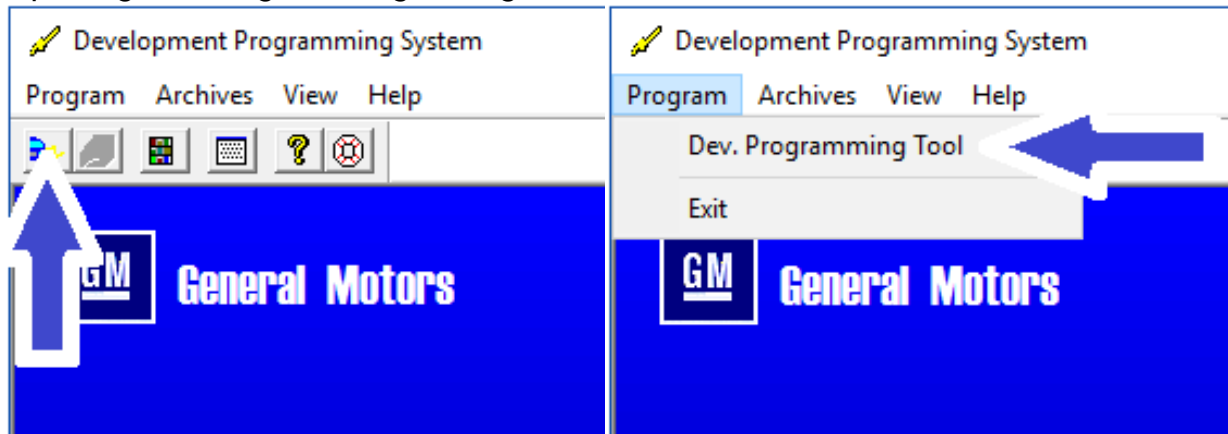
J2534 is an interface standard designed by the Society of Automotive Engineers (SAE) and mandated by the Environmental Protection Agency (EPA) for vehicle ECU reprogramming. Thus, J2534 has been adopted by all vehicle manufacturers, and allows the independent aftermarket the ability to program ECUs.

Examples of J2534 devices: MDI 2, Peak PCAN, Cardaq, Cardaq-Plus, Mongoose, CAT, Vector Pass Thru, NeoVi

Programming

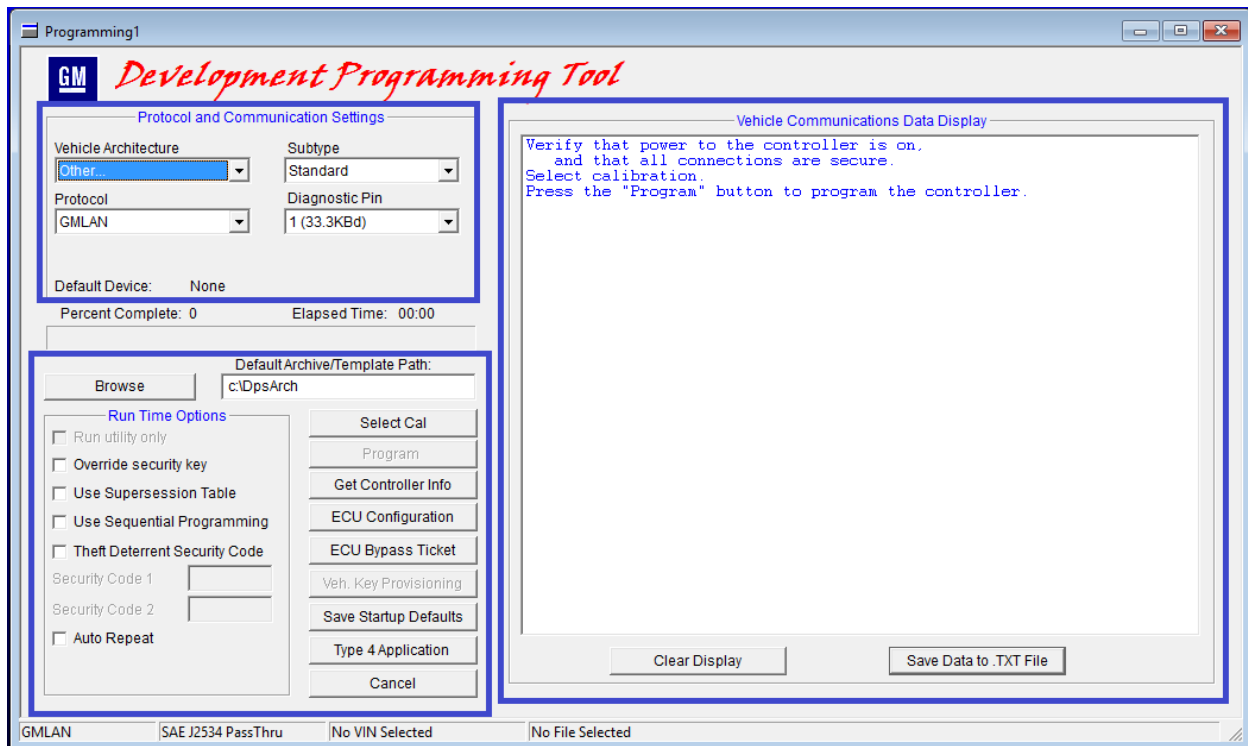
The programming window supports functionality with respect to communicating with the vehicle. This includes programming an ECU, clearing DTCs, reading ECU data, testing Type 4 applications, ECU configuration (XML), ECU Bypass Ticket writing, selecting a VIN for programming, and more. While the programming dialog is open, the menu bar changes to include new options.

Opening the Programming Dialog



Programming Dialog

The programming dialog is broken up into three sections. In the top left, the user can select the protocol and communication settings. These settings are used for all communication features supported in the programming dialog. The controls for the programming interface are in the bottom left of the dialog. On the right side, the dialog will display the communication messages.



Protocol/Pin Selection Comments

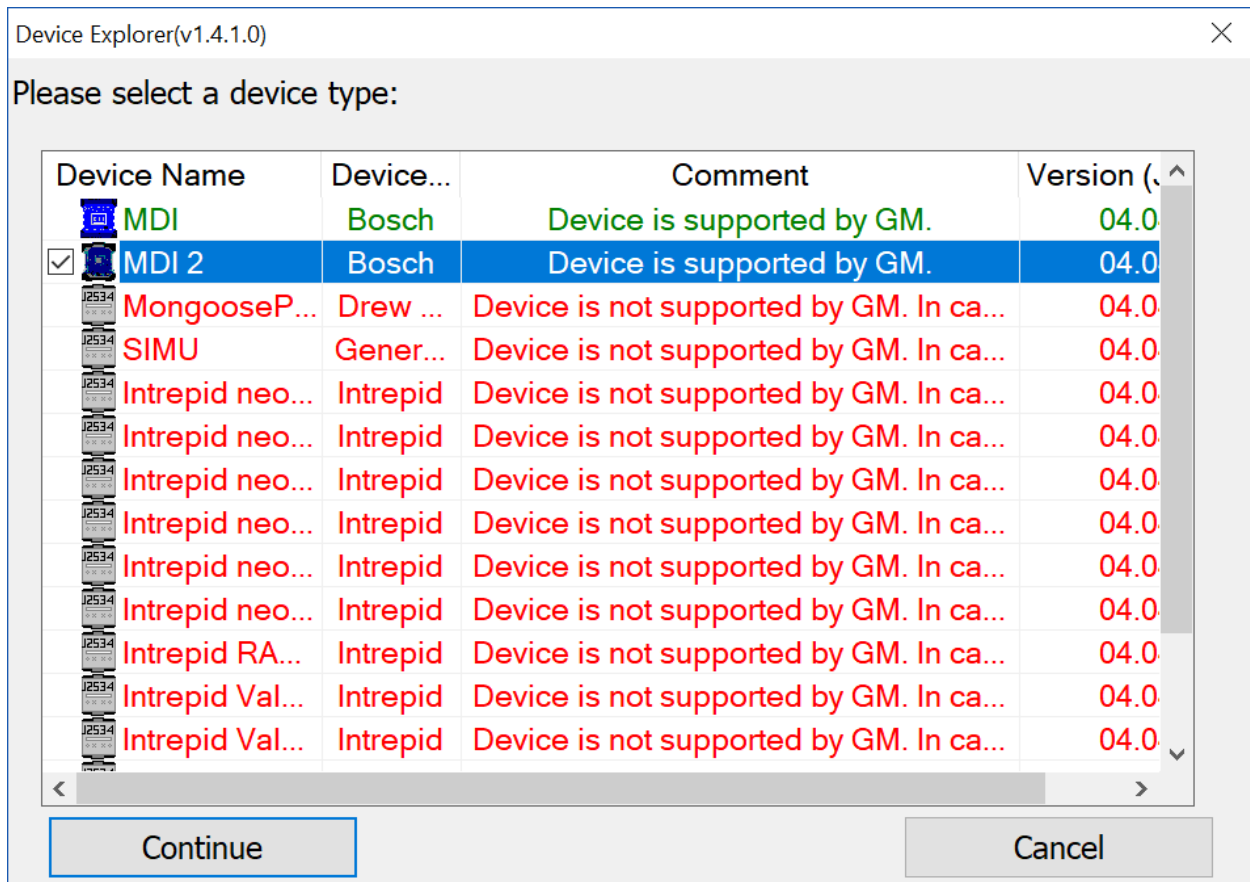
Some protocol/pin selections require additional installation supplied by 3rd parties.

Global B CAN 2.0 / CAN FD: DPS provides a no-gateway selection for programming cases in which a CGM is not present. In this case, DPS will send a default wake-up message or user-defined message in the VcsConfig.txt file in the DPS installation directory.

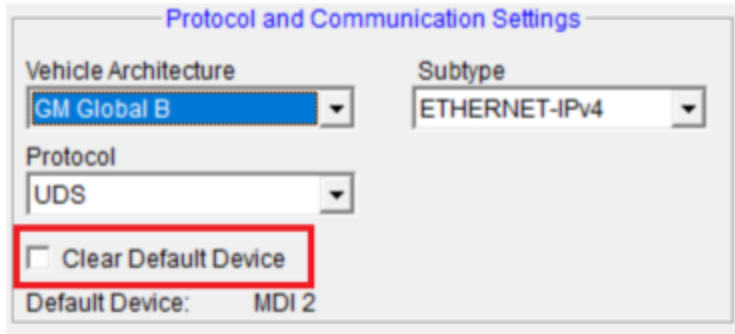
Global B DoIP: There are two configurations: 4K Block Limit and Optimal. Optimal will use the largest possible block size based on the UDS service mode \$34 response and the DoIP status response message. 4K will use the largest possible block but will not exceed 4095.

