Engineering Release Notice (ERN)	Location	Change Description A = Added W = Was U = Deleted	Document Release Status	
			Date 2022-05-12	Modification Count

Diagnostic Engineering Tool
User Manual

Revision history

Previous Version	Current Version	Version Description	Responsible	Date
N/A	001	1st Official Release	Jason Miller, Ruben Mendoza, Alfredo Martin Mendoza Trejo	2010-11-11
001	002	2nd Release	Jason Miller	2011-01-05
002	003	3rd Relaease	Jason Miller	2011-03-31
003	004	4th Release	Jason Miller	2012-02-14
004	005	5th Release	Jason Miller	2012-08-14
005	006	6th Release	Jason Miller	2012-10-09
006	006.1	Draft Release	Jason Miller	2013-01-16
006.1	006.2	Draft Release	Jason Miller	2013-03-07
006.1	006.2	Draft Release	Jason Miller	2013-07-16
006.2	006.3	Draft Release	Jason Miller	2013-10-31
006.3	006.4	Draft Release	Jason Miller	2014-05-27
006.4	006.5	Draft Release	Jason Miller	2014-07-22
006.5	006.6	Draft Release	Jason Miller	2014-09-04
006.6	006.7	Draft Release	Jason Miller	2015-11-28
006.7	006.8	Draft Release	Jason Miller	2016-05-19
006.8	006.9	Draft Release	Jason Miller	2016-10-25
006.9	006.10	Draft Release	Bill Waldeck	2019-09-xx
006.10	006.11	Draft Release	Michelle Matowski	2020-08-07
006.11	006.12	Draft Release	Michelle Matowski	2020-08-25
006.12	006.13	Draft Release	Michelle Matowski	2020-10-15
006.13	007	7th Release	Michelle Matowski	2021-03-29
007	008	8th Release	Michelle Matowski	2021-08-10
008	009	9th Release	Michelle Matowski	2021-09-27
009	010	10Th Release	Michelle Matowski	2021-12-13
010	011	11th Release	Michelle Matowski	2022-02-18
011	012	12th Release	Michelle Matowski	2022-03-16
012	013	13th Release	Michelle Matowski	2022-03-28
013	014	14th Release	Michelle Matowski	2022-05-12

Change log

Release	Section	Change Description
001	All	1st Official Release
002	10.1.6	Added details for relative timestamps based upon change in v5.2.0 of DET.
003	Annex C	Added Annex C to describe supported hardware
004	All	Updated requirements related to SAE J2534 support based upon changes in v5.8.0 of DET
004	11.1.7	Added details regarding STmin override allowance for microsecond intervals.
004	10	Added framework for new PATS Diag View (details to come in later release)
004	4.2.22	Added details for Generic Diagnostic Message 2
004	5.4	Added comment relating to ability to manually include relative path information in VBS files.
004	11.5.1	Added information about Vector hardware transmit queue size and potential for missing consecutive frames.
004	Annex D	Added a section for Frequently Asked Questions.
004.1	Annex C	Added a clarification to Kvaser install on Windows XP
004.1	Annex D	Added details regarding how to adjust display, especially for Windows 7 with Asian Language pack.
004.2	5.8	Added information to update in DET v5.10.3 to allow better support for relative paths
004.2	Annex C	Added more details to fix an issue related to Kvaser SAE J2534 driver install.
005	9	Added more details regarding DET v6.0.0 updates on DVS changes for new fields of ALWAYS_ANALYZE_ALL and VERIFY_NOMATCH.
005	Annex B	Added new DVS XML schema definition file for version 002.
006	Annex C	Added new SWDL programming time comparisons for latest hardware and drivers
006	Annex C	Updated support information for known supported hardware
006	Annex D	Added a new FAQ for supporting Sync USB programming on Windows 7.
006.1	Annex C	Added additional steps due to possible install issues with Kvaser hardware
006.1	16.7	Deleted legacy text that some PATS security levels required a password.
006.2	Annex C.2	Updated to latest programming times for newer Kvaser J2534 driver on Windows 7.
006.3	Annex E	Added new Annex to address solution for Japanese Windows 7 issue with the DET.
006.4	Annex C	Updated latest programming times for new Dearborn Group hardware
006.4	Annex C	Updated latest programming times for Bosch VCM II hardware
006.5	Annex C	Added direct link for Intrepid drivers
006.6	24	Updated command line arguments to detail support for option “-05” used to perform ECU Configuration.
006.7	Annex C	Updated EEPOD phone number
006.8	All	Added details on new command line arguments to detail support for option “-f” to load a user specified DCS file and option “-06” used to perform sending a user message sequence.
006.9	3.3.2	Updated J2534 quick connect details to remove outdated restriction that only the first installed J2534 device in the registry could be used.
006.10	3.5	Updated references to where DET can be obtained and where requests for access may be sent.
006.11	All	Updated many form graphics to reflect current DET implementation.
006.11	3.3.2	Added a graphic showing the drop down menu for Quick Connects.
006.11	4.1-4.7	Added sections for services 0x01, 0x03, 0x04, 0x06, 0x07, 0x09, 0x0A

Release	Section	Change Description
006.11	4.2.29	Added a picture for Generic Diagnostic Message 2
006.11	1	Updated Table of Contents
006.11	All	Replaced occurrences of “click” to “select” to account for users using touchscreens rather than a mouse.
006.11	5	Added descriptions for the new buttons in SWDL.
006.11	11.2	DCS, KEY, ECD, and VBS files will not be saved in CLI Mode.
006.11	24.3	Added a new section for using CLI Mode file placement
006.11	24.4	Added new CLI errors
006.11	D.2	Added information on how to resize controls. Removed information on the GUI Tweak as it has been removed.
006.11	D.5	Added the J2534 Device Troubleshooting Guide and System Information Report to the section regarding J2534 devices not being recognized.
006.11	12.7.2	Added the J2534 Device Troubleshooting Guide
006.11	12.7.4	Added the System Information Log
006.11	12.3	Added a V or I to the status bar to display the currently selected DLL in the DLL swap.
006.12	5.9	Added section for programming subnodes
006.13	Annex C	Added clarification that DET supports J2534 04.04 with Windows 32 bit
006.13	17.1	Added clarification of what ECU’s will be detected based on FD-CAN settings in Initialize Hardware or Quick Connects.
007	17.1	Removed Section 17.1 addition for Release 006.13 and replaced it with the instructions to use the option to detect all ECU’s when using a CAN-FD device.
007.1	4.2.31	Added clarification that the Generic CAN Frame is cannot be used for framed greater than 8 bytes.
007.1	C.1.2	Added ValueCAN 3 has reached end of life, and hardware should be updated.
007.1	D.9	Removed sections as it pertained to Windows 7 and DET does not support Windows 7
007.1	D.9	Added new section on S19 and hex files.
007.1	Annex E	Removed as it pertained to Windows 7 and DET does not support Windows 7. GUI has also been fixed with resizable controls and a standard Windows 10 font.
010	17.1.1	Added a note about vehicle architectures with multiple OBD connectors
010	18.1.1	Added a note about vehicle architectures with multiple OBD connectors
011	D2	Removed paragraph about Windows 7 and fonts. Updated the picture.
011	4.2.20	Added notes about the DataIdentifier combo box being loaded and when 0x2F works at < 2mph and not at > 2mph.
012	4.2.12	Added early implementation of service 0x19 for J1979-2
013	2.3	Added info about Apache Log4j Vulnerability
014	All	Unkind words have been replaced
014	21	Clarified that the Periodic Frame Tool is limited to a TX_DL of 8 bytes.

1 Table of Contents

2 Introduction

From a high level perspective, the Diagnostic Engineering Tool is a tool designed to allow individuals to perform the following diagnostic tasks on ECU's compliant with Ford's ISO 14229 based diagnostic specs:

- Transmit, receive and decode diagnostic messages (DIDs, DTCs, etc.).
- Download software to ECUs.
- Perform Direct ECU Configuration.
- Monitor and log CAN traffic, including CAN FD.
- Transmit non diagnostic CAN messages.
- Query all networks to perform user defined tasks (e.g., reading all hardware / software part info).
- Upload and save information from an ECU.
- Generic DTC Tracker functionality.
- Miscellaneous tasks for diagnostics, using the specific tools designed for each task (e.g., Unlock ECUs, Periodic Diagnostic Tool, Hot Key Messages, APIM USB programming, etc.).

2.1 Purpose/Scope

The purpose of this document is to provide guidance to all users of the Diagnostic Engineering Tool (Applications Engineers, D&R Engineers, Systems Engineers, Suppliers, etc.). The document contains specific information on the content of this tool, how to use it, how to set it up, and some examples for the different tools within DET.

Note that updates are made to the DET on an ad-hoc basis and the details of these updates are not always added promptly to this user manual.

2.2 Abbreviations/Acronyms

The following abbreviations are used throughout this specification:

CAN:	**C**ontroller **A**rea **N**etwork
DID:	**D**ata**Id**entifer
DCS:	**D**iagnostic **C**onfiguration **S**ettings
DET:	**D**iagnostic **E**ngineering **T**ool
DL:	**D**ownload
DLC:	**D**ata **L**ink **C**onnector (SAE J1962)
DTC:	**D**iagnostic **T**rouble **C**ode
ECU:	**E**lectronic **C**ontrol **U**nit
EESE:	**E**lectrical/**E**lectronic **S**ystems **E**ngineering
EOL:	**E**nd **O**f **L**ine
FMC:	**F**ord **M**otor **C**ompany
ISO:	**I**nternational **S**tandards **O**rganisation
GMRDB:	**G**lobal **M**aster **R**eference **D**ata **B**ase
NRC:	**N**egative **R**esponse **C**ode
OBD:	**O**n**b**oard **D**iagnostics (legislated requirements, e.g. OBD II, EOBD and JOBD)
SAE:	**S**ociety of **A**utomotive **E**ngineers
SBL:	**S**econdary **B**ootloader
SPRMIB:	**S**uppress**P**ositive**R**esponse**M**essage**I**ndication**B**it
SW:	**S**oftware
SWDL:	**S**oftware **D**ownload
UL:	**U**pload
UUDT:	**U**nsegmented **U**nacknowledged **D**ata **T**ransfer
VBF:	**V**ersatile **B**inary **F**ormat
VBS:	**VBF** **S**equence

2.3 Apache Log4j Vulnerability Statement

You may be aware of a security flaw in a common software library called Log4j that could affect large portions of the internet. Also known as CVE-2021-44228, the vulnerability is specific to Java-based services.

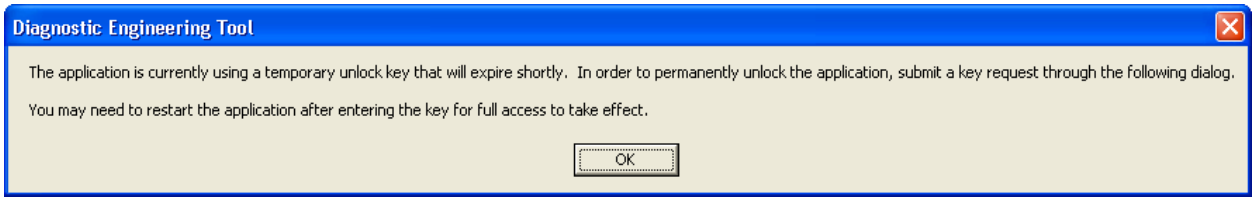
DET has been evaluated and it has been determined there is no vulnerability, in any version of DET, to the Log4j issue because neither DET nor any of its associated dll files utilize the Log4j software library.

3 Getting Started

To get started, the first thing to do is to become familiar with the interface of the Diagnostic Engineering Tool (DET). An overview of DET's sections is described in the following sections.

3.1 Submitting Unlock Key for DET

Starting with version 5.0.0, DET will require an unlock key to gain the full access to its tools and features. After installing it, DET will display the following message:



After selecting OK on the previous message, a new pop-up window will open, requesting the user to email a personalized code to Ford, so that Ford can generate a corresponding personalized Key and send it back to the user. The unlock key is personalized to the user and computer from which the request came.

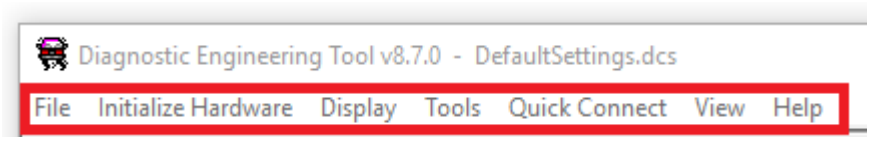
After performing this, DET will now have all the features it is built with. Note that this is a one time task and does not need to be done each time a new version is installed. The primary usage of this is for better tracking and support of the user community and to reduce the chance of individuals without a legitimate Ford business case from utilizing the functionality of the tool.

NOTE: The personalized DET unlock keys are version independent and do not expire. However, each DET version is set up to expire and no longer run one year after the release date. When submitting a personalized code or entering the Ford provided unlock key, it is critical that the **entire** string is copied and pasted. Screen shots are not acceptable and hand-typed requests often result in errors (e.g., capital i's, lower case L's, and number 1's all look alike).

3.2 DET Sections and Views

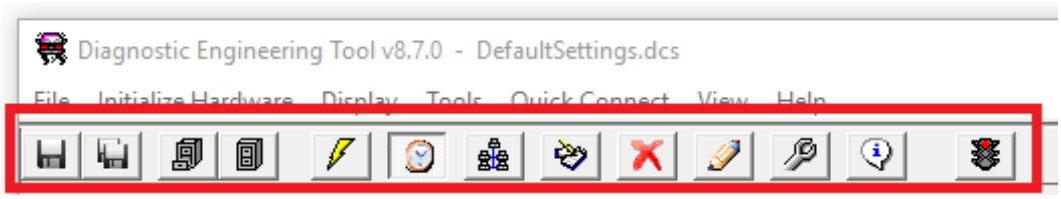
The DET contains menu options, a toolbar, and a view selector. Selecting the correct view is critical in order for the correct interface to be displayed to the user. Currently there are 6 different views as described in section 3.2.3. The view can be changed either by selecting the View menu option, or by selecting the view from the combo box near the right of the toolbar.

3.2.1 Menu Options



For more information on the content of the Menu refer to section 12.

3.2.2 Toolbar

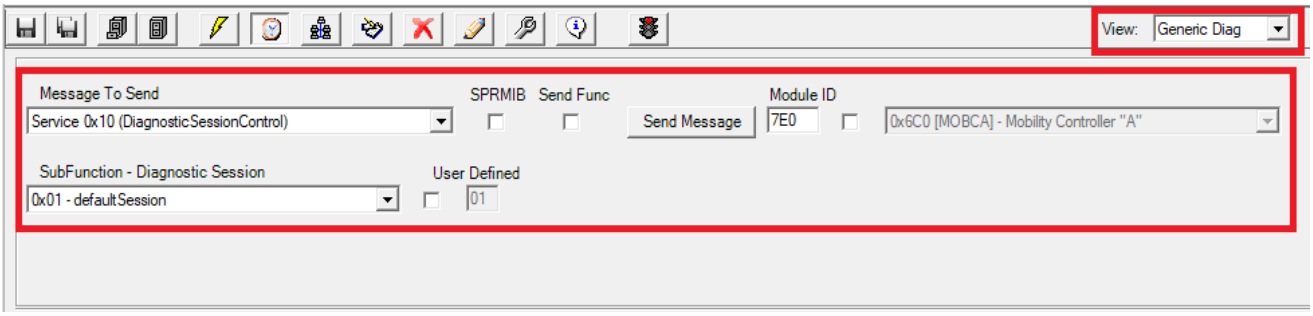


For more information on the content of the Toolbar refer to section 13.

3.2.3 Views

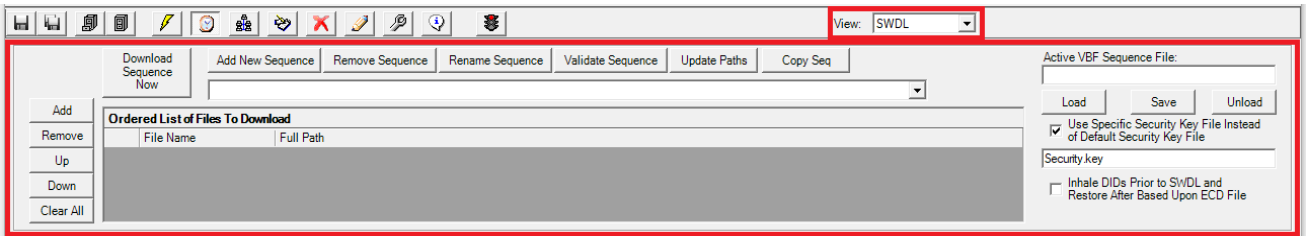
3.2.3.1 Generic Diag View

This view allows the user to send diagnostic messages to the ECU (change Diagnostic Session, read DIDs, unlock ECU, etc.). The messages can be sent with drop boxes, can be edited manually by the user or can be sent by pressing configurable buttons. Refer to section 4 to learn more about it.



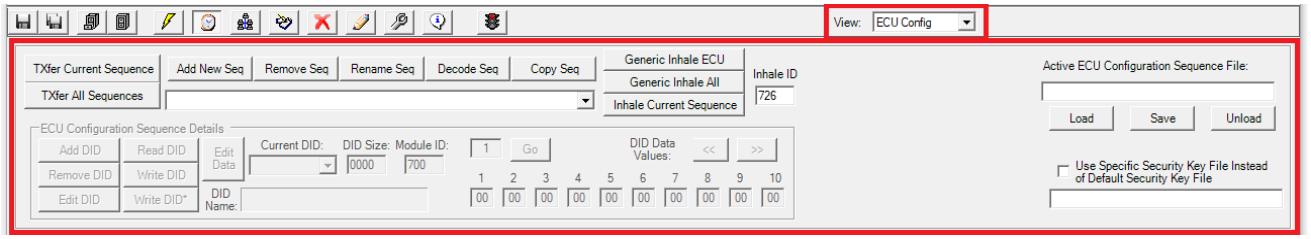
3.2.3.2 SWDL View

This view allows the user to download SW to the ECU. To learn more about what can be done in this section, please refer to section 5 of this document.



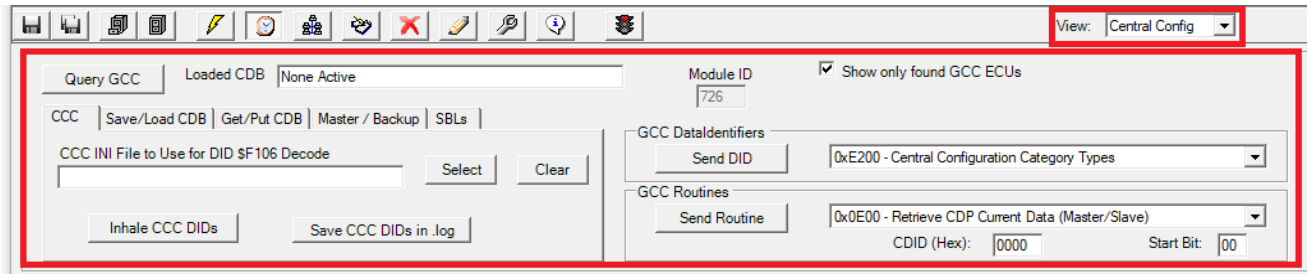
3.2.3.3 ECU Config View

This view allows the user to transfer direct ECU configuration to or from an ECU (using DIDs). To learn more about it, please refer to section 6 of this document.



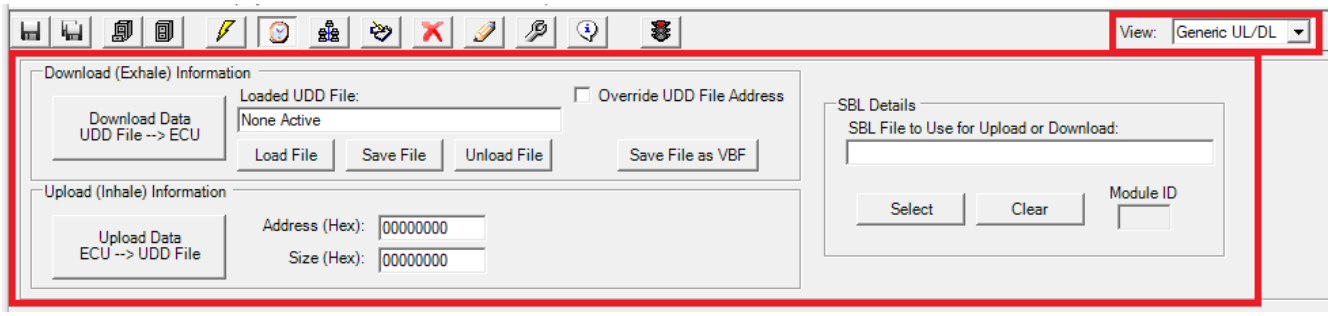
3.2.3.4 Central Config View

This view allows the user to work with modules that support the GCC spec (obsolete). It can inhale GCC data, save it to disk, and exhale it (to same ECU or one in a different vehicle). Users can query the bus to find what GCC compliant nodes are on the network. To learn more about it, please refer to section 7.



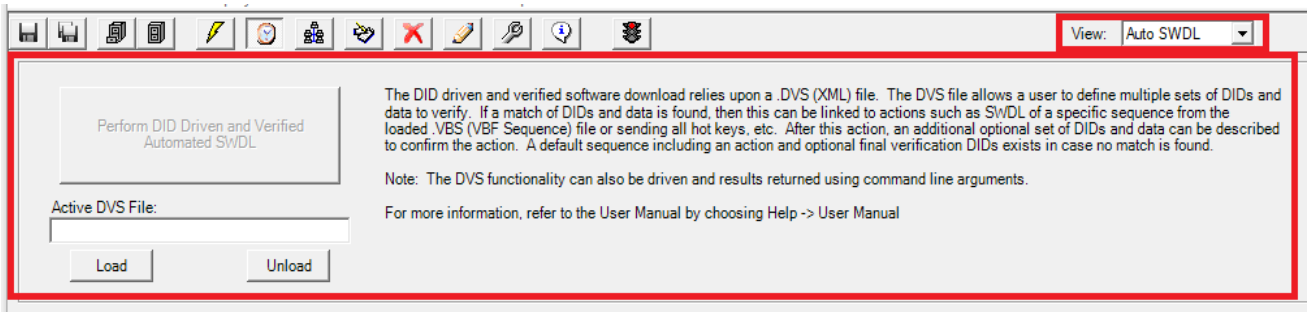
3.2.3.5 Generic UL/DL

This view allows the user to upload or download information to a specific memory address in an ECU. To learn more about it, please refer to section 8.



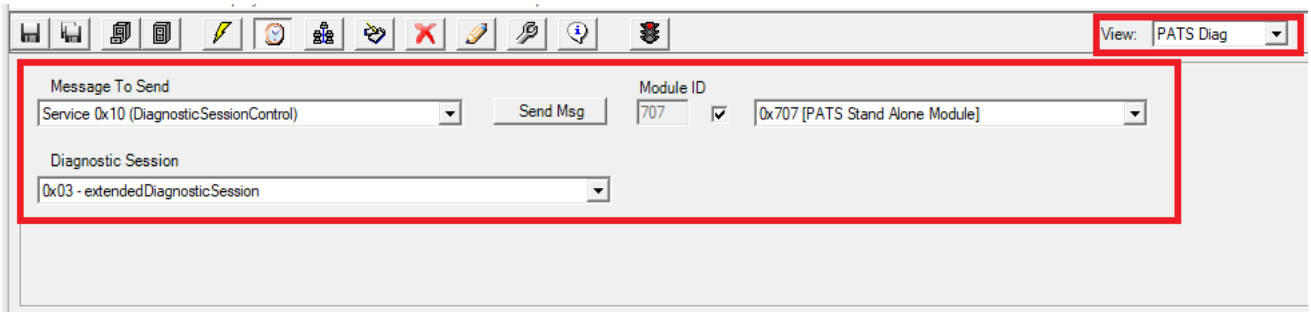
3.2.3.6 Auto SWDL View

This view allows the user to download software (in the form of VBF files) to an ECU, based on the content of a "*.DVS" file (XML). To learn more about it, please refer to section 9 of this document.



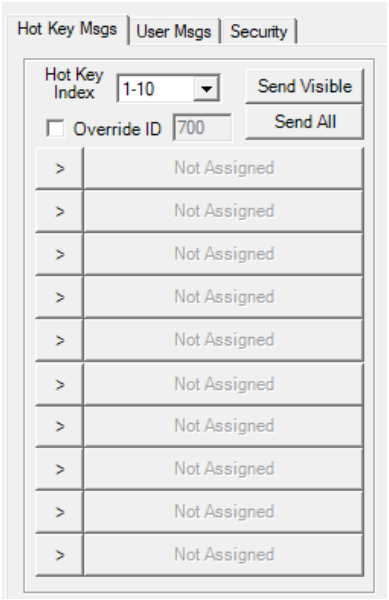
3.2.3.7 PATS Diag View

This view is intended to assist users who have a use case for performing some of the typical PATS diagnostic functionality, including simultaneous PATS timed security access. To learn more about it, please refer to section 10 of this document.



3.2.4 Hot Key Messages / User Messages / Security

This section, just as the main menu and the display, is available in all views. It allows saving messages in buttons and sending them all (at the same time) or separately. In the security tab, the security keys are managed (to unlock secured ECUs). The Security Challenge Bytes can be added manually or loaded from a previously created *.KEY file.



To learn more about this section please refer to sections 14, 15 and 16.

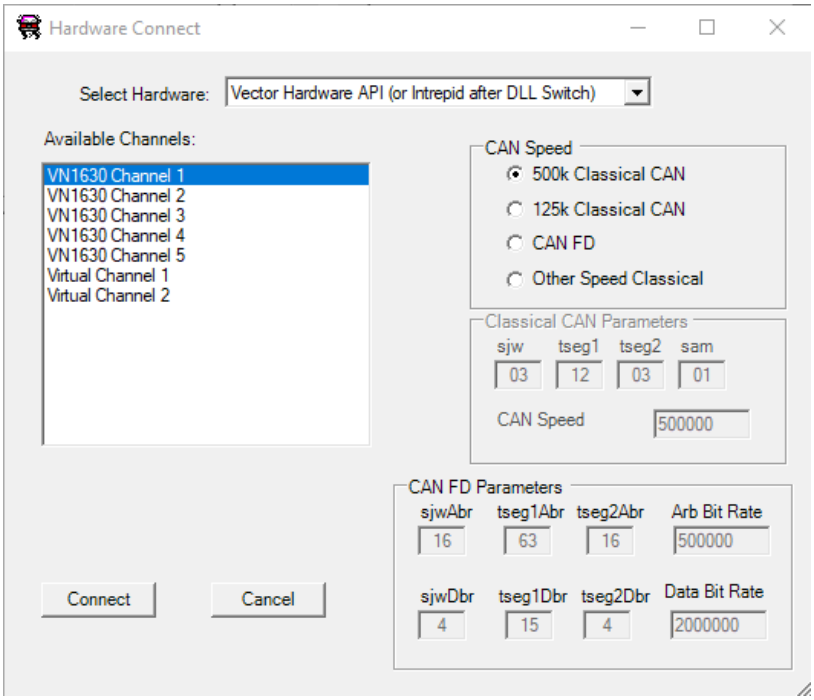
3.3 Connecting to Hardware

There are two options to connect the hardware: "Initialize Hardware" and "Quick Connect". With the "Initialize Hardware" option, a user can connect to a single CAN channel. With Quick Connects, a user can connect to more than one CAN channel.

Note that all available connections (whether connected using the "Initialize Hardware" mechanism described in 3.3.1 or the "Quick Connects" as described in 3.3.2) will appear in a drop down combo box in the toolbar. The selected connection represents the active one that a request to transmit will be sent on (see 3.4 for more details).

3.3.1 Initialize Hardware

Go to the Main Menu, select "Initialize Hardware", the following window will pop.

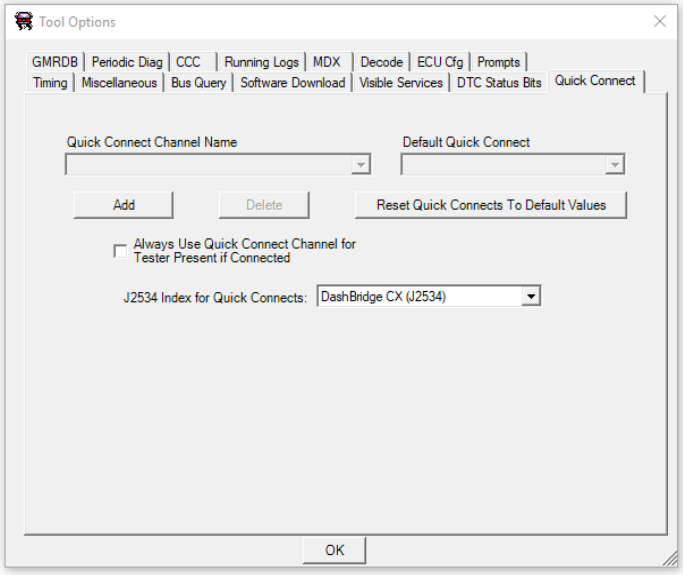


Select the Hardware you want to use (e.g. Vector, Intrepid, etc.), the channel you have configured for either HS-CAN or MS-CAN, and select the connect button.

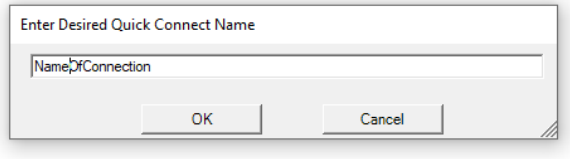
For more information to connect hardware using "Initialize Hardware" option, please refer to section 12.3 of this document.

3.3.2 Quick Connect

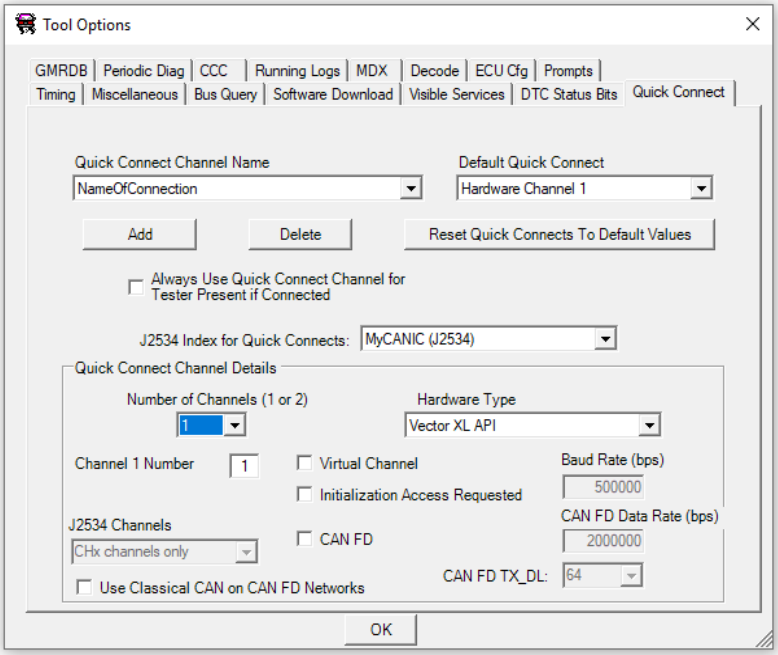
The main benefits to using Quick Connect is the 1) ability to connect to multiple channels, 2) one click connection after starting the program, and 3) ability to support command line arguments. To configure Quick Connect, go to the Main Menu, select Tools > Options (or select on the wrench in the toolbar) and select on the "Quick Connect" Tab. See section 11.7 for more information on the Quick Connect options.



Click on Add button and select the name of the connection.



Select the number of Channels to use (either 1 or 2 channels). Note that for almost ALL scenarios, users will only want to connect to 1 channel for a given Quick Connect connection. If there is a desire to connect to more than one CAN channel, then more than 1 Quick Connect should be setup. If a single Quick Connect is configured for two channels, then all transmits on that connection will always go out on both channels.

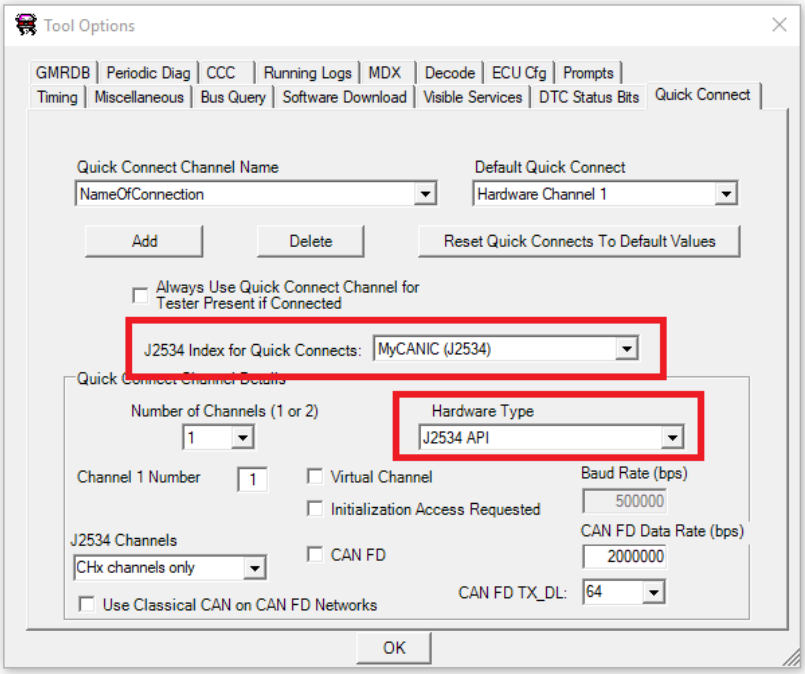


Quick Connect Channel names can be deleted by selecting it in the Quick Connect Channel Name, and clicking the delete button.

3.3.2.1 J2534 Device Setup

To setup a J2534 Quick Connect, first choose J2534 API in the Hardware Type dropdown. Then choose the device in the J2534 Index for Quick Connect. If the dropdown for your device does not appear, ensure that:

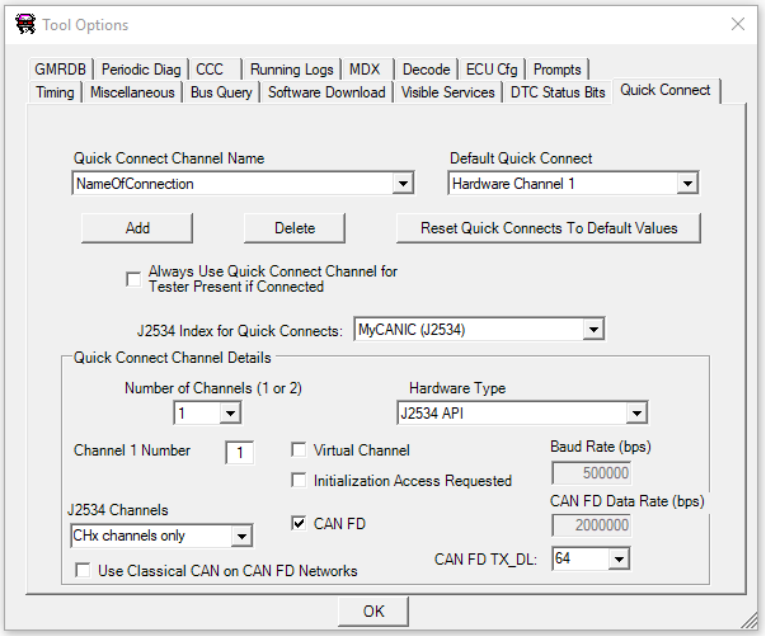
1. The J2534 device is correctly setup.
2. You may need to contact your local IT support for permission to install these devices.
3. For additional support, contact the device's manufacturer.



To use the J2534 CAN and PS channels, select CAN & PS Channels only in the J2534 Channels dropdown. To use the CHx channels, select CHx channels only.

The baud rate may be changed in the Baud Rate text box.

For CAN FD, check the box. The desired maximum CAN FD frame can be set in the CAN FD TX_DL dropdown. If using CAN FD, but desiring to use classical CAN on CAN FD, select the check box next to Use Classical CAN on CAN FD Networks.

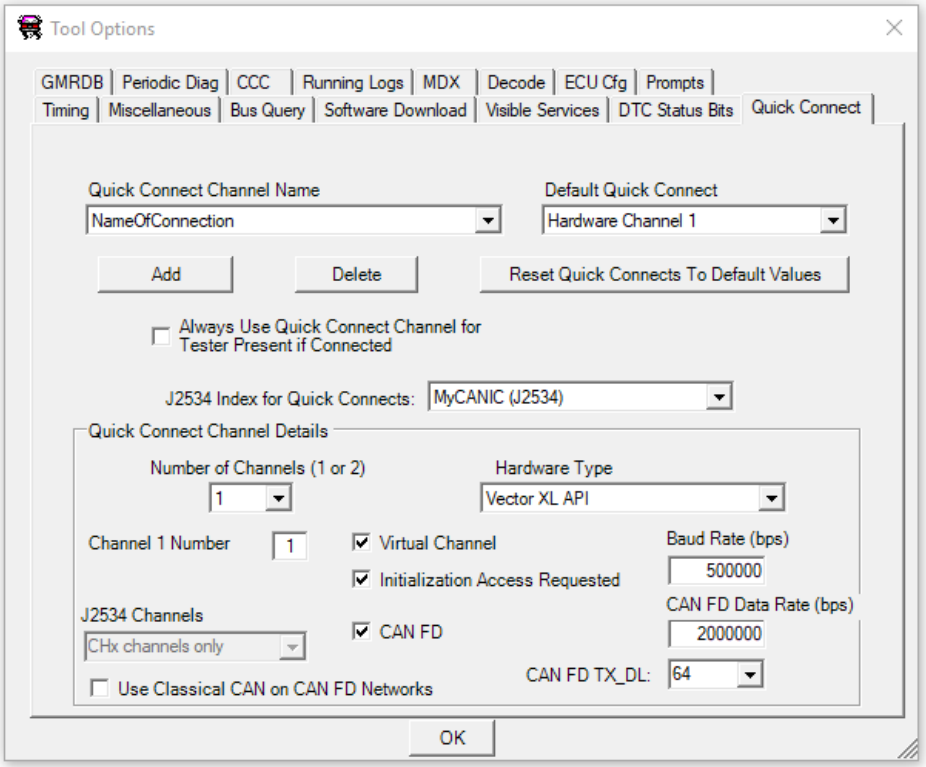


3.3.2.2 Virtual Channel Setup

Virtual channels are only available with Vector devices. Virtual channels can also be setup as a Quick Connect. Under the Hardware Type dropdown, select the correct Hardware Type, and then check the Virtual Channel checkbox.

The baud rate may be changed in the Baud Rate text box.

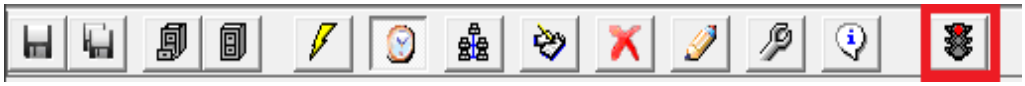
For CAN FD, check the box. The desired maximum CAN FD frame can be set in the CAN FD TX_DL dropdown. If using CAN FD, but desiring to use classical CAN on CAN FD, click the check box next to Use Classical CAN on CAN FD Networks.



3.3.2.3 Using Quick Connect

Once the Quick Connects are configured, to connect to all configured Quick Connects, perform the following steps:

1. Go to the "Toolbar Buttons" area



2. Click on the Traffic Light button (selected in red above). This will attempt to connect ALL quick connects that are configured. The connection selected by default will be the one configured in "Default Quick Connect".

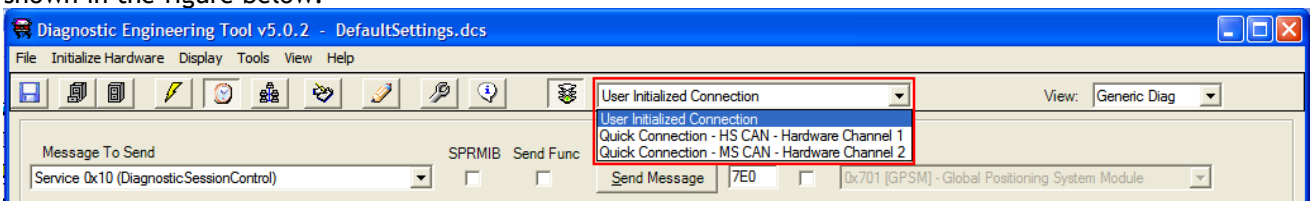
As with almost all settings of the DET, the Quick Connect settings are saved as part of the .DCS file. These settings can be made part of the user's default settings by selecting File → Make Current Settings Default Configuration, or by ensuring the Tools → Options → Miscellaneous tab choice of "Always Save DCS on Exit if Default Settings File Active" is selected.

Please note the following:

- Quick connections are required for command line functionality (see section 24).
- Quick Connections work with Vector or Intrepid (with DLL switch) hardware and also with generic J2534 connections. For J2534 connections, if multiple J2534 devices are registered, the index of the desired J2534 must be set correctly under the quick connect setup by selecting the corresponding hardware in the “J2534 Index for Quick Connects” dropdown box.

3.4 Active Connection

All active hardware connections (refer to section 3.3) are displayed in a drop down box in the toolbar as shown in the figure below.



When quick connects are used, this drop down box can contain more than one connection. When the manual initialize hardware approach is used, the drop down box will only ever contain a single connection. When sending individual diagnostic requests, etc. these will be transmitted on the connection that is selected in the drop down box shown in the figure above. Note that there are various options in the DET (e.g., periodic testerPresent, SWDL, Bus Query, etc.) that allow the user to configure whether or not it should send on all connections or not.

3.5 Tool Updates, Issues, and Expiration

Note that it is always important to utilize the latest version of the DET. New functionality is added on a periodic basis and all known issues are resolved as soon as possible. Issues submitted with older versions of the tool will not be addressed as it is most likely the issue has already been fixed in a later version. Only after an issue has been verified to exist with the latest version of the tool should it be submitted. Note that a detailed change log describing the changes from all versions is included with each release (see section 12.7.2).

In order to help ensure later versions of the tool are used as well as to prevent access to the tool to individuals that no longer have a business case, DET versions are set up to expire and no longer run one year after the date they are released. Note that attempts by the user to roll back the system date are recognized and will prevent the tool from running at all on the computer, even if the date is restored and even if the DET is uninstalled and reinstalled. The fix for this will require a special customized application and the turn-around time is expected to be 30 days.

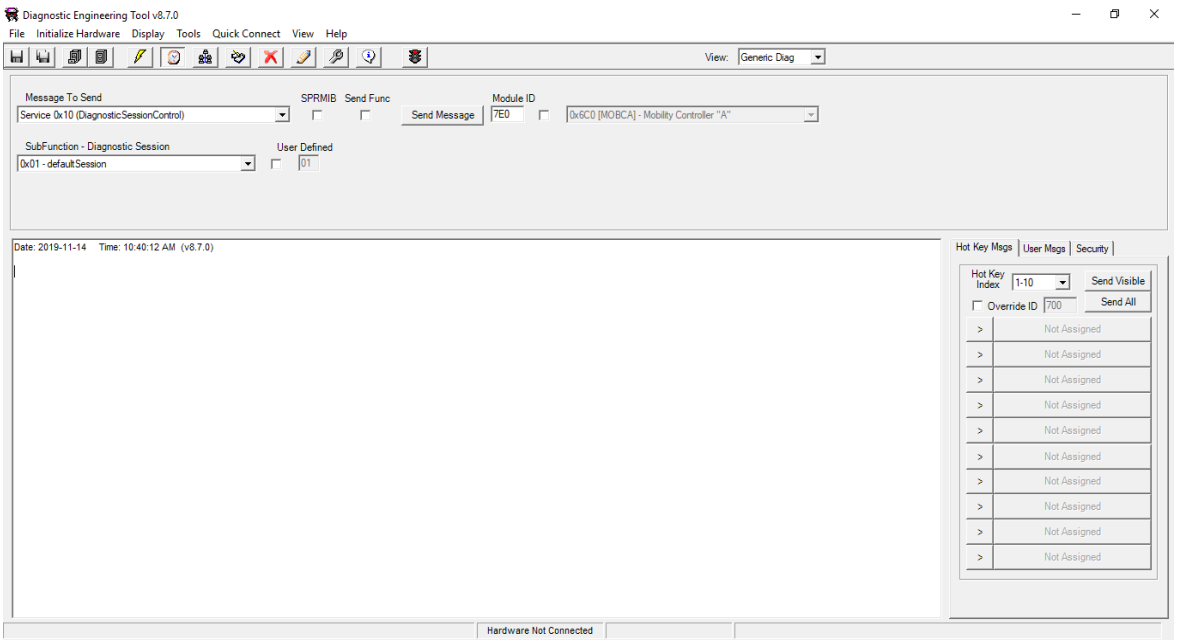
The official location of the DET releases is on the Ford NetCom SharePoint site. For those with direct intranet access, the latest DET may be obtained from the following link:
https://pd1.extspt.ford.com/sites/EENETCOM/netcomtools4/SitePages/Home.aspx

Then select the Diagnostic Engineering Tool (DET) link to get to the latest DET release.

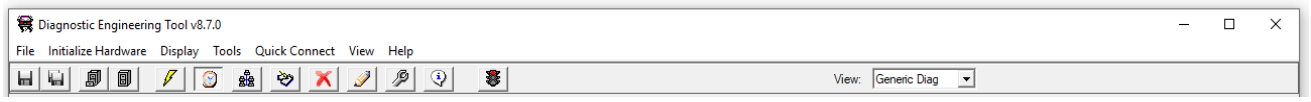
Requests to access the SharePoint site link may be submitted to corenetc@ford.com

4 Generic Diag View

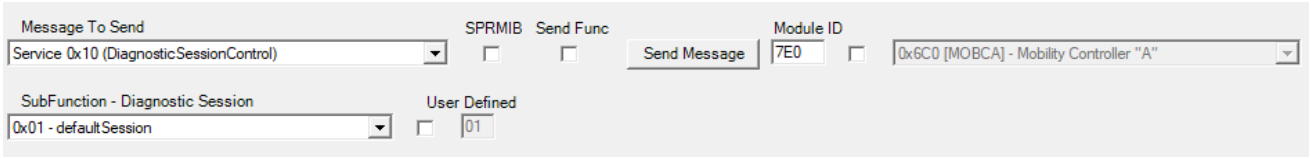
When you open your DET for the first time with no default .DCS file, the default view is the Generic Diag view. In this view you are able to easily configure all of the Ford utilized ISO 14229 services, and save these as Hot Keys, User Messages, or send them directly on one of the CAN connections.



For more information regarding the options at the header please refer to the section 3.2.

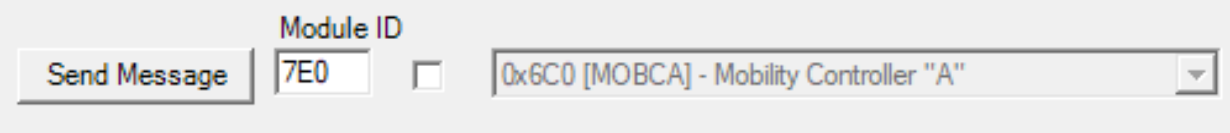


Just below the buttons shown in the previous picture, you will find a series of "drop down" boxes, these will allow you to choose the service you want to run along with the ECU that you want to communicate with.



4.1 Send Message Button

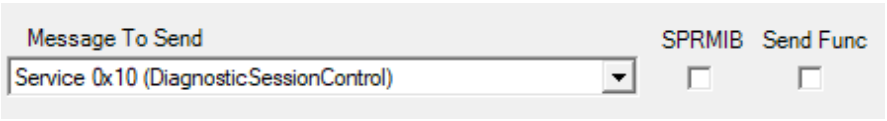
The "Send Message" button (see also 13.4) is used to actually transmit the currently configured "Message to Send" to the desired ECU on the Active Connection (see section 3.4). The ECU that the request is sent to is specified by the Module ID box. As with many of the settings on the Generic Diag view, you can either type the information in directly into the corresponding text box, or select the checkbox accordingly so that the information can be selected from a pre-populated list. For Module IDs, the pre-populated list comes from the GMRDB if it is loaded at start-up or imported manually by the user.



There are two different checkboxes that are present for some of the services in the Generic Diag view, dependent upon their applicability to the service. One checkbox is titled SPRMIB and the second is titled Send Func.

SPRMIB stands for Suppress Positive Response Message Indication Bit and is always the most significant bit of the subfunction byte for any service that uses a subfunction parameter. When the checkbox is checked, this SPRMIB bit is set to 1 and is used to request that the ECU suppress its response, but only if the response is positive. If the checkbox is not checked, this SPRIMB is set to 0 and is used to request that the ECU send its positive response. The SPRMIB is often used with functional requests such as testerPresent, when the tool does not have a desire to see all of the positive responses.

The "Send Func" checkbox is used when you want to send the message or service as a functional request, which means that all ECU connected in the network will respond to the message accordingly. Note that while the DET sends the request functionally (i.e., on CAN ID 0x7DF) so that all ECUs receive it, the DET will only receive and display the response from the ECU whose Module ID is configured in the request (i.e., the ECU who would receive the request if "Send Func" was not checked).



4.2 Message to Send

The message to send drop box, offers the opportunity to choose among the Ford supported ISO 14229 diagnostic services. Each service has a specific function, and because this aspect of the DET is user driven, the user must have a clear understanding of what they want to do. For further information and details relating to these services, it is recommended to review the Ford diagnostic specifications, ISO 14229 specification, and the ISO 14229 diagnostic training documents.

The list of visible services in the drop down box is controlled via the user options (see section 11.5 for more details). This way a user can customize the list to only display the services that they use on a regular basis.

Once a service is selected in the drop down box, the remaining information displayed on the Generic Diag view is customized to this specific "Message to Send" selected. For example, if the user selects "Read Data By Identifier" then a parameter to enter the 2 byte DataIdentifier to send appears. For most parameters, there are two ways to enter the information. One way is to select from a predefined list of parameters from a drop down box (sometimes these are hardcoded in the DET, sometimes they are imported from an MDX file, etc.)

and the other way is to select a checkbox next to the drop down box which will enable the user to manually type in the value to send directly into a textbox.

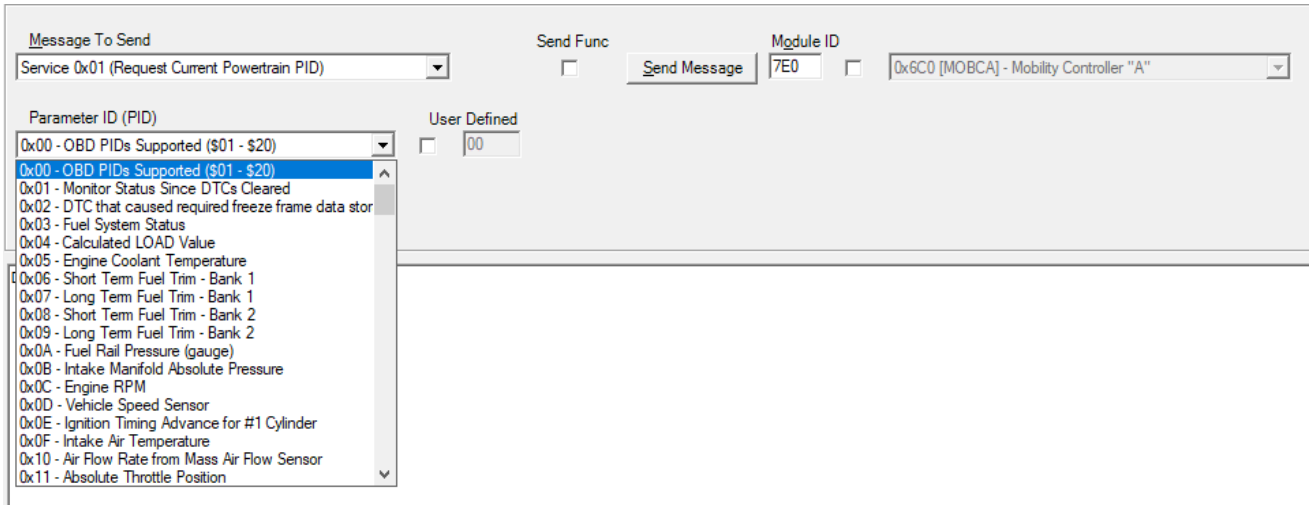
After defining a message, it can be transmitted immediately (see sections 4.1 and 13.4), saved as a hot key (see section 14), or saved as a user defined message (see section 15).

A summary of most of the available services are listed in the following sections.

4.2.1 User Labelled Diagnostic Message

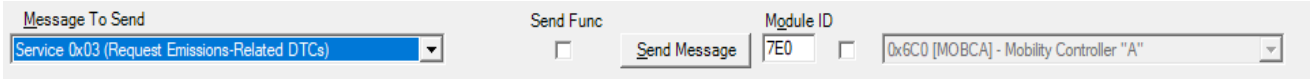
TODO: Add picture here and description.

4.2.2 Service 0x01 (Request Current Powertrain PID)



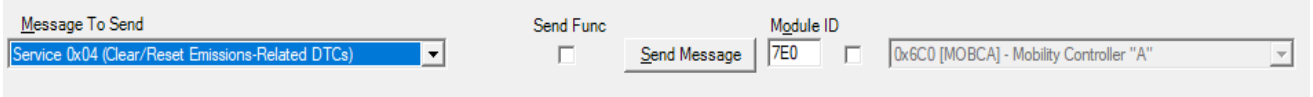
This service allows the user to read the OBD Mode 0x01 PIDs. Predefined PIDs can be chosen from the drop down box. Any other user defined PID can be read by selecting the checkbox under “User Defined” and then manually entering the one byte PID address.

4.2.3 Service 0x03 (Request Emissions-Related DTCs)



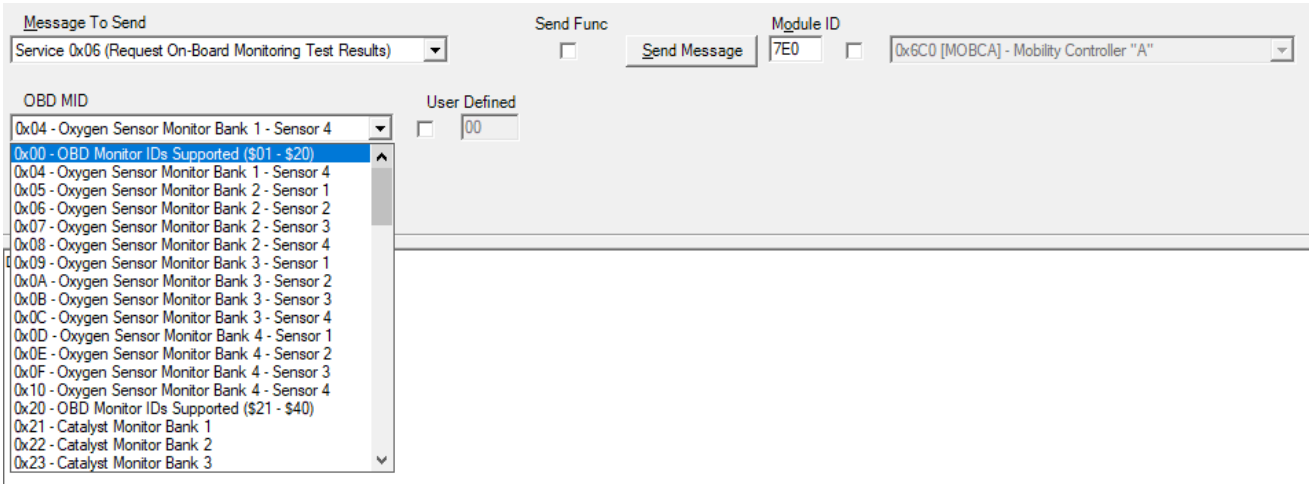
This service allows the user to request emissions-related DTC’s.

4.2.4 Service 0x04 (Clear/Reset Emissions-Related DTC’s)



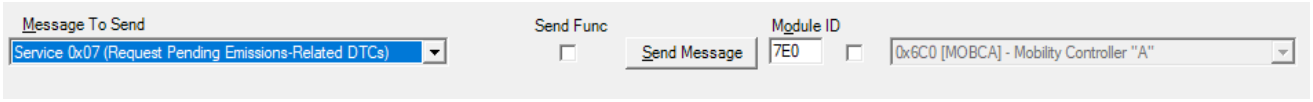
This service allows the user to clear/reset emissions-related DTC’s.

4.2.5 Service 0x06 (Request On-Board Monitoring Test Results)



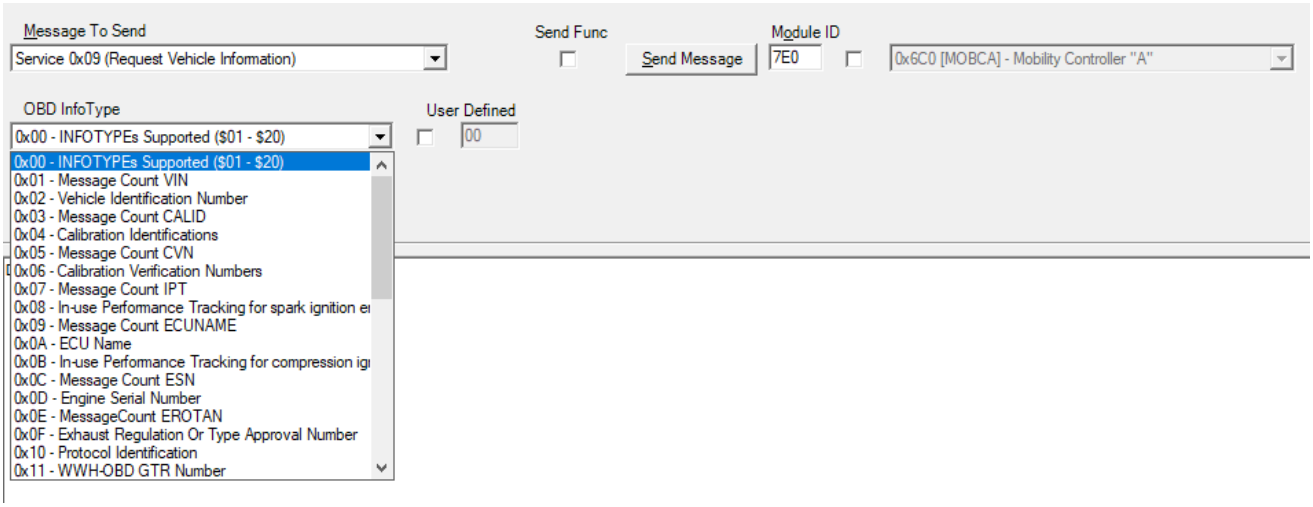
This service allows the user to read the OBD Mode 0x01 Monitor IDs. Predefined Monitor IDs can be chosen from the drop down box. Any other user defined Monitor ID can be read by selecting the checkbox under "User Defined" and then manually entering the one byte Monitor ID address.

4.2.6 Service 0x07 (Request Pending Emissions-Related DTCs)



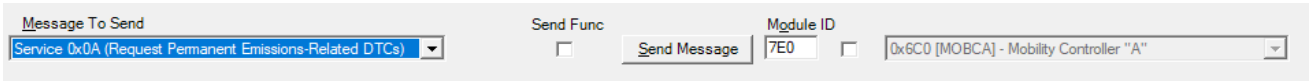
This service allows the user to request pending emissions-related DTC's.

4.2.7 Service 0x09 (Request Vehicle Information)



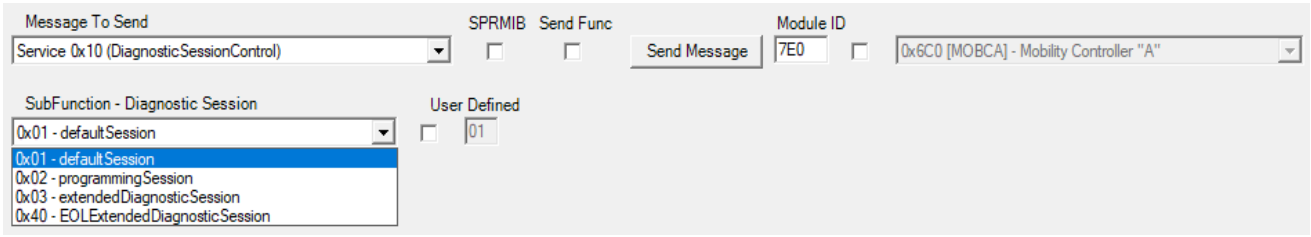
This service allows the user to read the OBD Mode 0x09 INFOTYPEs. Predefined INFOTYPEs can be chosen from the drop down box. Any other user defined INFOTYPE can be read by selecting the checkbox under "User Defined" and then manually entering the one byte INFOTYPE address.

4.2.8 Service 0x0A (Request Permanent Emissions-Related DTCs)



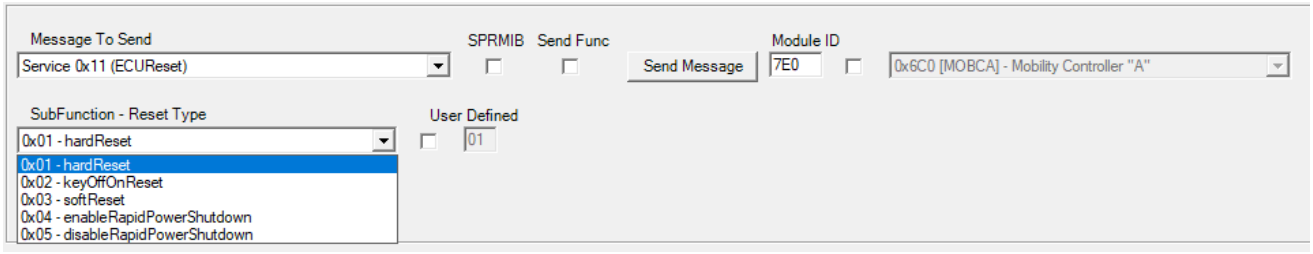
This service allows the user to request permanent emissions-related DTC's.

4.2.9 Service 0x10 (DiagnosticSessionControl)



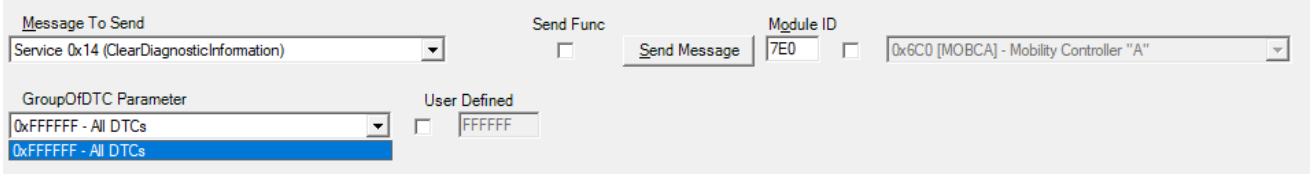
This service allows the user to change the active diagnostic session on the ECU. With this service, the only additional parameter is the diagnostic session to change to. The predefined drop down box contains standard sessions. Any other user defined session can be sent by selecting the checkbox under "User Defined" and then manually entering the one byte value to send for the session.

4.2.10 Service 0x11 (ECUReset)



This service allows the user to change the perform one of the defined ECU resets on the ECU. With this service, the only additional parameter is the reset type to use. The predefined drop down box contains standard reset types. Any other user defined reset type can be sent by selecting the checkbox under "User Defined" and then manually entering the one byte value to send for the reset type.

4.2.11 Service 0x14 (ClearDiagnosticInformation)



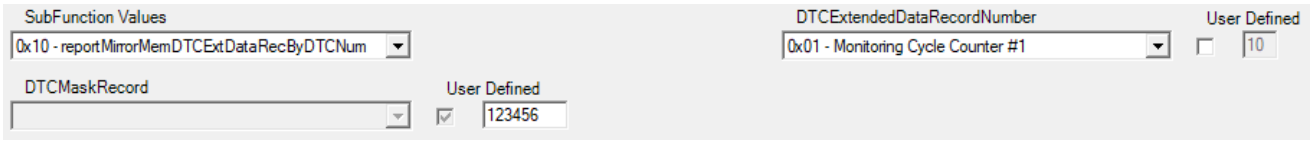
This service allows the user to clear DTCs in the ECU. With this service, the only additional parameter is the GroupOfDTC parameter to send. The predefined drop down box contains the only GroupOfDTC parameter supported by Ford which is 0xFFFFFF (corresponding to All DTCs). Any other user defined GroupOfDTC can be sent by selecting the checkbox under "User Defined" and then manually entering the three byte value to send

for the GroupOfDTC. As mentioned previously, if the "Send Func" checkbox is selected, this request will be sent on CAN ID 0x7DF and will therefore be received (and thus DTCs cleared) in all ECUs on the active connection.

4.2.12 Service 0x19 (ReadDTCInformation)

This service allows the user to perform any of the subfunctions related to reading DTC information. The information that appears with this service is dependent upon the specific subfunction value that is chosen. For example, if subfunction 0x02 (reportDTCByStatusMask) is selected, then the additional information needed is the DTC Status Mask. The DET will populate the DTC Status Mask drop down box with a list of some of the more common masks, but a user can enter their own user defined mask. However, if subfunction 0x04 (reportDTCSnapshotRecordByDTCNumber) is selected, then the additional information needed is the 3 byte DTC number (entered in the DTC Mask Record) as well as the one byte DTC Snapshot Record Number to send. For some subfunctions such as 0x03, there is no additional information at all to configure for the request.

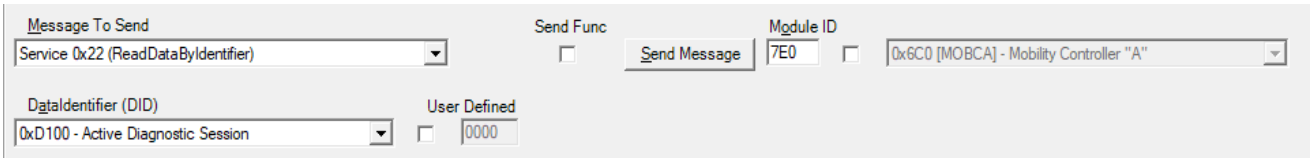
As an example of how the view adapts, for subfunction 0x10 the view will appear as shown below.



For J1979-2, the following have been implemented but should still be considered under development.

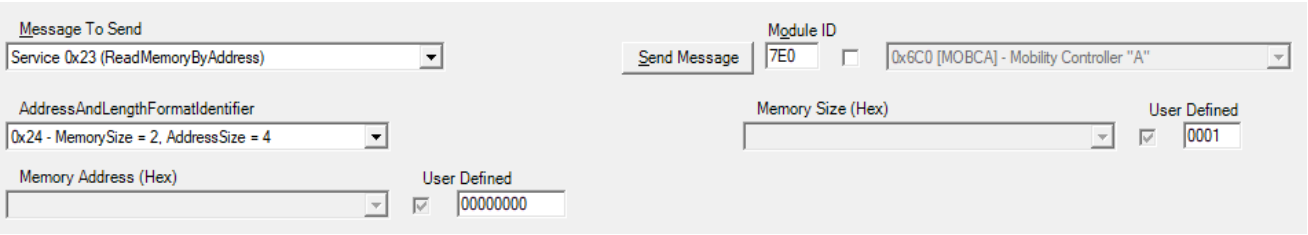
SAE J1979		SAE 1979-2	
0x03	Request Emission-Related DTCs	0x19 42 (08)	reportWWHOBDDTCByMaskRecord (confirmed)
0x07	Request Emission-Related DTCs Detected During Current or Last Completed Driving Cycle	0x19 42 (04)	reportWWHOBDDTCByMaskRecord (pending)
0x0A	Request Emission-Related DTCs with Permanent Status	0x19 55	reportWWHOBDDTCWithPermanentStatus
0x02	Request Powertrain Freeze Frame Data	0x19 04	reportDTCSnapshotRecordByDTCNumber 0x00 (first), 0xF0(latest)
0x04	Clear/Reset Emission-Related Diagnostic Information	0x14	ClearDiagnosticInformation
-	-	0x19 56	reportDTCInformationByDTCReadinessGroupIdentifier (list of DTCs for a readynessgroup)
-	-	0x19 1A	reportSupportedDTCExtDataRecord
-	-	0x19 06 DTC 91	reportDTCExtDataRecordByDTCNumber (single IUMPR data)
-	-	0x19 06 DTC 92	reportDTCExtDataRecordByDTCNumber (test result data)
		0x19 06 DTC 93	reportDTCExtDataRecordByDTCNumber (IUMPR-lite)

4.2.13 Service 0x22 (ReadDatabyIdentifier)



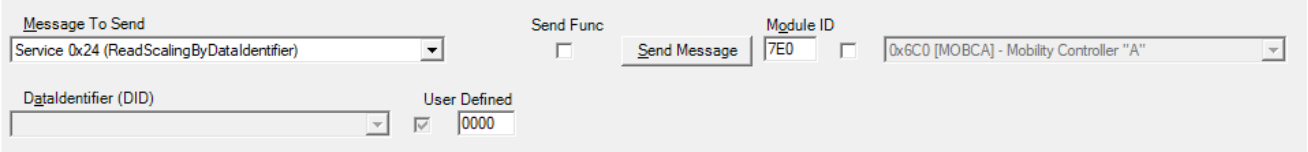
This service allows the user to read information based upon a specific DID. With this service, the only additional parameter is the DID to read. The predefined drop down box contains some of the standard DIDs used at Ford. However, if the MDX file for the ECU in the Module ID field is loaded, then this predefined drop down box contains all of the DIDs supported in the MDX file. As is typical, any other user defined DID can be sent by selecting the checkbox under "User Defined" and then manually entering the 2 byte DID value to send in the request.

4.2.14 Service 0x23 (ReadMemorybyAddress)



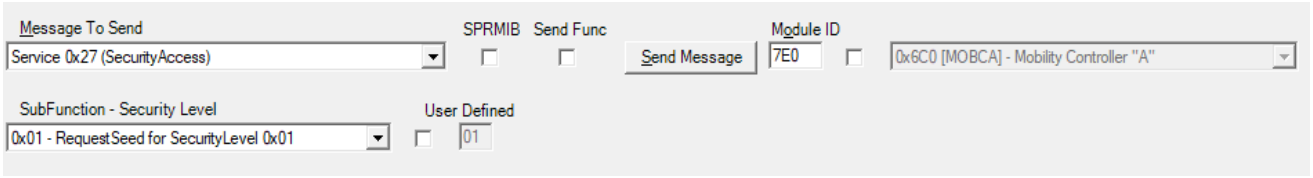
This service allows the user to read information based upon a physical memory address in the ECU. With this service, there are three additional parameters. One is the AddressAndLengthFormatIdentifier which contains the predefined values supported by Ford. The other parameters are the Memory Address to read and the number of bytes (Memory Size) to read. With this service, because the ECU addresses, etc. are specific to the ECU being diagnosed, the drop down boxes for address and size do not contain any predefined values.

4.2.15 Service 0x24 (ReadScalingbyDataIdentifier)



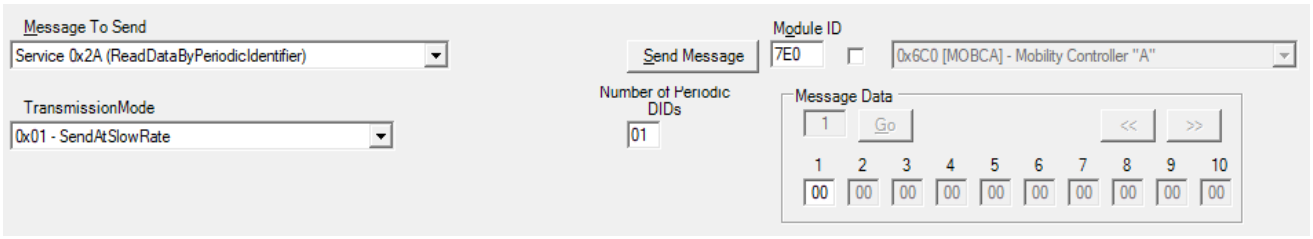
This service allows the user to read the scaling information for a particular DID. With this service, the only additional parameter is the DID to read the scaling information for. Note that this service is not commonly used by ECUs at Ford and even for ECUs which support the service, not all DIDs are supported. The ECU's Part 2 should be referenced to determine if the service is supported and for what DIDs.

4.2.16 Service 0x27 (Security Access)



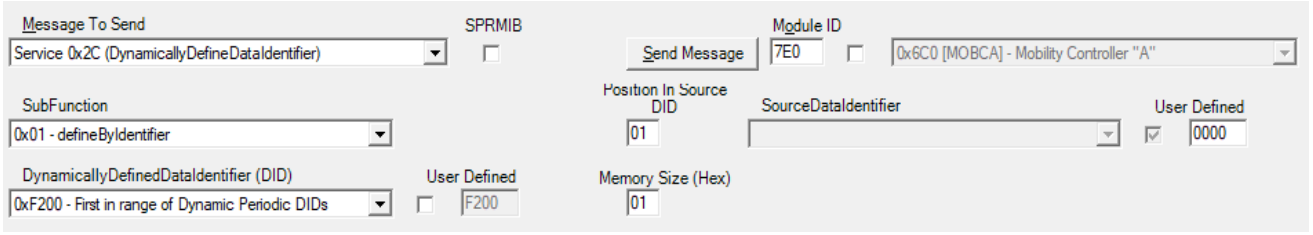
Many diagnostic functions are protected behind security. This service allows the user to send either a seed request or a key submittal. Seed requests are always odd values and key submittals are always even values. Note that security access is typically dependent upon an algorithm which is a specific sequence consisting of a seed request following by a key submittal, where the key is calculating from the seed. Therefore, the manual sending of the seed/key has limited usage for typical engineers. Note that if the MDX file is loaded for the ECU and supports the standard Ford security algorithm for a given level, then a new "Unlock Security" button will appear immediately above the "Send Message" button that will perform the security sequence. However, for more details on how security is typically performed with the DET, refer to section 16.

4.2.17 Service 0x2A (ReadDatabyPeriodicIdentifier)



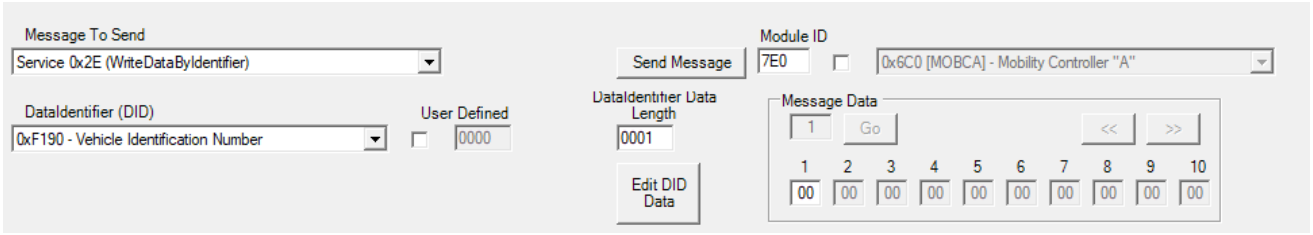
This service allows the user to request that the ECU start reporting rapid data packers (or UUDT frames). With this service, the additional data needed is the transmission mode, the number of periodic DIDs being requested, and the IDs of the periodic DIDs being requested. Note that this service is typically only supported by powertrain ECUs and some chassis ECUs. Refer to Ford specifications for more details.

4.2.18 Service 0x2C (DynamicallyDefineDataIndentifier)



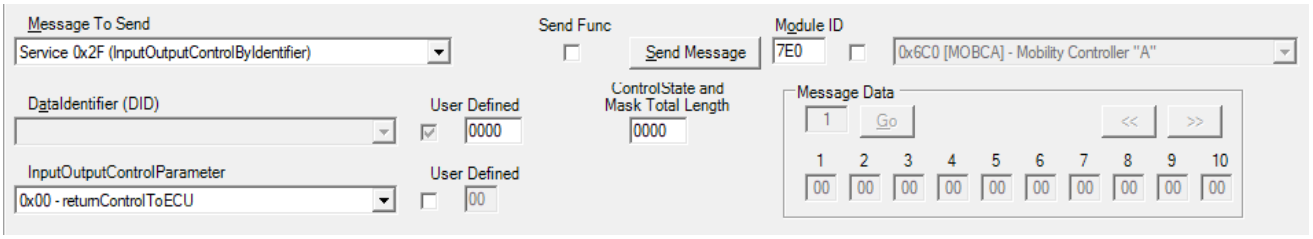
This service allows the user to dynamically define a DID based upon combinations from other DIDs or from memory addresses. This is typically used for advanced functionality at EOL and service on powertrain ECUs. Refer to Ford specifications for more details.

4.2.19 Service 0x2E (WriteDatabyIdentifier)



This service allows the user to write information based upon a specific DID. With this service, the additional parameters are the DID to write, the length of the DID data to write, and the actual data to write. The predefined drop down box for the DID contains some of the standard DIDs used at Ford. However, if the MDX file for the ECU in the Module ID field is loaded, then this predefined drop down box contains all of the DIDs supported in the MDX file. As is typical, any other user defined DID can be sent by selecting the checkbox under "User Defined" and then manually entering the 2 byte DID value to send in the request. When the DID data length is chosen, the corresponding number of bytes will become unlocked in the array to the right. The user can then enter in all data bytes to write.