Selecting a J2534 Device

When performing an action that initiates a communication event, the Device Explorer interface will prompt the user to select a device type if no device is selected. This dialog allows a user to select a default device to which DPS will attempt to connect.

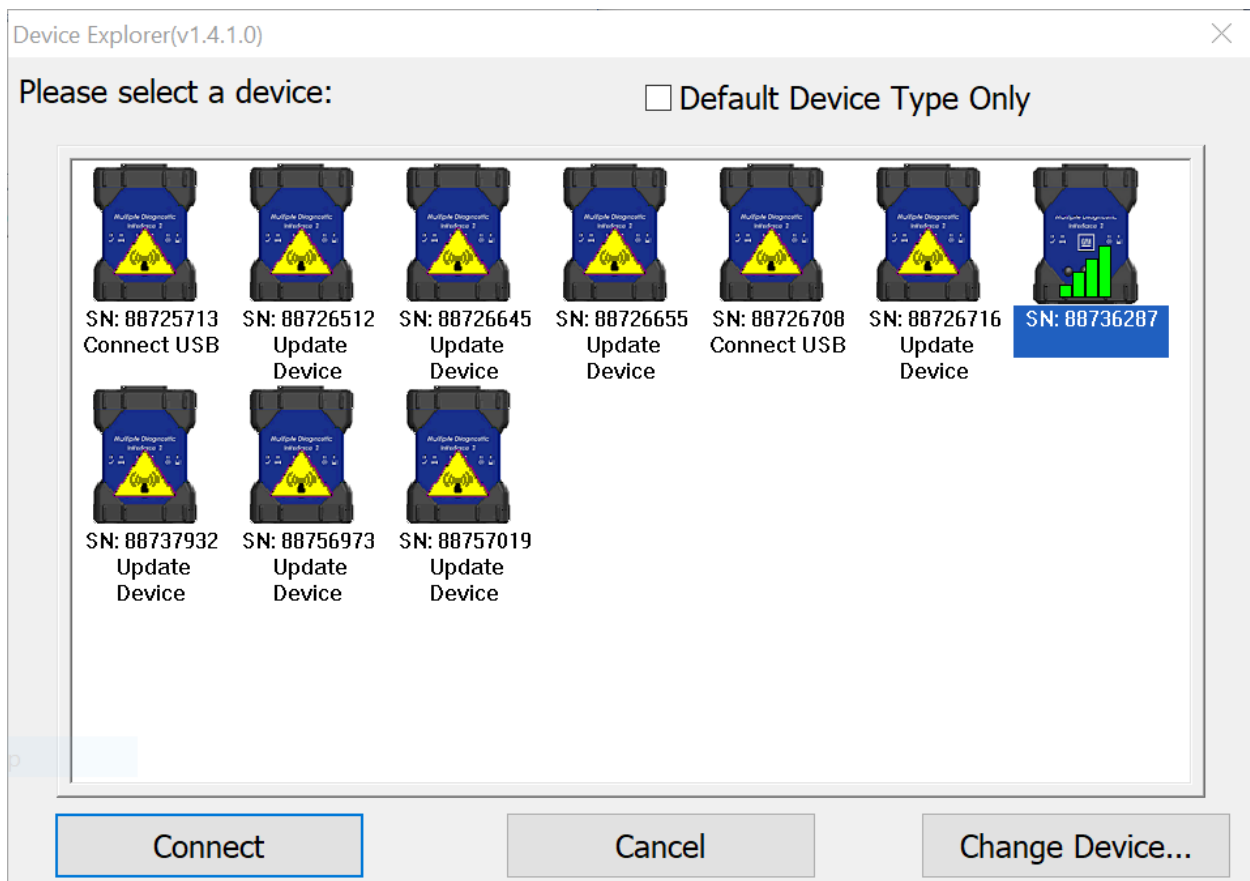


To select a different device type, click the **Clear Default Device** checkbox in the **Protocol and Communication Settings** area of the programming interface prior to the next communication event.



For issues with most devices, contact the device manufacturer for support. If the "Device is support by GM" then the DPS team provides support for issues with those devices.

If MDI or MDI 2 device type is selected, the user may be prompted to select a specific device to which to connect:

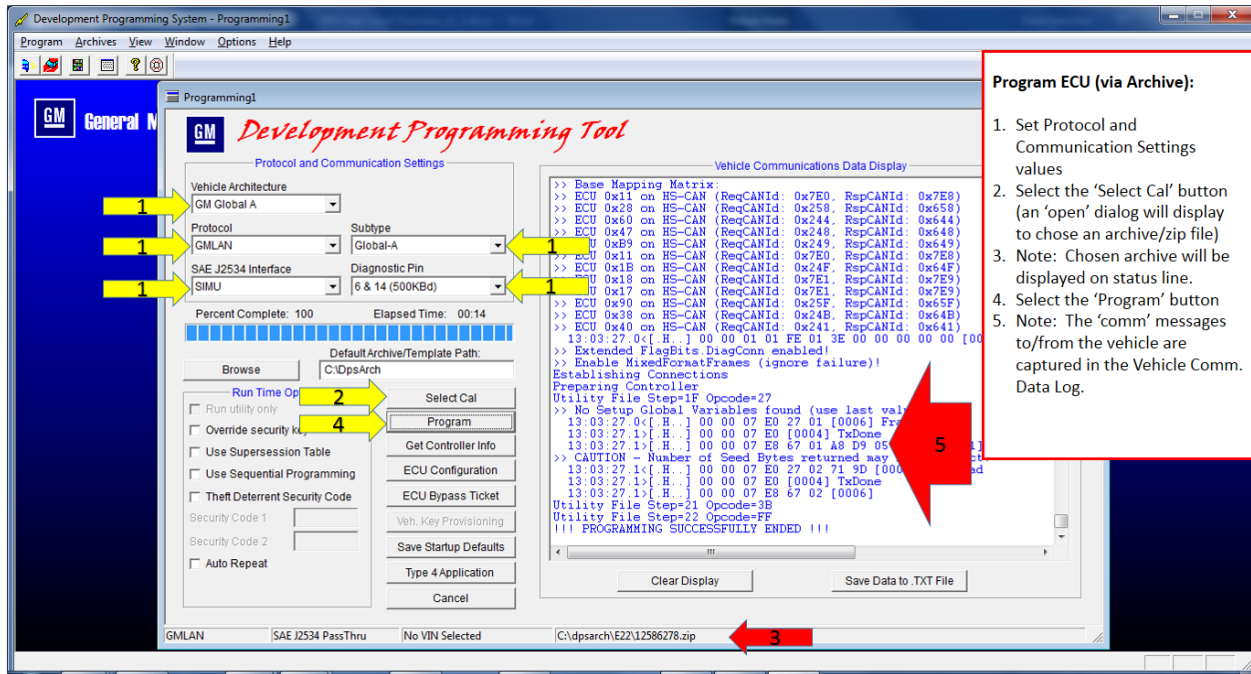


Program an ECU

To program an ECU, follow these steps:

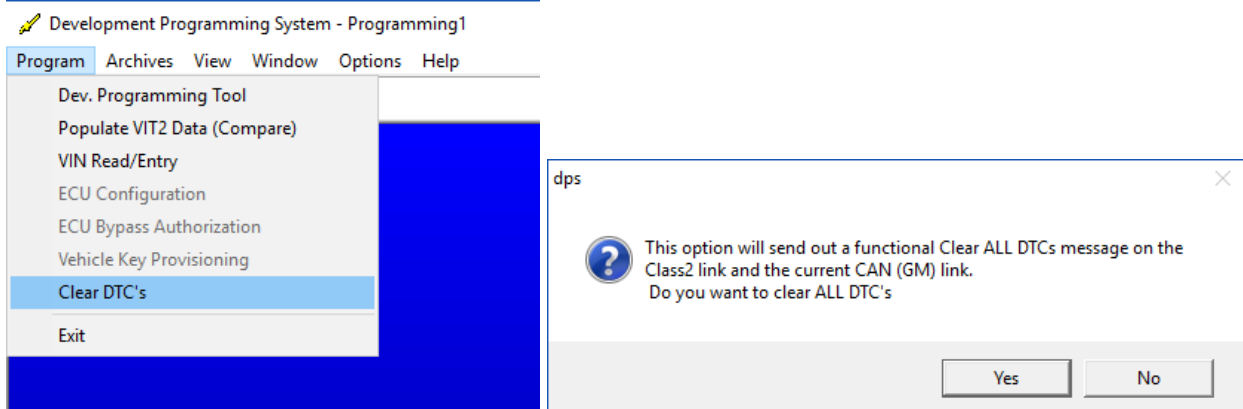
1. Set the protocol and communication settings.
2. Click the **Select Cal** button, which displays the open file dialog.
3. Click **Program**.

The communication messages to/from the vehicle are captured in the vehicle communications data display.