4.2.20 Service 0x2F (InputOutputControlByIdentifier)

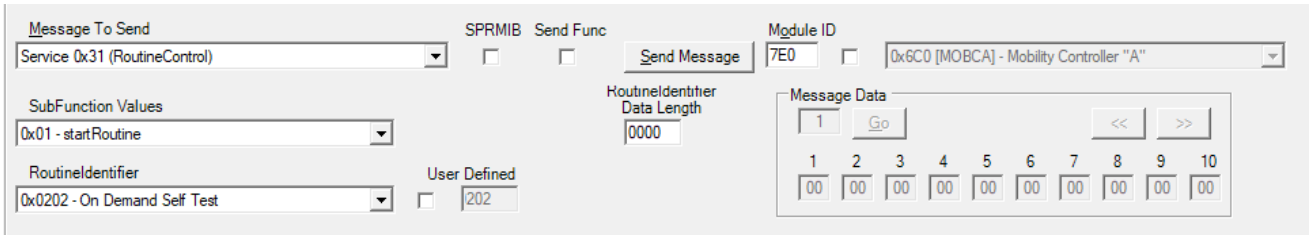


This service allows the user to temporarily control inputs or outputs based upon a specific DID. With this service, the additional parameters are the DID to write, the input output control parameter, the length of the DID data and control mask to include, and the actual data to include. Refer to Ford specifications for more details regarding which input output control parameters require to include the DID data and which DID data types require a control mask, etc.

The DataIdentifier (DID) combo box will enable only when an MDX with DIDs that support 0x2F are loaded.

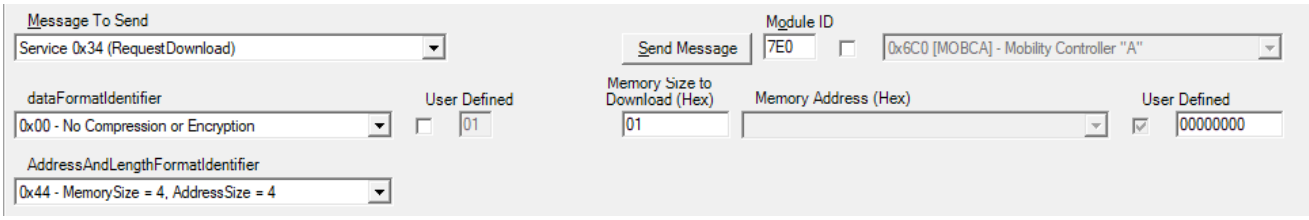
NOTE: If service 0x2F works at speeds less than 2 mph, and does not work at speeds greater than 2 mph, the ECU software is blocking it, not DET. There is no workaround for this other than a new calibration for that ECU.

4.2.21 Service 0x31 (RoutineControl)



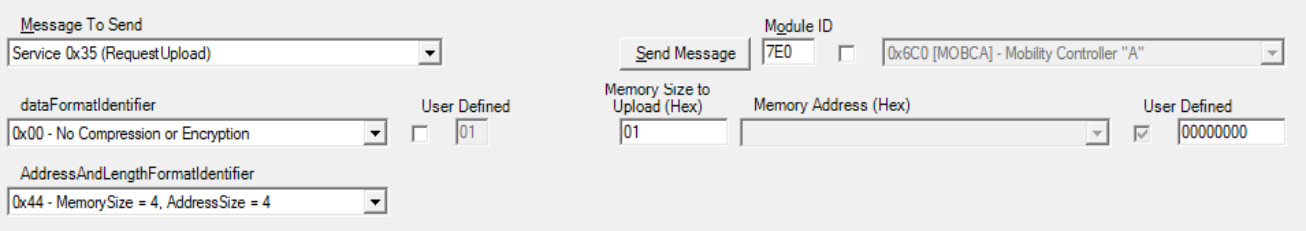
This service allows the user to execution a routine in an ECU. With this service, the additional parameters are the subfunction (which contains the 3 predefined values), the routineIdentifier to send, the length of any request additional data to include (dependent upon routineIdentifier and subfunction), and the actual request additional data. For more details on a particular routine, refer to the GMRDB or the ECU's Part 2 specification.

4.2.22 Service 0x34 (RequestDownload)



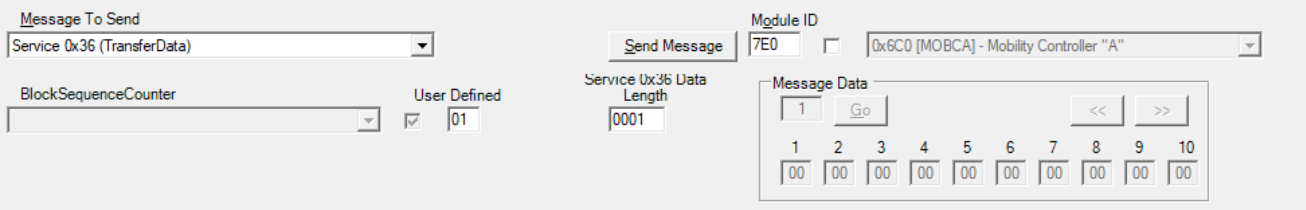
This service allows the user to send a request download service. With this service, the additional parameters are the dataFormatIdentifier, the AddressAndLengthFormatIdentifier, the memory size, and the memory address. Note that the actual downloading of an ECU typically follows a well structured sequence of events driven by a VBF file at Ford and therefore the manual sending of this service has limited usage for typical engineers.

4.2.23 Service 0x35 (RequestUpload)



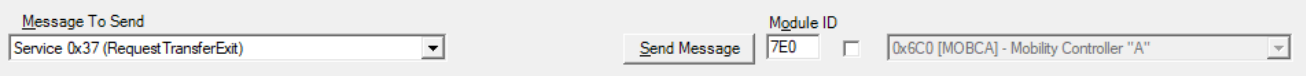
This service allows the user to send a request upload service. With this service, the additional parameters are the dataFormatIdentifier, the AddressAndLengthFormatIdentifier, the memory size, and the memory address. Note that the actual uploading of information from an ECU typically follows a well structured sequence of events and therefore the manual sending of this service has limited usage for typical engineers.

4.2.24 Service 0x36 (TransferData)



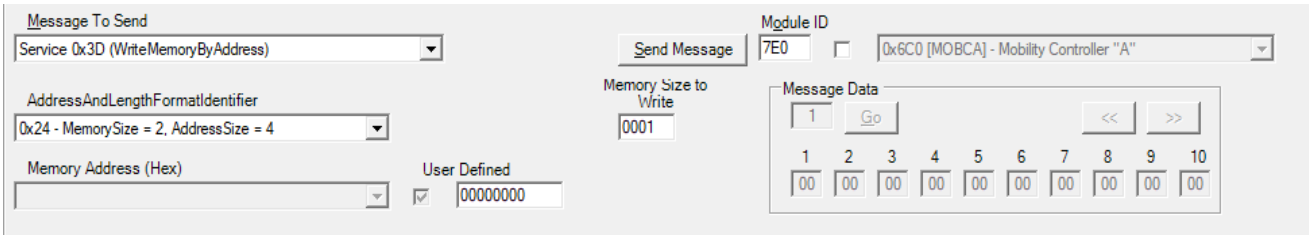
This service allows the user to send a transfer data service. With this service, the additional parameters are the length of the data included in the transferData request and the actual data itself. Note that this service is used in the actual downloading of an ECU which typically follows a well structured sequence of events driven by a VBF file at Ford. Therefore the manual sending of this service has limited usage for typical engineers.

4.2.25 Service 0x37 (RequestTransferExit)



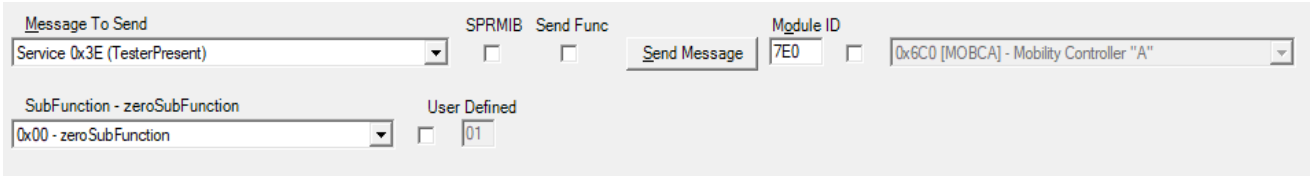
This service allows the user to terminate the uploading or downloading of a contiguous block of data. With this service, there are no additional parameters to configure. Note that this service is used in the actual downloading of an ECU which typically follows a well structured sequence of events driven by a VBF file at Ford. Therefore the manual sending of this service has limited usage for typical engineers.

4.2.26 Service 0x3D (WriteMemorybyAddress)



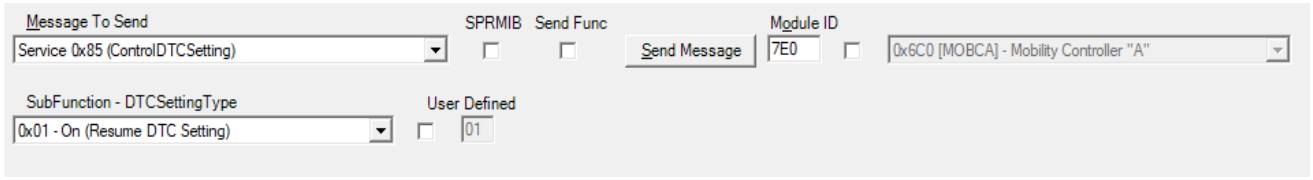
This service allows the user to write information starting at a physical memory address. With this service, the additional parameters are the AddressAndLengthFormatIdentifier, the physical memory address to begin writing at, the length of the data to write, and the actual data to write. When the data length is chosen, the corresponding number of bytes will become unlocked in the array to the right. The user can then enter in all data bytes to write.

4.2.27 Service 0x3E (TesterPresent)



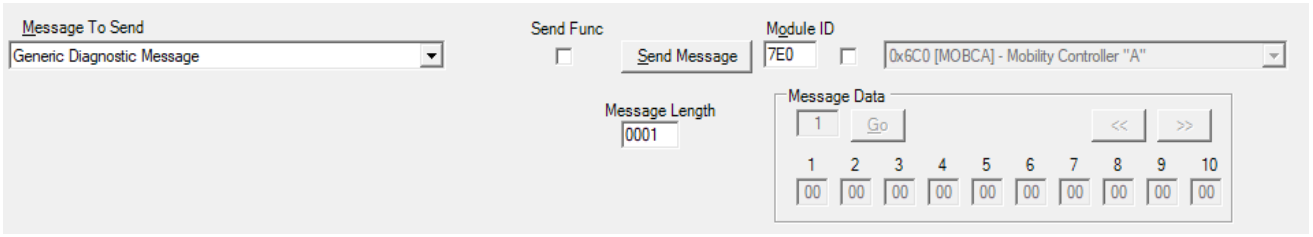
This service allows the user to send the testerPresent request which is normally used to prevent an ECU from timing out without invoking any specific diagnostic functionality within the ECU. With this service, the only additional parameter is the subfunction. The predefined drop down box contains the only standard subfunction which is the zero subfunction. Any other subfunction values can be sent by selecting the checkbox under "User Defined" and then manually entering the one byte value to send for the subfunction.

4.2.28 Service 0x85 (ControlDTC Settings)



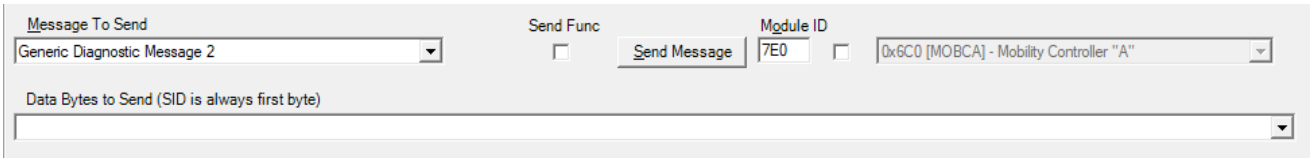
This service allows the user to temporarily stop and then resume the updating of DTC status bits for all DTCs. With this service, the only additional parameter is the DTC setting type to use. The predefined drop down box contains the 2 standard DTC setting types of 0x01 On and 0x02 Off. Any other user defined DTC setting type can be sent by selecting the checkbox under "User Defined" and then manually entering the one byte value to send.

4.2.29 Generic Diagnostic Message



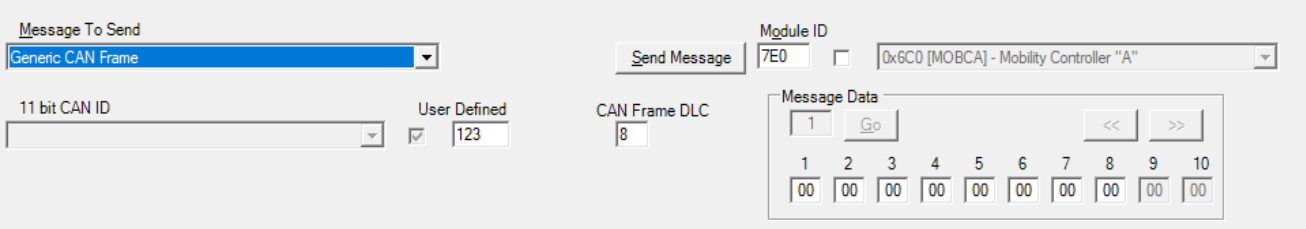
This allows the user to send any ISO 15765-2 compliant diagnostic message desired. With this approach the user simply enters the length of the request, which may be between 1 and 4095 bytes and then the X bytes corresponding to the length. The tool will correctly add any and all necessary network layer details and handle any segmentation over the CAN bus. For example, if one wanted to read DID 0x1234 then the message length would be set to 0003 with message data of 0x22 0x12 0x34. This also allows for sending invalidly formatted diagnostic requests from an application layer protocol perspective (e.g., a 4 byte DID read request).

4.2.30 Generic Diagnostic Message 2



This also allows the user to send any ISO 15765-2 compliant diagnostic message desired. However, the data entry for the request is different as compared to the other Generic Diagnostic Message. With this approach the user simply types the entire request (from an application layer perspective) into a text box in hexadecimal format. The tool will automatically determine the length of the request, which may be between 1 and 4095 bytes, and correctly add any and all necessary network layer details and handle any segmentation over the CAN bus. For example, if one wanted to read DID 0x1234 then the user would simply type "22 12 34" in the text box labeled "Data Bytes to Send (SID is always first byte)". This also allows for sending invalidly formatted diagnostic requests from an application layer protocol perspective (e.g., a 4 byte DID read request).

4.2.31 Generic CAN Frame



This allows the user to send any single CAN frame desired. This is typically used for non-diagnostic messages as no 15765-2 network layer details are added. With this approach the user simply enters the DLC of the CAN frame, which may be between 0 and 8 bytes and then the X bytes corresponding to the length. This approach can be used to simulate normal vehicle messages such as ignition status, vehicle speed, PRNDL, etc.

This will not work for CAN frames greater 8 bytes.

4.2.32 Predefined Sequence

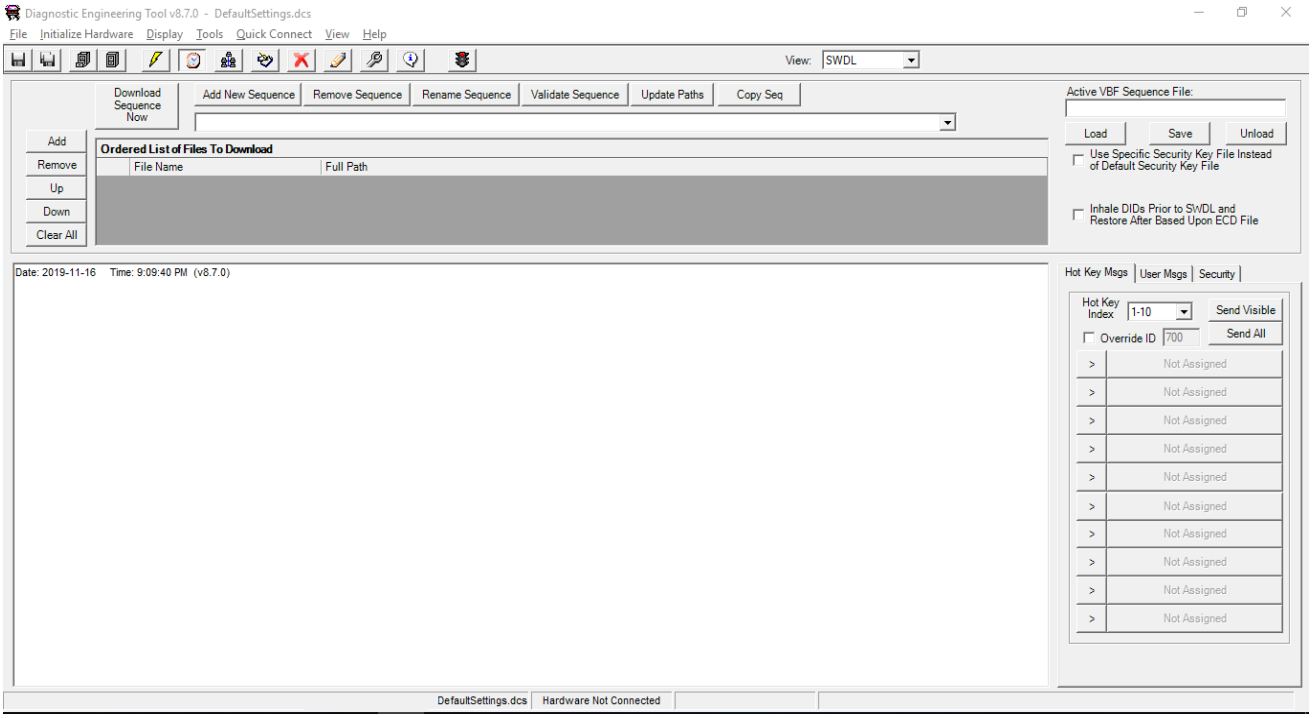
TODO: Add info here

5 SWDL View

Software Download view allows to create sequences to download SW to ECUs, the sequences can be saved (VBS extension) and loaded back any time it is needed. SWDL also allows one to Inhale information from configurable DIDs (based on ECD file loaded in DET, please refer to section 6) and write this inhaled information back after the software download was performed.

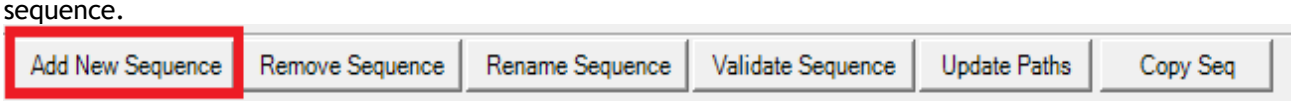
Please note the following when creating the VBS files:

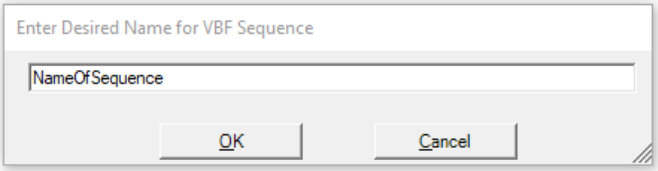
- The only supported files in the VBS files should have VBF extension.
- A VBS file can contain one or more sequences of VBF files.
- VBF files for more than one ECU can be in a single sequence (e.g., it is possible to download all ECUs on the vehicle with a single sequence), and will be processed from top to bottom.
- The order of the files for each ECU in the sequence should be as follows, from top to bottom:
    - SBL
    - Application
    - Calibration



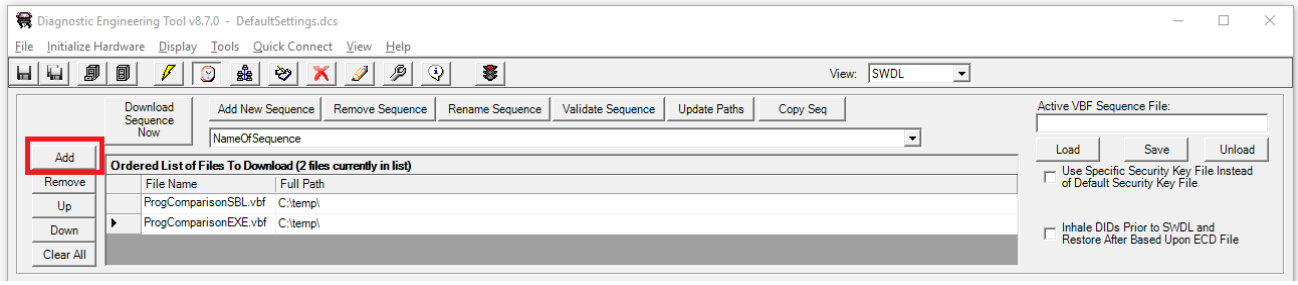
5.1 Add New Sequence

To add a new VBS file, use the "Add New Sequence" button. Simply select the button and enter a name for the sequence.



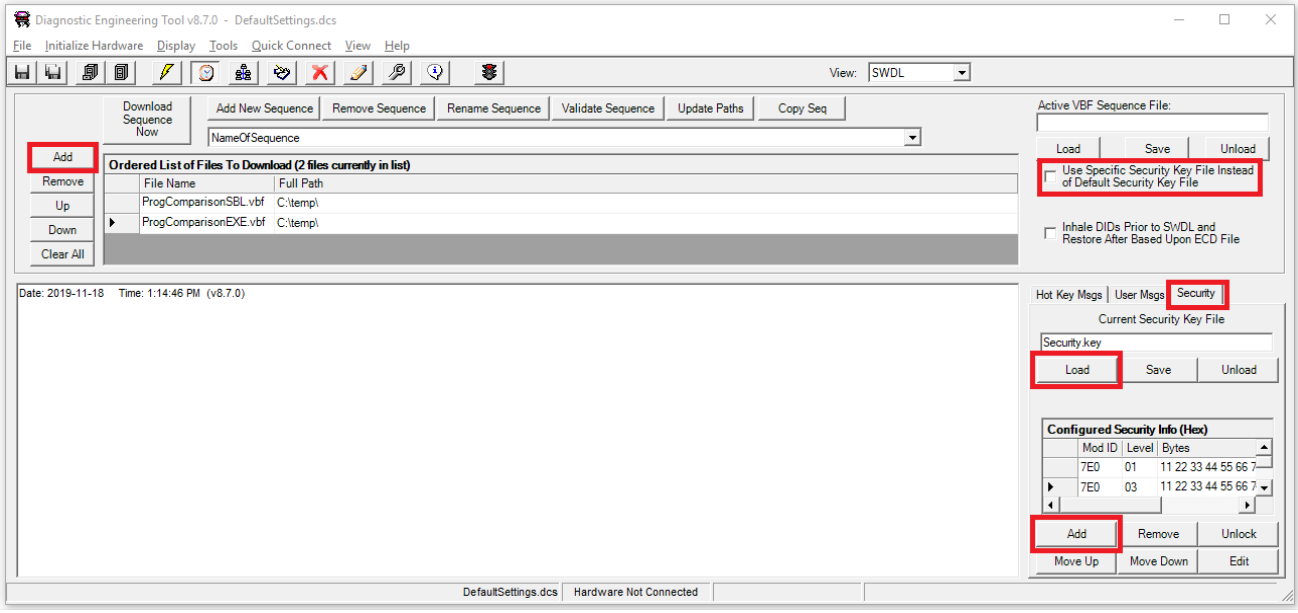


Click on the "Add" button and select the VBF files of the VBS sequence.

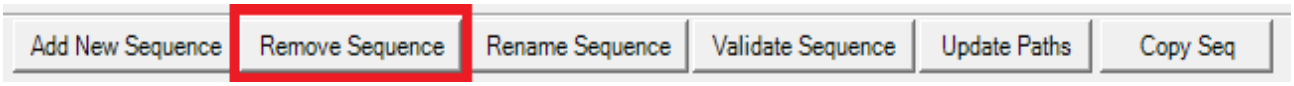


Add / load the Security Key (level 0x01), in the Security Tab, or enable the "Use Specific Security Key File Instead of Default Security Key File" checkbox.

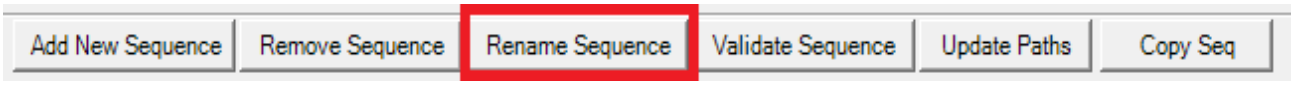
Note: If downloading to subnode through a gateway, please refer to special instructions in 5.9.1 Through a Gateway. If downloading to a gateway on a bench see the special instructions in 5.9.2 On a Bench.



Remove Sequence When a created or loaded sequence needs to be removed from the DET, select "Remove Sequence" button and the sequence will be removed from the SWDL drop box.



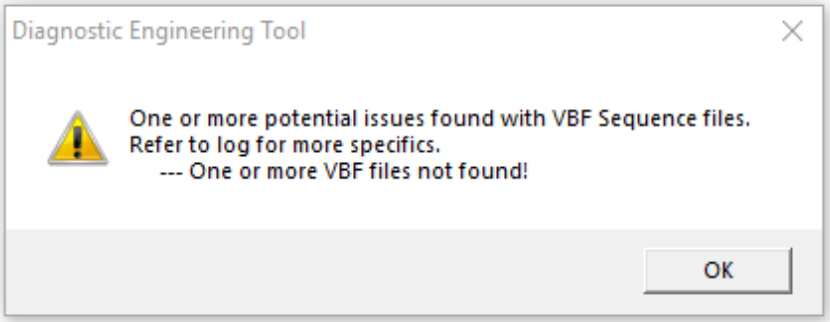
Rename Sequence If there is a need to rename the sequence, select "Rename Sequence" and introduce the new name desired for the sequence (selected sequence on the drop box).



5.2 Validate Sequence

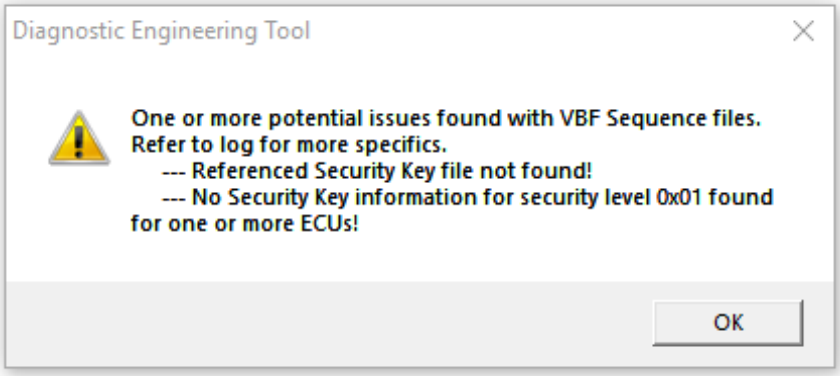
This button will allow the users to check basic configuration of the VBS file.

Sometimes a VBS file can be loaded, but the VBF files are not on the same path as they were when the original VBS file was saved (this mostly presents when opening a VBS on a different computer than the one that originally saved the VBS file). If this occurs, DET won't be able to successfully load the VBF files as it will be searching for them on a different path. The following error will be displayed:

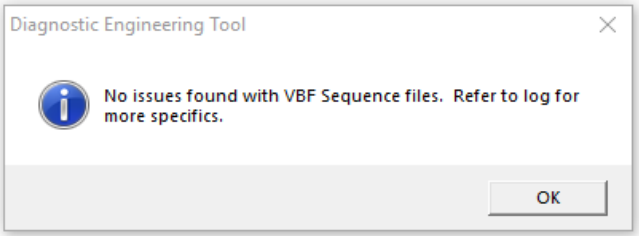


NOTE: Although the DET (prior to v5.10.3) does not directly support creating VBS files with relative paths, it is possible to manually edit the XML .VBS file to replace absolute paths with relative paths as compared to the loaded .VBS file. Using this approach allows one to ZIP and send a VBS file and associated VBF files that exist within subdirectories and have the receiving user unzip to a different absolute directory and still find the files. See section 0 for more information on modifying the VBS file through DET with relative paths.

For each ECU being programmed, a security Key (level 0x01) must be loaded. Either using a "Security Key File" (with checkbox "Use Specific Security Key File Instead of Default Security Key File" enabled), or loading/adding a Security Key file on the Security Tab. If no Security Key file is found, the following error will be displayed:



If the VBF files are set up correctly, the following message will be displayed:

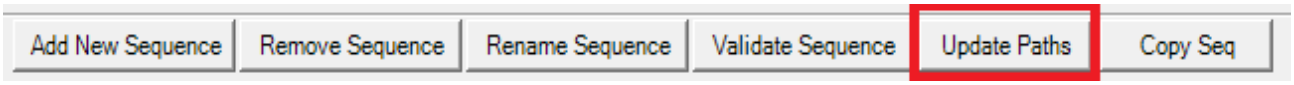


Please note that "Validate Sequence" does not verify the following:

- The Security Key has the correct challenge bytes for the module that needs to be flashed.
- The VBF files are ordered as: SBL, Application and Calibration.

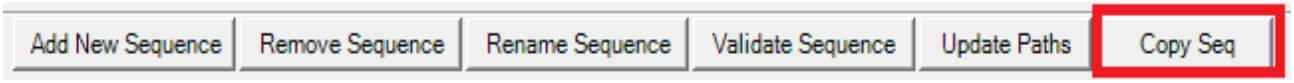
5.3 Update Paths

In DET v5.10.3 a new button was added titled “Update Paths”. This button allows a user to update the file path information for all VBF files and other files associated with a VBS sequence. The primary usage is to make it easier for others to utilize the same VBS file on a computer other than the one the file was created on, since relative paths can be used which allows the VBF files to be placed in a location relative to where the VBS file resides.



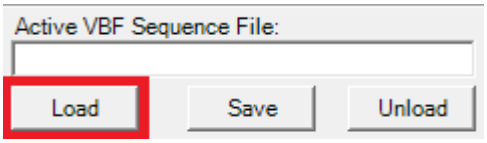
5.4 Copy Sequence

Copy Sequence will allow you to make a copy of the selected sequence on the drop box.



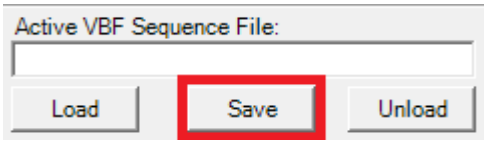
5.5 Load

Load will load a previously saved sequence. Click load and it will ask for the .VBS file.



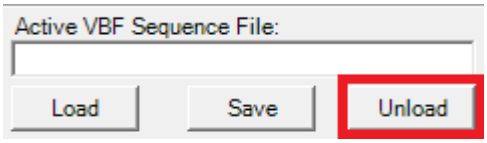
5.6 Save

Once a New Sequence is created, it can be saved with the Save button. Click on it and select the path where it is going to be saved.



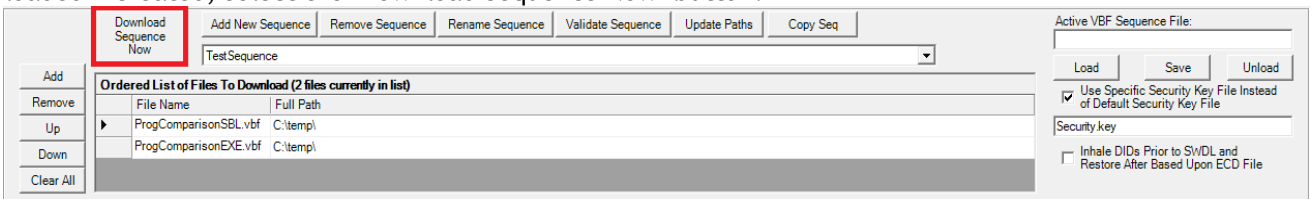
5.7 Unload

A loaded VBS file can be unloaded with the "Unload" button. This will unload all the sequences of the VBS file located at the drop box.

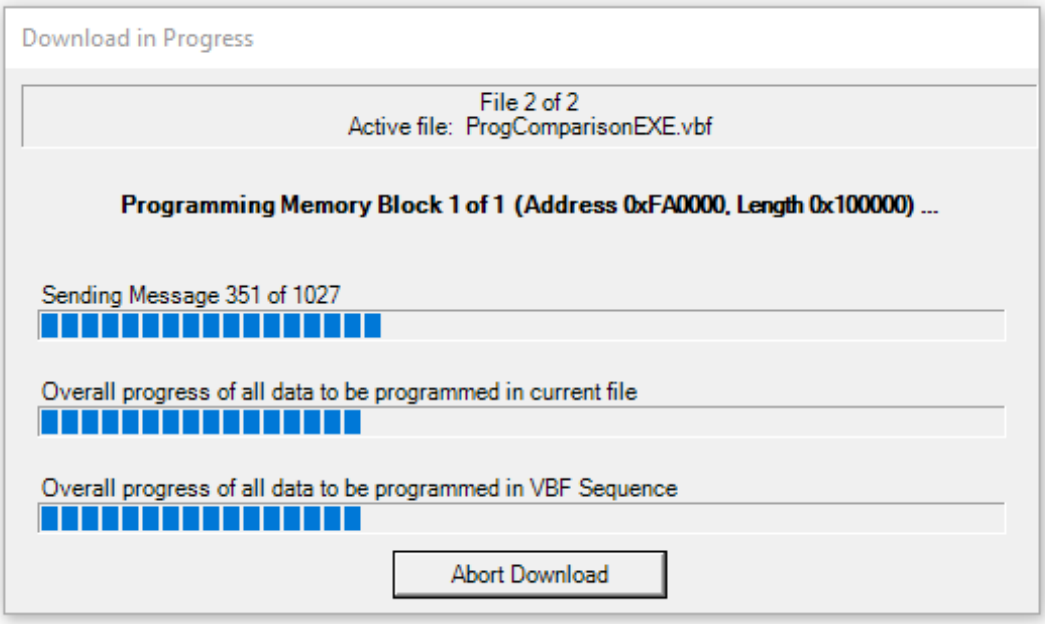


5.8 Download Sequence Now

To download the VBF sequence, once all the files are in the List of files to download and the Security Key loaded / created, select the "Download Sequence Now" button.



A window will appear, specifying the status of the download. Additionally, the display will contain information on the process of the download. If a problem occurred, the details appear on the Display section (displayed in red colored text).



Once the download is finished, DET performs a Functionally Addressed Hard Reset. This can be modified in "Options" on the Main Menu. To see how to modify this please refer to the Software Download tab section.

5.9 Programming a Subnode

DET can program subnodes if extra steps are performed.

5.9.1 Through a Gateway

5.9.1.1 VBF Files

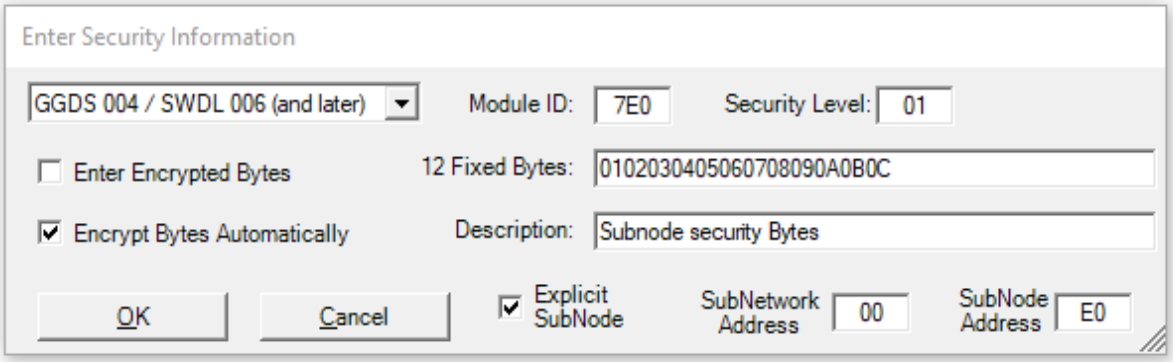
Open the VBF files in a text editor such as Notepad or Notepad++. The ecu_address must appear in the header. The data below is used only as an example. The 0x7E0 is the gateway address. The 0x00 and 0xE0 are the SubNetwork Address and the SubNode Address respectively.

```
// ecu_address or list
ecu_address = { 0x7E0, 0x00, 0xE0};
```

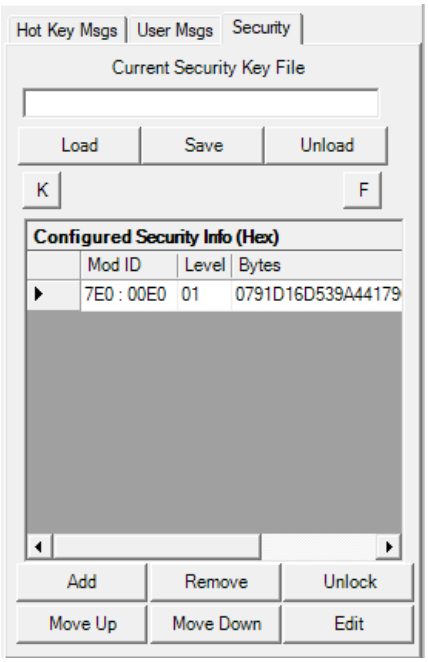
5.9.1.2 Security Bytes

On the Security tab, add the Security Info by clicking Add.

Enter the Security Info except check the Explicit SubNode box. The SubNetwork and SubNode boxes will appear. Enter the same information in these boxes as contained in the VBF files.



Click ok, and the Security Info tab should now look like this:



The Mod ID column now lists the SubNetwork and SubNodeAddresses.

See to continue programming.