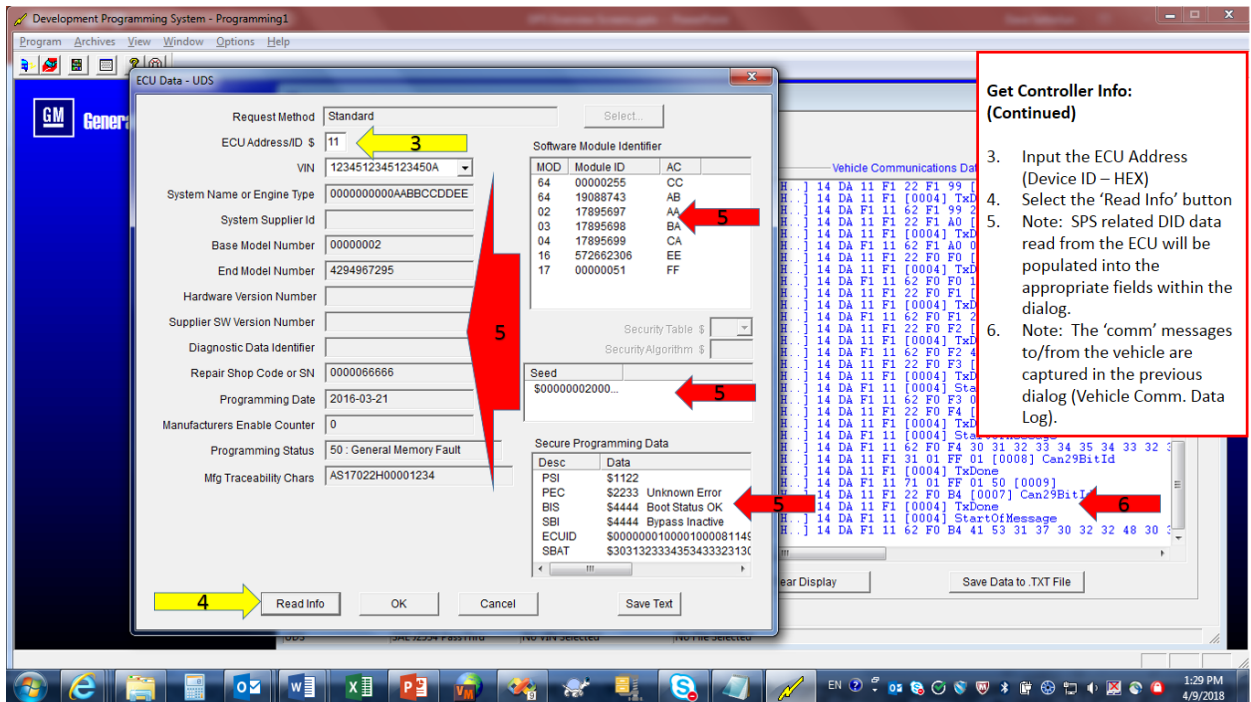
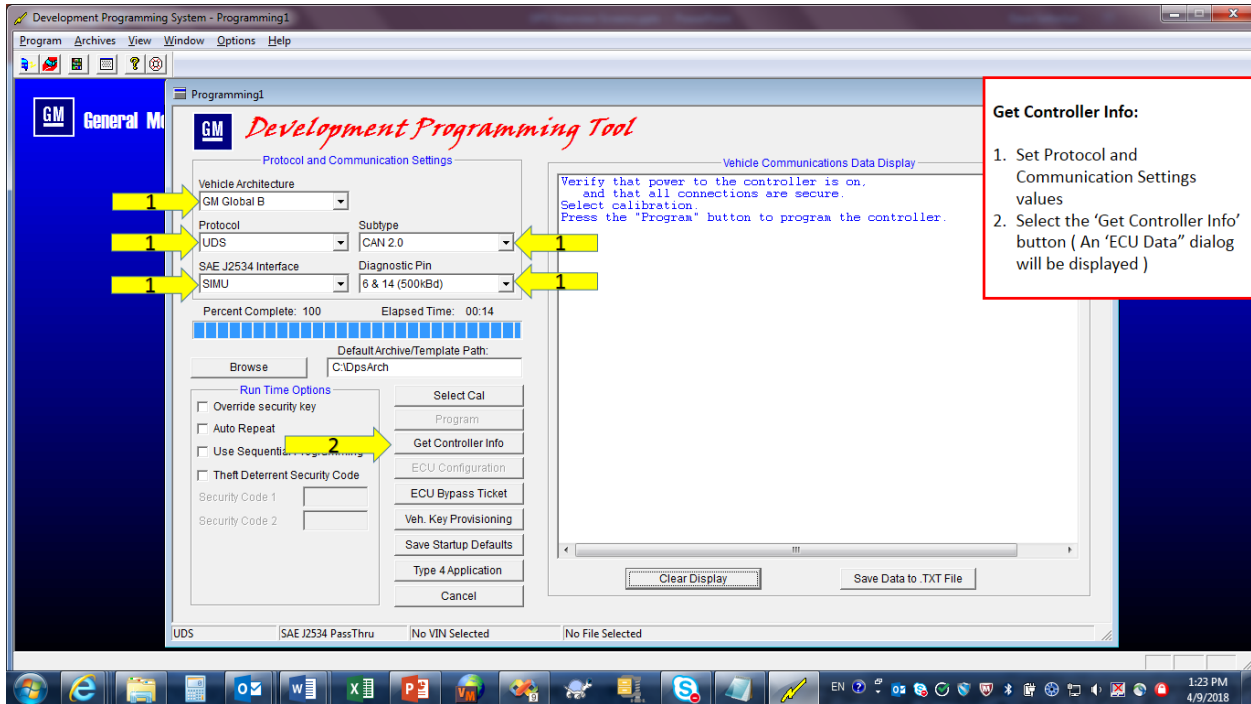
Clearing DTCs

While the program dialog is open and the communication settings are set, the user can clear DTCs using the **Clear DTC's** button in the program menu. Clearing DTCs is only supported for Class II, GMLAN, and GM UDS. The data display will display the messages.



Get Controller Info (Reading ECU Data)

Set the communication settings in the programming dialog and click the **Get Controller Info** button. Input the ECU Address (Device ID – HEX) and select the **Read Info** button. SPS related DID data read from the ECU will populate the appropriate fields within the dialog.



Example dialog:

ECU Data - UDS
✕

Request Method Select..

ECU Address/ID \$

VIN ▼

System Name or Engine Type

System Supplier Id

Base Model Number

End Model Number

Hardware Version Number

Supplier SW Version Number

Diagnostic Data Identifier

Repair Shop Code or SN

Programming Date

Manufacturers Enable Counter

Programming Status

Mfg Traceability Chars

Software Module Identifier

MOD	Module ID	AC
64	00000255	CC
64	19088743	AB
02	17895697	AA
03	17895698	BA
04	17895699	CA
16	572662306	EE
17	00000051	FF

Security Table \$

Security Algorithm \$

Seed

\$00000002000...

Secure Programming Data

Desc	Data
PSI	\$1122
PEC	\$2233 Unknown Error
BIS	\$4444 Boot Status OK
SBI	\$4444 Bypass Inactive
ECUID	\$0000000100001000081149
SBAT	\$3031323334353433323130

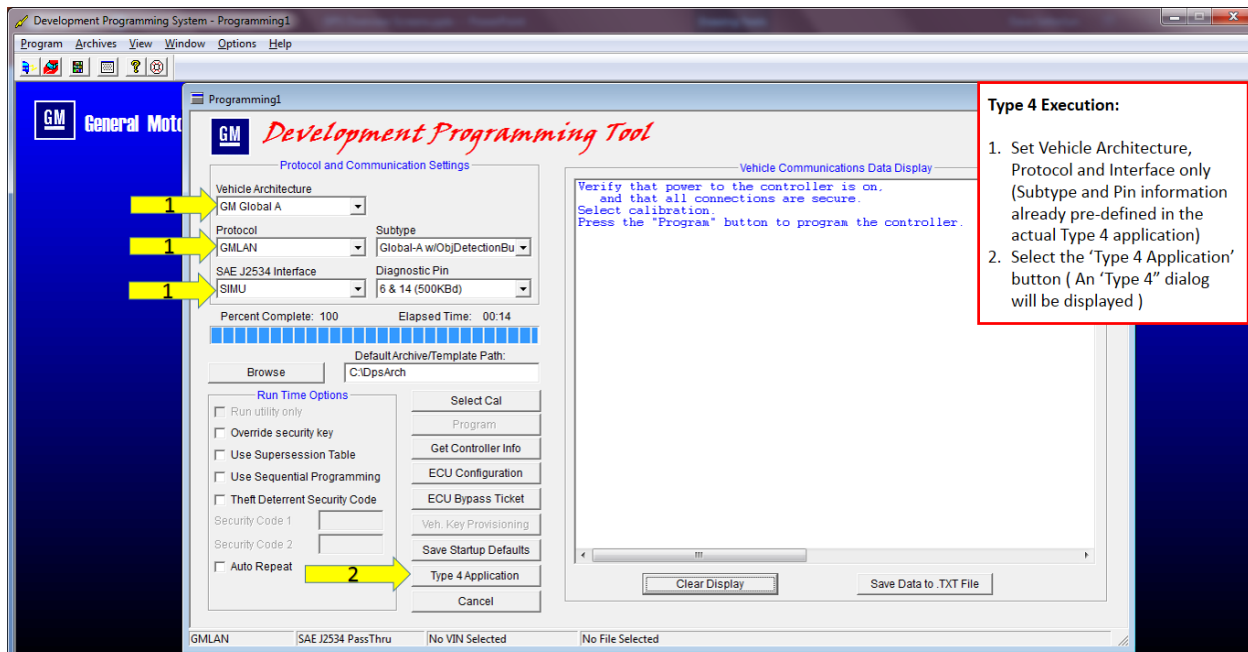
< >

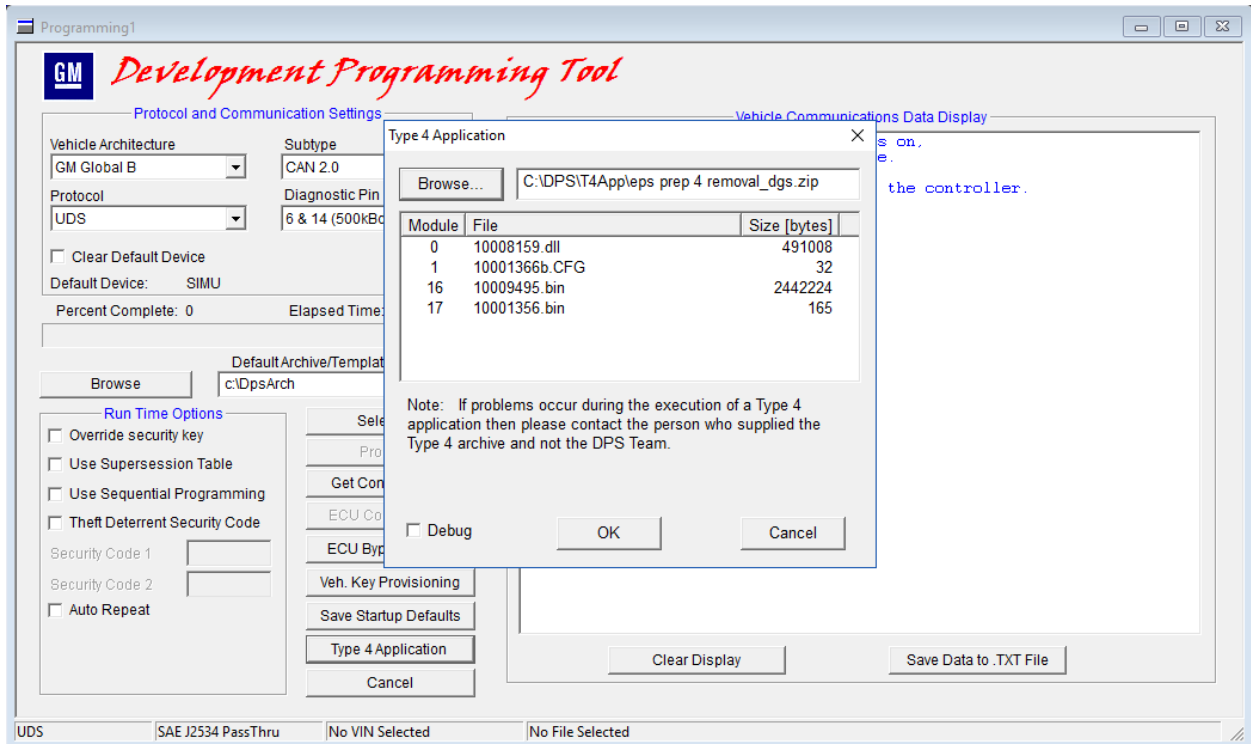
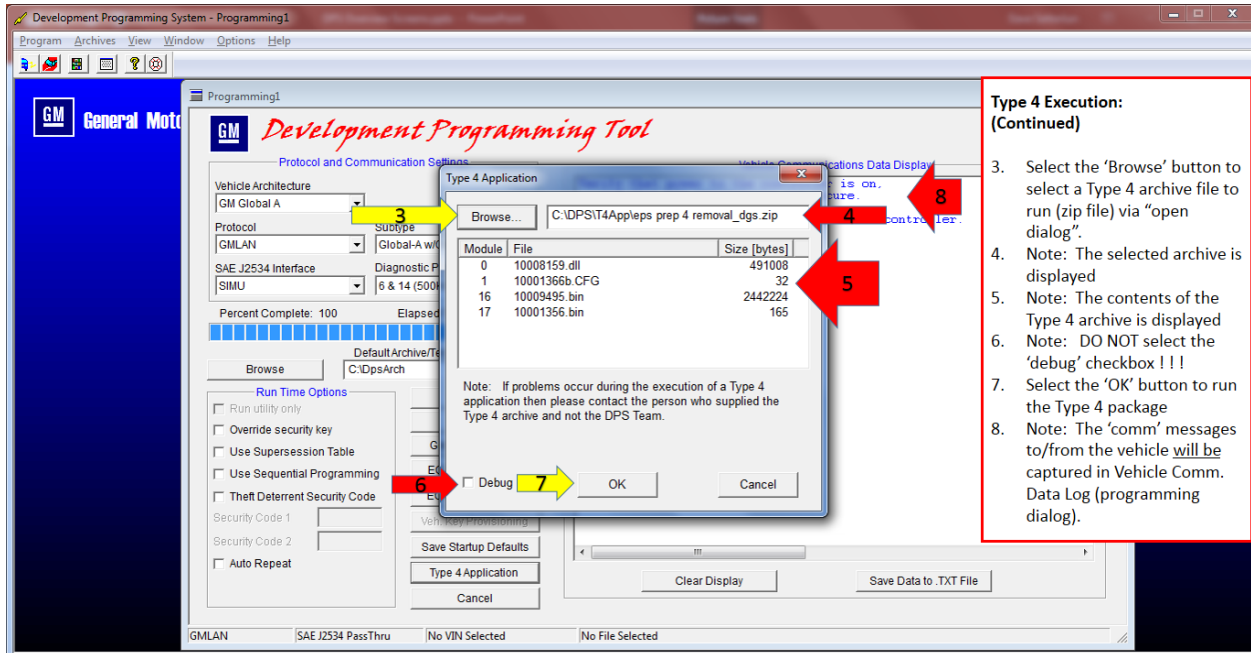
Read Info
OK
Cancel
Save Text

Testing Type 4 Applications

The Type 4 Application functionality within DPS allows the user to run a pre-defined software procedure (DLL) on an ECU. These procedures typically execute a setup/learn function which requires user interaction (examples: Oil Life Setup, BPP Learn, Clutch Position Learn, VCAP – Vehicle Data Capture). This DPS functionality is mostly used during the software development of the actual Type 4 application code that is used in Tis2Web. For this reason, no Type 4 packages are provided with the DPS installation. On rare occasion, GM engineers do obtain one of these procedures for internal validation on development vehicles. Most DPS users will never see this button displayed because there are some unique dependencies. Unless the DPS user is a Type 4 software developer or Type 4 development validator, the DPS Team will not provide any support. The DPS team does not own or distribute any of the type 4 packages.

To execute the Type 4 application, set the Vehicle Architecture and Protocol under communication settings. The subtype and pin information are predefined in the actual Type 4 application. Select the **Type 4 Application** button. In the Type 4 Application dialog, the user can click the browse button to select a Type 4 archive file to run (.zip). Click **OK** to run the Type 4 package. Do not use the Debug checkbox.





ECU Configuration (XML)

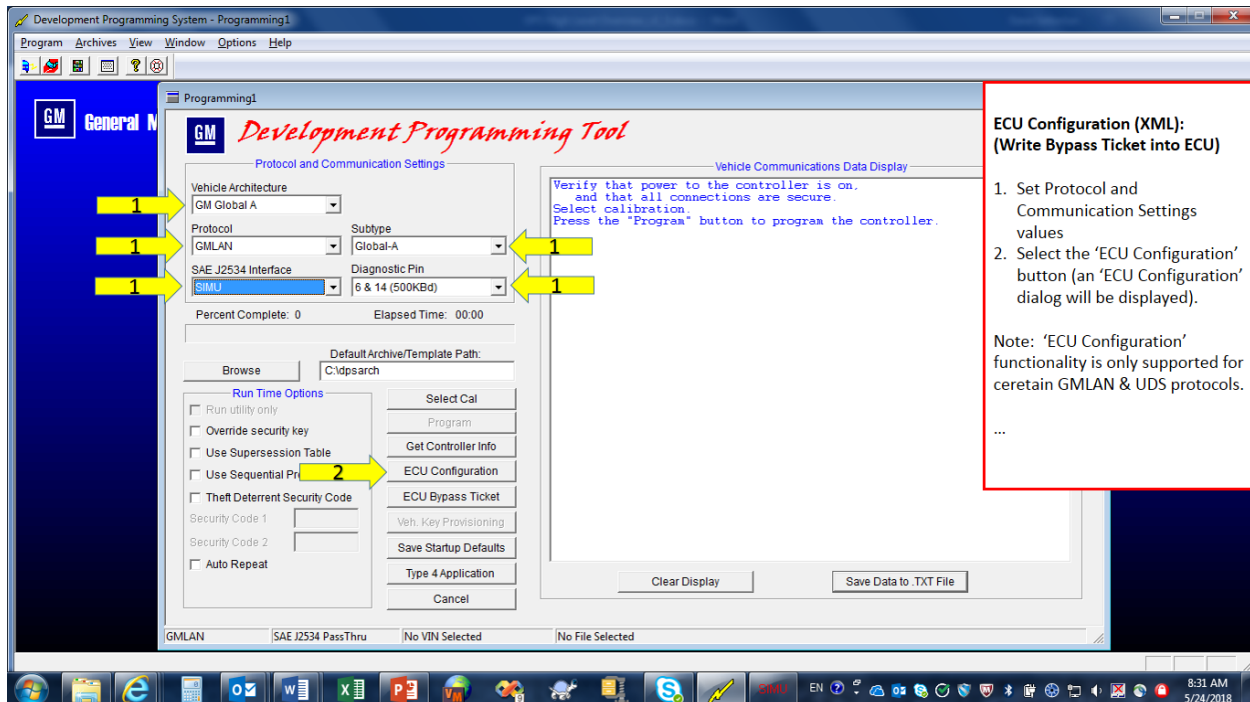
An ECU configuration event allows a user to perform bit/byte setting within an ECU which is based on RPO / vehicle content. The logic to drive this event is determined from an XML configuration file. DPS allows an engineer to validate their XML logic based on various RPO settings (user defined). This DPS functionality is mostly used during the development of the XML file. For this reason, no "configuration event" files are provided with the DPS installation.

XML Configuration files are provided by the DPS user. The XML file must be copied to the \DPS\Config\ folder location where DPS is installed (this typically is C:\DPS\Config\ **). Any user defined Build Record Data file (.txt or .xml), if any, also should be copied to this same folder location.

** This is the location where DPS installs the .XSD (XML schema files).

To use the ECU Configuration feature, set the communication settings, then click the **ECU Configuration** button. Note: ECU Configuration is only supported for certain GMLAN and UDS protocols.

In the ECU Configuration dialog, click the **Select XML Config. File** button. Click the **Select Build Data File** button. Click the **Execute** button. The results of the XML Configuration event are recorded.



ECU Configuration (XML): (continued)

3. Select the XML Config File button to pick the XML file.
4. Note: Selected XML Config file will be displayed.
5. Select the XML Build Data File button to pick the Build file.
6. Note: Selected XML Build Record data file will be displayed.
7. Select the "Execute" button..
8. Note: Results of the XML Config. event are recorded.
9. Note: The 'comm' messages to/from the vehicle are captured in the previous dialog (Vehicle Comm. Data Log).

ECU Configuration

Configuration Selections

"Execute" button:
The XML Configuration file and Build Data (RPO) file must be selected prior to executing an ECU Configuration Event.

"Create Build Data File" button:
The XML Configuration file must be selected in order to create a Build Data file.

Select XML Config. File

XML File:
C:\DPS\Config\XMLFile.xml

Select Build Data File

Build Data File:
C:\DPS\Config\BuildRecord.txt

Options

Simulation Mode (No Communications at Run-Time)

Select Build Data File (XML Format)

Environment Selection:

Service Only

Manufacturing Only

Create Build Data File (.TXT)

Configuration Data Display

Item Name	Type	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value 9	Value 10	Value 11	Value 12
- Adaptive_Cruise_Control_ACC	F	2	3	1	00								
- Semi_Active_Damping_System_SADS	T	2	6	1	00								
- Rear_Drive_Control_Module_RDCM	T	2	7	1	00								
- Supervisory_Control_Module_SCM	F	3	0	1	00								
- Electric_Limited_Slip_Differen..	F	3	1	1	00								
- Active_Roll_Control_ARC	F	3	3	1	00								
- Active_Front_Steering_AFS	F	3	5	1	00								
- Pinion_Angle_Sensor_PAS	F	3	6	1	00								
- _DID_BF_Subnet_Config_List_Cha..	F	0	7	8	None								Using D

- ECU Configuration Complete

```

09:55:24.3<[.H..] 00 00 01 01 FE 01 20 00 00 00 00 00 [0012] FramePad
09:55:24.3>[.H..] 00 00 01 01 FE 01 20 00 00 00 00 00 [0012] TxMsgType
09:55:24.3>[.H..] 00 00 07 E8 01 60 [0006]
09:55:24.3>[.H..] 00 00 06 58 01 60 [0006]
09:55:24.3>[.H..] 00 00 06 42 01 60 [0006]
09:55:24.3>[.H..] 00 00 06 48 01 60 [0006]
09:55:24.3>[.H..] 00 00 06 49 01 60 [0006]
09:55:24.3>[.H..] 00 00 07 E8 01 60 [0006]
09:55:24.3>[.H..] 00 00 06 58 01 60 [0006]
09:55:24.3>[.H..] 00 00 06 42 01 60 [0006]
09:55:24.4>[.H..] 00 00 06 48 01 60 [0006]
09:55:24.4>[.H..] 00 00 06 49 01 60 [0006]
09:55:24.4>[.H..] 00 00 07 E8 01 60 [0006]
09:55:24.4>[.H..] 00 00 06 58 01 60 [0006]
09:55:24.4>[.H..] 00 00 06 42 01 60 [0006]
09:55:24.4>[.H..] 00 00 06 48 01 60 [0006]
09:55:24.4>[.H..] 00 00 06 48 01 60 [0006]
    
```

Clear Display Save Data to .TXT File

Execute Exit

Create Build Data Record File

Users can build their own custom build data file based on the criteria supported in the selected XML Configuration file (RPOs and Model Designator).