5.9.2 On a Bench

5.9.2.1 VBF Files

When flashing a subnode on a bench and not through a gateway, the VBF files must contain only the gateway address and not the SubNetwork and SubNode Addresses. Here is an example:

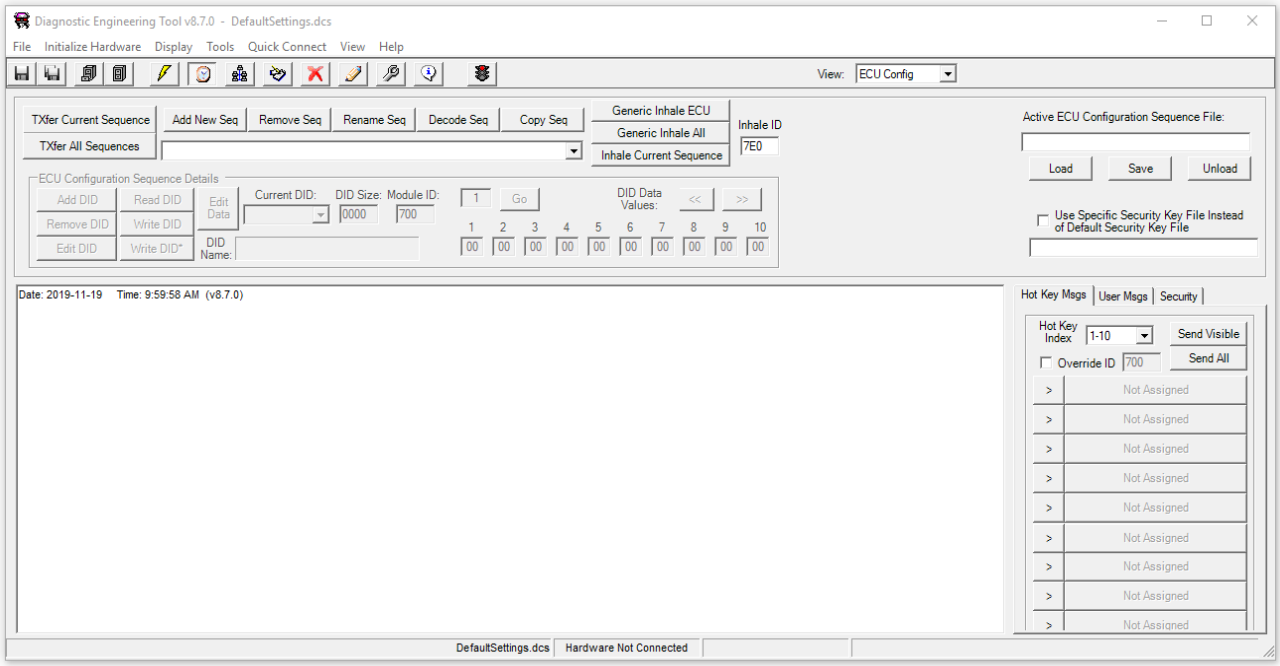
```
// ecu_address or list
ecu_address = { 0x7E0};
```

5.9.2.2 Change the ECU Transmission ID Offset

Go to Tools -> Options and click on the GMRDB tab. Enter the ECU Transmission Offset ID. In this case, it is 16. Click ok. See **Error! Reference source not found. Error! Reference source not found.** to continue programming.

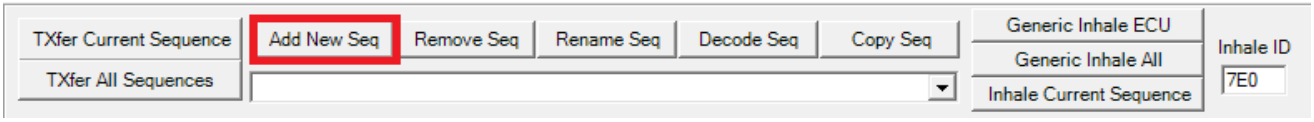
6 ECU Config View

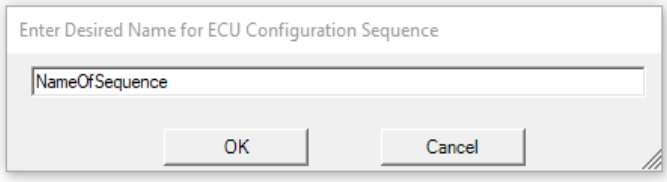
This view allow the users to create a Configuration Sequence, which is intended to inhale (if necessary) the information contained in User defined configurable DIDs, modify the information (if necessary) of configurable DIDs, and write the information to configurable DIDs.



6.1 Add New Sequence

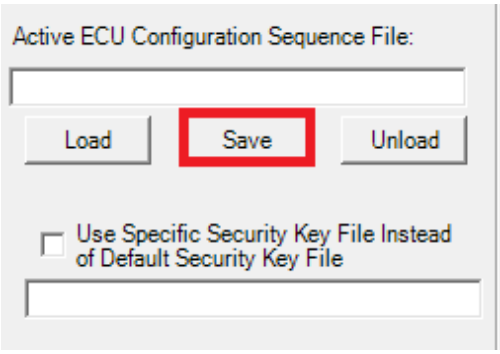
To add a new DID configuration sequence, click in the "Add New Sequence" button, introduce the name of the sequence and proceed to complete the information in the "ECU Configuration Sequence Details" section.





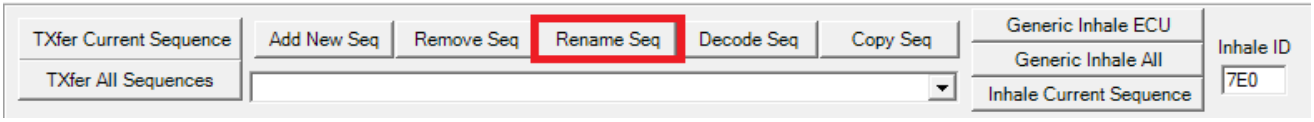
6.2 Save a Sequence

After introducing the information of the configurable DIDs in "ECU Configuration Sequence Details" (section 6.7), the sequence(s), can be saved (ECD extension) with:



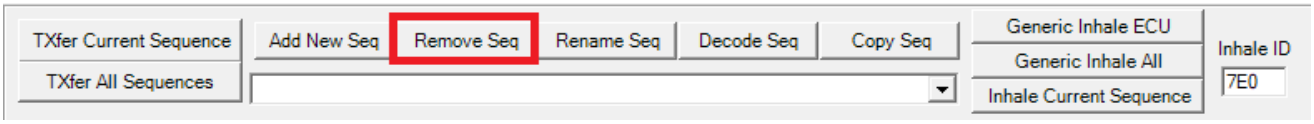
6.3 Remove Sequence

If a sequence needs to be removed, the "Remove Sequence" button should be used. This option will remove the sequence from the drop box list.



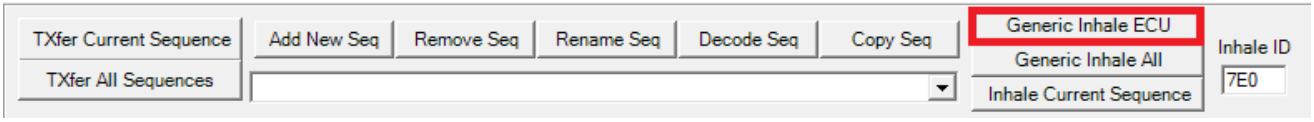
6.4 Rename Sequence

If a sequence needs to be renamed, the "Rename Sequence" button should be used. This option will open a new window that allows selecting a new name for the selected sequence on the drop box list.

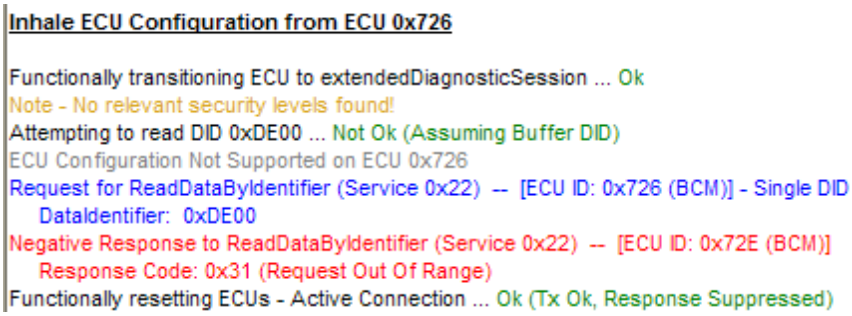


6.5 Generic ECU Configuration Inhale from ECU

Use this option to download the information contained in the generic ECU Configuration DIDs of the ECU selected on "Inhale Module ID" text box. The generic ECU Configuration DIDs (per ECU Configuration Spec) are in the range: DE00 – DEFF. Note that every time an inhale is performed, a NEW sequence is created which contains the data for all of the information that was inhaled.

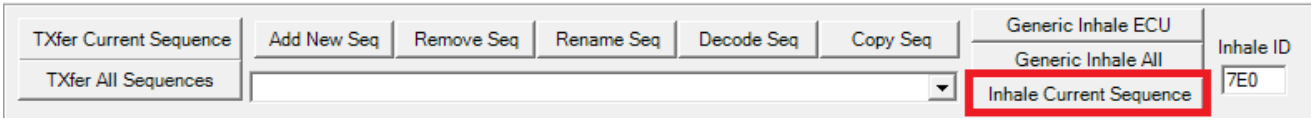


If the tool does not receive a positive response for the request to read these DIDs, the information is not displayed and the following warning is displayed instead:

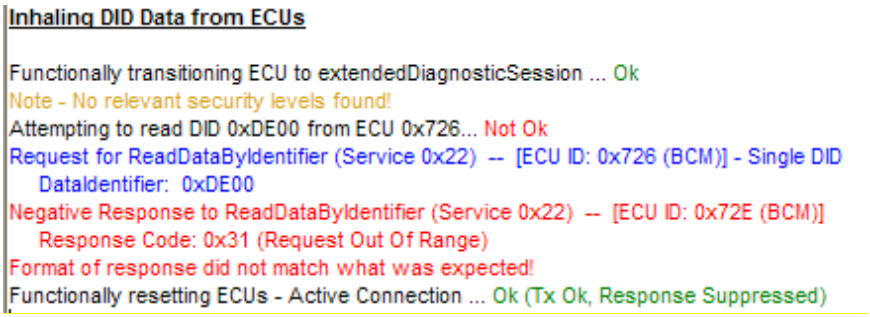


6.6 Inhale Current DID Data for Sequence

Use this option to inhale the information contained on all specific DIDs (those DIDs contained in the currently selected sequence). Because the DIDs in the sequence also contain the Module ID associated with the DID, this option can be used to inhale DIDs from more than one ECU on the network at once. Note that every time an inhale is performed, a NEW sequence is created which contains the data for all of the information that was inhaled.

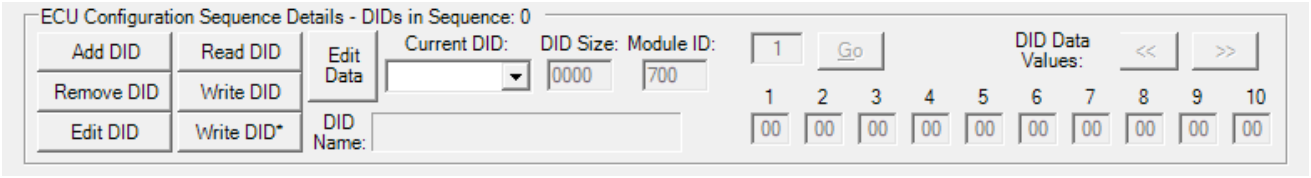


If the tool does not receive a positive response for the request to read the DID, the information is not displayed and the following warning is displayed instead:



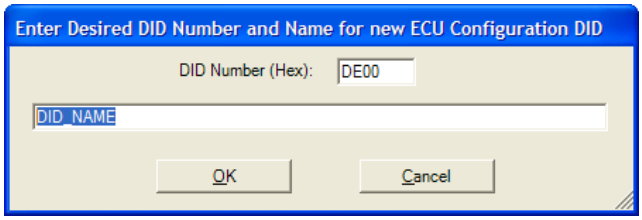
6.7 ECU Configuration Sequence Details

This section contains details on how DIDs are manually added to each configuration sequence. Once the DIDs are added, the user data values can be written to each DID or the information can be inhaled from the ECUs on the vehicle.

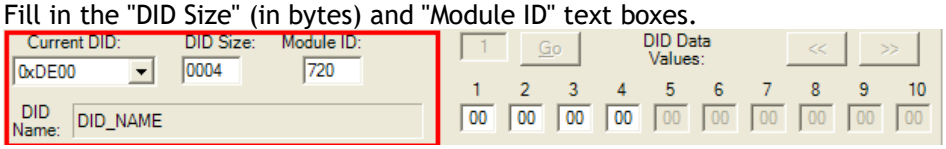


Steps to add a DID:

- Click the "Add DID" button, introduce the specific information of the configurable DID in the pop window and click the OK button.

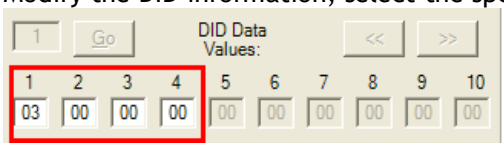


- Fill in the "DID Size" (in bytes) and "Module ID" text boxes.



Once all this information is complete, the following tasks can be performed:

- Inhale the DID information (see section 6.6). Again, this will create a new sequence containing all of the DIDs inhaled, but with the new data from the ECU(s) on the network.
- Modify the DID information, select the specific bytes to be updated, and manually modify them.



- Write new values (no matter what value the DID had before), following same procedure as step above. To transfer the new information to the ECU in the vehicle's network select the "Transfer ECU Configuration Sequence Now" button. Please refer to section 6.8.

6.8 Transfer ECU Configuration Sequence Now

This feature downloads into the module the information contained in the actively selected "ECU Configuration Sequence Details" section. It writes all of the information of the DIDs for the selected sequence in the drop box.

The transfer might not always be successful. The following are the most common reasons why an ECU transfer Configuration might fail:

- There is no Security Key, wrong Security Key level, or wrong challenge bytes for the Security Key.
- Error on the DID configuration: Wrong DID, wrong Size, wrong Module ID or information to be written is not supported.
- Pre-conditions to write to a specific DID are not met (Refer to Part 2 spec document, to identify the pre-conditions needed to write to each configurable DID).

7 Central Config View

This function is no longer supported with the DET tool, so no further information about the functionality of this function will be provided.

8 Generic UL/DL View

Information for this view is not yet added and will be included in a future release of this user manual.
TODO: Add this info

9 Auto SWDL View

The "Auto SWDL" view is used to automatically query specific DIDs and based upon the values returned by these DIDs, download a specific set of VBF files to the vehicle. In addition, the process can verify that a set of DIDs report a specific value after the VBF files are downloaded. The information used to drive this automated software download is contained within an XML file with an extension of .DVS (DID Driven and Verified SWDL). The DIDs to read, the data to compare, and the files to download are completely specified by the creator of the .DVS file.

A simple example of where this may be useful include replacing a specific calibration (e.g., -AB) with a new calibration (e.g., -AC), replacing a different calibration (e.g., -BC) with a different new calibration (e.g., -BD), but leaving an existing calibration (e.g., -BD) unchanged. Wildcards can also be used if desired so that any specific calibration path (e.g., -A*) can be replaced with a new calibration (e.g., -AF).

9.1 File Structure (.DVS)

The creator of the DVS file (which must be done manually today) can define a series of sequences that are evaluated by the Diagnostic Engineering Tool in the order they are found in the file. Whether or not the tool processes only the first sequence it matches or continues to the next depends upon an XML element in the DVS file of ALWAYS_ANALYZE_ALL. If this element is not present, or set to a value other than True, then the tool will only process the first sequence it matches. In either case, ordering of sequences is important as the tool may process only the first sequence it matches, or if all sequences or analyzed, the action performed by one sequence can affect the data returned and therefore whether a subsequent sequence is matched or not. Generically, each sequence within the DVS XML file is structured with the following hierarchy.

- SEQUENCE
  - NAME
  - PRE_SWDL
    - DID_DATA
      - ID
      - DATA_LENGTH
      - DATA
      - MASK
      - ECU_ID
      - VERIFY_NOMATCH
  - ACTION
  - VBS_INDEX
  - POST_SWDL
    - DID_DATA
      - ID
      - DATA_LENGTH
      - DATA
      - MASK
      - ECU_ID
      - VERIFY_NOMATCH
  - FINAL_TEXT_SUCCESS
  - FINAL_TEXT_FAIL
  - POPUP_SUCCESS
  - POPUP_FAIL

9.1.1 DVS Schema

Refer to Annex B for an XML schema (.xsd) that can be used to help ensure the structure of any DVS file is compliant. XML fields of ALWAYS_ANALYZE_ALL and VERIFY_NOMATCH were added in version 2 of the DVS schema.

Note that the ID, DATA_LENGTH, and ECU_ID can either be in base 16 (if prefixed by "0x") or base 10 (if not prefixed with "0x"). However, the DATA and MASK elements are required to always be in base 16 (no "0x" is included), and the number of ASCII characters shall always be twice as many as the numerical value that DATA_LENGTH decodes to. As an example, if DATA_LENGTH contains a value of 2, then DATA shall consist of ZZZZ where each Z is a value between 0-9 or A-F and will be interpreted as base 16 (i.e., valid values for DATA would be 0000, 5A5A, FFFF, etc.).

Note that at the time this document was written, there are no specific tools available to assist in the creation of DVS files and therefore these files must be created manually.

9.2 DVS Evaluation Flow

The NAME element of the sequence is text displayed to the user to describe the sequence that is being evaluated for a match. The PRE_SWDL element of the sequence contains the list of 0 to n DIDs including the 2 byte DID number, the data length to expect, the actual data to expect, an optional data mask, and the CAN ID of the ECU to read the DID from. The DET will read all DIDs under PRE_SWDL and only if ALL DIDs have an exact match to the data content (taking into account the MASK), then the ACTION associated with the sequence takes place. If ALL DIDs do not have an exact match (taking into account the MASK), then the next sequence will be evaluated to see if the PRE_SWDL DIDs match.

NOTE: Each DID can optionally have a VERIFY_NOMATCH field present. If the value in this field is True, then the DID will be considered matched only if the ECU returns a positive response with the expected DID number, expected data length, but with data that is at least one bit **different** from what is expected. If the field is not present, or contains any value other than true, the DID will be matched as normal. An example usage of this field in a sequence may be to have 2 DIDs, where the first DID checks the hardware using the normal match, and then the second DID programs the new software if it does not match the already expected value.

If no sequences have all of their PRE_SWDL DID data matched, there is the ability to contain a default sequence whose ACTION gets executed. After performing an ACTION for the matched or default sequence, the POST_SWDL element of the sequence is analyzed. This element may contain 0 to n DIDs in an XML structure exactly the same as the PRE_SWDL element. If the ACTION was successful and all POST_SWDL DIDs exactly match (taking into account the MASK) to the data content reported by the ECU, then the process will end with a result of success. Otherwise, the process will end with a result of fail.

Whether or not the tool processes only the first sequence it matches or continues to the next depends upon an XML element in the DVS file of ALWAYS_ANALYZE_ALL. If this element is not present, or set to a value other than True, then the tool will only process the first sequence where ALL PRE_SWDL DIDs are matched. Setting ALWAYS_ANALYZE_ALL to True may be useful when simplifying the scenario of programming the entire vehicle using the automated software download approach. Regardless, ordering of sequences is important as the tool may process only the first sequence it matches, or if all sequences or analyzed, the action performed by one sequence can affect the data returned and therefore whether a subsequent sequence is matched or not.

9.2.1 Data Mask

Before the DATA from the DVS file is compared byte by byte against the data returned in the DID by the ECU, both are logically ANDed with the MASK. This allows one to selectively set specific bits or bytes to "don't

cares". If the data MASK element is not present, it is assumed to be all 0xFF so that the DATA must exactly match.

As an example, consider the XML extract below:

```
<DID_DATA>
        <ID>0xF188</ID>
        <DATA_LENGTH>24</DATA_LENGTH>
        <DATA>424333542D3134433138342D414B000000000000000000000000</DATA>  <!--
BC3T-14C184-AK -->
        <MASK>FFFFFFFFFFFFFFFFFFFFFFFF00FFFFFFFFFFFFFFFFFFFF</MASK>
        <ECU_ID>0x726</ECU_ID>
</DID_DATA>
```

In this case, the data returned in DID 0xF188 by ECU 0x726 must be 24 bytes in length with a value of BC3T-14C184-A*. In this case, * is a don't care byte. In addition, the remaining bytes are required to be 0x00 (ASCII NULL).

9.3 DVS Actions

The ACTION associated with a sequence shall be one of the following:

- None = No Action
- SWDL = Automatic programming of one of the sequences defined in the currently loaded .VBS file.
- SendAllHotKeys = Send all defined hot keys (from 1 – 100) in the currently loaded .DCS file.
- ForcedFail = Force a failure due to unexpected state
- SWDLAndSendAllHotKeys = Automatic programming of one of the sequences defined in the currently loaded .VBS file, followed by sending all defined hot keys (from 1 – 100) in the currently loaded .DCS file.

9.3.1 None

If the ACTION is None, then the DVS elements of VBS_INDEX, POST_SWDL, FINAL_TEXT_FAIL, and POPUP_FAIL are not relevant. Ideally these should not be included in a sequence with an ACTION equal to None, but doing so will not cause issues. When an ACTION equal to None is performed, it simply causes FINAL_TEXT_SUCCESS to get displayed in green. This is normally used when the PRE_SWDL DIDs exactly match the desired state. FINAL_TEXT_SUCCESS may display information to the user to indicate that the ECU / vehicle software is up to date and that no further action is necessary. Furthermore, if POPUP_SUCCESS element is present and contains a value of "True", a dialog box will appear with a clear indication of the pass result.

9.3.2 SWDL

If the ACTION is SWDL, then VBS_INDEX element is required to exist and the tool will automatically attempt to download that index of the currently loaded VBS file. Note that a value of 1 corresponds to the first index in the currently loaded VBS file. Note that the referenced VBS sequence is downloaded as normal, so that any associated ECD files or KEY files are used as normal. If the requested index does not exist in the VBF file then the tool will give an error indicating that the index is not within range (note that no further sequences will be evaluated, but the POST_SWDL DIDs will be evaluated). After the SWDL of the corresponding set of VBF files is downloaded, the tool will then read any DIDs under POST_SWDL and verify the data values are reported exactly (taking into account the MASK) to further validate the download process. If any extra DIDs are desired for data logging purposes only (e.g., VIN or serial number), the DATA_LENGTH can be set to 0 so that no data comparison is done (alternatively the MASK can be set to all 00s). If both the SWDL and final DID value verification are successful, the DET will display FINAL_TEXT_SUCCESS in green. Furthermore, if POPUP_SUCCESS element is present and contains a value of "True", a dialog box will appear with a clear indication of the pass result. If either the SWDL action or the final DID verification are not successful, the DET will display FINAL_TEXT_FAIL in red. Furthermore, if POPUP_FAIL element is present and contains a value of "True", a dialog box will appear with a clear indication of the fail result.

9.3.3 SendAllHotKeys

If the ACTION is SendAllHotKeys, then the DVS elements of VBS_INDEX is not relevant and ideally should not be included in a sequence with an ACTION equal to SendAllHotKeys. When an ACTION equal to SendAllHotKeys is performed, the DET will loop through all 100 possible user defined Hot Keys from the currently loaded DCS file and send in order (from 1 to 100). After the transmission of all defined Hot Keys, the tool will then read any DIDs under POST_SWDL and verify the data values are reported exactly. If any extra DIDs are desired for data logging purposes only (e.g., VIN or serial number), the DATA_LENGTH can be set to 0 so that no data comparison is done. If both the Hot Key transmissions and final DID value verification are successful, the DET will display FINAL_TEXT_SUCCESS in green. Furthermore, if POPUP_SUCCESS element is present and contains a value of "True", a dialog box will appear with a clear indication of the pass result. If either the SWDL action or the final DID verification are not successful, the DET will display FINAL_TEXT_FAIL in red. Furthermore, if POPUP_FAIL element is present and contains a value of "True", a dialog box will appear with a clear indication of the fail result.

9.3.4 SWDLandSendAllHotKeys

If the ACTION is SWDLAndSendAllHotKeys, then the tool effectively performs the actions of both SWDL (see 9.3.2) as well as SendAllHotKeys (see 9.3.3). The VBS_INDEX element is required to exist and the tool will automatically attempt to download that index of the currently loaded VBS file, followed by sending all hot keys. After performing these actions, the POST_SWDL DIDs will be validated as normal.

9.3.5 ForcedFail

If the ACTION is ForcedFail, then the DVS elements of VBS_INDEX, POST_SWDL, FINAL_TEXT_SUCCESS, and POPUP_SUCCESS are not relevant. Ideally these should not be included in a sequence with an ACTION equal to ForcedFail, but doing so will not cause issues. When an ACTION equal to ForcedFail is performed, it simply causes FINAL_TEXT_FAIL to get displayed in red. This is normally used for the default sequence when none of the expected sequences are met and it is unclear how to proceed. FINAL_TEXT_FAIL may display information to the user to indicate that the ECU / vehicle was not expected to be in this state and that it requires manual investigation. Furthermore, if POPUP_FAIL element is present and contains a value of "True", a dialog box will appear with a clear indication of the fail result.

9.4 Example DVS

9.4.1 Example DVS File #1

DVS files can be as simple or as complex as desired. An example of simple case is where the need exists to always download a fixed set of files, but verify that certain DID values were updated following the software download.

Essentially, the way the example below works is as follows.

- Only a default sequence exists, so it always gets matched. The action is to always perform a SWDL of index 1 of the loaded VBS file. Following the software download, a DID is read to further verify the SWDL was successful.

```
<DVS VERSION='1.0' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xlink='http://www.w3.org/1999/xlink' xsi:noNamespaceSchemaLocation='DVS-001.xsd'>
<!-- This simple DVS file can be used to always perform a SWDL of the first index of the VBS
file and verify DID is updated with specific value -->
<DEFAULT_SEQUENCE>
          <NAME>Any sequence name you want</NAME>
```

```
            <ACTION>SWDL</ACTION>
            <VBS_INDEX>1</VBS_INDEX>
            <POST_SWDL>
               <DID_DATA>
                      <ID>0xF188</ID>
                      <DATA_LENGTH>24</DATA_LENGTH>
`                     <!--  BC3T-14C184-BD -->
                      <DATA>424333542D3134433138342D424400000000000000000000</DATA>
                      <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
            </POST_SWDL>
            <FINAL_TEXT_SUCCESS>Any success text you want</FINAL_TEXT_SUCCESS>
            <FINAL_TEXT_FAIL>Any fail text you want</FINAL_TEXT_FAIL>
            <POPUP_SUCCESS>TRUE</POPUP_SUCCESS>
            <POPUP_FAIL>TRUE</POPUP_FAIL>
</DEFAULT_SEQUENCE>
</DVS>
```

9.4.2 Example DVS File #2

The example in this section is an actual DVS file that has been used to update the message list in a scenario where many parts (but not all) were accidentally shipped with old message lists and a parking lot flash was necessary. There was also a desire to not waste time downloading those ECUs which were already up to date. In addition, the single DVS file was used across two platforms and the message list was different for the two platforms.

Essentially, the way the example below works is as follows.

- 1st sequence looks for DID 0xF188 = BC3T-14C184-BD and DID 0xF108 = BT4T-14C08-AB. If this matched, then it was desired state so no action was taken.
- 2nd sequence looks for DID 0xF188 = BC3T-14C184-BD and if this matched, then VBS index 1 was downloaded. VBS index 1 consisted of the SBL and file BT4T-14C08-AB. In other words, if "-AB" was not in 0xF10A (since sequence 1 was not matched), but 0xF188 = "-BD", then always download "-BD" message list. The post SWDL DIDs were to verify F188 = BC3T-14C184-BD and F108 = BT4T-14C08-AB after the download.
- The 3rd and 4th sequences work very similar to the 1st and 2nd, but verified a different strategy level and corresponding message list.

```
<DVS VERSION='1.0' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xlink='http://www.w3.org/1999/xlink' xsi:noNamespaceSchemaLocation='DVS-001.xsd'>

<!-- This sequence looks for DID F188 = BC3T-14C184-BD and DID F108 = BT4T-14C08-AB.
If so, no action needed -->
<SEQUENCE>
            <NAME>Checking for desired U38X. DID F188 = BC3T-14C184-BD and DID F108 = BT4T-
14C08-AB.</NAME>
            <PRE_SWDL>
<!-- All Pre SWDL DIDs are read to get a match. If ALL match, action is performed -->
               <DID_DATA>
                      <ID>0xF188</ID>
                      <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C184-BD -->
                      <DATA>424333542D3134433138342D424400000000000000000000</DATA>
                      <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
               <DID_DATA>
                      <ID>0xF108</ID>
                  <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BT4T-14C408-AB -->
                      <DATA>425434542D3134433430382D414200000000000000000000</DATA>
```

```
                    <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
          </PRE_SWDL>
          <ACTION>NONE</ACTION>
          <FINAL_TEXT_SUCCESS> U38X Strategy and FNOS Postbuild are up to date.  No further
action necessary.  </FINAL_TEXT_SUCCESS>
          <POPUP_SUCCESS> true </POPUP_SUCCESS>
          <POPUP_FAIL> true </POPUP_FAIL>
</SEQUENCE>

<SEQUENCE>
          <NAME>Checking for U38X.  DID F188 = BC3T-14C184-BD.  Download FNOS RP Postbuild
(BT4T-14C08-AB) if this occurs.</NAME>
          <PRE_SWDL>
               <DID_DATA>
                    <ID>0xF188</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C184-BD -->
                    <DATA>424333542D3134433138342D424400000000000000000000</DATA>
                    <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
          </PRE_SWDL>
          <ACTION>SWDL</ACTION>
          <VBS_INDEX>1</VBS_INDEX>
          <POST_SWDL>
               <DID_DATA>
                    <ID>0xF188</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C184-BD -->
                    <DATA>424333542D3134433138342D424400000000000000000000</DATA>
                    <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
            <DID_DATA>
                    <ID>0xF108</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BT4T-14C408-AB -->
                    <DATA>425434542D3134433430382D414200000000000000000000</DATA>
                    <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
          </POST_SWDL>
          <FINAL_TEXT_SUCCESS> U38X Strategy and FNOS Postbuild are up to date.  NO further
action necessary.</FINAL_TEXT_SUCCESS>
          <FINAL_TEXT_FAIL> Error occurred during download of U38X FNOS Postbuild.  Retry
or manual investigation required! </FINAL_TEXT_FAIL>
          <POPUP_SUCCESS> true </POPUP_SUCCESS>
          <POPUP_FAIL> true </POPUP_FAIL>
</SEQUENCE>


<!-- This sequence looks for DID F188 = BC3T-14C184-AK and DID F108 = BC3T-14C408-AA.
If so, no action needed -->
<SEQUENCE>
          <NAME>Checking for desired P473. DID F188=BC3T-14C184-AK and DID F108=BC3T-
14C408-AA.</NAME>
          <PRE_SWDL>
               <DID_DATA>
                    <ID>0xF188</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C184-AK -->
                    <DATA>424333542D3134433138342D414B000000000000000000000</DATA>
                    <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
               <DID_DATA>
```

```
                    <ID>0xF108</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
                <!--  BC3T-14C408-AA -->
                    <DATA>424333542D3134433430382D414100000000000000000000</DATA>
                   <ECU_ID>0x726</ECU_ID>
                </DID_DATA>
            </PRE_SWDL>
            <ACTION>NONE</ACTION>  <!-- Allowed values for ACTION are NONE, SWDL, and
FORCEDFAIL -->
            <FINAL_TEXT_SUCCESS>P473 Strategy and FNOS Postbuild are up to date.  No further
action necessary.</FINAL_TEXT_SUCCESS>
            <POPUP_SUCCESS> true </POPUP_SUCCESS>
            <POPUP_FAIL> true </POPUP_FAIL>
</SEQUENCE>

<SEQUENCE>
            <NAME>Checking for P473 DID F188 = BC3T-14C184-AK.  Download FNOS RP Postbuild
(BC3T-14C408-AA) if this occurs.</NAME>
            <PRE_SWDL>
               <DID_DATA>
                    <ID>0xF188</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C184-AK -->
               <DATA>424333542D3134433138342D414B000000000000000000000</DATA>
               <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
            </PRE_SWDL>
            <ACTION>SWDL</ACTION>
            <VBS_INDEX>2</VBS_INDEX>
            <POST_SWDL>
               <DID_DATA>
                    <ID>0xF188</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C184-AK -->
                    <DATA>424333542D3134433138342D414B000000000000000000000</DATA>
                    <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
               <DID_DATA>
                    <ID>0xF108</ID>
                    <DATA_LENGTH>24</DATA_LENGTH>
               <!--  BC3T-14C408-AA -->
                   <DATA>424333542D3134433430382D414100000000000000000000</DATA>
                   <ECU_ID>0x726</ECU_ID>
               </DID_DATA>
            </POST_SWDL>
            <FINAL_TEXT_SUCCESS>P473 Strategy and FNOS Postbuild are up to date.  NO further
action necessary.</FINAL_TEXT_SUCCESS>
            <FINAL_TEXT_FAIL>Error occurred during download of U38X Strategy and FNOS
Postbuild.  Retry or manual investigation required!</FINAL_TEXT_FAIL>
            <POPUP_SUCCESS> true </POPUP_SUCCESS>
            <POPUP_FAIL> true </POPUP_FAIL>
</SEQUENCE>

<DEFAULT_SEQUENCE>
            <NAME>Module is not reporting BC3T-14C184-BD or BC3T-14C184-AK.  Not good!</NAME>
            <ACTION>FORCEDFAIL</ACTION>
            <FINAL_TEXT_FAIL>Module was not expected to be in this state.  Requires retry or
manual investigation.</FINAL_TEXT_FAIL>
            <POPUP_SUCCESS> true </POPUP_SUCCESS>
            <POPUP_FAIL> true </POPUP_FAIL>
</DEFAULT_SEQUENCE>

</DVS>
```

10 PATS Diag View

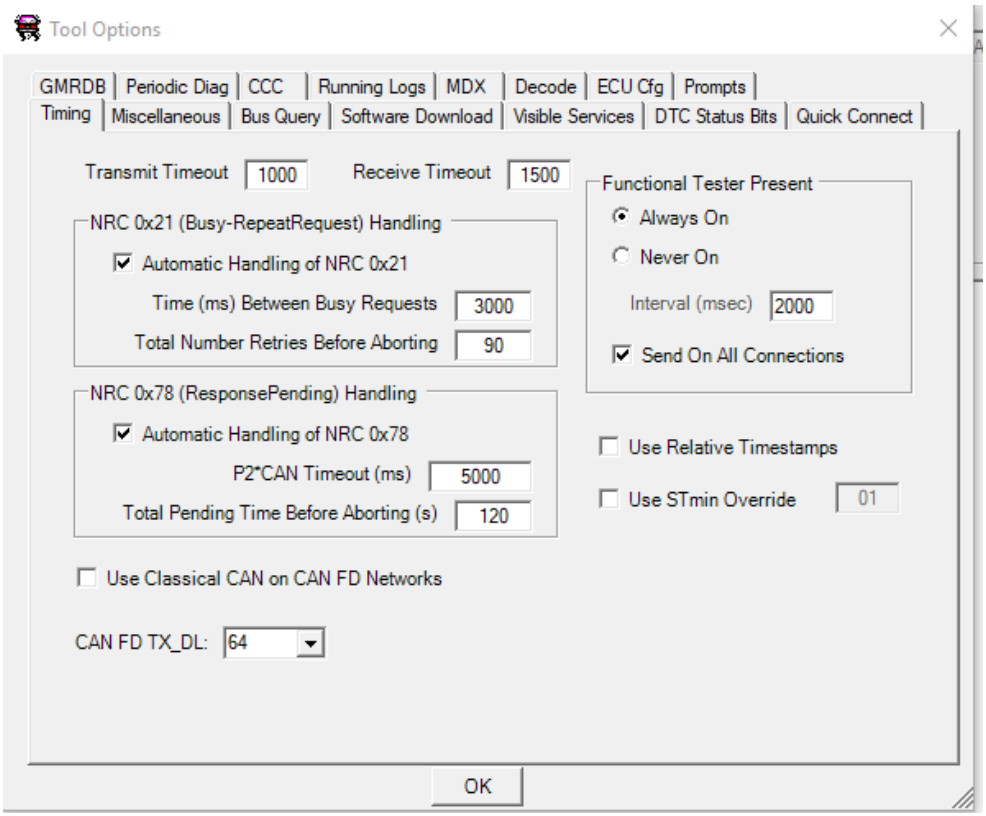
Information for this view is not yet added and will be included in a future release of this user manual.

11 Configurable Tool Options

This tool provides access to several configurable options within the DET. The tool is available at: Main Menu > Tools > Options or by selecting the wrench icon in the toolbar.

11.1 Timing Tab

The timing tab contains the configuration of the diagnostics timing parameters for the DET tool.



11.1.1 Transmit Timeout

Time the tool will keep trying to send a frame on the network prior to quitting. Default value is 1000ms.

11.1.2 Receive Timeout

Time the tool will keep waiting for a diagnostic response. Default value is 500ms.

11.1.3 Automatic Handling of NRC 0x21

If enabled, when a NRC 0x21 (busyRepeatRequest) is received as a response to a diagnostic message request, the DET will repeat the request "Total Number Retries Before Quitting" times with a delay between requests of "Time (ms) Between Busy Requests". By default, the tool will retry 90 times with a delay of 3 seconds in between for a total time of 270 seconds if the ECU responds to each request with a NRC 0x21. This is typically useful when requesting results to a routine such as On-Demand Self-Test immediately after starting the routine. If this feature is disabled, the DET will not automatically retry a request for which a NRC 0x21 is received.

11.1.3.1 Time (ms) Between Busy Requests

Time the DET waits, after the re-transmission of the request that was responded to with a NRC 0x21. This parameter is only available for its configuration when "Automatic Handling of NRC 0x21" is enabled.

11.1.3.2 Total Number Retries Before Quitting

DET retries the transmission of the request that was responded to with a NRC 0x21, when "Automatic Handling of NRC 0x21" is enabled. The number of retries before the tool stops automatically resending the request is defined by this parameter.

11.1.4 Automatic Handling of NRC 0x78

NRC 0x78 (RequestCorrectlyReceived-ResponsePending) is used when the action associated with the diagnostic request takes longer than P2can time (e.g., 50ms) to complete. When an ECU responds with NRC 0x78, it is required to continue to respond periodically (e.g., typically every 5000ms) until a final response is complete. When "Automatic Handling of NRC 0x78" is enabled, DET will continue waiting for the information requested after the reception of a NRC 0x78 according to the parameters below. If this is not enabled, the DET will treat the NRC 0x78 as the final response and will not continue to wait for an additional response.

11.1.4.1 P2*CAN Timeout (ms)

P2*CAN is the performance requirement for the ECU to either send another NRC 0x78 or send the final response message after the transmission of a negative response message with response code 0x78 hex (enhanced response timing). If the ECU exceeds this user configurable timeout then the DET will assume no further response is coming and will not continue to wait.

11.1.4.2 Total Pending Time Before Quitting (s)

This is the overall time the DET will continue to wait for a final response when NRC 0x78 handling is active. This timeout comes into play if the ECU is periodically sending the NRC 0x78, but is doing so for a prolonged period of time. The biggest usage of this timeout is to prevent an ECU that is stuck in an infinite loop of periodic NRC 0x78s from locking up the tester.

11.1.5 Functional Tester Present

Enables or disables the functionally addressed tester present message (0x3E) sent by the DET to the ECUs. This message is intended to reset ECU's session timers to prevent all ECUs from timing out of any active diagnostic session and any active time functionality, without causing any other diagnostic functionality to occur.