**ECU Configuration (XML):
"Create an XML Data Record File"**

** XML file must be first be selected in order to perform this feature.

- A. Select "Create..." button
- B. Note: Build dialog will display.
- C. Enter or Select VIN.
- D. Select Model Designator (one).
- E. Select RPOs (one or more).
- F. Select "Write RPO..." button.
- G. Note: Output File Text data is updated based on the user selections.
- H. Note: Output File name where the data is written (this name can be changed by the user).
- I. Select "Exit" to close dialog.
- J. Note: the Build data file name information is automatically populated for "Build Data"

ECU Configuration - Build Data File Creation

VIN: 1G1YL3D76E5127633

RPOs for: EBCM Part Number: 00CB9E50

Model Designator: <select one>

- 0GA00
- 4GA00
- EGA00
- FGA00

RPO Selections: <select one or more>

- EF7: COUNTRY UNITED STATES OF AMERICA (USA)
- F45: CHASSIS CONTINUOUSLY VARIABLE REAL TIME DAMPING
- F46: CHASSIS ALL WHEEL DRIVE (AWD)
- FX3: RIDE AND HANDLING AUTOMATIC ELECTRONIC CONTROLLED
- G96: AXLE POSITRACTION LIMITED SLIP, ELECTRONIC
- J71: BRAKE PARKING POWER OPERATED
- KB5: CONTROL ECM GRADE BRAKING
- LBW: ENGINE GAS, 6 CYL, 2.8L, SFI, V6, TURBO LOW, ALUM, GM
- LDK: ENGINE GAS, 4 CYL, 2.0L, DI, DOHC, TURBO-HO, 158 KW VARIABLE
- MM1: MERCHANDISED TRANS AUTO EQUIPMENT
- MM3: MERCHANDISED TRANS MANUAL EQUIPMENT
- TR7: HEADLAMPS CONTROL LEVELING SYSTEM, AUTOMATIC
- TT2: HEADLAMPS DIRECTIONAL, HIGH INTENSITY DISCHARGE, LOW BE
- TT6: HEADLAMPS HIGH INTENSITY DISCHARGE
- UE1: COMMUNICATION SYSTEM VEHICLE, G.P.S. 1
- UJM: TIRE PRESS INDICATOR MANUAL LEARN
- UJN: TIRE PRESS INDICATOR AUTO LEARN
- Z49: COUNTRY CANADA

Build Data Output Text File Data:

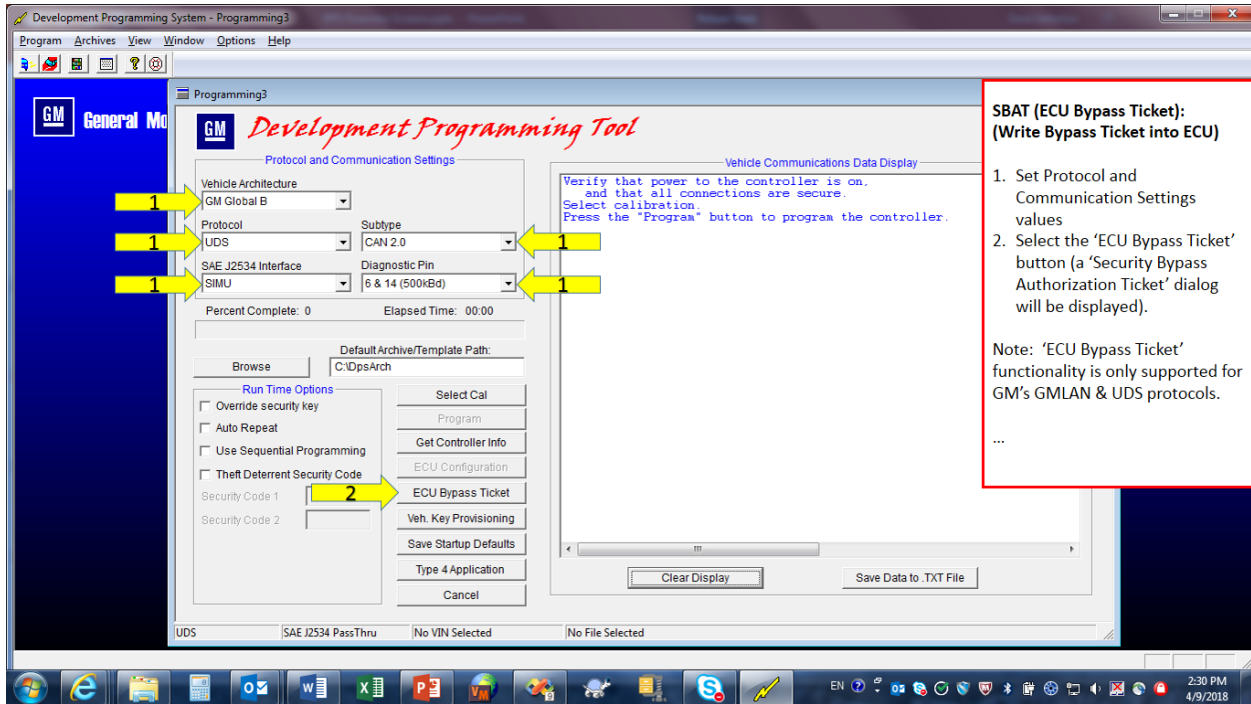
C:\DPS\Config\XMLFile.txt

1G1YL3D76E5127633
EGA00
FX3
KB5
LDK
TT2
UE1
UJN
Z49

Reset RPO Selections Write RPO Build Data Exit

ECU Bypass Ticket

To write an ECU Bypass Ticket, start by clicking the **ECU Bypass Ticket** button. Note: This functionality is only supported for GMLAN & GM UDS. Type in the ECU Device ID. Click the **Select SBAT File** button. Click the **Execute** button. Note: The results are displayed in the SBAT Data Display and can be saved to a .txt file. Binary SBAT files are obtained from the GM Cyber SBAT ticket server, which is independent of DPS.

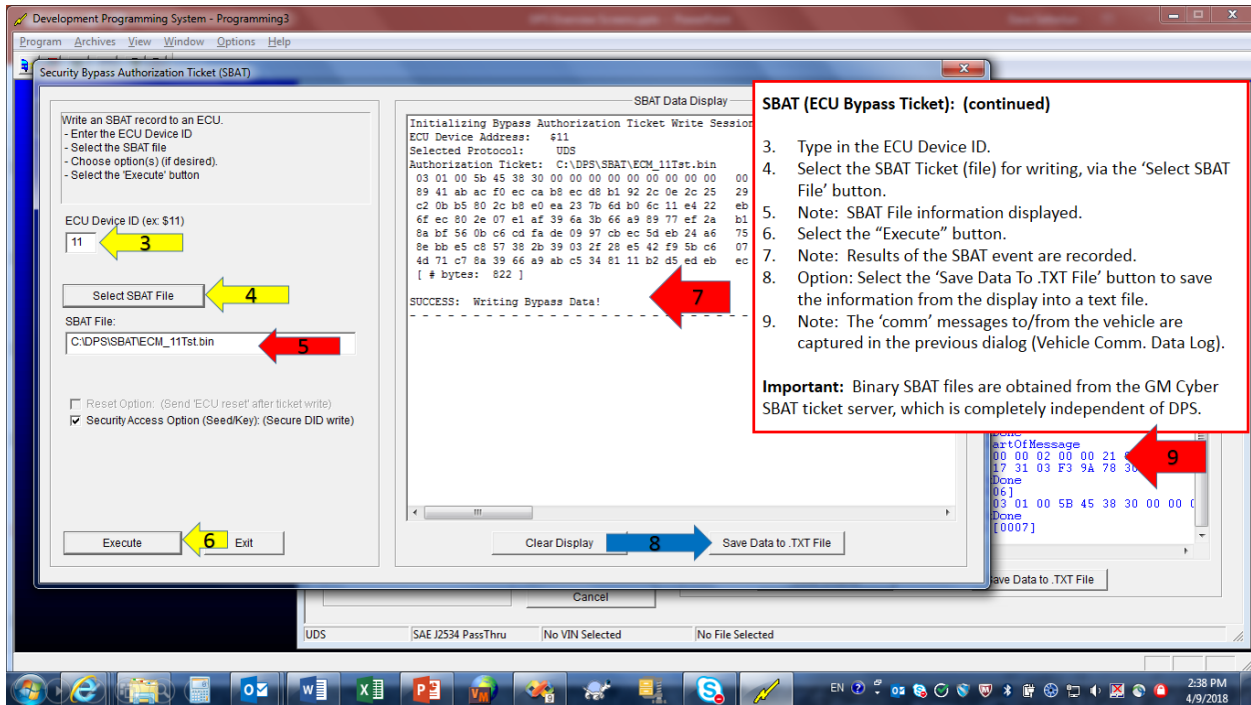


**SBAT (ECU Bypass Ticket):
(Write Bypass Ticket into ECU)**

1. Set Protocol and Communication Settings values
2. Select the 'ECU Bypass Ticket' button (a 'Security Bypass Authorization Ticket' dialog will be displayed).

Note: 'ECU Bypass Ticket' functionality is only supported for GM's GMLAN & UDS protocols.

...



SBAT (ECU Bypass Ticket): (continued)

3. Type in the ECU Device ID.
4. Select the SBAT Ticket (file) for writing, via the 'Select SBAT File' button.
5. Note: SBAT File information displayed.
6. Select the "Execute" button.
7. Note: Results of the SBAT event are recorded.
8. Option: Select the 'Save Data To .TXT File' button to save the information from the display into a text file.
9. Note: The 'comm' messages to/from the vehicle are captured in the previous dialog (Vehicle Comm. Data Log).

Important: Binary SBAT files are obtained from the GM Cyber SBAT ticket server, which is completely independent of DPS.

Selecting a VIN for use during Programming Event

After selecting the communication settings and choosing the programming archive (just prior to selecting the “Program” button), select the “Get Controller Info” button and the ECU Data dialog will display, then either type in the 17-digit VIN that is needed (VIN edit field) or select the VIN from a pull-down list of the previously used VINs. Then select the ‘OK’ button from this dialog. The VIN will be displayed on the programming dialogs status bar.

ECU Data - GMLAN
✕

Request Method Select...

ECU Address/ID \$

VIN (\$90)

System Name or Engine Type (\$97) ^

System Supplier Id (\$92) v

Base Model Number (\$CC)

End Model Number (\$CB)

Broadcast Code (\$B5)

Supplier SW Version Number (\$95)

Diagnostic Data Identifier (\$9A)

Repair Shop Code or SN (\$98)

Programming Date (\$99)

Manufacturers Enable Counter (\$A0)

Programming Status

Partial VIN (\$28)

History: RSCOSN (\$9F)

Mfg Traceability Chars (\$B4)

Engine Serial Number

Software Module Identifier

DID	Module ID

Security Table \$ v

Security Algorithm \$

Seed	Key

Secure Programming Data

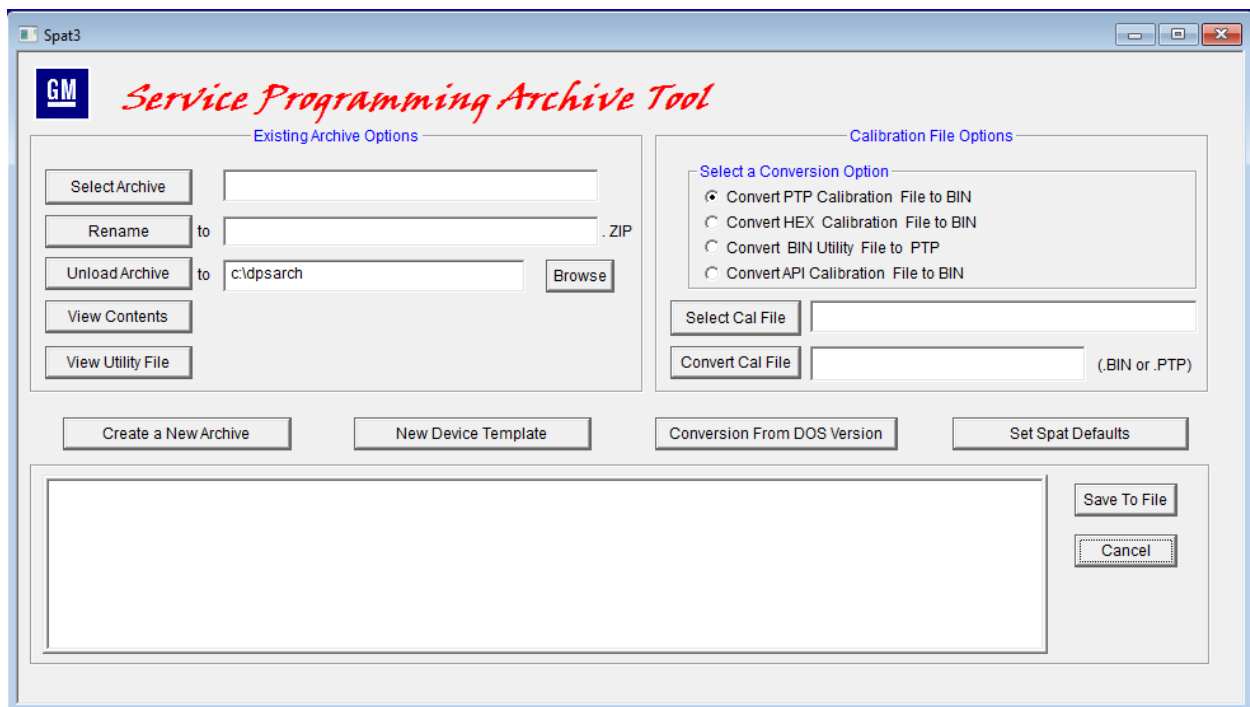
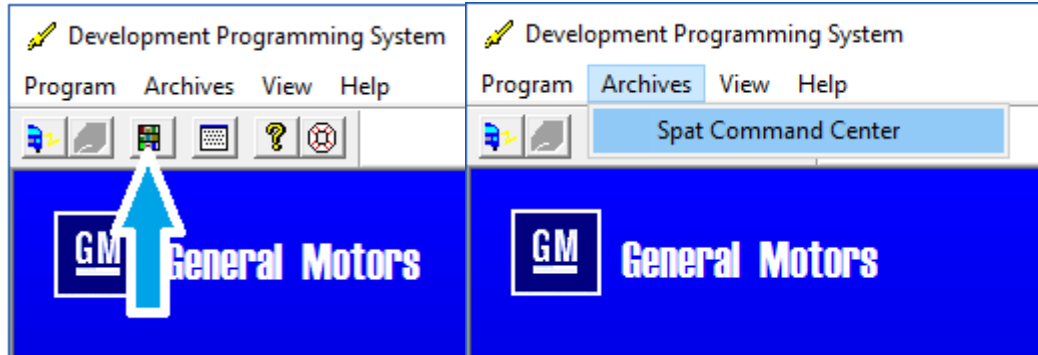
Desc	Data

Read Info
OK
Cancel
Save Text

Service Programming Archive Tool

The SPAT Dialog supports functionality with respect to:

- Viewing the contents of an existing Archive file (zip file)
- Viewing the Utility File portion of an existing Archive file
- Converting Files from type PTP, HEX, API to binary, ...
- Creating a new Programming Archives (zip files)



View Contents of an Archive

To view the contents of an archive, click the **Select Archive** button. Then click the **View Contents** button. The content of the archive will be shown at the bottom. Users can click the **Save To File** button to save the display printout.

SPAT – View Contents:
(of an archive – zip file)

1. Select the 'Select Archive' button to choose an archive
2. Note: The chosen archive is displayed.
3. Select the 'View Contents' button
4. Note: The contents of the programming archive will be displayed (display section).
5. Option: Select the 'Save To File' button to save the information from the display section into a text file.

Archive Signature: DAUE
 SetTermin - always Program Everything - Do not skip OpSw
 AND - Set STMN to 01 (ms) OpCode \$04 1Sep2015

CALIBRATION	MODID	LENGTH	FMT	DESCRIPTION
13578486ds2.bin	00	0003002	BIN	Programming Control File
13587440.bin	01	0405248	BIN	operating system
123122007.bin	02	0012002	BIN	System

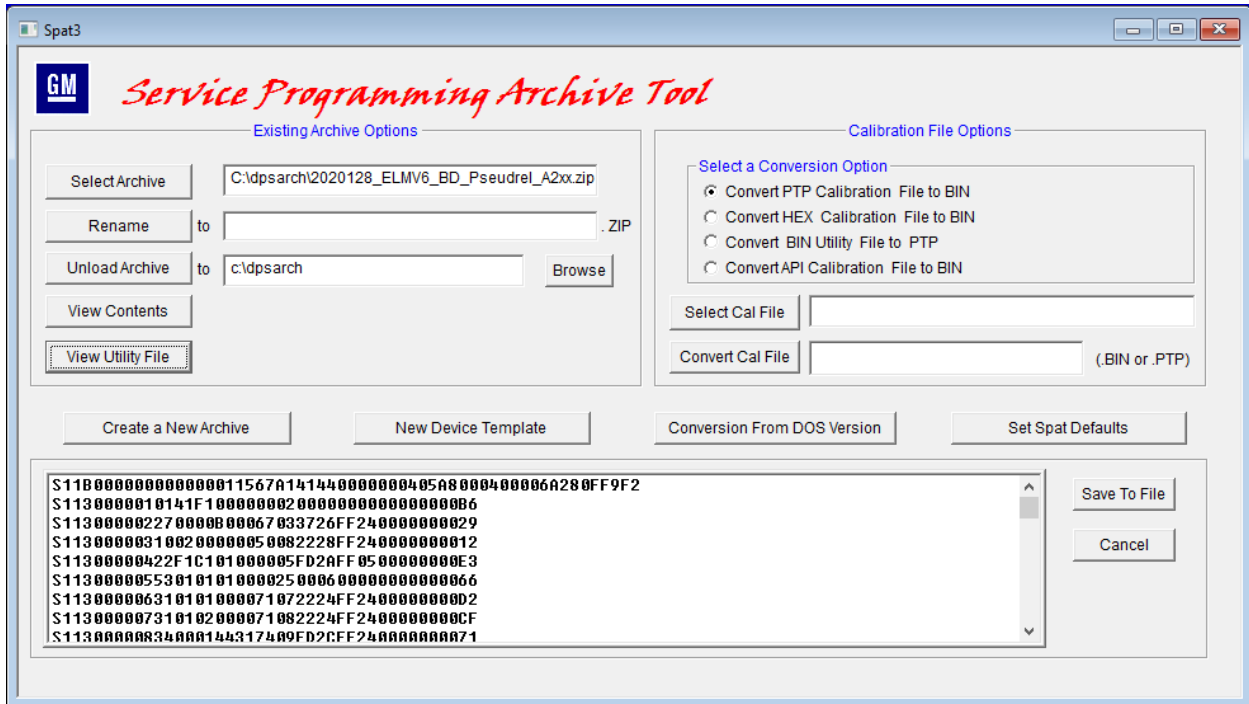
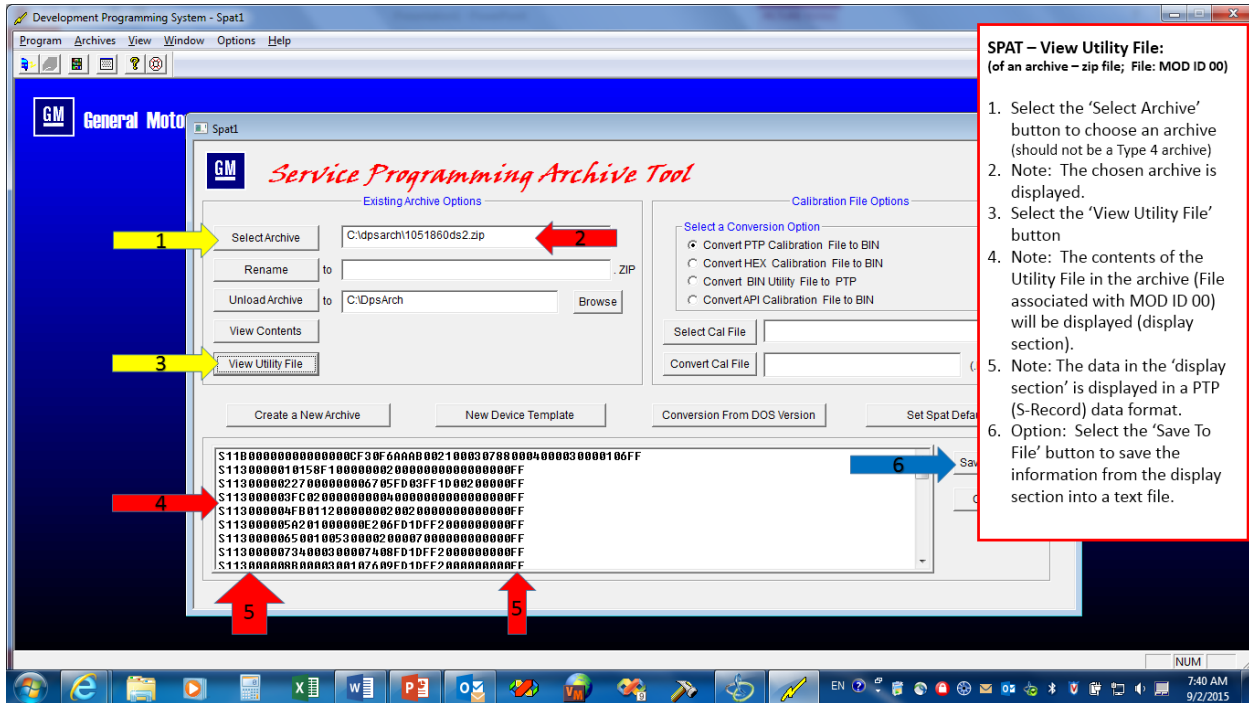
Archive Comments