If "Tester Present" is configured as "Never On", the periodic message 0x3E will not be sent. If "Tester Present" is configured as "Always On", the periodicity of its transmission can be modified in the "Interval (msec)" section.

"Send On All Connections" option, will allow to send the tester present on all connections, not only the active one. This allows sending the tester present message in more than one network at a time when using quick connects.

11.1.6 Use Relative Timestamps

When selected, the relative timestamp feature displays the time difference between the frame and the previously displayed frame. This affects both the message frame log as well as the generic bus monitor and may be useful when monitoring the period of frames or if troubleshooting timing issues. If this is not selected, then the absolute timestamp is displayed.

11.1.7 Use STmin Override

Whenever a segmented request is sent from the DET to an ECU, the ECU responds with a flow control frame that contains a minimum separation time parameter (STmin). The usage of this parameter is described in ISO 15765-2, but essentially reflects the minimum time the DET must wait between transmitting consecutive frames during the segmented request. If this STmin override feature is selected, the DET will be have "as if" the ECU returned this STmin value instead of using the actual STmin value the ECU returned. The normal use case of this parameter is to either speed up the download on an ECU that reports a non-zero STmin yet can truly handle back to back consecutive frames, or to slow down a download where a non-compliant ECU reports an STmin time that it can not actually handle. For example, if the ECU normally returns a STmin of 1ms in the programmingSession, this override value could be set to 0ms to shorten the minimum wait time and therefore speed up the download. Conversely, this override value could be set to 2ms which would effectively increase the minimum wait time and therefore slow down the download. STmin override values can also use the values that represent microsecond intervals from ISO 15765-2 (e.g., $F3 = 300 microseconds, etc.).

11.1.8 Use Classical CAN on CAN FD Networks

Normally, when DET is connected to a CAN FD network it will send all requests as CAN FD and expect CAN FD responses. When “Use Classical CAN on CAN FD Networks” is enabled DET will send all requests on the CAN FD network as Classical CAN and expect all responses to be Classical CAN.

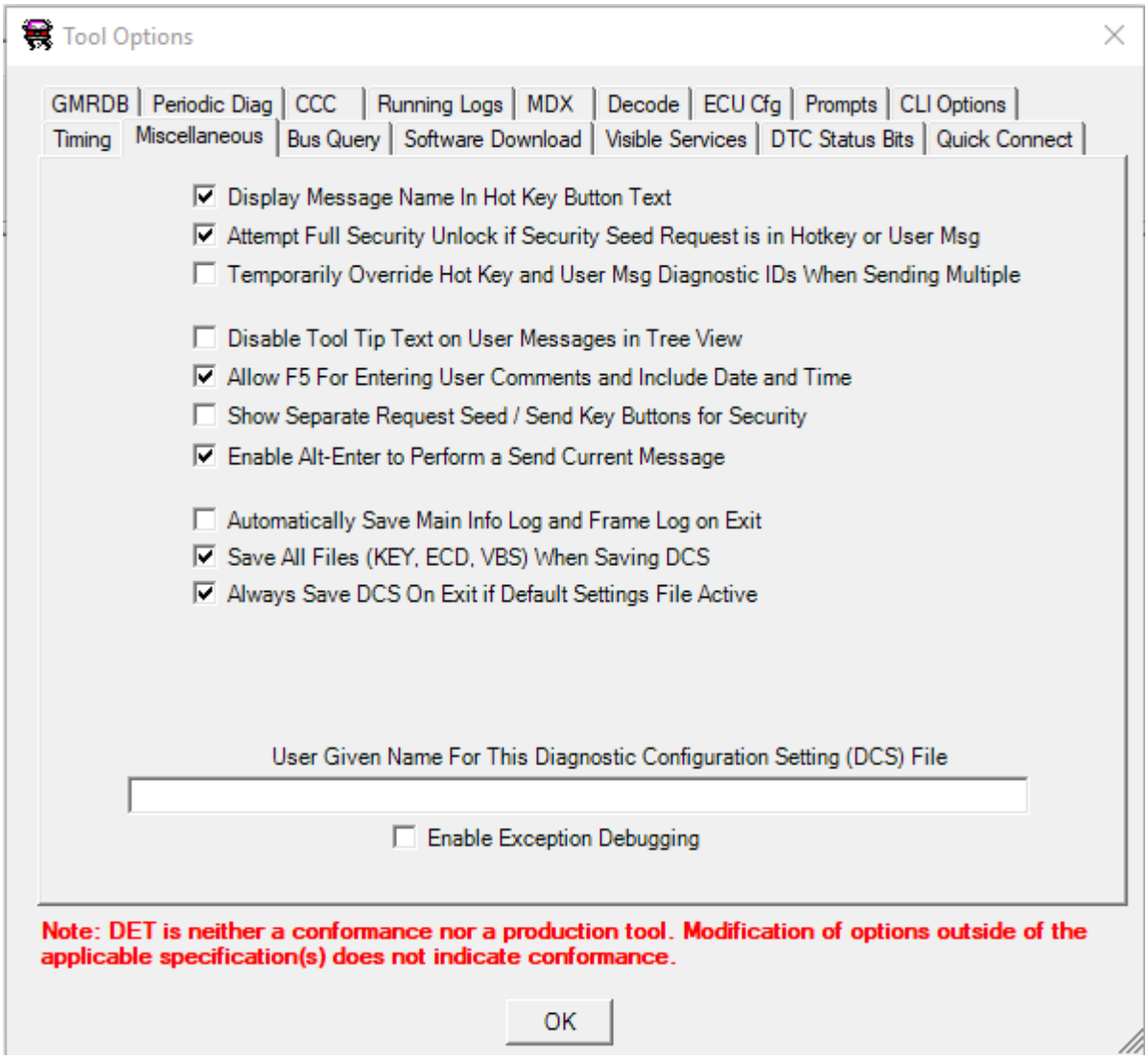
Note: When DET is physically connected to a CAN FD network and it is desired to send / received Classical CAN messages this feature must be used instead of connecting as a Classical CAN.

11.1.9 CAN FD TX_DL

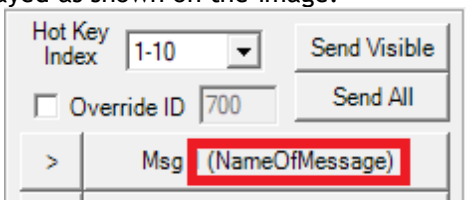
This sets this maximum size CAN FD frame size (in bytes) for transmitting single frame messages or consecutive frames. Normally, the maximum size of 64 bytes is uses, but if a another maximum size is desired this allows setting sizes of 8, 12, 16, 20, 24, 32, 48, or 64 bytes.

11.2 Miscellaneous Tab

This section includes different configuration options on the way DET displays certain information and how it behaves it certain situations.



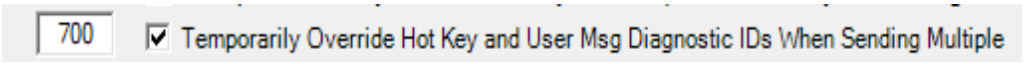
Display Message Name In Hot Key Button Text: A name can be assigned to each Hot Key button, the name assigned to the button will be displayed as shown on the image:



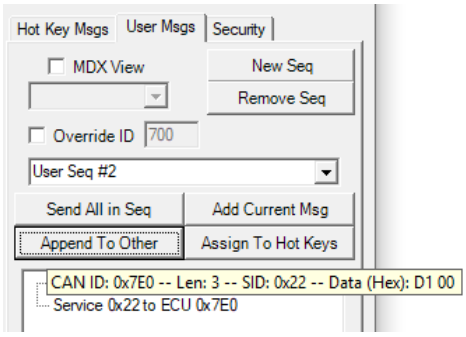
Attempt Full Security Unlock if Security Seed Request is in Hotkey: If this option is checked, then a seed request in the hotkey will not simply send the request message, but will calculate the key and send the response automatically based upon information within the loaded security key (.key) file. This is useful especially when sending all hotkeys whether manually or through the command line interface and security needs to be handled automatically.

Temporarily Override Hot Key and User Msg Diagnostic IDs When Sending Multiple:

If selected, this will temporarily override the diagnostic CAN ID within all hot key messages and user messages with the user selected CAN ID here (only when sending multiple at once).

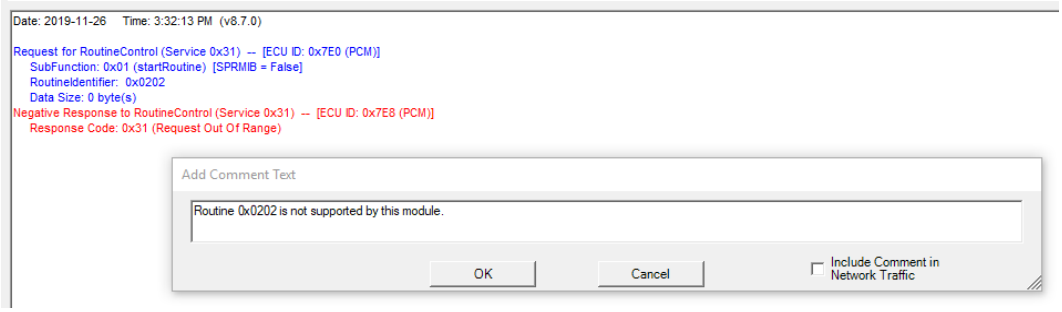


Disable Tool Tip Text on User Messages in Tree View: When placing the cursor over the "User Messages" in the tree view (see image below), a Tip Text will be displayed, displaying specific information of the message. If this Checkbox is enabled, the "Tip Text" message will not be displayed.

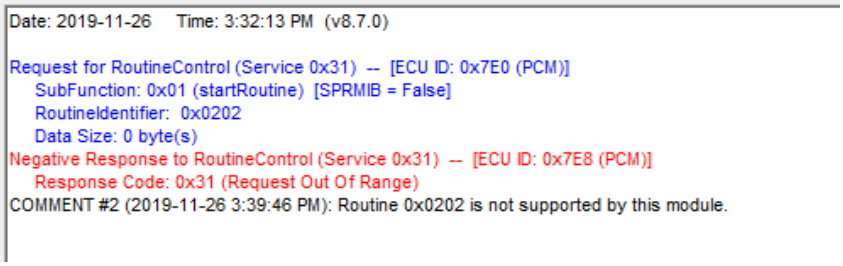


Allow F5 For Entering User Comments and Include Date and Time: Allows F5 to pop up the user comment dialog and includes the date and time when adding the comment to trace windows.

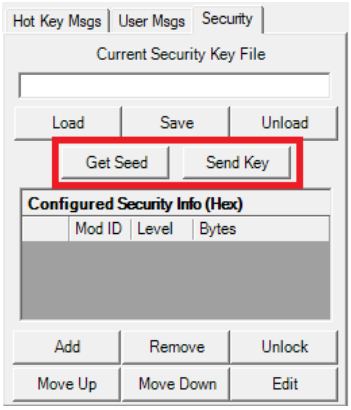
Pressing F5 will bring up a pop up window. User comments can be entered into the text box. To include the comments in the network traffic, check the box on the lower right.



After selecting OK, the comment appears in the trace window, along with the date and time.



Show Separate Request Seed / Send Key Buttons for Security: Add two buttons to the Security tab, to request the seed and send the new key. This is primarily useful when performing detailed Part 2 level testing and other diagnostic requests / data need to be interjected between a seed request and its corresponding key submittal.



Enable Alt-Enter to Perform a Send Current Message: Selecting this allows a user to press Alt-Enter to perform the equivalent of hitting the Send Msg button in the toolbar (lightning bolt).

Automatically Save Main Info Log and Frame Log on Exit: Selecting this allows automatic saving of the main info log and frame log in the My Documents \ Ford NetCom \ Diagnostic Engineering Tool \ Log Files direction when the program is exited.

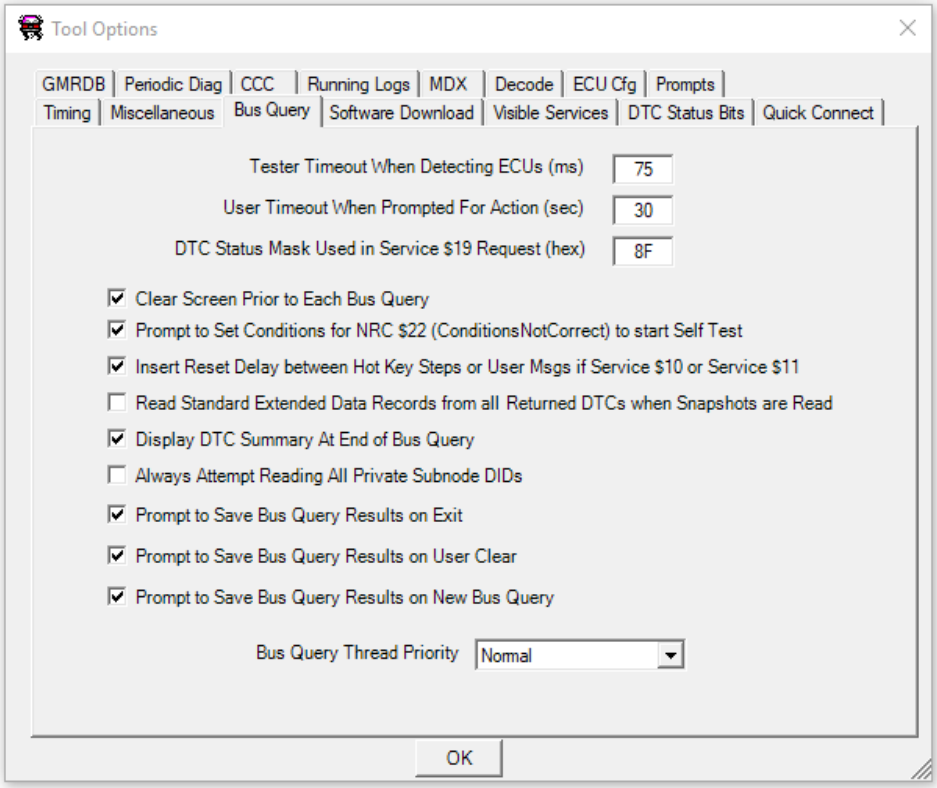
Save All Files (KEY, ECD, VBS) When Saving DCS: If the feature is enabled, when saving a Diagnostic Configuration Settings file (see section 12.2.3), the currently loaded KEY file, ECD file, and VBS file will also be saved so that any changes made will be kept (all this information will be available when loading the DCS file). In CLI mode, these files will not be saved.

Always Save DCS On Exit if Default Settings File Active: If the feature is enabled; when exiting DET and the Default DCS file is loaded, all the changes will be automatically saved into the Default DCS file. This is useful when the user desires to always keep their changes they made without having to remember to save. In CLI mode, these changes will not be saved.

Enable Exception Debugging: Enables exception debugging for code failures on the DET. By default it is disabled. This is not typically useful to the normal user but may provide useful information to the DET author (e.g., the function name of where an exception was handled) when unexpected operation / behavior is reported.

11.3 Bus Query Tab

Contain the options to configure the Bus Query tool.



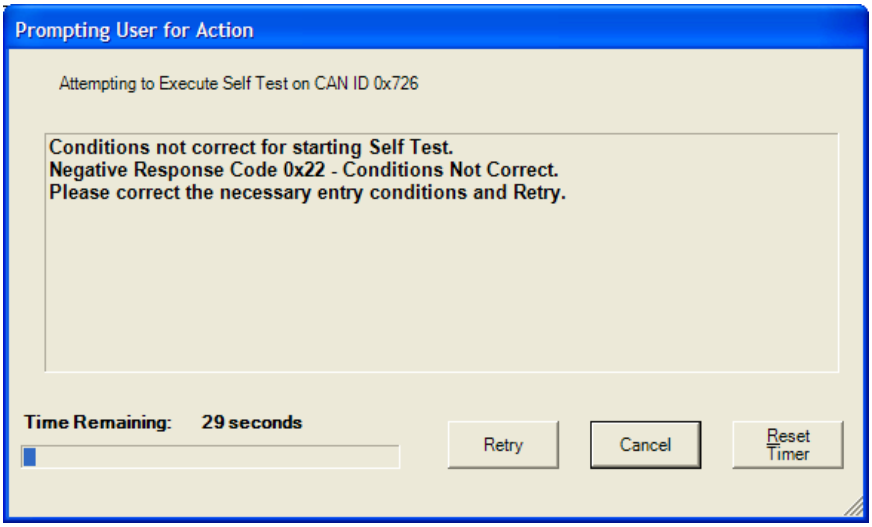
Tester Timeout When Detecting ECUs (ms): The "Query Bus" tool has very options where it generically detects all ECUs on the network by sending requests to each possible ECU request CAN ID in the 0x700 – 0x7FF range. This value is the timeout that the DET waits in receiving a response before it moves on to the next possible ECU ID. As ECUs are required to respond within 50ms, this should not need to be changed unless non-compliant ECUs are on the network.

User Timeout When Prompted For Action (sec): Some tasks within the "Query Bus Tool" require that the user perform different actions (e.g. while executing a Self-Test, the pre-conditions might not be correct so DET will ask the user to set the correct conditions on the vehicle to execute the Self-Test). A window will be displayed, asking for the user to perform the corresponding action, a timer countdown (based upon this value), and a button to retry or cancel (before the timer expires).

DTC Status Mask Used in Service $19 Request (hex): When using the "Request Continuous DTCs" of the "Query Bus Tool", the mask the tool we use when sending the Service 0x19 request will be the one defined in here. By default the mask is $8F which matches what is normally sent by Ford service and EOL tools.

Clear Screen Prior to Each Bus Query: Automatically clears the screen when performing a new bus query if selected.

Prompt to Set Conditions for NRC $22 (ConditionsNotCorrect) to start Self Test: If an ECU on the network responds with NRC 0x22 after requesting to start routine 0x0202 (Self-Test), DET can prompt the user to set the correct pre-conditions on the ECU to execute this routine, and then will resend the request.



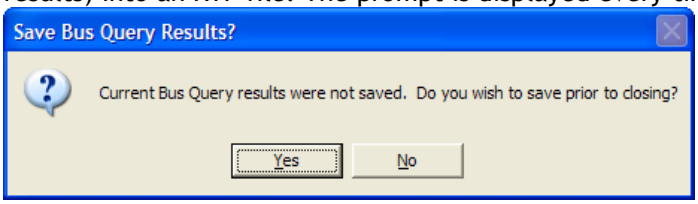
Insert Reset Delay between Hot Key Steps if Service $10 or Service $11: If selected, this automatically inserts a 1000ms delay between any service 0x10 or 0x11 request and the following request when sending hotkeys through an automated function. Because service 0x10 and 0x11 have the capability to cause a reset, the intent of this option is to account for the 500ms period where ECUs are not required to respond to requests after they perform a reset.

Read Standard Extended Data Records from all Returned DTCs when Snapshots are Read: This will attempt to read the standard Ford extendedDataRecords during bus query when reading DTCs and snapshot data option is selected.

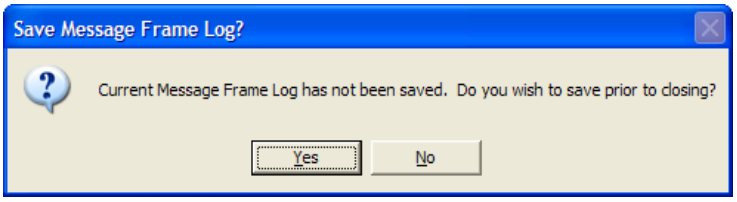
Display DTC Summary At End of Bus Query: This will display a DTC summary at the end of a bus query if DTCs are read.

Always Attempt Reading All Private Subnode DIDs: This will attempt to read all possible private sub node DIDs when standard DIDs are read instead of quitting after the first unsupported one.

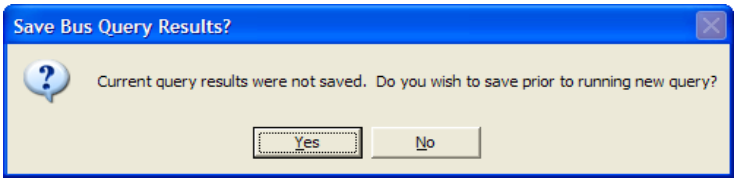
Prompt to Save Bus Query Results on Exit: Prompts if the user wants to save the "Query Bus" results (display results) into an RTF file. The prompt is displayed every time the "Query Bus tool" is closed.



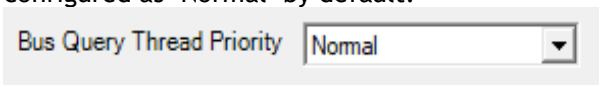
Prompt to Save Bus Query Results on User Clear: If the option is enabled, every time a user wants to clear the information using the "Clear" option in the Menu of the tool, a pop window will ask the user if the results should be saved or not.



Prompt to Save Bus Query Results on New Bus Query: If the option is enabled, every time a new query is performed, "Query Bus tool" will ask if the results need to be saved.



Bus Query Thread Priority: Assigns resources of the computer, depending on the priority this tool has. It is configured as "Normal" by default.

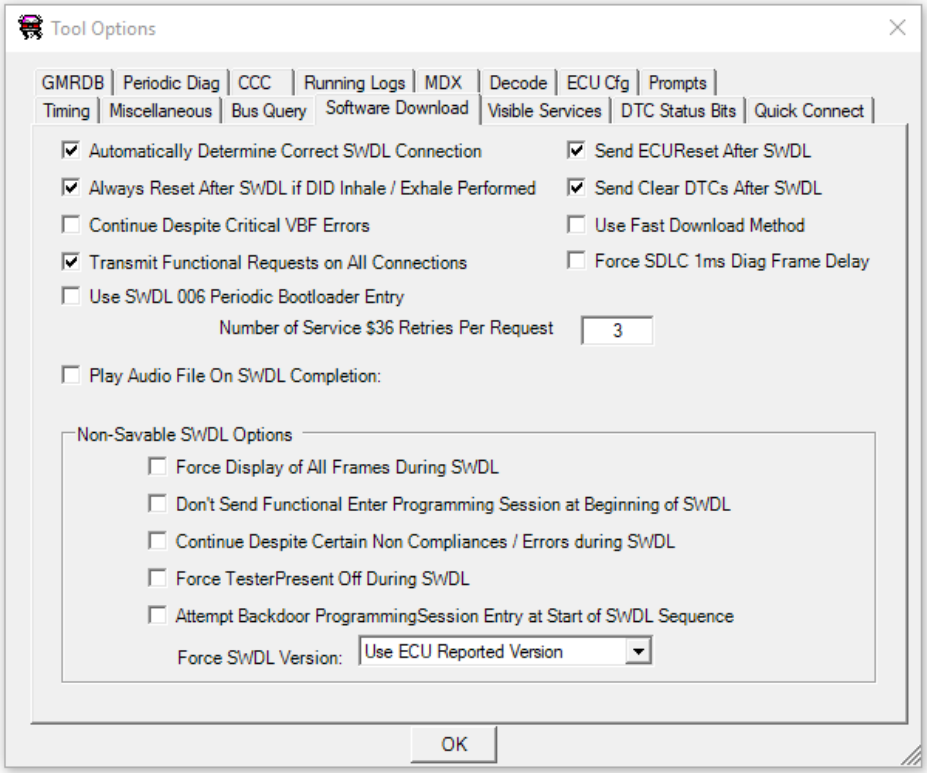


Available options are:

- Lowest
- Below Normal
- Normal
- Above Normal
- Highest

11.4 Software Download Tab

This section allows configuring different options of the Software Download section.



11.4.1 Software Download Configuration

Automatically Determine Correct SWDL Connection: When this option is checked, the DET will search for the ECU it is programming on all connections. The main use case is for when users always connect to all

vehicle networks, and do no want to have to change to the correct connection when downloading ECUs on different networks or want to have a single software download sequence with ECUs on multiple networks.

Always Reset After SWDL if DID Inhale / Exhale Performed: If checkbox "Inhale DIDs Prior to SWDL and Restore After Based Upon ECD File", is enabled in "SWDL View", a Functionally Hard Reset will be performed after the SWDL.

Continue Despite Critical VBF Errors: DET performs a validation of the VBF files before starting the software download. While some errors do not normally prevent a SWDL (e.g., filename does not match software part number in file), critical errors such as incorrect file checksums, incorrect block checksums, etc. normally prevent a SWDL from occurring. If this checkbox is checked, the tool will attempt to download the VBF files despite these errors. This of course doesn't mean a successful download will occur.

Transmit Functional Requests on All Connections: When this option is checked, functional requests (e.g., enter programmingSession, ECU reset, clear DTCs) during the SWDL will be transmitted on all connections. This is especially useful to keep the entire vehicle (all networks) in the programmingSession to prevent DTCs during the SWDL process.

Use SWDL 006 Periodic Bootloader Entry: If selected, tool will send a periodic functional enter programmingSession request approximately every 100ms for 2s at the beginning of a SWDL event.

Force SDLC 1ms Diag Frame Delay: When checked, the SWDL process will attempt to put the gateway module into a mode where frames are transmitted with at least a 1ms delay between them on the subnetwork.

Send ECU Reset After SWDL: After the SWDL is completed, a Hard Reset will be performed when this option is enabled. This is normally useful to uncheck if testing needs to be done with the SBL downloaded and active.

Send Clear DTCs After SWDL: DTCs will be cleared after SWDL is performed, when this option is enabled.

Use Fast Download Method: When this option is checked, the tool will not wait for the transmit confirmation from a consecutive frame prior to starting its STmin timer during segmented message transmissions within the SWDL process. Normally, after transmitting a consecutive frame, the tool will first confirm the frame was transmitted by obtaining the frame through its own receive queue prior to starting the STmin timer and queuing the next frame. This process can take a significant amount of time when certain hardware is used (e.g., CAN hardware with USB 1.1 connection to PC). However, if this option is checked, the DET assumes that when a consecutive frame is queued, it has been transmitted and therefore it immediately starts the STmin timer in order to queue the next frame. The usage of this option can significantly improve the overall software download time, but is not guaranteed to work 100% with ECUs that utilize a non-zero STmin.

NOTE: When using Vector hardware along with this "fast download" mechanism described above, it is possible to overflow the Vector transmit queue if it is set to low. The transmit queue size value is in the registry and is set through the Vector Hardware option in the control panel (see General Information → Settings). This is normally defaulted to 256. However, with fast download and an STmin of 0, if each request is large (e.g., more than this default number of frames) it is possible to overflow the hardware's transmit buffer since the software fills this faster than the buffer empties on CAN. The hardware does not return an error code, but the result will be lost consecutive frames and therefore no response from the ECU. If this occurs, the transmit size needs to be increased (which is recommended regardless). The recommended transmit queue size is 1024.

Number of Service $36 Retries Per Request: Service 0x36 is TransferData, which is used to transfer the information to the ECUs during a SWDL event. If a transferData request is not responded to, the DET will re-send the request this number of times. By default this value is configured to 3, but it can be increased or

decreased as desired. The main usage of this parameter is to make the download more robust so that a single issue in transmitting the data does not quit the entire download.

11.4.2 Non-Savable SWDL Options

The various options listed here were purposely made non-savable because the only time they should be required to be used would be to support ECUs with major deviations, or if they could adversely affect the normal SWDL process. Therefore, while the ability to support the non-compliant ECU is supported, it is purposely made more painful as a reminder to those responsible that the ECU has a fundamental issue that needs to be resolved.

Force Display of All Frames During SWDL: During SWDL, the frame log (which captures the raw bus traffic) will normally display all traffic EXCEPT the service 0x36 (transferData) requests. This is to prevent overly long captures and to prevent a slow down of the entire software download process. It is normally recommended to use the Generic Bus Monitor Tool to log the full traffic during a SWDL event. If this option is selection, then the normal diagnostic frame log will also contain the full service 0x36 requests and responses.

Don't Send Functional Enter Programming Session at Beginning of SWDL: By default, at the beginning of SWDL process DET sends a "Functionally Addressed Request" to start Programming Session (0x7DF 10 02), this can be changed and send a "Physically Addressed Request" instead ($XXX 10 02; where XXX is the module ID of the ECU receiving the SW). An early use case of this was a vehicle program where the functional request caused a module that went into programmingSession to remove power from one of the ECUs which needed programming.

Continue Despite Certain Non Compliances / Errors during SWDL: Normally, once an error such as an unexpected response occurs during the SWDL event, the entire process will be quitted. When this option is selected, the DET will still continue in its attempt to program the ECU(s) for certain errors. The main errors that are addressed include invalidly formatted positive responses to routine requests such as eraseMemory and activate SBL, as well as incorrect checksums returned in the positive response to requestTransferExit.

Force Tester Present Off During SWDL: Normally, during a SWDL event the DET will automatically enable the functional and periodic testerPresent message to keep all ECUs in the programmingSession. ECUs being programmed are required to be able to handle receiving these functionally addressed testerPresents interleaved with segmented requests and responses. If this option is selected, the DET will not send the periodic functional testerPresent, which would cause other ECUs to timeout of programmingSession and greatly slow down the programming time on a vehicle (due to normal communication being present). The use case for this would be to support early ECUs which have a MAJOR non-compliance in not handling the interleaved functional testerPresent requests. Note that this would be a critical deviation and would not be approved for production.

Attempt Backdoor ProgrammingSession Entry at Start of SWDL Sequence: When this option is selected, instead of the single functionally addressed enter programmingSession sent at the beginning of the SWDL sequence, the DET will send it periodically at a user defined rate and for a user defined amount of time. By default, it will send it every 5ms for 10s. See section 22 for details on how to change these values. The primary usage for this is the scenario where an ECU is programmed with software that it has deemed valid (and therefore passed control from the bootloader), yet the software itself does not correctly support diagnostics or the ability to transition back to the programmingSession in order to be programmed. The periodic request attempts to hit the 20ms window that is required by the SWDL specification. Note that the tool will prompt the user to cycle power to the ECU when it begins the fast periodic enter programmingSession request.

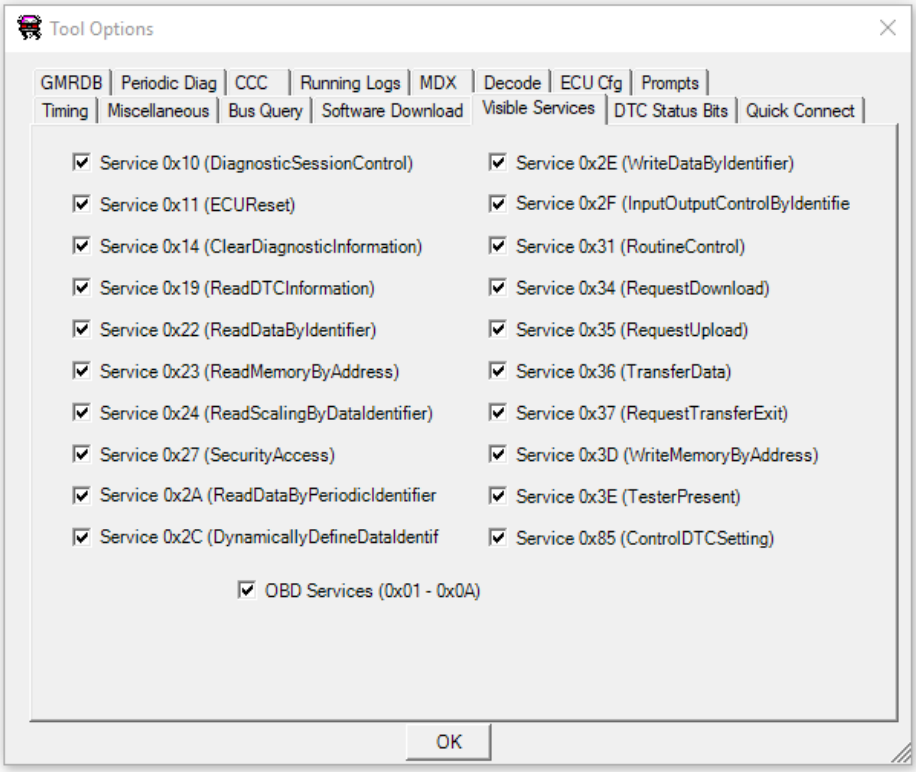
Force SWDL Version: The DET typically reads DID 0xF162 (SWDL Specification Version) during the SWDL event and adapts based upon the value returned. As SWDL implementations prior to 004 did not require the DID, the DET assumes that no response to the DID indicates an implementation of SWDL 003. If a value other than "Use ECU Reported Version" is selected here, the DET will assume the ECU responded with the selected

version and will adapt the programming process and expected responses accordingly. The allowed values as of the writing of this user manual are:

- Use ECU Reported Version
- 001
- 002
- 003
- 004
- 005

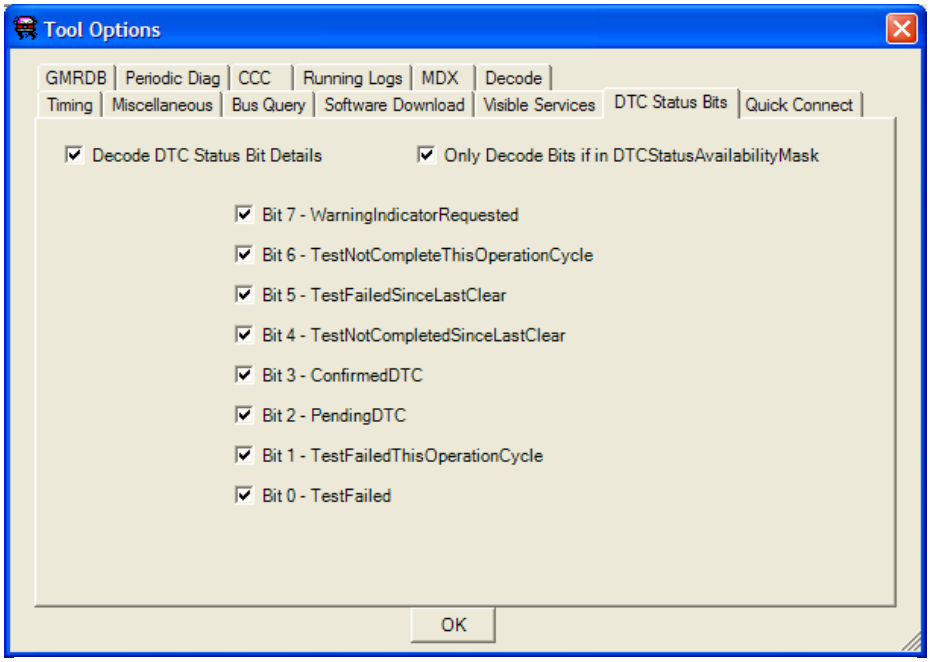
11.5 Visible Services Tab

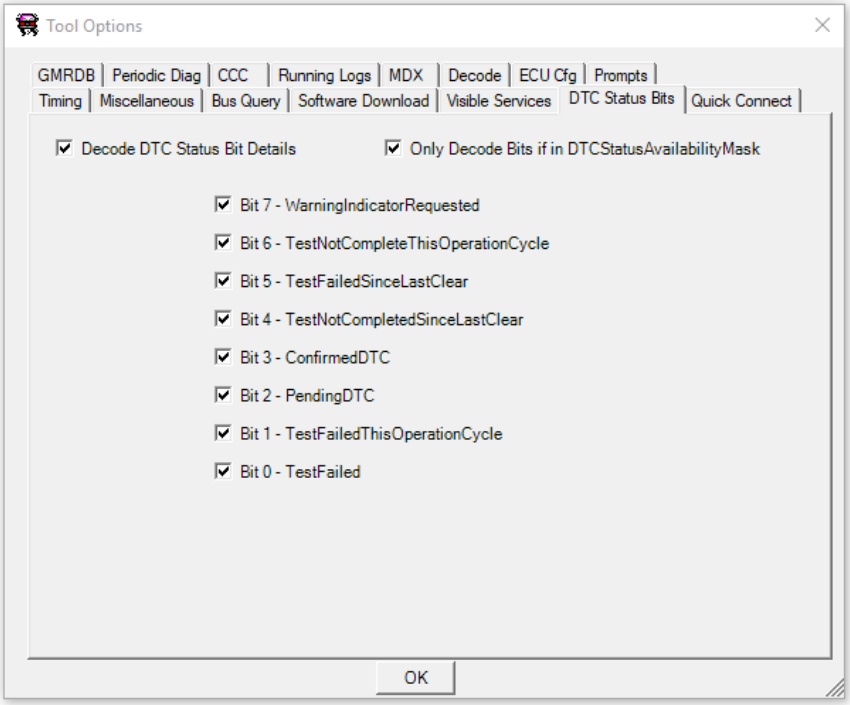
The supported services that will be visible in the "Generic Diag View" drop box (see section 4) are defined in this tab. By default, all of the ISO 14229 services are displayed. However, most users may likely only use a subset of all Ford services and therefore can make selection of the services easier by not displaying services they do not need to send manually on a regular basis.



11.6 DTC Status Bits Tab

This tab is used to customize which DTC status bits are decoded in the positive response to service 0x19 (readDTCByStatus) requests. By default, all of the DTC status bits are selected to be decoded.



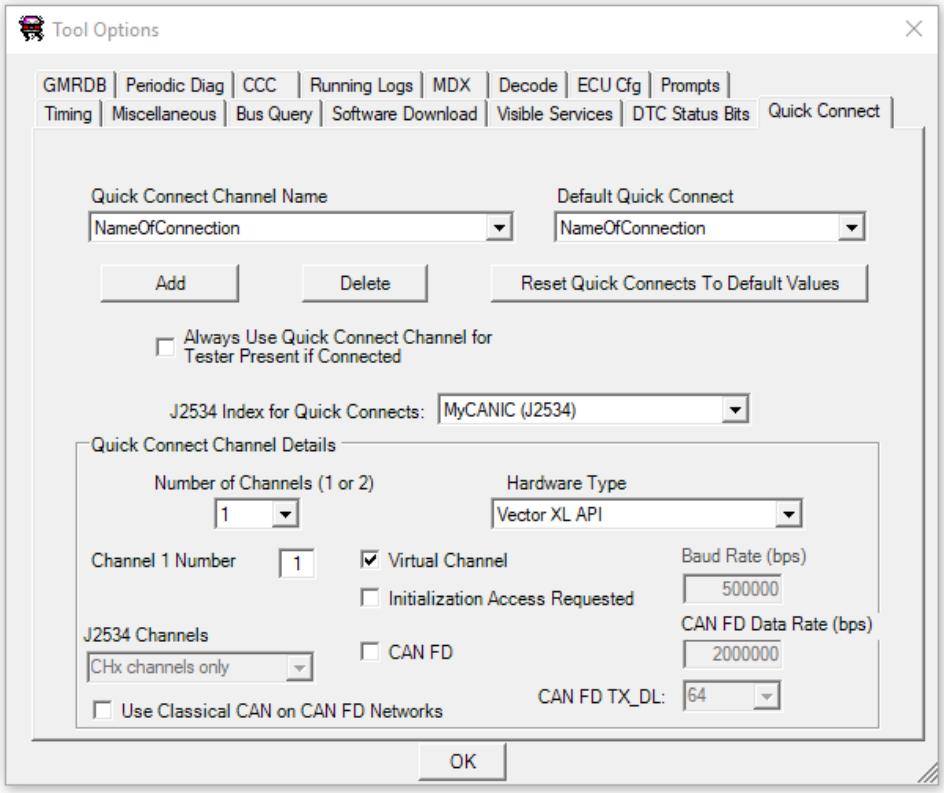


Decode DTC Status Bit Details: Enables or disables the entire decoding of the DTC status bits within the positive response to a service 0x19 request.