CALIBRATION	MODID	LENGTH	FMT	DESCRIPTION
18180001.AD	00	0009238	BIN	Utility File
13509112.bin	01	0492081	BIN	Description of Cal
bsp1.bin	02	0002912	BIN	Description of Cal
hso2.bin	03	0002912	BIN	Description of Cal

View Utility File

To view the utility file, click the **Select Archive** button. Then click the **View Utility File** button. The content of the Utility File will be shown at the bottom. Users can click the **Save To File** button to save the display printout.

Note: The data in the display section is displayed in a PTP (S-Record) data format.



Convert Files

To convert files, the user need only use the Calibration File Options section of the dialog. To convert a file, start by selecting a conversion option. Click **Select Cal File** to select the actual file to get converted (PTP, BIN, HEX, or API). Type in the name of the new file next to the **Convert Cal File** button. Click **Convert Cal File** to generate the new file.

Note: The conversion status (success/failure) and the output file information are displayed in the display section. As always, the use can click save to file to save the data display.

SPAT – File Conversion:

1. Select one of the file conversion options for the File Conversion.
2. Select the actual file to get converted (PTP, BIN, HEX, or API) – ‘Select Cal File’ button.
3. Note: The chosen file is displayed.
4. Type in the name (without the path information) into the ‘output’ filename field.
5. Select the ‘Convert Cal File’ button
6. Note: The conversion status (success/failure) and the output file information will be displayed.
7. Option: Select the ‘Save To File’ button to save the information from the display section into a text file.

The screenshot shows the SPAT2 interface with the following details:

- Existing Archive Options:** Select Archive, Rename to, Unload Archive to (C:\DpsArch), View Contents, View Utility File.
- Calibration File Options:**
 - Select a Conversion Option:
 - Convert PTP Calibration File to BIN
 - Convert HEX Calibration File to BIN
 - Convert BIN Utility File to PTP
 - Convert API Calibration File to BIN
 - Select Cal File: C:\NewPTI\24242756.pti
 - Convert Cal File: 23232756
- Display Section:**

```
>> Converting API File -> C:\NewPTI\24242756.pti
to Binary File -> C:\DpsArch\23232756.bin
>> The File Conversion was Successful !
```
- Buttons:** Create a New Archive, New Device Template, Conversion From DOS Version, Set Spat Defaults, Save To File, Cancel.

The second screenshot shows the same interface after conversion:

- Calibration File Options:**
 - Select a Conversion Option:
 - Convert PTP Calibration File to BIN
 - Convert HEX Calibration File to BIN
 - Convert BIN Utility File to PTP
 - Convert API Calibration File to BIN
 - Select Cal File: C:\PTI\12616887.pti
 - Convert Cal File: new_file_name (BIN or .PTP)
- Display Section:**

```
>> Converting API File -> C:\PTI\12616887.pti
to Binary File -> c:\dpsarch\new_file_name.bin
>> The File Conversion was Successful !
```
- Buttons:** Save To File, Cancel.

Create New Archive

To create a new archive, start by clicking the **Create a New Archive** button in the SPAT dialog. This will open a new SPAT dialog. In the new dialog, type in the name of the new file without an extension (32 characters). Click on the **Template** button to select a predefined device template to use for creating the archive. The number of parts to the archive is displayed based on the template. Type in any comments. There are four lines of comments. Each comment can hold up to 77 characters. Click the **Next** button to open a new dialog and proceed onto the next step.

SPAT – Create New Archive:

1. Select the 'Create a New Archive' button from the main SPAT dialog
2. Note: A SPAT 'new archive' dialog will open.
3. Type in the new Archive name without any extension (max = 32 chars)
4. Select the 'Template' button to choose a predefined Device Template to use for creating the archive
5. Note: After a template is chosen, the number of parts to the archive is displayed (including the utility file)
6. Type in any comments – Optional (Four lines x 77 chars each)
7. Select the 'Next' button


Note: 'Select Base Archive' can be selected after Step 4 when many of the data elements for the 'new archive' are identical to an existing archive and that existing archive is already built (zip file)

SPAT – Create New Archive: (continued)

8. Note: The 'Build Calibration Archive/Template File' dialog will pop up and display. The number of ROWs displayed is directly related to the device template that was chosen on the previous dialog. For ex: CalFil02 is 2 calibration files PLUS one utility file – thus 3 rows.
9. (Optional) Update one or more of the 'Mod ID' fields (if so desired)
10. (Optional) Update one or more of the Description fields (if so desired)
11. For each and every column, click in the Utility/Calibration File column -> a 'Select File' button will be displayed. Click on this button to select the applicable file (from the hard drive) for building the archive.
12. When all of the calibration files have been chosen, the select the 'Build' button. FYI: The 'success' or 'failure' of the archive build will be displayed.

Verify the MOD_ID and Description fields, then select a file for each Utility/Calibration File entry. When all entries have been completed, press the BUILD button to complete the building of the archive. Press Cancel to return.

Service Programming Archive Tool(SPAT)

 *Service Programming Archive Tool*

New Archive Options

New Archive Name: .ZIP (Build a .ZIP archive)
 .TM (Build a .TM template)

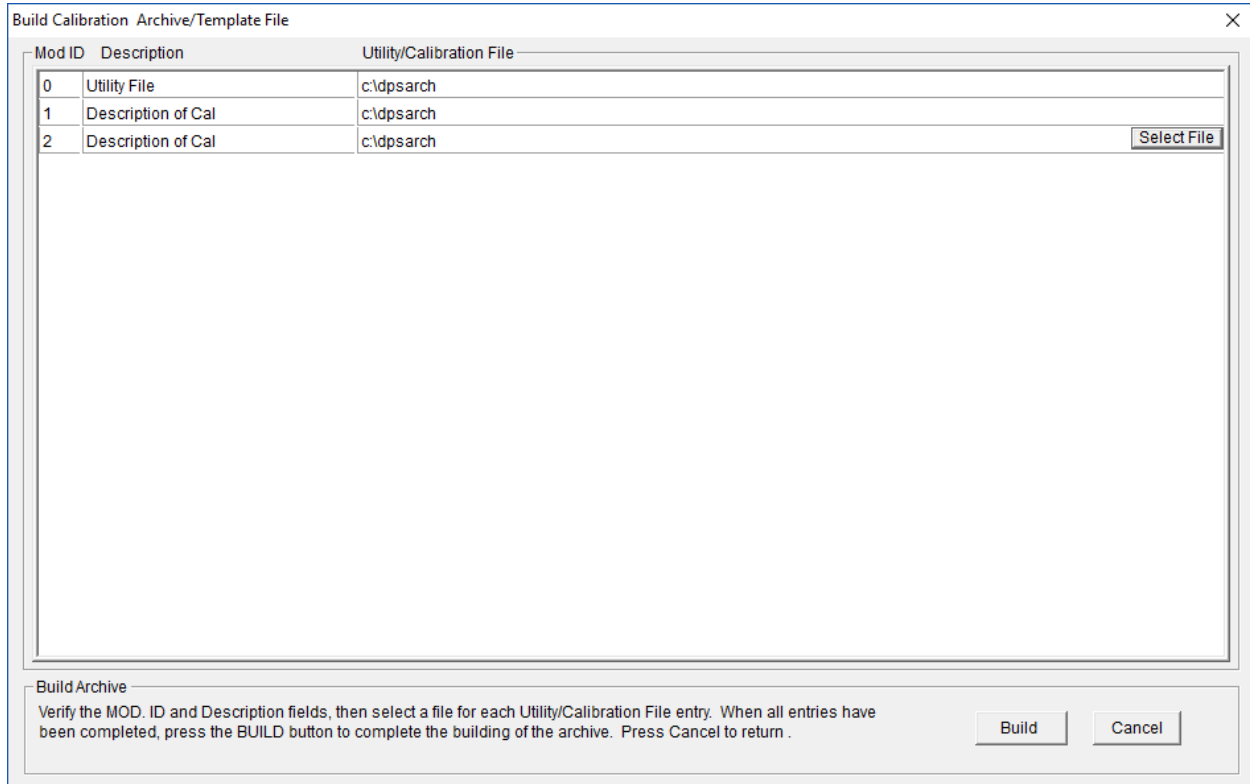
Folder Where New Archive Will Be Stored:

Select an Archive Layout Template (.DTM Extension)

Number of Files to be used in the build of this archive (including Utility File):

Archive Comments

The Build Calibration Archive/Template File dialog shows the user the files going into the archive. The user can update the Mod ID and Description fields if desired. For each row, click in the Utility/Calibration File column to display a Select File button. Click the **Select File** button to select the applicable file for building the archive. When all files are selected, click the **Build** button. The status of the build process will display.



Appendix A: Other Specialized DPS Functions

Supersession Programming (SST = Supersession Table)

Used mostly for 'plant spill' type of events where a bunch of SIMILAR vehicles need to get updated. How this works is that the SST event will first read the Part Number information out of an ECU (which tell the application which software calibrations are in the ECU) and then based on the SST file contents, the replacement calibration parts are identified for the programming event. Kind of like dynamically building a programming archive (piece by piece) based on the data currently in the ECU.