Only Decode Bits if in DTCStatusAvailabilityMask: With this option, the only decoded bits will be the ones that are actually supported by the ECU (i.e., those that are contained within the DTCStatusAvailabilityMask of the positive response). This will prevent the ECU reported values for unsupported status bits from being decoded to the user.

11.7 Quick Connect Tab

The tab allow the users to create Quick Connect channels (see section 3.3.2 for quick connect procedure), which makes a hardware connection faster than the traditional Hardware connection. The main benefits to using Quick Connect is the 1) ability to connect to multiple channels, 2) one click connection after starting the program, and 3) ability to support command line arguments.



Quick Connect Channel Name: In this section several Quick Connections can be created. Clicking the add button will open a window asking for a connection name. The drop box will display the name of all the created connections. The Delete button will remove the selected Quick Connection from the drop box.

Default Quick Connect: This drop box allows selecting the Quick Connection that will be displayed on top of other created connections, when clicking the traffic light in the "Toolbar buttons" section.

Reset Quick Connects To Default Values: Will remove all the Quick Connections and will add the following 2 Quick Connects:

- HS CAN – Hardware Channel 1: With channel 1 configured as 500k.
- MS-CAN – Hardware Channel 2: With channel 2 configured as 125k.

Always Use Quick Connect Channel for Tester Present if Connected: This option was created in an earlier version of the DET tool, and its purpose was to have a designated connection to send tester present messages (mostly intended to have a Quick Connection for HS-CAN and MS-CAN and send tester present on both channels). In later versions of DET this feature will be overridden by the configuration in the Timing tab, on Functional Tester Present section, "Send On All Connections", which is described on section 11.1.5 of this document. Therefore, this should normally NOT be checked.

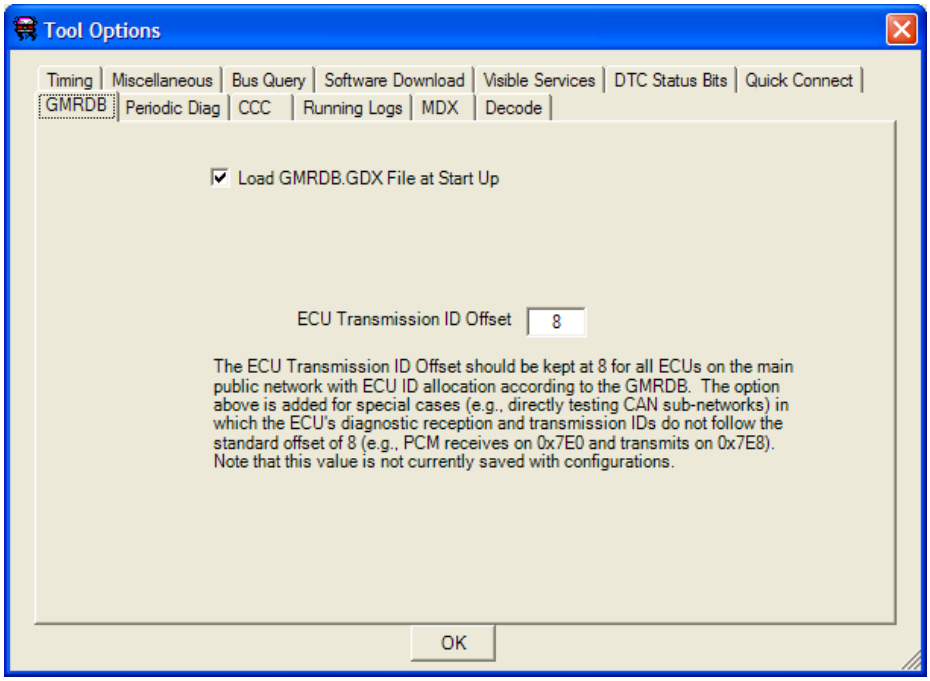
J2534 Index for Quick Connects: This applies only when the Hardware Type is J2534 API. This is the default index for the desired J2534 device.

Quick Connect Channel Details: In this section the Quick Connect is configured. The following can be configured:

- The number of channels (1 or 2).
  IMPORTANT: Note that for almost ALL scenarios, users will only want to connect to 1 channel for a given Quick Connect connection. If there is a desire to connect to more than one CAN channel, then more than 1 Quick Connect should be setup. If a single Quick Connect is configured for two channels, then all transmits on that connection will always go out on both channels, which can cause confusion.
- Hardware type:
  - There are currently 3 options. "Vector XL", "Vector (Old API)", and "J2534 API". The Vector XL selection should be used for any of the Vector XL hardware such as CANcardXL. The Vector (Old API) selection should be used for any pre Vector XL hardware such as CANcardX. Note that either of these work with Intrepid hardware when the DLL switch is performed. SAE J2534 support always connects to the first found SAE J2534 device in the Windows registry. This device can be determined by choosing Initialize Hardware and then looking at the first hardware device in the dropdown combo box that has (J2534) in the name.
- Channel number.
  - This selects which hardware channel that the quick connect will connect to.
- Virtual Channel.
  - This selects whether or not to connect to a virtual channel. Most users will not need this. This is normally used for testing purposes where no physical transceivers or hardware is needed during testing.
- Initialization Access Requested.
  - When connecting to the hardware, the software can either simply connect and not change the baud rate, or it can request initialization access (which if granted) allows the software to change the baud rate of the connection. Note that if multiple connections exist to the same hardware channel (e.g., DET and CANalyzer), that only one software connection will be given initialization access and therefore the ability to set the baud rate. Normally, the user would want this selected.
- Baud Rate.
  - This allows the user to define the speed of the Quick connect channel (assuming initialization access was both requested and granted). If 2 channels are selected for a single quick connect, this text box is disabled. The baud rate will be defined by the default hardware configuration in the system (or by another program which received initialization access).
- J2534 Channels
  - To use the J2534 CAN and PS channels, select CAN & PS Channels only in the J2534 Channels dropdown. To use the CHx channels, select CHx channels only.
- Use Classical CAN on CAN FD Networks
  - Use this option to do classical CAN on CAN FD netowrks.
- CAN FD
  - Check this box for CAN FD.
- CAN FD Data Rate (bps)
  - This sets the CAN FD data rate in bps.
- CAN FD TX_DL
  - This sets the maximum CAN FD frame.

11.8 GMRDB Tab

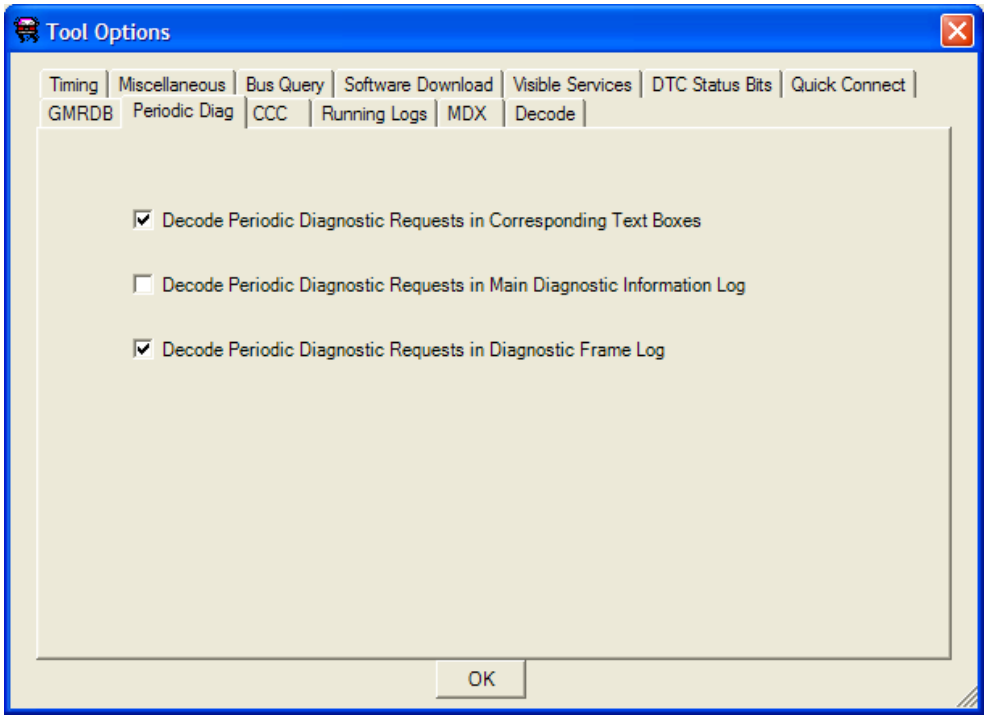
By default, the GMRDB is loaded every time when the DET start up. This can be disabled if "Load GMRDB.GDX File at Start Up" is unchecked. This always loads the GMRDB.gdx file from the directory where the application is installed and allows decoding of ECU IDs, DTCs, DIDs, etc.



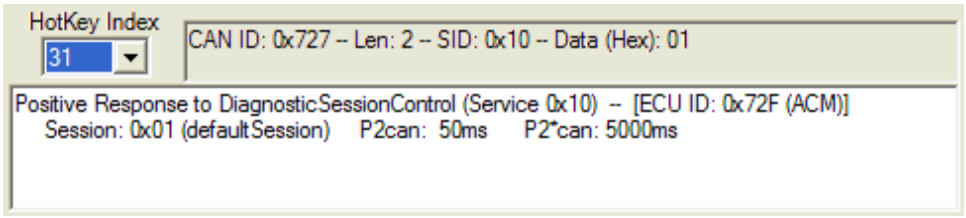
ECU Transmission ID Offset: This value is the value that the DET adds to the diagnostic request ID when looking for a response. For example, if the diagnostic request ID is 0x7E0, then the DET will look for a response on 0x7E8. On Ford main public networks (and as described in the GMRDB), the offset between request and response ID is always 8 and therefore this parameter should not need to be modified. The main use case is when the DET is directly connected to certain private CAN subnetworks, etc. where the offset is not always equal to 8. PLEASE NOTE THIS PARAMETER IS NOT SAVED WITH *.DCS FILES.

11.9 Periodic Diag Tab

This tab configures the way the Periodic Diagnostic Tool will display the periodic messages it transmits. Note that the more locations the data is displayed in, the less likely the tool will be able to meet the user desired period.



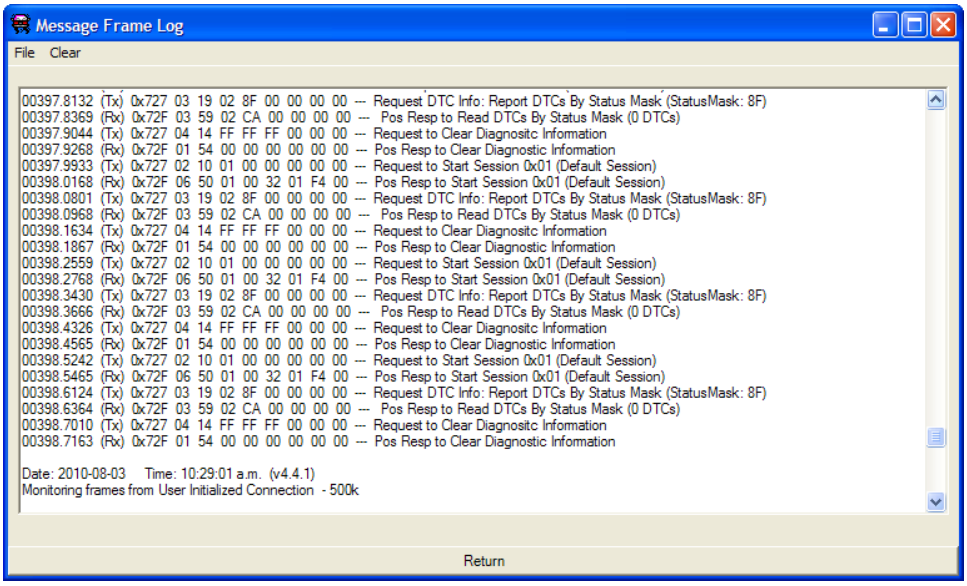
When "Decode Periodic Diagnostic Requests in Corresponding Text Boxes" is enabled, the text of the periodic message being sent is decoded in the text box on the main page of the periodic diagnostic tool (see image below); if disabled, the text boxes will not display the information sent by the Periodic Diagnostic Tool.



When "Decode Periodic Diagnostic Requests in Main Diagnostic Information Log" is enabled, the messages sent by the Periodic Diagnostic Tool will be displayed in the Main Diagnostic Information Log (see image below); when disabled, the messages sent by the tool, won't be displayed on the Main Diagnostic Information Log area.

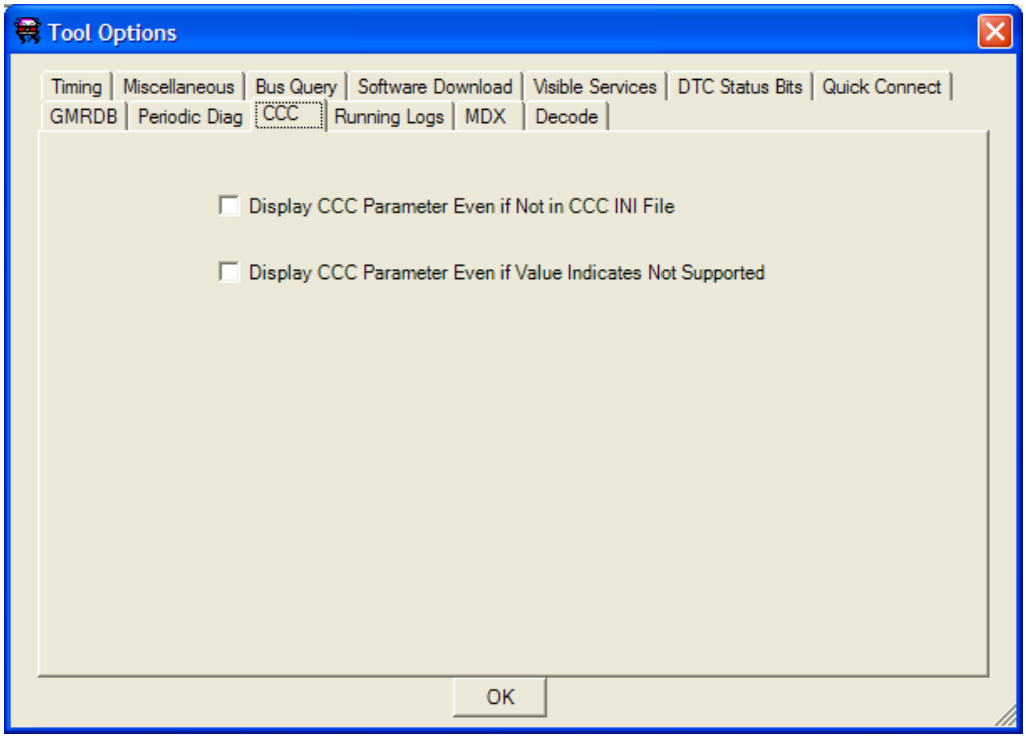
```
Request for DiagnosticSessionControl (Service 0x10) -- [ECU ID: 0x727 (ACM)]
   Session: 0x01 (defaultSession) [SPRMIB = False]
Positive Response to DiagnosticSessionControl (Service 0x10) -- [ECU ID: 0x72F (ACM)]
   Session: 0x01 (defaultSession)    P2can: 50ms    P2*can: 5000ms
Request for ReadDTCInformation (Service 0x19) -- [ECU ID: 0x727 (ACM)]
   SubFunction: 0x02 (reportDTCByStatusMask) [SPRMIB = False]
   DTCStatusAvailabilityMask: 0x8F
Positive Response to ReadDTCInformation (Service 0x19) -- [ECU ID: 0x72F (ACM)]
   SubFunction: 0x02 - reportDTCByStatusMask
   DTCStatusAvailabilityMask: 0xCA
   Number of Returned DTCs: 0
Request for ClearDiagnosticInformation (Service 0x14) -- [ECU ID: 0x727 (ACM)]
   GroupOfDTC = 0xFFFFFF (All DTCs)
Positive Response to ClearDiagnosticInformation (Service 0x14) -- [ECU ID: 0x72F (ACM)]
   No Additional Information
Request for DiagnosticSessionControl (Service 0x10) -- [ECU ID: 0x727 (ACM)]
   Session: 0x01 (defaultSession) [SPRMIB = False]
Positive Response to DiagnosticSessionControl (Service 0x10) -- [ECU ID: 0x72F (ACM)]
   Session: 0x01 (defaultSession)    P2can: 50ms    P2*can: 5000ms
Request for ReadDTCInformation (Service 0x19) -- [ECU ID: 0x727 (ACM)]
   SubFunction: 0x02 (reportDTCByStatusMask) [SPRMIB = False]
   DTCStatusAvailabilityMask: 0x8F
Positive Response to ReadDTCInformation (Service 0x19) -- [ECU ID: 0x72F (ACM)]
   SubFunction: 0x02 - reportDTCByStatusMask
   DTCStatusAvailabilityMask: 0xCA
   Number of Returned DTCs: 0
Request for ClearDiagnosticInformation (Service 0x14) -- [ECU ID: 0x727 (ACM)]
   GroupOfDTC = 0xFFFFFF (All DTCs)
Positive Response to ClearDiagnosticInformation (Service 0x14) -- [ECU ID: 0x72F (ACM)]
   No Additional Information
```

When "Decode Periodic Diagnostic Requests in Diagnostic Frame Log" is enabled, the messages sent by the Periodic Diagnostic Tool are displayed on the Diagnostic Frame Log (see image below); when disabled, messages sent by this tool are not displayed on the Diagnostic Frame Log.



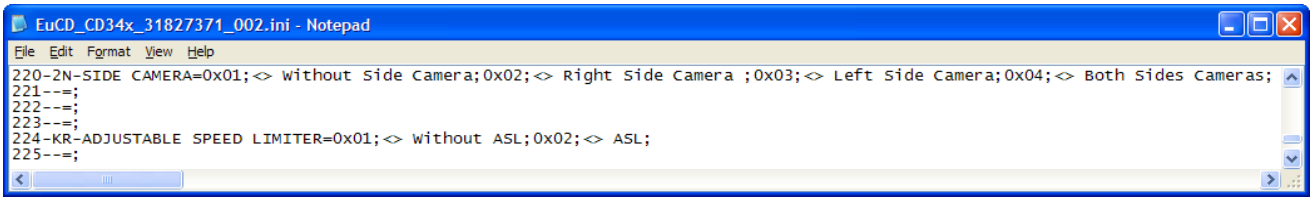
11.10 CCC Tab

This section allows configuring the way the DET displays the parameters of DID 0xF106 (Central Car Configuration parameters). This decoding is based upon loading of a CCC .ini file (see section 12.2.10). As CCC was never a globally accepted solution, and has been phased out as part of Ford's CGEA 1.3, this is primarily provided for support of legacy programs.



By default, the 2 options above are disabled and that means that only supported information is displayed when decoding DID 0xF106. If "Display CCC Parameter Even if Not in CCC INI File" is enabled, the DET will display the ECU's reported value for the particular CCC parameter number (despite not being in the INI file). If "Display CCC Parameter Even if Value Indicates Not Supported" is enabled, the DET will display the ECU's reported value even though the INI file indicates that the parameter itself has a value indicating it is not supported. If both options are unchecked, the DET will show only parameters defined in CCC "INI" file and that are supported by the ECU.

INI FILE



MAIN DIAGNOSTIC INFORMATION LOG

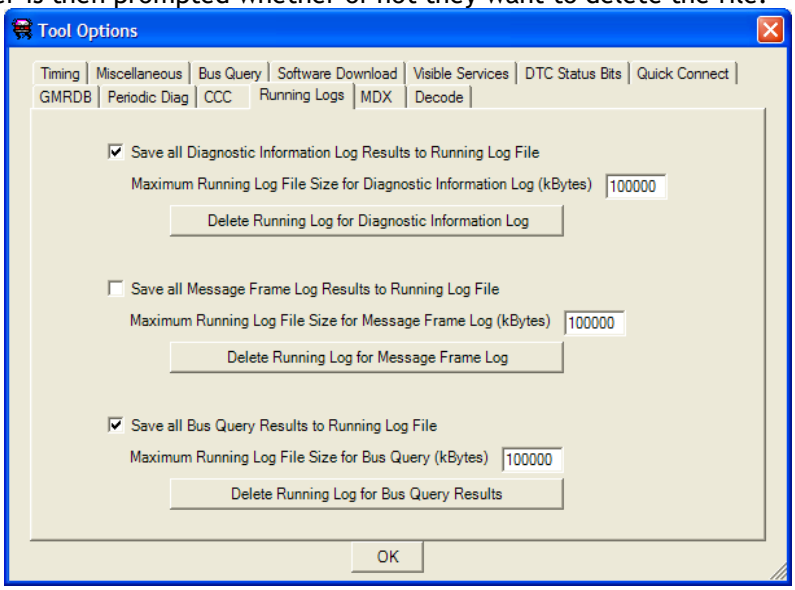
221, 0x00 = Parameter name not in CCC INI file, Not Supported
222, 0x00 = Parameter name not in CCC INI file, Not Supported
223, 0x00 = Parameter name not in CCC INI file, Not Supported
224, 0x00 = ADJUSTABLE SPEED LIMITER, Not Supported

11.11 Running Logs Tab

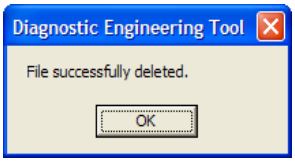
Running logs are .txt files that can store a history of all of the information displayed by the DET. There are three separate possible running logs. One for the main diagnostic information log, one for the message frame long, and one for the bus query results. The maximum size of the log files created by the DET can be configured in this section. By default, the Diagnostic Information Log results and the Bus Query Results are saved in a running log file with a 100000Kb limit (Message Frame Log results are not saved by default, but can be enabled in this section by selecting the checkbox). The purpose of the logs are to have backup information of Diagnostics information with DET, in case the user did not save information or the information was otherwise lost. When a user does not save their diagnostics data and wish to recover that information, they

could just pull the logs and get the data needed. Note that the date/time is included in the log to make it more convenient to correlate the information.

If a user is having disk space problems, he can disable this feature and no running log files will be created. A configurable maximum size is included for each file to prevent filling up the hard disk for users who use the tool frequently. Note that whether or not the maximum size is exceeded or not is only checked when the DET first starts and the user is then prompted whether or not they want to delete the file.

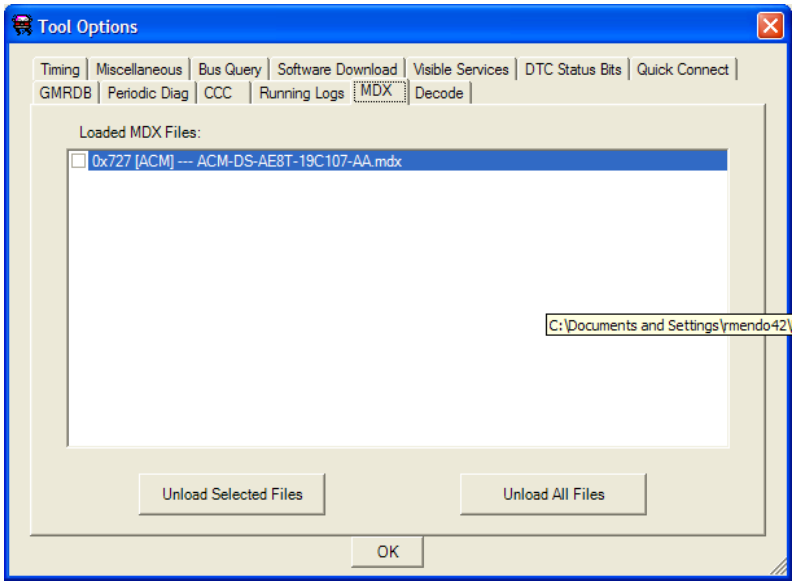


Please note users can delete the running file by clicking the buttons "Delete Running Log…" on each section. A confirmation window will pop, when the deletion is completed successfully.



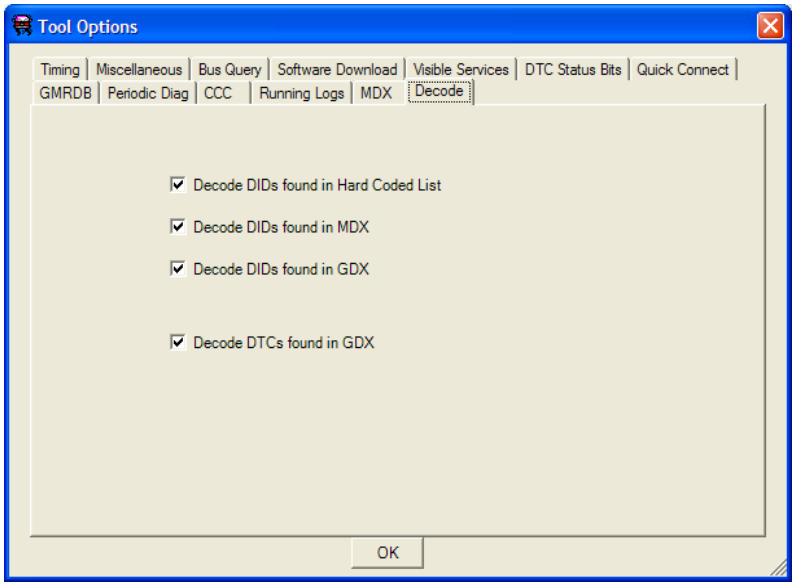
11.12 MDX Tab

In this section the user can unload previously imported MDX files (see section 12.2.6). Imported MDX files are normally saved within the .dcs file so that they are loaded automatically when a .dcs file is loaded (or at startup if part of the default settings .dcs file). This tab provides the mechanism to no longer import one or more MDX files. Unloading the MDX file here will effectively remove the association of the MDX when the .dcs file is saved.



11.13 Decode Tab

This section allows configuring the decoding options of the diagnostic messages displayed on the screen (see image below).



By default these are all enabled and will behave as described below. If none of these options are enabled, or the DID is not found, only the raw data will be displayed.

Decode DIDs found in Hard Coded List: The DET has a hard coded list of certain core identification DIDs such as 0xF111 (ECU Core Assembly Number) that the tool can decode independent of any imported MDX or GDX files. If this option is selected, DIDs will first be checked to see if they are in the hard coded list prior to looking in the relevant MDX file or the GDX file.

Decode DIDs found in MDX: If this option is selected, DIDs which are not in the hard coded list (or were selected not to be decoded based upon this) will be checked to see if they are in the relevant MDX file, and will be decoded to "English" if so.

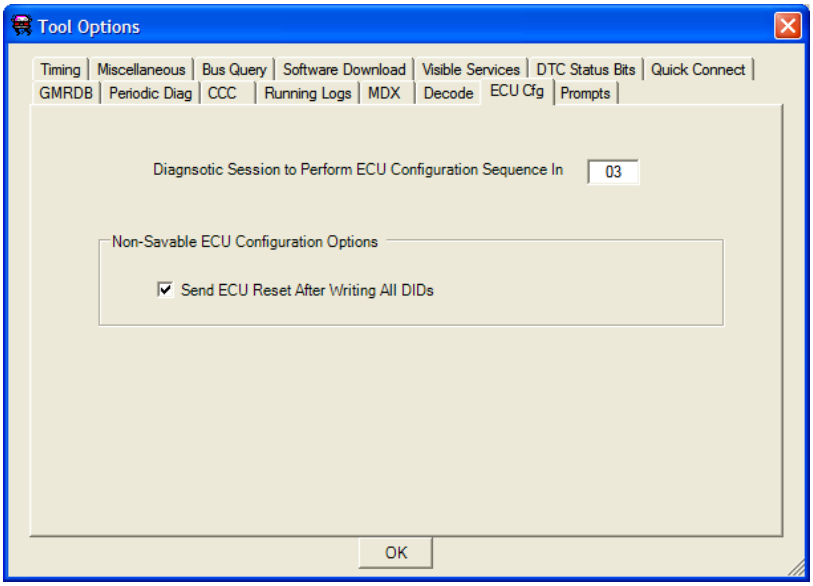
Decode DIDs found in GDX: If this option is selected, DIDs which are not in the hard coded list (or were selected not to be decoded based upon this) and DIDs not found in an associated MDX will be checked to see if they are in the loaded GDX file, and will be decoded to "English" if so.

```
Request for ReadDataByIdentifier (Service 0x22) -- [ECU ID: 0x726 (BCM)] - Single DID
   DataIdentifier: 0xA45F
Positive Response to ReadDataByIdentifier (Service 0x22) -- [ECU ID: 0x72E (BCM)] -- Assuming Single DID
   DataIdentifier: 0xA45F   Data Size: 4 byte(s)
   Data (Hex): 04 00 00 00
   Key Switch System Input Signal Status (GDX):   ---   (Bitmapped DID)
      Ignition On Relay A Feed Back: Off
      Ignition/Accessory: Off
      Ignition/Crank: Off
      Crank Switch: Off
      Ignition Switch: On
      Auxilary Switch: Off
      Key In Switch: Off
      Request For Engine Start Is Being Received Over CAN (May Persist For A Few Seconds For Readability): False
```

Decode DTCs found in GDX: The DET will decode the first 2 bytes of DTCs based upon the loaded GDX file if this option is selected. The third DTC byte is the FailureTypeByte and this is decoded from a hardcoded list within the DET.

11.14 ECU Cfg Tab

This section allows configuring some customization options that are done during the ECU Configuration process of reading / writing DIDs as described in section 6.

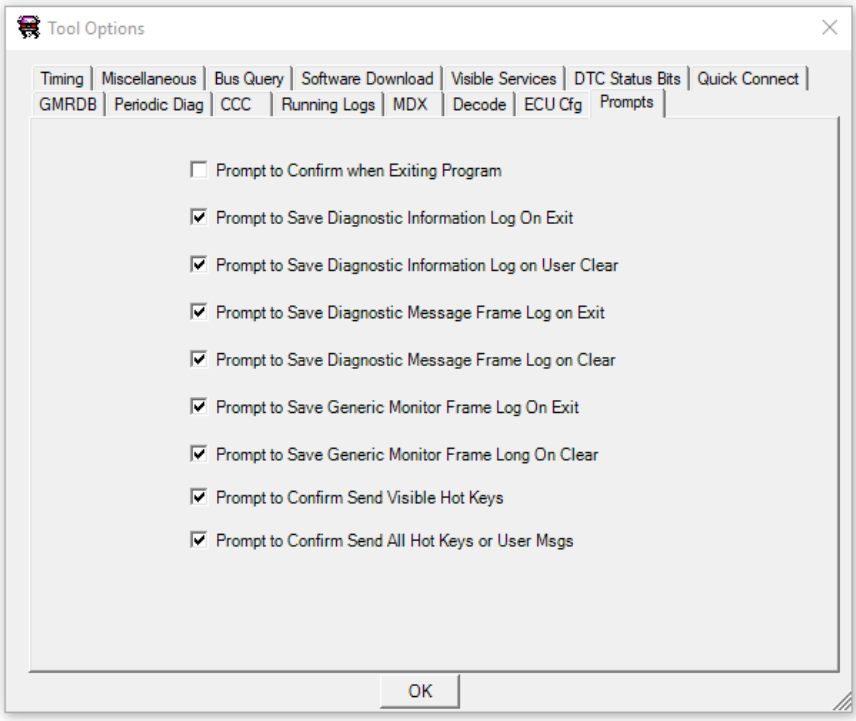


Diagnostic Session to Perform ECU Configuration Sequence In: This value reflects the diagnostic session that the DET will enter when reading and writing DIDs in the ECU Configuration view. It is defaulted to session 0x03 which is where the standard Ford ECU Configuration DIDs are required to be supported. The main usage for changing this option is for scenarios such as reading and writing supplier specific DIDs that are only supported in a supplier specific session.

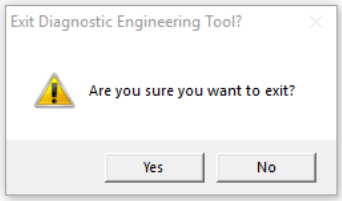
Send ECU Reset After Writing All DIDs: Normally, after an ECU Configuration event the DET will automatically send an ECU reset request to ensure that the ECU begins to utilize the updated configuration. If this option is not selected, the DET will not send this ECU reset. The use case for this would be to support early ECUs which have a MAJOR non-compliance in losing the ECU configuration data upon reset. Note that this would be a critical deviation and would not be approved for production. The option is purposely non-savable.

11.15 Prompts Tab

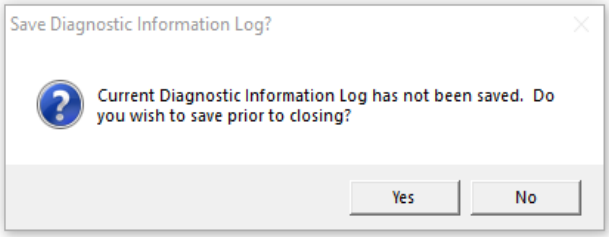
This section includes different options that can be customized based upon user preference for when to automatically prompt for certain actions.

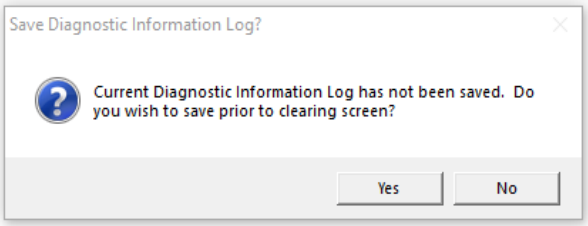


Prompt to Confirm when Exiting a Program: Enables the prompt displayed below, when exiting DET.

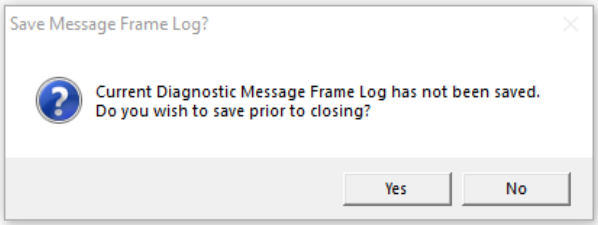


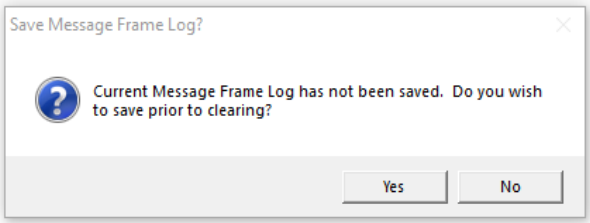
***Prompt to Save Diagnostic Information Log On Exit / Prompt to Save Diagnostic Information Log on User Clear*:** Enables the prompts displayed below when exiting DET or manually clearing the Display via "Display > Clear".



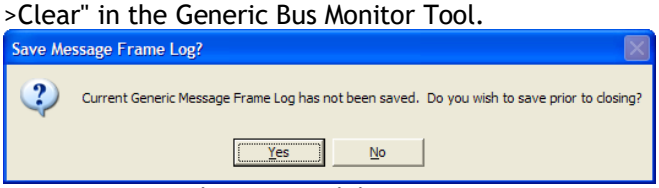


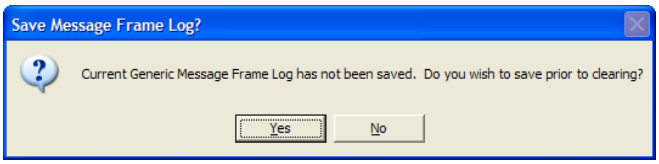
Prompt to Save Diagnostic Message Frame Log on Exit / Prompt to Save Diagnostic Message Frame Log on Clear: Enables the prompts displayed below when exiting DET or manually clearing the Display clicking on "Clear" in the Message Frame Log Display.



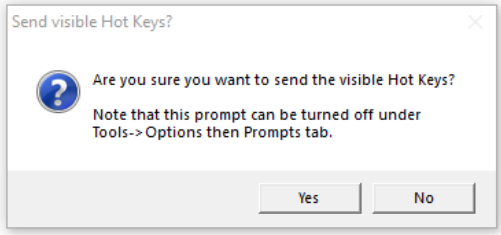


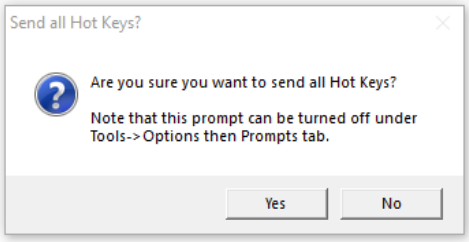
Prompt to Save Generic Monitor Frame Log On Exit / Prompt to Save Generic Monitor Frame Log On Clear: Enables the prompts displayed below when exiting DET or manually clearing the Display via "Display >Clear" in the Generic Bus Monitor Tool.





Prompt to Confirm Send Visible hot keys / Prompt to Send All Hot Keys or User Msgs: Enables the prompts displayed below when sending hot keys or sending all hot keys or user messages.





12 Menu Options

12.1.1 Main Menu

Main menu is available in all "Views" (DET sections) and it provides access to the following:

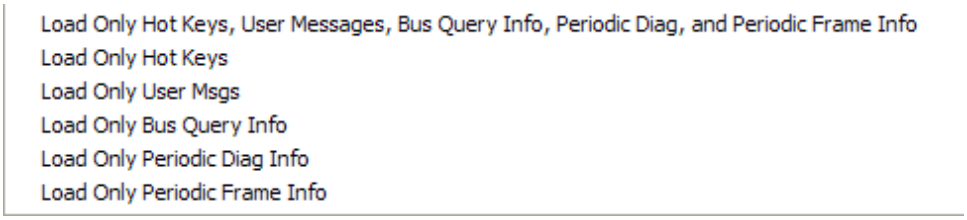
12.2 File

12.2.1 Open Diagnostics Configuration Settings File

Load configuration settings file, whose extension is *.dcs. These files, allow saving almost all user options and DET settings including Hot Key Messages, User Messages, etc. as well as links to the currently loaded .mdx, .ecd, .vbs, .key files, etc. etc.

12.2.2 Load Subset of Diagnostics Configuration Settings File

Load a Diagnostic Configuration File (*.dcs) and select a specific subset of the information contained in it. That information can be:



A primary use case of this would be if you receive someone's .dcs file that contains all of their settings, but you only want to load a subset of the information (e.g., Hotkeys) without affecting your other personal settings.

12.2.3 Save Diagnostic Configuration Settings File (As)

Save a *.dcs file, with the current information loaded in DET.

12.2.4 Make Current Setting Default Configuration

The current settings loaded in DET will be saved as DefaultSettings.dcs, which is the file that the DET will attempt to load every time it is opened (unless command line arguments to load a specific DCS file are used).

12.2.5 Import GMRDB File (.gdx)

This option imports a specific GMRDB, which might contain updated DIDs, DTC, Routines or ECUs. Note that if it is desired to load this GMRDB version automatically then this file should be renamed to gmrdb.gdx file and placed in the install directory of the tool.

12.2.6 Import MDX File (.mdx)

Imports the XML Part 2 which allows the DET to decode certain Diagnostic information from this file (e.g., DIDs) as well as auto-populate certain drop-down boxes such as the list of DIDs in the service 0x22 request message in the Generic Diag view.

12.2.7 Load VBF Sequence File (.vbs)

This loads a Versatile Binary Format Sequence File (*.vbs). This file type contains one or more ordered sequences of Versatile Binary Format (*.vbf) files to download, which is utilized in the SWDL view. The files loaded can be viewed at the SWDL section.

12.2.8 Load ECU Cfg Sequence File (.ecd)

This loads an ECU Configuration File (*.ecd). This file type contains one or more sequences of DIDs, that can be automatically read and written to.

12.2.9 Load Security Key File (.key)

This loads a file that contains one or more ECUs and security levels with the challenge bytes.

12.2.10 Load CCC INI File (.ini)

This loads a CCC INI file that allows for decoding of the legacy Central Car Configuration DID of 0xF106.

12.2.11 Save Diagnostic Information Log

A Rich Text format (*.rtf) file will be created including all the information that is contained in the Display section of this window. The usser will be prompted to select the directory and filename of where to save.

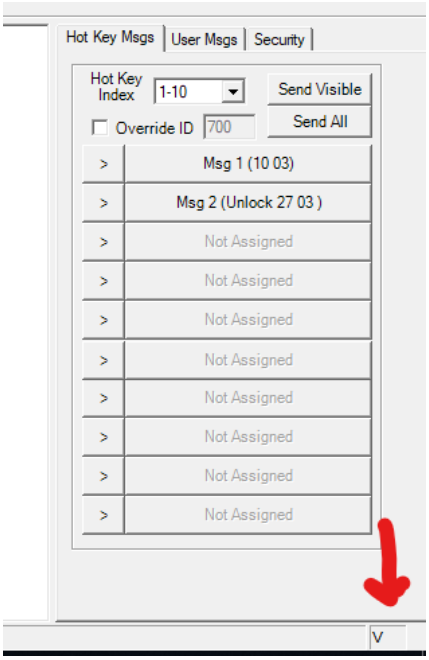
12.3 Initialize Hardware

This option only allows connection of a single channel. There are currently 3 Application Programming Interfaces (APIs) supported by the tool. Those are:

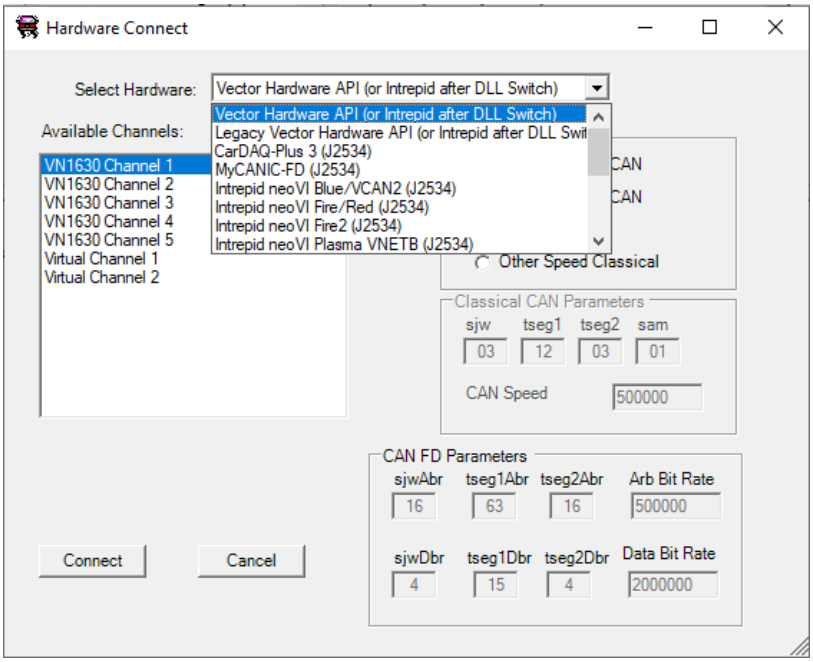
- Vector XL Hardware (to be used for any of the Vector XL hardware such as CANcardXL)
  - Note: This uses the vxlapi.dll interface to the hardware.
- Vector CANcardX (to be used for any of the pre Vector XL hardware such as CANcardX)
  - Note: This uses the vcand32.dll interface to the hardware.
- SAE J2534 API
  - Note: The DET will scan the Windows Registry for all installed SAE J2534 hardware devices and will dynamically populate the list of available hardware based upon the SAE J2534 hardware devices installed on the computer. If your particular device is not appearing, then it is not correctly installed on the laptop as a registered SAE J2534 device.

By default, DET tool works with Vector hardware when selecting Vector for the hardware type. However, Intrepid hardware is also supported through this same approach, but requires that the DLL swap procedure described in section 12.3.1 is performed. When the DLL swap is performed, the DET behaves no differently than when communicating to Vector hardware. The difference is that the DLL (either vcand32 or vxlapi) which is called by the DET is Intrepid's DLL instead of Vector's DLL and effectively intercepts the calls and converts them to the attached Intrepid hardware.

In the lower right hand corner, the currently selected DLL will be displayed. V is for Vector and I is for Intrepid.

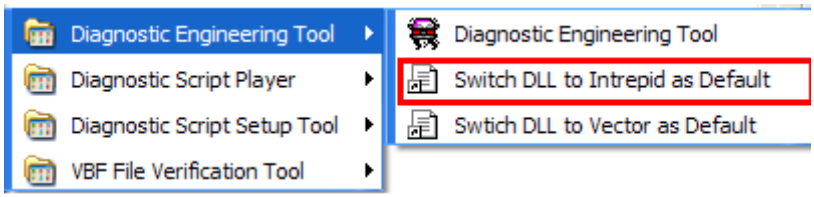


After clicking "Initialize Hardware" on the main menu, the following window will pop. The drop box will allow the selection of different hardware for the connection of the DET.

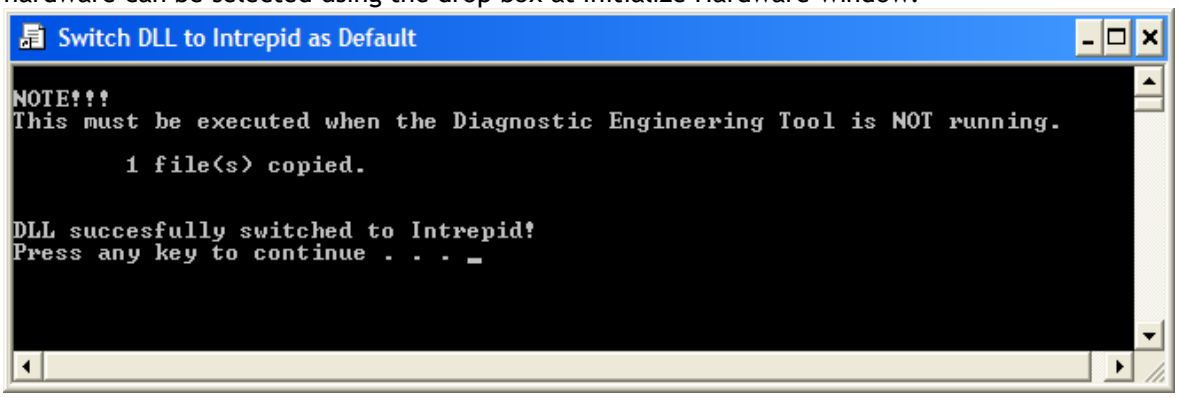


12.3.1 Intrepid DLL Switch

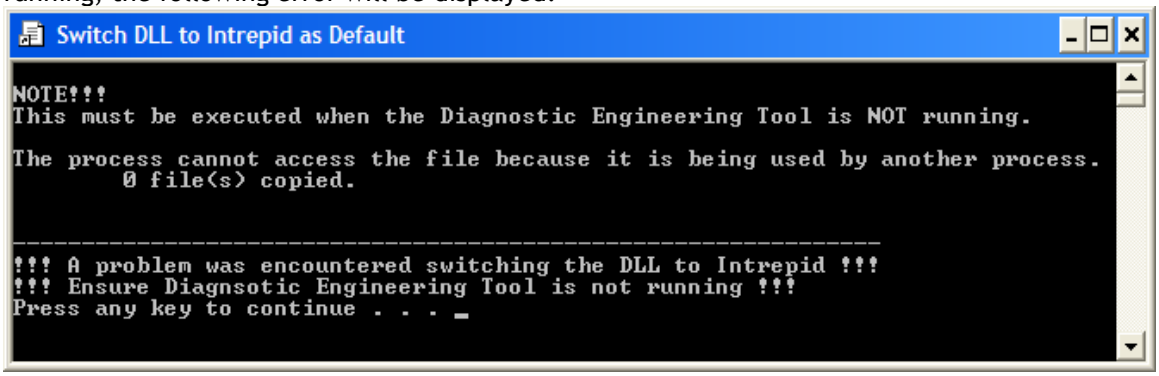
Go to Windows Start menu, All Programs, Ford NetCom and Select Diagnostic Engineering Tool (as shown on the image). Click on "Switch DLL to Intrepid as Default". Note that starting with v5.0.0 of the DET, this swap only needs to be performed one time and no longer after each time a new version of the tool is installed.



A window will open (see image below) specifying the DLL switch was successful. After doing this, Intrepid hardware can be selected using the drop box at Initialize Hardware window.



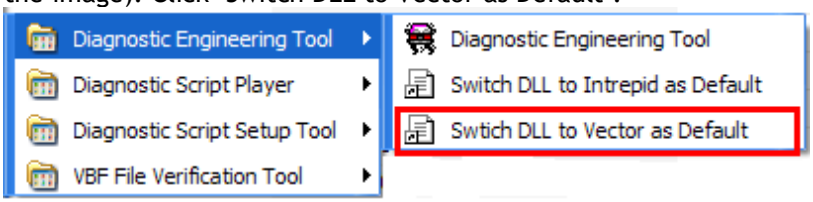
To perform the DLL switch, DET should not be running. If the task is performed while the application is running, the following error will be displayed:

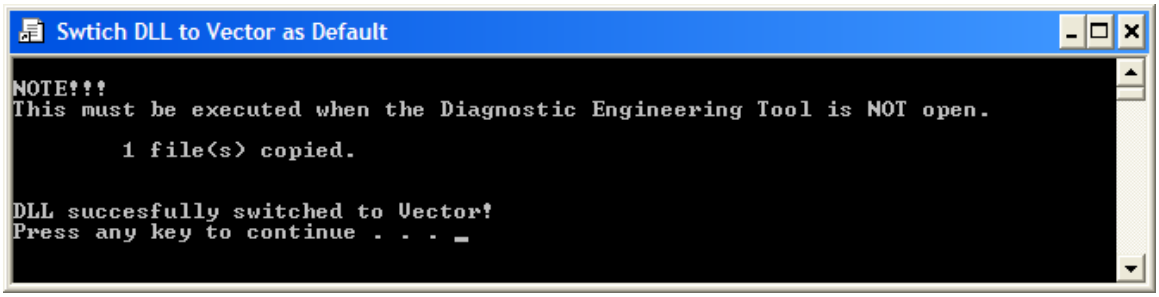


12.3.2 Vector DLL Switch

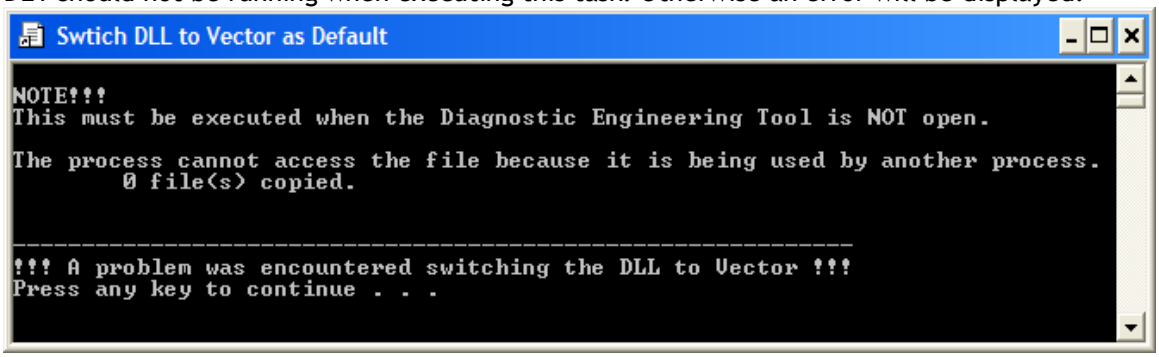
For hardware connection, Vector hardware is the default on DET. It could be possible that a user may want to use the Vector hardware once an Intrepid DLL switch has been performed. To do this the following steps should be performed:

Go to Windows Start menu, All Programs, Ford NetCom and Select Diagnostic Engineering Tool (as shown on the image). Click "Switch DLL to Vector as Default".





DET should not be running when executing this task. Otherwise an error will be displayed.



12.3.3 J2534 interface

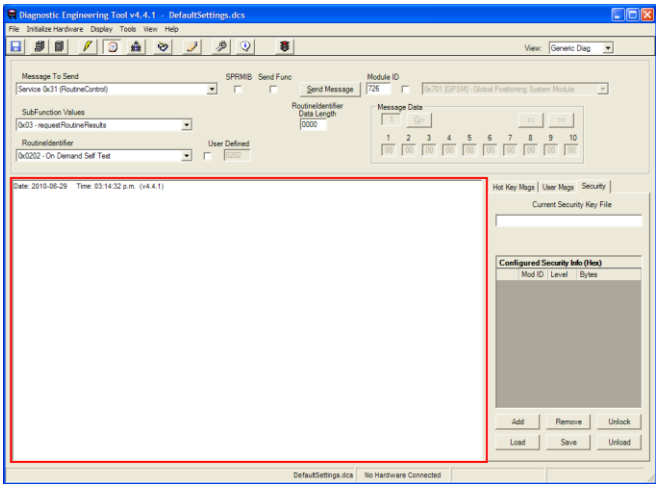
When a J2534 module was correctly installed in the system, DET will automatically detect it and show it as an available hardware to perform a J2534 connection. Examples are: MCS1, Intrepid (J2534), PUMA, etc.

Note: In theory, any compliant SAE J2534 tool should work. However, the tested SAE J2534 devices and any known limitations are described in Annex C. To ensure full support (e.g., multi-channel connections, etc.), the latest drivers and J2534 DLLs should always be obtained directly from the tool supplier. One of the limitations with the SAE J2534 interface is that each software connection to a channel does not necessarily get its own receive queue and its own acceptance filter settings. This means that the functionality of connecting to diagnostics via the main form and also connecting to view all network traffic with the generic bus monitor will not necessarily work (as it will when using the Vector API).

12.4 Display

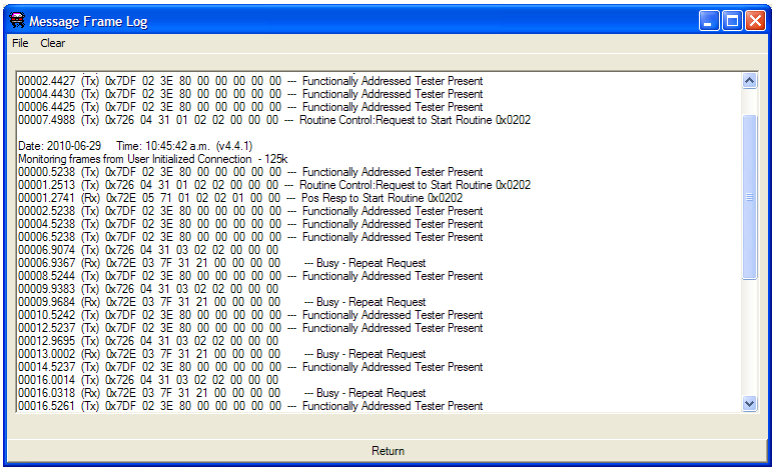
12.4.1 Clear

Clears the current Diagnostic Information Log of the DET.



12.4.2 Show Frame Log

This makes the Message Frame Log Display visible. Note that the DET normally logs the diagnostic CAN frame details regardless of whether or not this form is visible.



The information displayed can be saved into an "*.RTF" file, by selecting File > Save. To clear the information of the display, select the Main Menu, Clear.

12.5 Tools

This menu item allows the user to open one of the many individual tools available as part of the DET. This includes the Bus Query Tool, Generic Bus Monitor, DTC Tracker Tool, Periodic Diagnostic Tool, Periodic Frame Tool, Force Bootloader Entry, and the MGM Image Reflash. For more information on these tools, see sections 17 - 22.

12.5.1 Clear All User Messages

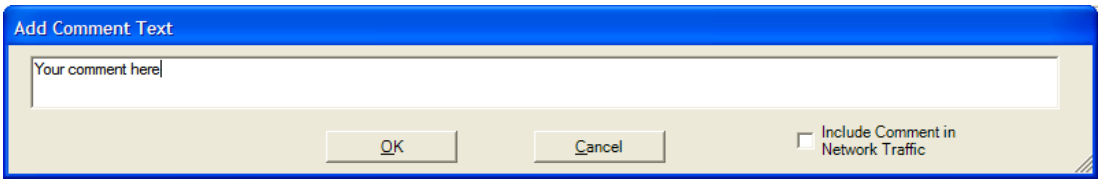
This option clears all of the defined User Messages (see section 15).

12.5.2 Clear All Hot Key Messages

This option clears all of the defined Hot Key Messages (see section 14).

12.5.3 Add Comment

This option allows the user to enter their own text comment which will be displayed within the Diagnostic Information Log to provide further details on what they were doing when they sent the request. The comment will be given a unique number that is automatically incremented. The user will also be given the choice as to whether or not to include the same comment in the network traffic. Selecting this makes it more convenient to align the decoded output to the raw CAN traffic.



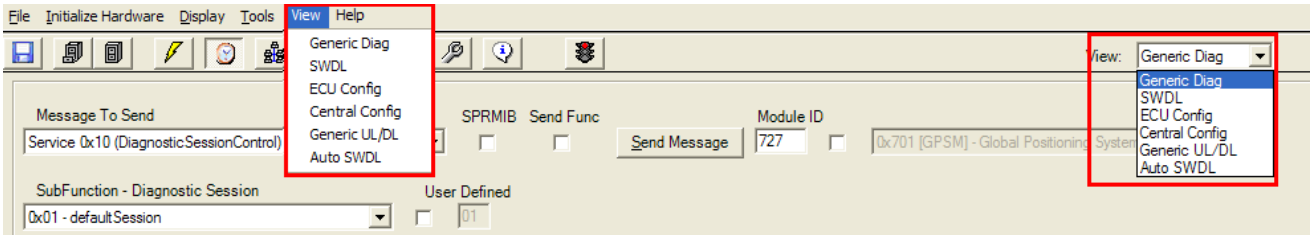
12.5.4 Options

This launches the user configuration options form (see 10 for more details).

12.6 View

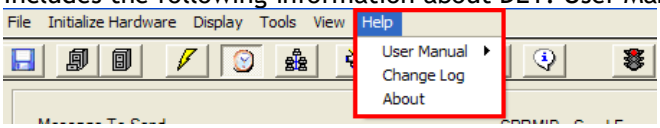
It is available in the MAIN MENU or at the right of the Toolbar buttons. It allows the user to switch between DET sections, which are:

- Generic Diag (see section 4)
- SWDL (see section 5)
- ECU Config (see section 6)
- Central Config (see section 7)
- Generic UL/DL (see section 8)
- Auto SWDL (see section 9)



12.7 Help

Includes the following information about DET: User Manual, Change Log and About.



12.7.1 User Manual

This option opens the latest "Diagnostic Engineering Tool User Manual" file, for the version the user is working with.

12.7.2 J2534 Device Troubleshooting Guide

This will launch a guide with J2534 troubleshooting tips.

12.7.3 Change Log

This option opens an Excel file that describes all of the changes made in each version of the DET tool.

12.7.4 System Information Report

This will read information from the user's PC and generate a report that can be emailed to Net Com when support is requested.

12.7.5 About

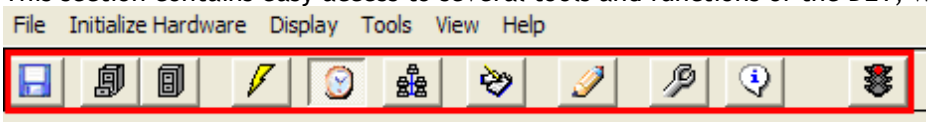
Displays a window that provides version information for the DET including supporting DLLs:

- DET Version
- GGDS DLL version
- FordNetcom DLL version
- FordNetCom XML DLL version
- Contact of the developer.

See example below:

13 Toolbar Buttons

This section contains easy access to several tools and functions of the DET, which are defined below.



13.1 Save Diagnostic Information Log

Saves to a "*.rtf" file, all the information contained on the Display. Refer to section 12.2.11, for more information about this file.

13.2 Open Diagnostic Configuration Setting File

Open a Diagnostic Configuration Setting File, "*.dcs". Refer to sections 12.2.1 and A.1, for more information about this file.

13.3 Save Configuration File

Save the Diagnostic Configuration Setting file. Refer to section 12.2.3, for more information about this file.

13.4 Send Message

Sends the information based upon the current message configured in the "Message to Send" drop down box on the Generic Diag view. Refer to section 4.2, for more information.

13.5 Enable / Disable Periodic Tester Present

Enabled by default, this option enables or disables the transmission of tester present messages from the DET to the modules connected on the network. This performs the same change as described in section 11.1.5.

13.6 Query Bus

Open the Query Bus tool. For more information, refer to section 17.

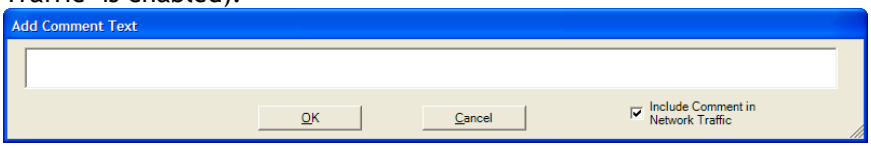
13.7 Toggle Frame Display

Open the Frame Log window. For more information about the Frame Log window, refer to section 12.4.2.

13.8 Add User Comment to Window

This tool allows the addition of comments into the Diagnostic Information Log and the Message Frame Log. When selecting the button a new window will open, allowing the user to introduce the comments in the

Diagnostic Information Log (and in Message Frame Log, if the check box: "Include Comment in Network Traffic" is enabled).



13.9 User Settings

Open the "Tool Options" window. To learn more about "Tool Options", refer to section 10.

13.10 About

Open "About" window. Refer to section 12.7.4, to learn more about this section.

13.11 Quick Connect to Predefined Channels

This button automatically connects all user defined quick connections to allow easy and multiple connections to the hardware. To learn more about the "Quick Connect" capabilities, refer to section 3.3.2 and 11.7.

14 Hot Keys

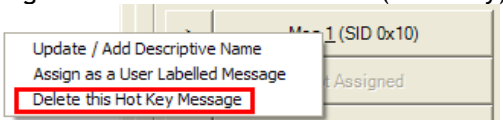
"Hot Key Msgs" section allows the user to assign specific "Diagnostic Messages" to a button, so they can be sent by just clicking the button. Up to 100 hot key messages can be defined. These hot key messages can then be sent individually, all at once, or can also be linked to Bus Query Tool functionality, Periodic Diagnostic Tool functionality, as well as command line arguments.

14.1 Assign a Message to a Button

First of all, the desired message to assign must be completely set up as desired in the Generic Diag view (see section 4.2). Instead of sending the message as would normally be done, one can instead assign it to a hot key by selecting the Arrow located at left hand of the hot key button where you want the message to be assigned. Once the message is assigned, the button will be enabled (not greyed out, as image below show). Note that all of the message to send information including ECU ID, etc. will be saved.

14.2 Delete an Assigned Message

- Right click on the Button (Hot Key) containing the message that needs to be deleted.



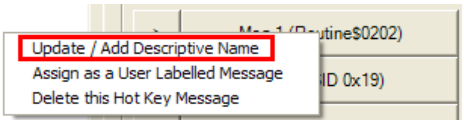
- Select "Delete this Hot Key Message". Once it is deleted, the button should be disabled (greyed out).

Refer to section 12.5.2 for a mechanism to clear all 100 hot key messages at once.

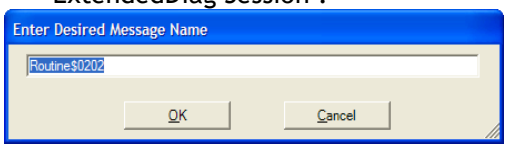
14.3 Update / Add Descriptive Name

Once a message has been assigned to a Hot Key, the description displayed on the button can be modified as follows:

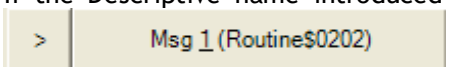
- Right click on the button with a message assigned.
- Select "Update / Add Descriptive Name."



- A window will pop up, where the user can introduce the desired descriptive name such as "Put PCM in ExtendedDiag Session".

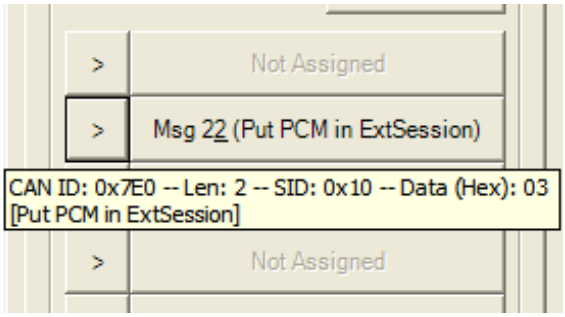


- If the Descriptive name introduced was: "Routine$0202". The button will be displayed as follows:



14.3.1 Tool Tip Text Help

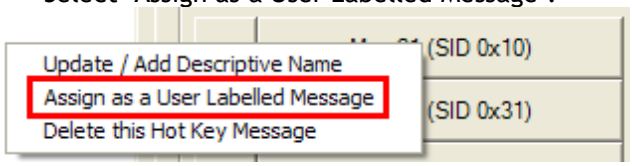
Prior to sending a particular hot key message, if the mouse is allowed to hover over the particular button the DET will display the details of what specifically is in the request in case there exists a need to confirm the details of the request.



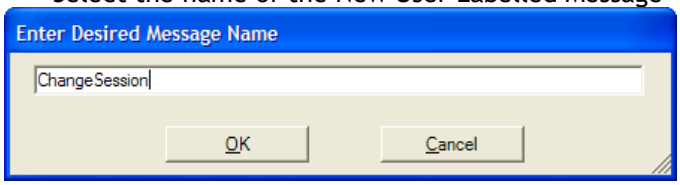
14.4 Assign as a User Labelled Message

Defined Hot Key messages (has a message assigned to the button) can also be assigned as a user labeled message (refer to section 15 for more information on User labelled messages) by performing the following steps:

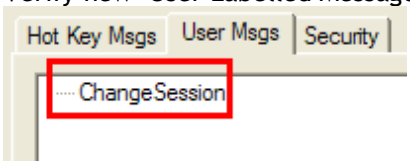
- Right click on Active Hot Key.
- Select "Assign as a User Labelled Message".



- Select the name of the New User Labelled Message



- Verify new "User Labelled Message" on "User Messages" tab.

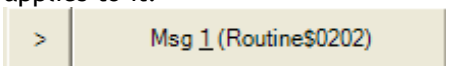


14.5 Sending an Individual Hot Key Message

Once a hot key message is defined, it can be sent on the active connection simply by selecting the button that contains the message number and hot key name (e.g., Msg 1 (Put PCM in ExtSession)).

14.5.1 Sending Hot Keys Using Shortcuts

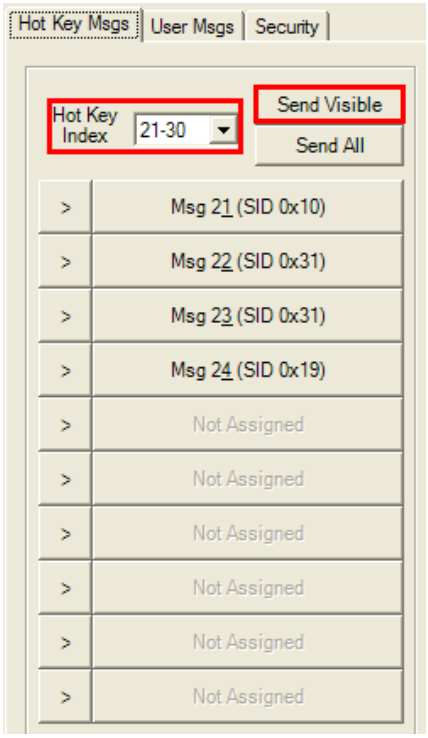
Ten hot key messages are always displayed at a time. As an alternative to selecting the hot key button to send, the "Alt" shortcuts can be used to send each message by entering Alt-1 to send the first hot key message, Alt-5 to send the fifth hot key message, etc. Each hot key is underlined with the Alt shortcut that applies to it.



Msg 1 (Routine$0202)

14.6 Send Visible

This option sends all the visible assigned Hot Key Messages (up to 10) on the active connection. Only the messages assigned and in the selected "Hot Key Index" range will be sent. The example below will send Message 21 to Message 24, as the selected "Hot Key Index" is 21-30, but only Hot Keys 21, 22, 23 and 24 are active.



14.7 Send All

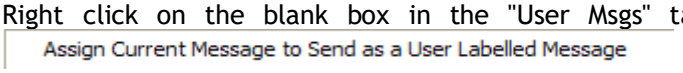
This option sends all the assigned Hot Key Messages (up to 100) on the active connection.

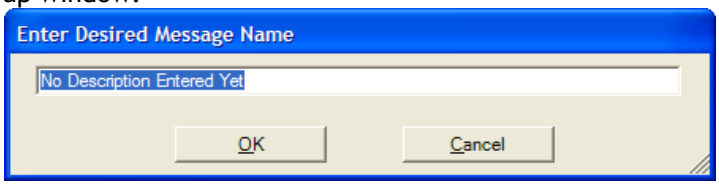
15 User Messages

User messages section, allow the users to save specific messages in this section; which can then be sent to the network directly by double clicking the request from the "User Messages" tab.

15.1 Adding New User Labelled Message

There are three ways to create a user labelled message:

1. Right click on the blank box in the "User Msgs" tab. The following box will be displayed:
   

   When clicking on the box above, the current message selected on the "Generic Diag View" will be assigned as a "User Labelled Message". To finalize, only select the name of the message, on the popup window.
   

2. Right click on the "Message to Send" drop down box in the "Generic Diag" view and choose "Assign this as a User Labelled Message".
3. See details within section 14.4.

15.2 Deleting a User Labelled Message

Right click on the user labelled message. Select "Delete the User Labelled Message".

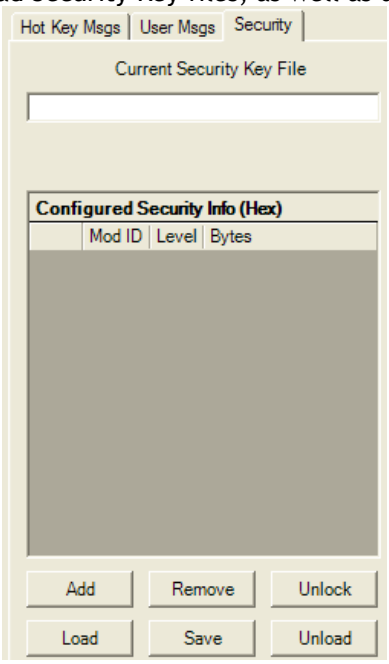
Refer to section 12.5.1 for a mechanism to clear all user defined messages at once.

15.3 Update Descriptive Name of the User Labelled Message

Right click on the user labelled message. Select "Update Descriptive Name of the User Labelled Message". The window that requests to "Enter Desired Message Name" will open again, and the name can be modified.

16 Security

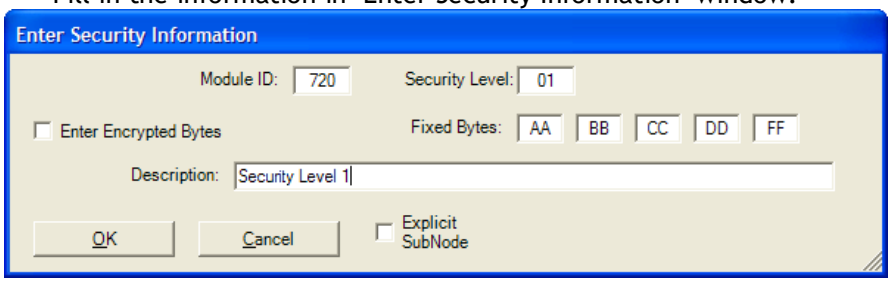
In this section, users can create and load Security Key files, as well as use them to unlock ECUs.



16.1 Add

A new Security Key can be created, using this button, the steps to do so are:

- Click on the add button.
- Fill in the information in "Enter Security Information" window.

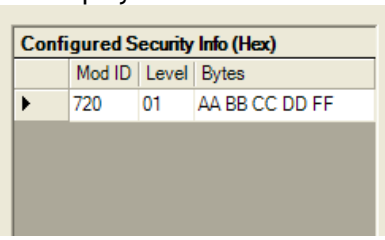


- Module ID:
  - 11 bit CAN ID of the module the security information is for (in hexadecimal format). The supported range is 0x700 - 0x7FF.
- Security Level:
  - This is the security level for which the security information is being added. This is always a one byte odd value entered in hexadecimal format.
- Fixed Bytes:
  - These are the standard 5 fixed bytes used with the Ford Security Algorithm.
- Description:
  - Any description selected by the user for the current security level.
- Enter Encrypted Bytes:
  - Instead of entering the 5 fixed bytes directly, this allows an encrypted version of the 5 fixed bytes to be entered. Encrypted bytes consist of 16 bytes using 32 ASCII characters. If the

format or the length is not correct, DET will display a warning, and the security information will not be kept.

- Note that Ford supports 3 encryption levels. The first 2 are intended for engineering usage and are supported by the DET. The third encryption level is intended for production usage and is only supported in the DET on an as needed basis.
  - Unvalidated MDX 003 files will typically contain the fixed bytes encrypted in one of the first 2 encryption levels. Validated MDX 003 files will typically have this encryption converted to the production encryption level.

- Click OK, the security level information and fixed bytes should be displayed on the Security Tab display.



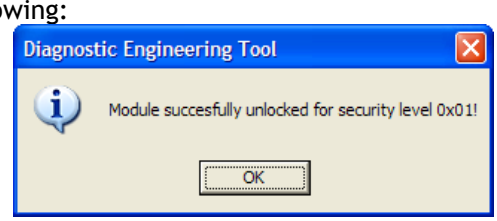
16.2 Remove

Remove the selected security level

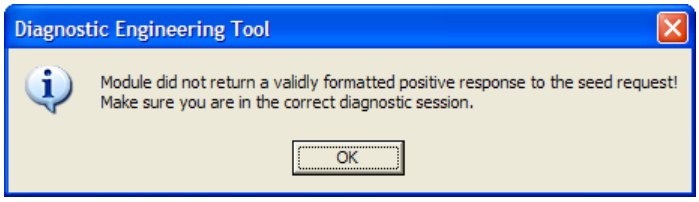
16.3 Unlock

This will attempt to unlock the currently selected security level. It is important that the user first ensure they have placed the ECU in the correct diagnostic session in which the security level is supported. By default, the DET gets the security seed from the module and sends the calculated key to the module, when pressing the Unlock button.

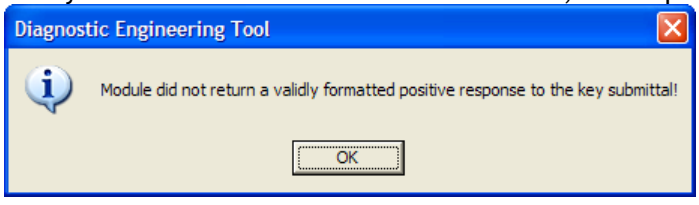
If the module was in the correct Diagnostic Session and the challenge bytes of the key are the correct, the module will respond with the following:



If the module was not in the correct session (i.e. Unlocking with level $01 in session $03), the response will be as follows:



If the Challenge bytes of the key are not the correct ones for the module, the response will be as follows:

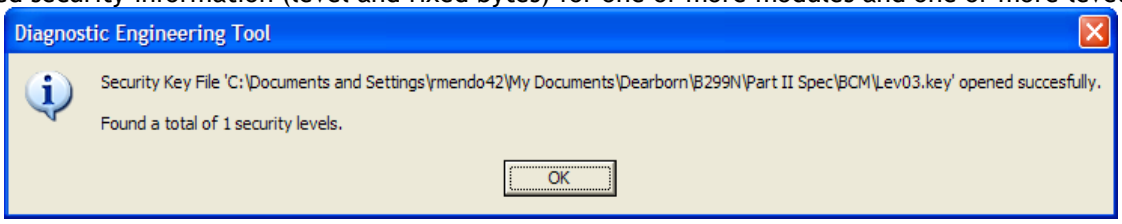


16.4 Save

This saves all security level information in the display into a .key file that can then later be manually loaded or loaded automatically when saved as part of a .dcs file.

16.5 Load

This manually loads a previously created Security Key file (*.key). These security key files contain ECU IDs and associated security information (level and fixed bytes) for one or more modules and one or more levels.