BATCH Supersession Programming

This truly is a 'batch' file where multiple SST files (see above) are executed sequentially. Used when a bunch of SIMILAR vehicles need to have MANY ECUs updated at a single time.

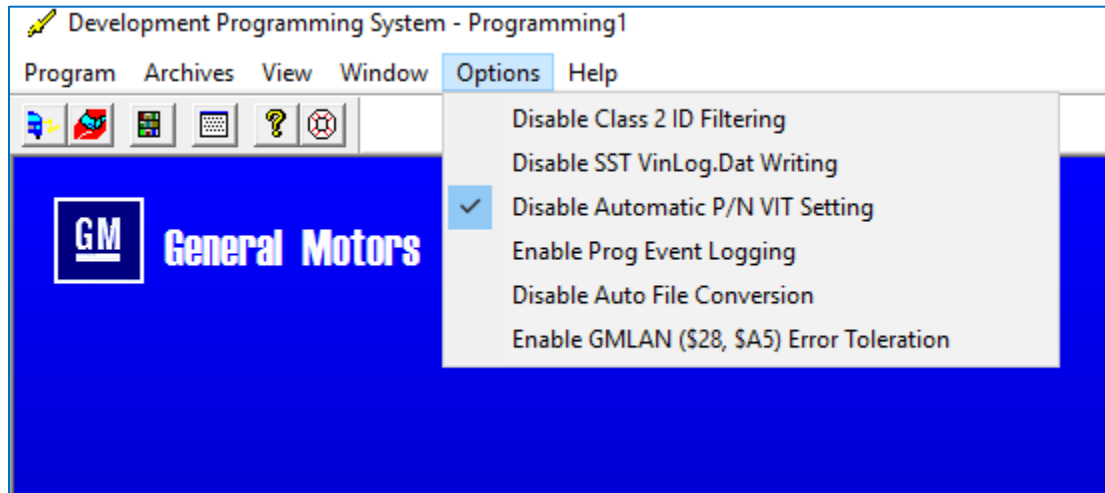
Sequential Programming

Used when someone wants to update multiple ECUs (programming) in one shot without having to pick for example - ECM_Archive.zip, then BCM_Archive.zip, then TCM_Archive.zip - they can list these archives in a file (much like a batch file) and then just pick ONE sequence file and program multiple ECU events at a single time sequentially.

Vehicle Key Provisioning (Global B/GEM only)

Vehicle Key Provisioning is the assignment of unique 'message authentication' security keys to the ECUs on a vehicle which participate in Secure Message Authentication (secure normal mode messaging). Key Provisioning is required to take place across the whole vehicle (via the gateway module).

Appendix B: Other DPS “Option” Functionality



Disable Class 2 ID Filtering

Class 2 protocol communication messages are filtered based on the ECU ID information within the message. This option is used to enable/disable this filter mechanism. This is typically used for non-compliant ECUs. The default setting upon starting up DPS is not checked.

Disable SST VinLog.Dat Writing

Supersession Table (SST) programming events will write a supersession data record to a log file to record the event. This option is used to enable/disable that reporting mechanism. The default setting upon starting up DPS is enabled not checked.

Disable GMLAN Automatic P/N VIT Setting

Programming events (utility file logic) can reference the Vehicle Information Table (VIT) data structure to determine if individual calibration downloads can be skipped. This option allows the user to enable/disable the population (setting) of the VIT structure. The default setting upon starting up DPS is checked. The option is explained in more detail in another section of this course.

Enable Prog Event Logging

The information from a programming event can also be captured (into a '.dat' file). The type of information recorded is success/fail, VIN, date/time and programming archive name. This option allows the user to enable/disable this logging. The default setting upon starting up DPS is not checked.

Disable Auto File Conversion

When archives are built using the DPS SPAT functionality, the input files are automatically converted to a binary format during the creation of the archive 'zip' file. The known formats that are converted are: PTP (S-record), API (.pti), and Hex. This option allows the user to create a programming archive without performing this file conversion. The default setting upon starting up DPS is not checked.

Enable GMLAN (\$28, \$A5) Error Toleration

Some ECUs are non-compliant to the GM specification for GMLAN Service \$28 and/or Service \$A5. When these ECUs are non-compliant, then programming will always fail. This option allows the user to ignore these non-compliances so that programming can be executed. The default setting upon starting up DPS is not checked.

Appendix C: What is “Populate VIT”

Populate VIT (Vehicle Information Table)

The “Populate VIT” functionality in DPS places the “to be” programming state information (Part Number / Module ID) into an internal VIT data structure. The ‘Automatic’ population of the VIT will automatically take this information from the programming archive; while a ‘manual’ population of the VIT allows the user to define this data manually.

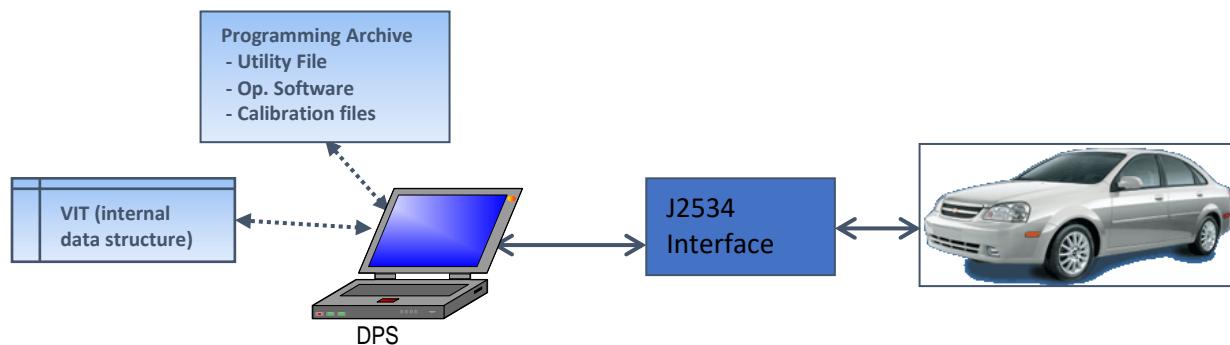
01	22182143
02	22182144
03	22182145
07	22182210
08	22182212
--	--
--	--
--	--

VIT: Populated with Data

--	--
--	--
--	--
--	--
--	--
--	--
--	--
--	--

VIT: Un-Populated (blank)

When a programming event is executed, the programming logic flow is determined by the Operation Code instructions that are written within the Utility File (the Utility File is contained within the Programming Archive). The utility file may include instructions that reference the data in the VIT data structure; if it does, then the programming event may execute differently based on the data populated within the VIT. (for example: Utility files typically are written so that the programming of the Op. Software can be bypassed if it is determined that the vehicle already contains this ‘new’ version of Op. Software. This greatly reduces programming time.)



Appendix D: What is a Utility File

A Utility File is a binary file that contains a list of step by step instructions on how to reprogram a control module. Utility Files were developed to keep the proliferation of tool reprogramming software variants to a minimum. The Utility File is viewed as three distinct sections: the header information (24 bytes), the interpreter instructions (size varies) and the device specific programming routines (a.k.a. routine data). Even though the Utility file is viewed as three distinct parts, it will always be treated as a single entity necessary for reprogramming a Control Module.

The Interpreter Concept used in a Utility File allows reprogramming support of new products, without having to hard code or create independent software packages. An Interpreter is a module that understands the format of Utility Files as well as the use of each of its Op-Codes. The Interpreter follows the Interpreter Instructions in the Utility Files until an exit point is reached. There are two Op-Codes that will end the programming event:

If the 'EE' Op-Code (End with Error) is executed by the interpreter, "Programming has Failed".

If the 'FF' Op-Code (End with Success) is executed by the interpreter, "Programming was Successful".

The Utility File is one contiguous block of data consisting of three distinct sections as depicted below:

Header Information (24 bytes - Used to setup the programming session)	The Header Information defines information that remains constant during the entire reprogramming event. An example of this information is the "Type of Interpreter", once the software starts using an Interpreter it will not change to an Interpreter using another communications protocol.
Interpreter Instructions (A set of sequentially numbered step to control the programming session)	The Interpreter Instructions are the Op-Codes that guide the terminal application through a reprogramming event. Each instruction line is 16 bytes long and consists of four sections: 1-byte Step Number, 1-byte Op-Code, 4-byte Action Field, and 10 bytes of goto fields.
ECU Specific Control Routines and Service Request Data Routine	The Device Specific Control Routines are programming routines used for performing various functions during the reprogramming event. The number of routines varies, depending on how each ECU. Examples of control routines are: erase flash memory, turn on reprogramming voltage, or read flash manufacturer and ID. Service Request Data Routines provide a means to pass additional data in Service Request. Examples of this data are Routine Entry Options and Record Values.

Even though the Utility File is viewed as three sections, it must be handled as a single file. The routine section of the utility file is an optional section and is controller specific.

Appendix E: Program an ECU that requires a Security Code

The typical way a utility file is designed with respect to the Security Code logic is to check to see if the VIN in the ECU matches the VIT VIN via the TOOL (DPS, Tis2Web). When this VIN check (comparison) fails, then SECURITY CODE LOGIC is executed.

When Security Code logic is executed within a utility file, the Security Code Data will need to be entered into DPS; otherwise, default Security Code data of 0xFF 0xFF 0xFF 0xFF data will be used (DPS does NOT access the GM Security Code database) – the user is EXPECTED to know the SECURITY CODE when using this Development tool.

See steps and screen below.

Setting up the VIN and Security Code data for DPS:

After picking the Protocol, Interface, Subtype and Diagnostic Pins - - Pick the Archive for programming, then.....

- 1) Pick The “Get Controller Info” button
- 2) The ECU-Data dialog will display
- 3) Pick “Read Info”
- 4) The data will be populated INCLUDING THE VIN INFORMATION
**** Note: The user can skip steps 3 & 4 and MANUALLY enter the VIN into the “VIN (\$90)” edit box.**
- 5) Pick the “OK” button (Note: ECU Data dialog will close)
- 6) The VIN INFORMATION from the “ECU Data” dialog will now be displayed on the PROGRAMMING dialog
(see arrow 6) - - this is what is used in “VIT” for comparison.
- 7) If Security CODE data needs to be entered, then the Run-Time Option “Theft Deterrent Security Code” must get checked
- 8,9) AND, then the SECURITY CODE for the VEHICLE (Code 1) and the Security Code for the ECU (Code 2) must be entered by the user.

Then attempt programming via the "Program" button