If the security key file loaded does not contain valid information (format) in it, different set of errors will appear:

- This is an unexpected token. The expected token is 'EndElement'. : The Security Key file does not have the correct format, verify in the file, all elements have their corresponding end tag (i.e. </KEY>, </SECURITY_INFO>, </MODULE_ID>, </DESCRIPTION>, </LEVEL>, </FIXED_BYTE>).
- There is an unexpected end tag. There is one additional end tag in the file, which means an open tag is missing, or an additional (not required) end tag is in the file. DET will give further details on the location of the unexpected end tag.
- Error Loading Security Key Data: Cast from string "&H?" to type 'Integer' is not valid. : Content of the file that should contain integer values, contain string values. The character "?" in the error description, displays the character that the tool could not convert to integer.
- If the MODULE_ID is missing in the file, DET will open the file successfully, but no Security Key Levels will be loaded.
- If LEVEL is missing, DET will open the file successfully and assign the level "00" to the Key, if the challenge bytes are populated.
- If any FIXED_BYTE (challenge byte) is missing, DET will open the file successfully and assign "00" to the missing bytes. Please note DET cannot determine which byte is missing, so will take the available information and will assign it as the first bytes of the challenge bytes, the missing bytes will be at the end of the FIXED_BYTES chain and will be filled with "00".

For more information about the KEY file format, please refer to section A.2.

16.6 Unload

This complete clears the Security tab display and all security information, including that any key file is associated with the .dcs.

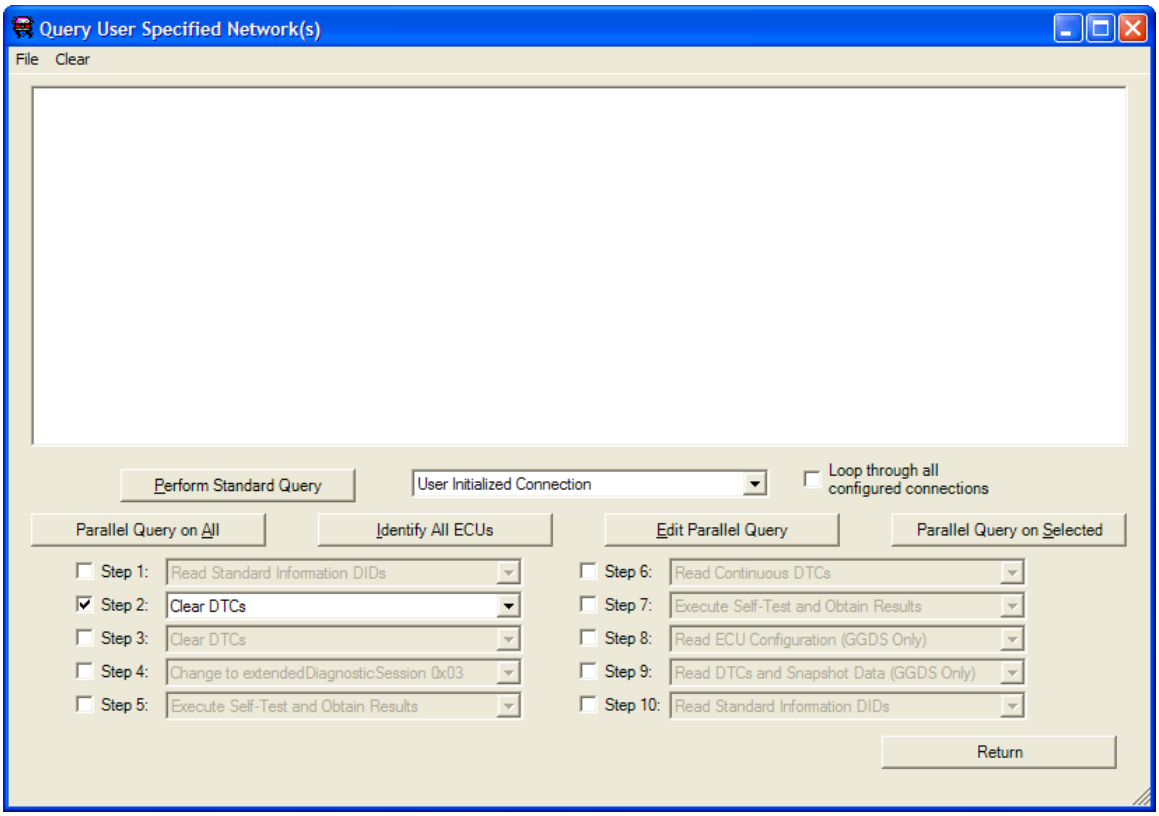
16.7 Special Security Levels

Security Levels 0x01 – 0x1F are required to support the standard Ford security algorithm. There are some additional non-standard security levels supported by the tool (e.g., PATS). When configuring these security levels, the Fixed Byte information is not necessary. The DET knows based upon the security level number, which security type it is.

17 Bus Query Tool

With this tool, users can send a set of Diagnostic Messages to all the modules connected in the networks of the vehicle. The messages can be transmitted on only a single network or on all connected networks. There are also options for performing the bus query sequentially on all ECUs, or in parallel on all or just the selected modules. The tool is located at: Tools > Bus Query Tool or by selecting the Bus Query icon in the toolbar.

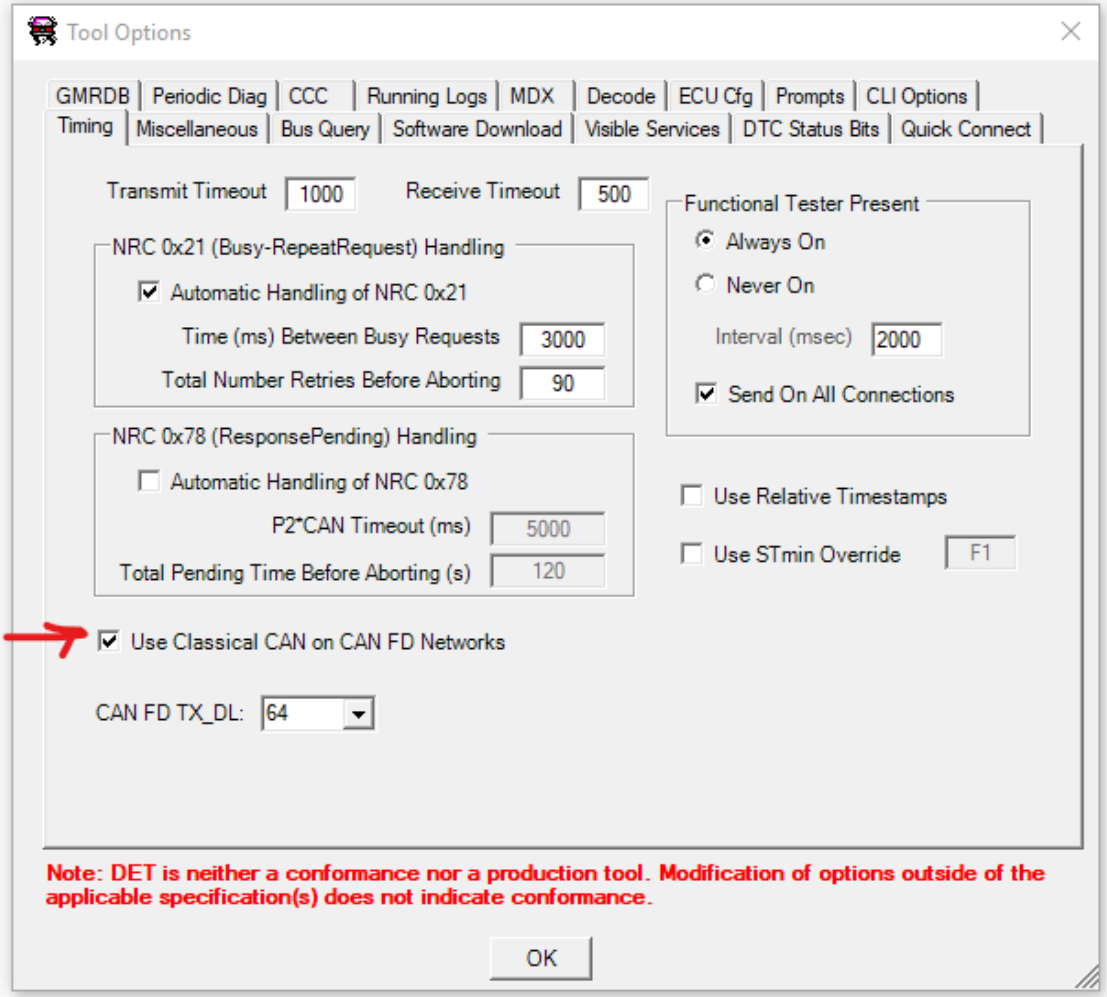
Note: The bus query tool supports not only the ISO 14229 based Ford diagnostic ECUs, but also the previous generation Ford diagnostic ECUs implementing to the CAN Generic Diagnostic Specification (CAN GDS).



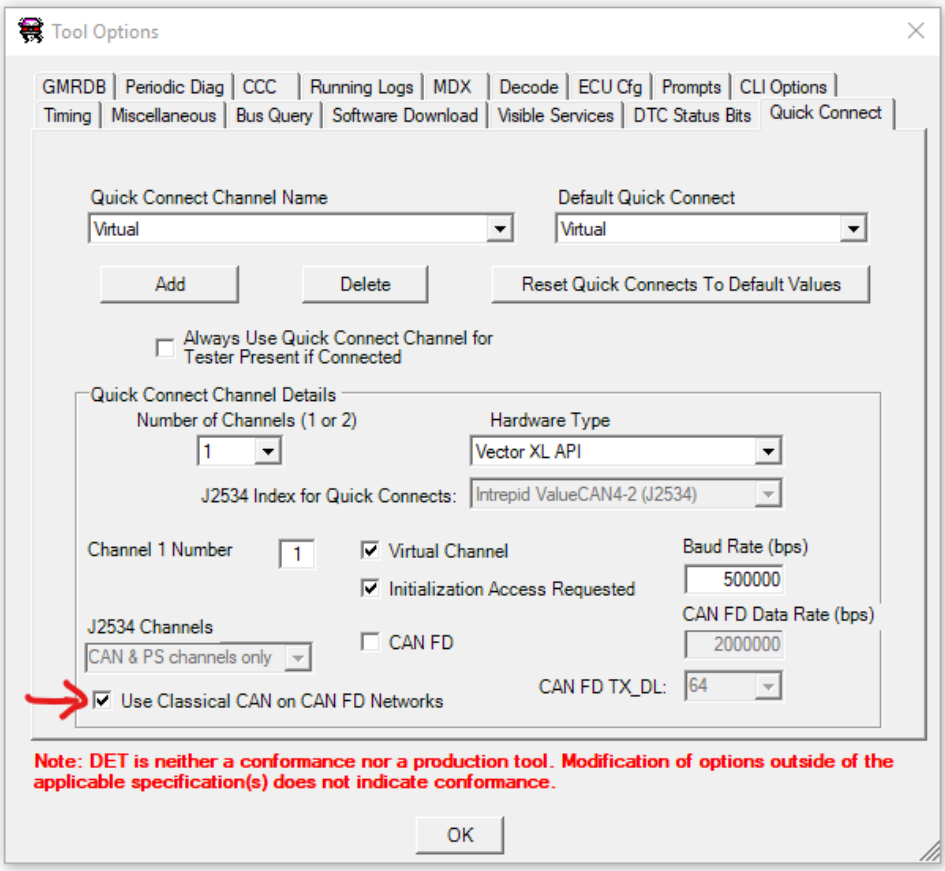
17.1 Bus Query Connections

If the "Loop through all configured connections" checkbox is selected, then all of the bus query functions will be attempted on all hardware connections. This allows performing a bus query on all vehicle networks with a single click. If this checkbox is not selected, then all requests will only be sent on the connection in the drop down box to the left of the checkbox.

When using a CAN-FD capable device, ensure that that the Use Classical CAN on CAN FD Networks is enabled on the Tools > Options > Timing tab.



If you are using Quick Connects with a CAN-FD capable device, ensure that that the Use Classical CAN on CAN FD Networks is enabled on the Tools > Options > Quick Connect tab.



17.1.1 Choosing the Correct OBD-II Connector

Depending on vehicle architecture, especially on CGEA 1.3c, FNV2.x, FNV3, and development vehicles, which can be a mix of different architectures, there can be 2 or more OBDII connectors, and not all ECU's are connected to the regulatory OBDII connector.

Behind the gateway module, there may be additional connectors to the vehicle bus. If there are 2 or more connectors behind the gateway module, they may connect to different ECU' and different results will occur on each connector.

It is the responsibility of the user to know and confirm the architecture of the vehicle. NetCom will not provide support in identifying the vehicle architecture.

17.2 Perform Standard Query

The standard bus query generically scans the user selected networks for all possible ECUs by starting at CAN ID 0x700 and continuing until CAN ID 0x7F7. Whenever a response is found to a given request, the standard query will then perform up to 10 pre-defined steps based upon the boxes that are checked and the values within those boxes.

Note that the standard bus query tests each ECU sequentially. For example, if it first finds ECU 0x720 it will perform the 10 selected step actions on that ECU. After that, it will continue searching (0x721, 0x722, etc.) and upon finding the next ECU it will perform all 10 selected steps on that ECU. The networks are also tested sequentially so that ECUs are first scanned on the first connection, then the next, etc.

17.3 Identify All ECUs

This generically queries all ECUs on all connections without performing any of the 10 bus query steps. This is a quick and easy way to see what ECUs are on each network, and also populates the list of ECUs that will be executed when the "Parallel Query on Selected" is performed.

17.4 Parallel Query on All

Unlike a standard bus query, parallel bus queries will perform each step on each ECU prior to moving to the next step. For example, if 10 ECUs are found on connection 1 and 5 ECUs are found on connection 2, the parallel bus query will send bus query step 1 (if selected) to all 15 ECUs, and then step bus query step 2 (if selected) to all 15 ECUs, etc. The parallel bus query can save significant time when performing potentially lengthy steps such as "Execute Self-Test and Obtain Results" as the self-test will run in parallel.

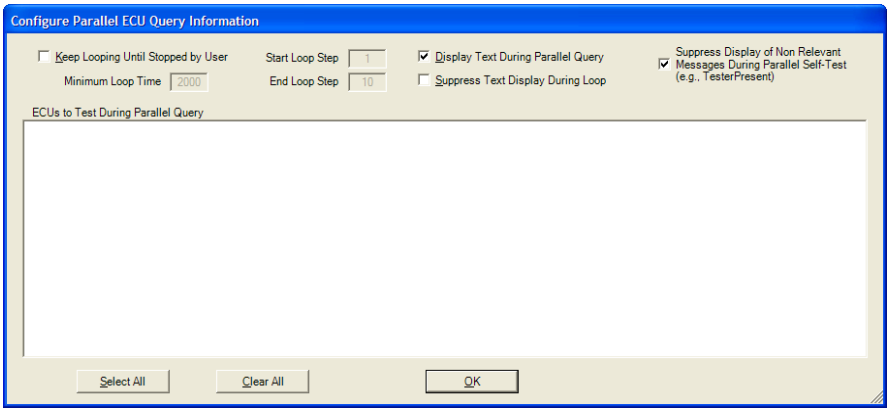
Note that when Parallel Query on All is selected, the DET will first find all ECUs on the connections by performing the "Identify All ECUs" option as described in section 17.3.

The parallel bus query can also be customized as described in 17.6.

17.5 Parallel Query on Selected

This button performs the same task as Parallel Query on All, but ECU IDs that are unchecked for testing (see section 17.6) will not be sent any requests. Example use cases for this would be to test permutations of ECU self-tests that can and can not be executed in parallel by selecting turning certain ECUs "on" and "off". Once ECUs are identified (see 17.3) this option can be used to do a parallel query on all ECUs (without doing a re-identification) by not unselecting any ECUs.

17.6 Edit Parallel Query



This menu allows the user to customize how the parallel bus query behaves.

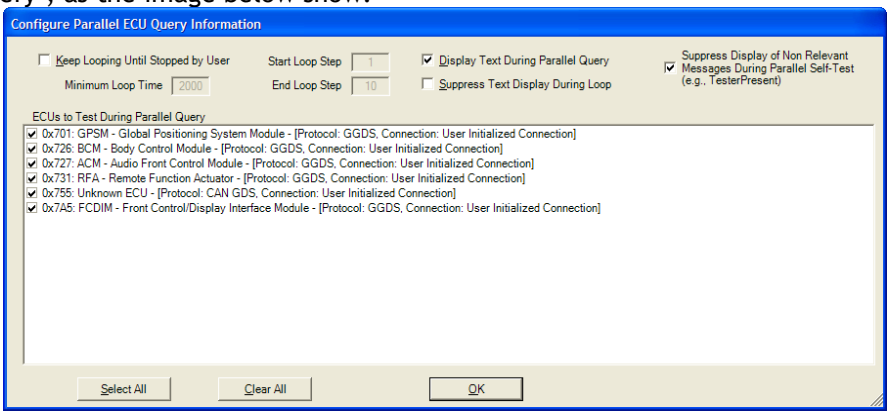
Details of the options are as follows:

- Keep Looping Until Stopped by User:
  - If this is unchecked, the parallel query performs the 10 selected steps in order as described in 17.4, doing each step once. If this is checked, the user can then specify one or more steps that the parallel bus query will continue to perform until stopped by the user (using minimum loop time, start loop step, stop loop step). See example in 17.6.1.
- Start Loop Step / End Loop Step
  - If "Keep Looping Until Stopped by User" is selected, then the parallel bus query will repeat all steps between the Start Loop Step value and the End Loop Step value until the bus query is stopped by the user. Once it is stopped, the bus query will finish any remaining steps.
- Minimum Loop Time

- If "Keep Looping Until Stopped by User" is selected, then this time is used to force a specific minimum delay between consecutive loops of sending the steps between Start Loop Step and End Loop Step. In other words, the timer is started once the first Start Loop Step is sent and after completing the End Loop Step, the next Start Loop Step will not begin until this timer has expired.

- Display Text During Parallel Query:
  - If this is selected then the parallel bus query will decode relevant step results and display in the user display. If unselected, no text will be displayed to the user.
- Suppress Text Display During Loop:
  - If this is selected then any bus query steps that are within the loop (only applicable if "Keep Looping Until Stopped by User" is selected) will not be decoded and displayed (but other will). An example use case would be if the loop is used to send a testerPresent physically to all ECUs that support prior Ford CAN diagnostic requirements (which did not use functional addressing). In this case, displaying the testerPresents to all modules which may be sent every 2000ms does not provide much value.
- Suppress Display of Non Relevant Messages During Parallel Self-Test:
  - When this is selected, all intermediate responses (i.e., primarily the NRC 0x21 busyRepeatRequest responses to routine results) will not be displayed during the parallel self-test query. While this may be useful sometimes for troubleshooting, it is generally not necessary and only complicates viewing the results.

In addition, the parallel query allows one to select only the specific ECUs to perform the query on. To accomplish this, first identify the ECUs (see 17.3) and then select the modules available in the "ECU to Test During Parallel Query", as the image below show.



17.6.1 Example Customized Parallel Bus Query

The following is an example real world use case and how the parallel bus query can be used to achieve it. During programming of Ford ISO 14229 based ECUs, normally the tester will functionally put all ECUs into the programmingSession to automatically have them stop logging DTCs and stop transmittal of non-diagnostic messages. However, some vehicle programs have many legacy ECUs that implement an older Ford diagnostic protocol (i.e., CAN GDS) which did not support functional addressing. In order to prevent these ECUs from logging DTCs and to disable them from transmitting requires a tool to send requests to each ECU and also periodically send requests (to each ECU) to prevent them from timing out, followed by a reset at the end. This can be accomplished as follows with the customized parallel bus query.

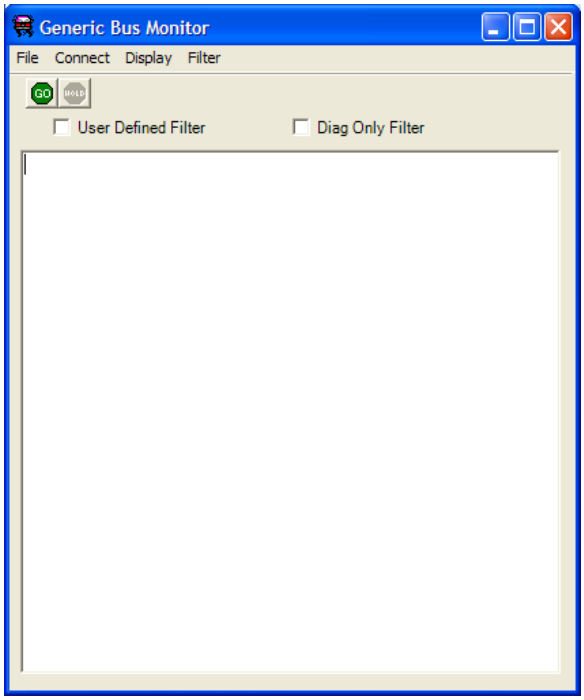
- Enable Step 1 and choose "Control DTC Setting "
- Enable Step 2 and choose "Communication Control"
- Enable Step 3 and choose "Tester Present"
- Enable Step 4 and choose "ECU Reset"
- Disable all other bus query steps
- Select "Keep Looping Until Stopped by User"

- Select Start Loop Step = 3
- Select Stop Loop Step = 3
- Select Minimum Loop Time = 2000

With the above setup, the user can first identify all of the ECUs and then select ONLY the CAN GDS ECUs to be included with the parallel bus query. When "Parallel Query on Selected" is performed, the DET will send Control DTC Setting service to each of the ECUs, it will then send Communication Control to each of the ECUs, it will the send TesterPresent at step 3 every 2000ms until the user manually stops it, and will find send the ECU reset to all ECUs. Note that multiple copies of the DET can be executed simultaneously when using the Vector hardware API, but not using SAE J2534 API.

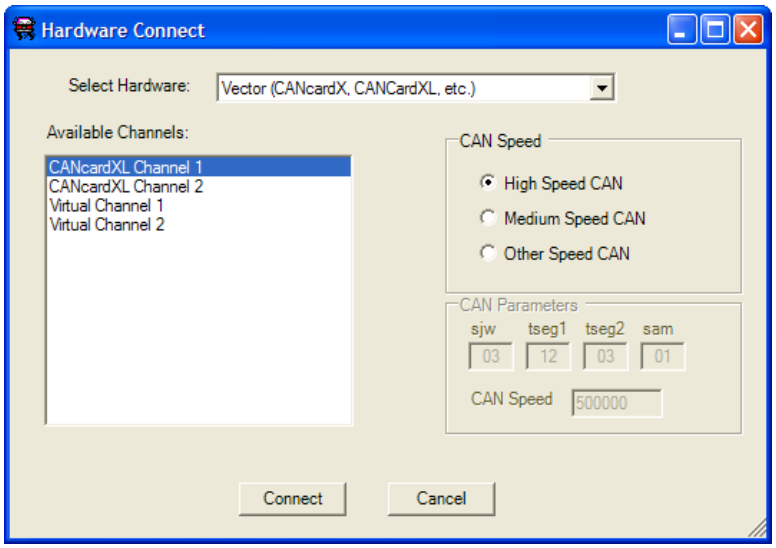
18 Generic Bus Monitor Tool

The Generic Bus Monitor is a tool capable of monitoring / recording all CAN traffic on a network. The latest versions of the DET tool save the information in ASC format (that is compatible with Vector CANalyzer software). Earlier versions of DET saved the information as an RTF file. The tool is located at: Tools > Generic Bus Monitor. Filtering can be performed by a user defined request mask or a quick and easy diagnostic filter.



18.1 Connect to a network

Click on "Connect" on the main Menu of the "Generic Bus Monitor" tool. A window will open (User Connection window), where the Hardware and Speed needs to be selected. The working of this form is the same as described in 3.3.1, but it creates a different logical connection.



Please note that the only available connection for "Generic Bus Monitor" tool is "User Connection", Quick Connections are currently not available on this tool. When using the Vector API, this tool can be used simultaneously with a connection to the same physical hardware channel through the DET main form. However, this support is not guaranteed through SAE J2534 API.

18.1.1 Choosing the Correct OBD-II Connector

Depending on vehicle architecture, especially on CGEA 1.3c, FNV2.x, FNV3, and development vehicles, which can be a mix of different architectures, there can be 2 or more OBDII connectors, and not all ECU's are connected to the regulatory OBDII connector.

Behind the gateway module, there may be additional connectors to the vehicle bus. If there are 2 or more connectors behind the gateway module, they may connect to different ECU's and different results will occur on each connector.

Additionaly, some gateway modules will not pass all the messages out to the regulatory OBDII connector. One of the connectors behind the gateway will need to be used.

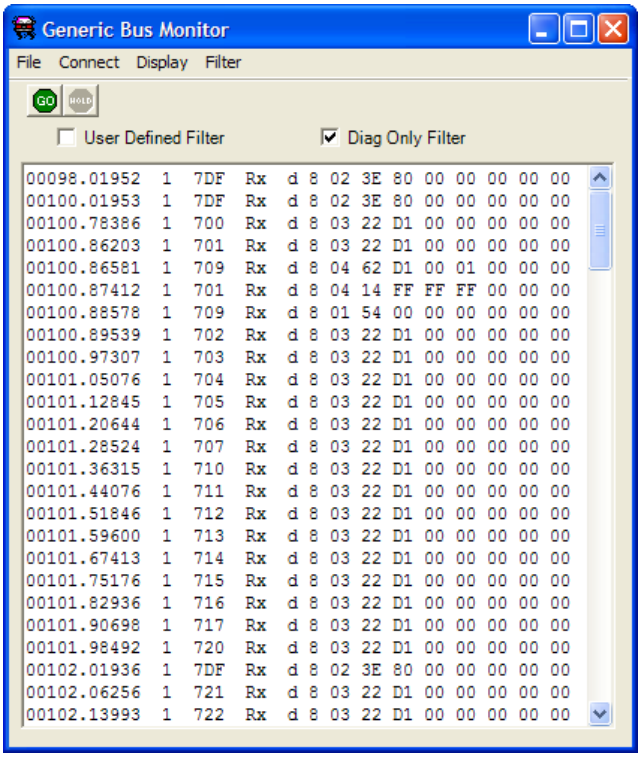
It is the responsibility of the user to know and confirm the architecture of the vehicle. NetCom will not provide support in identifying the vehicle architecture.

18.2 Start Bus Monitoring

Preconditions:

- Generic Bus Monitor opened.
- Connection to the network established.

Click on the GO button, if any messages are present on the network, they will be displayed on the display (depending on the configuration of the filters, if any). In the example below, the filter to display only diagnostic messages is selected which includes only those in the 0x700 – 0x7FF range.



18.3 Hold Bus Monitoring

When monitoring of the bus has started, the display of the CAN frames can be placed on hold in order to more easily look at the data. The "Hold" button (to the right of the "Go" button) is enabled when bus monitoring is active and will allow pausing the display of frames. Note that all frames will still be collected and stored while the display is on hold.

18.4 Stop Bus Monitoring

When monitoring of the bus is active, the queuing and display of the frames is stopped by pressing the Stop button, which is enabled when bus monitoring has started.

18.5 Clear Bus Monitor Display

- Select Display on the Main Menu
- Select Clear.
- A window will open asking if the information should be saved. If so, click YES and create the file where the information is going to be saved. After saving, the display will clear. If the information is not going to be saved, simply click NO and the display will clear.

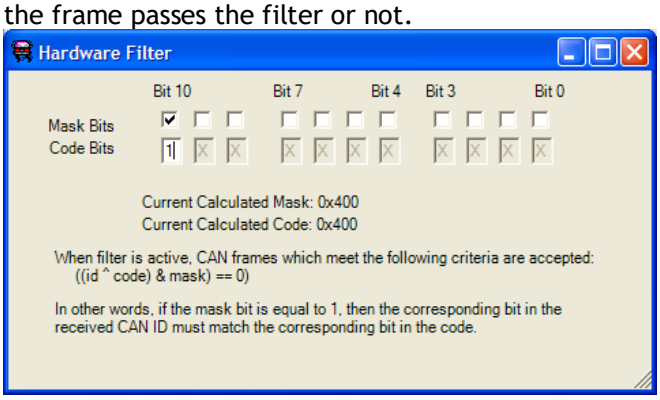
18.6 Filter

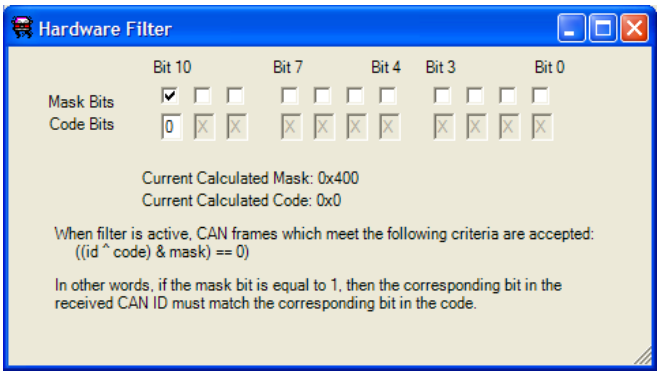
The tool has the ability to implement a filter to queue and display just a specific set of messages based on their message ID (11-bit identifier). To do so, follow the next steps:

- In the main menu of the tool, select Filter.

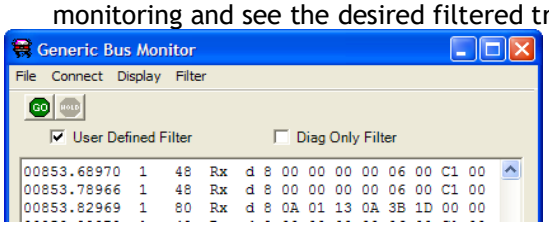
- The "Hardware Window" will pop (see image below). To select the desired filter, 2 sections need to be filled: "Mask Bits" and "Code Bits"
  - Mask Bits: Determine which bits of the message identifier the tool will select, to filter the messages.
  - Code bits: Determine the values (0,1) each bit of the mask should contain to filter the messages.

Examples:
The example to the left below would only allow CAN frames in the range of 0x400 – 0x7FF to pass through the filter. This is because it is only ensuring that the most significant bit is 1. The example to the right below would only allow CAN frames in the range of 0x000 – 0x3FF to pass through the filter. This is because it is only ensuring that the most significant bit is 0. If one only wanted to let a single ID through the filter, then the mask bits would be all checked and the code would equal the module ID. This is because it essentially ANDs the CAN frame with the mask and determines if it is equal to the code in order to determine whether the frame passes the filter or not.



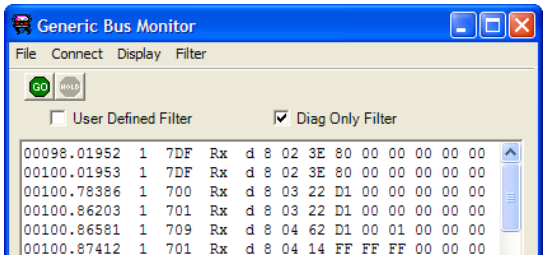


- Finally, click on "User Defined Filter" checkbox, to enable the configured filter. Start the bus monitoring and see the desired filtered traffic on the display of the tool.



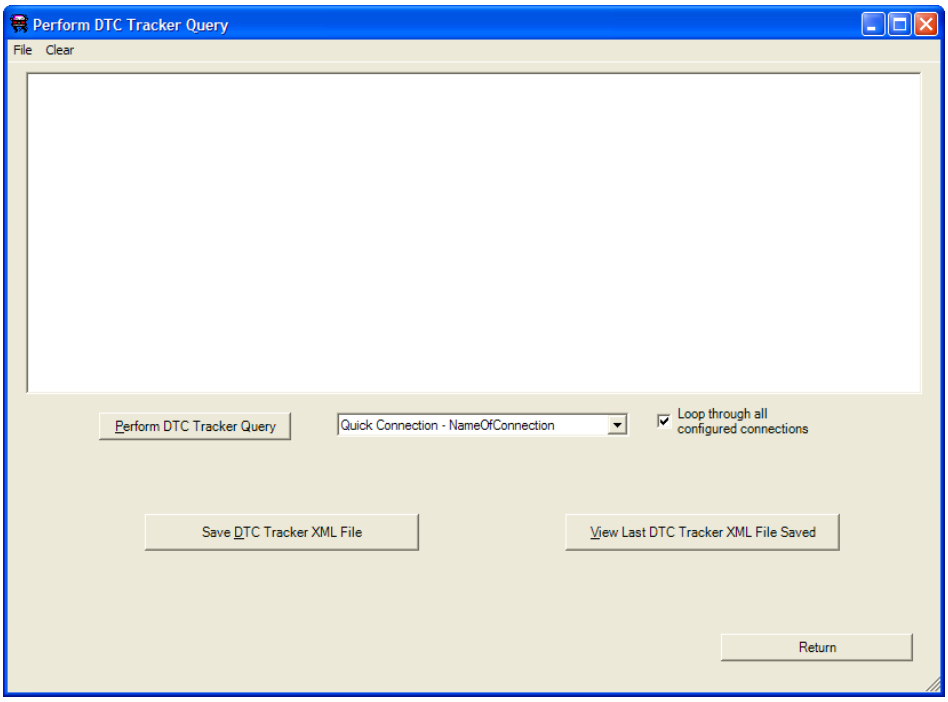
18.6.1 Diag Only Filter

In addition to the User Defined Filter, a predefined filter is already set up on the Generic Bus Monitor Tool that will only pass through diagnostic frames. When selected, the filter passes only the diagnostic frames (range 0x700 – 0x7DF), which are displayed on the screen while running a bus monitor. This is effectively accomplished by using a mask of 0x700 and a code of 0x700.



19 DTC Tracker Tool

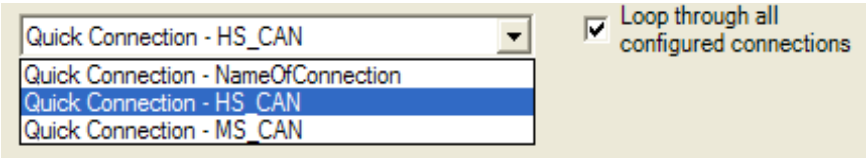
Various areas within Ford utilize DTC Tracker functionality as part of the development process in order to track the number of DTCs set. The "normal" DTC tracker functionality is part of another tool, but various users requested the functionality be put into the DET so that they could use a single tool and use the same hardware they are already using. This effectively scans the vehicle connections for DTCs and saves the results in a predefined XML file format required by the back end DTC Tracker system. The tool is located at: Tools > DTC Tracker Tool.



19.1 Connections

If the "Loop through all configured connections" checkbox is selected, then the scanning of DTCs will be attempted on all hardware connections. This allows performing a DTC tracker event on all vehicle networks with a single click. If this checkbox is not selected, then all requests will only be sent on the connection in the drop down box to the left of the checkbox.

In the example below, the tool will loop between the 3 preconfigured "Quick Connections" (see section 3.3.2) and return the DTCs of all those networks.



If "Loop through all configured connections" is disabled, the tool will just query the ECUs of the active connection.

19.2 Save DTC Tracker XML File

This button allows saving the information obtained through the DTC Tracker Tool into the predefined XML format. This is typically then imported into a different system for processing. After clicking the button, a

window will pop up, requesting the user to introduce the vehicle ID (*this information is optional, the user can click OK leaving this field in blank*). Finally the user will select where to save the DTC Tracker XML file.

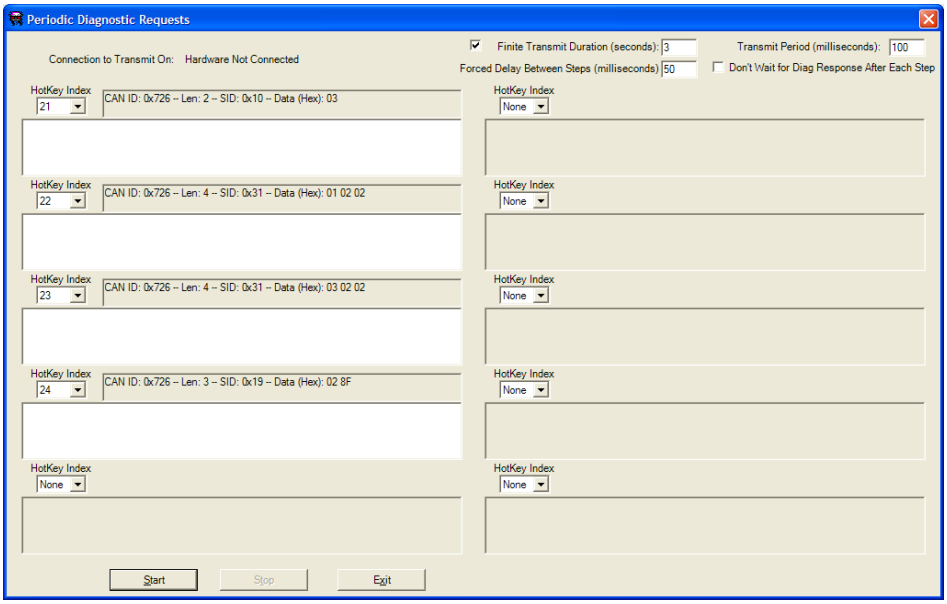
19.3 View DTC Tracker XML File

The last DTC tracker XML file will open (the application used to open the XML file varies depending upon the default program used for viewing XML files in each computer).

20 Periodic Diagnostic Tool

The "Periodic Diagnostic Tool" is a tool used to send one or more diagnostic messages periodically on the network. Non-diagnostic frames are also possible to be sent. The tool is located at: Tools > Periodic Diagnostic Tool.

The Periodic Diagnostic Tool essentially loops in order from the top left down and then the top right down. A full loop is considered the transmitting of all 10 messages (assuming 10 are selected).



20.1 Send Diagnostic Requests

The steps to start Periodic Diagnostic requests are:

1. The requests to send must first be added as Hot Keys (see section 14).
2. Open the Periodic Diagnostic Tool, located at: Tools > Periodic Diagnostic Tool.
3. Assign the requests to send, by choosing the corresponding Hot Keys in the "HotKey Index" drop box. A maximum of 10 requests (Hot Keys) can be selected. In example below, Hot Keys 21 and 22 were selected:

4. Modify the options of the periodic transmission. The available options are:

- Finite Transmission Duration (seconds):
  - If selected, the periodic diagnostic tool will keep sending messages (up to 10) periodically until this overall time is reached. The total time is checked at the end of each full loop. If disabled, the tool will continue to loop through all messages until the "Stop" button is pressed.

- Transmit Period (milliseconds):
    - The total period to transmit the whole set of enabled requests. The tool will send all configured messages (up to 10) and then wait for this timer to expire before starting the next loop. After this timer has expired, the tool will start to send the next loop of messages. This is useful if one wanted to send a request (e.g., read DTCs) but only once a second instead of as fast as possible.
- Forced Delay Between Steps (milliseconds):
    - If this value is non-zero, the tool will wait this amount of time in milliseconds after transmitting the request for one step prior to transmitting the request for the next. Note that the time could be longer than this if the tool is waiting for a diagnostic response.
- Don't Wait for Diag Response After Each Step:
    - If the option is enabled, after sending a message that is a diagnostic message, the tool will not wait for a response before moving onto the next transmission. However, the tool will still respect the "Forced Delay Between Steps" value. If the option is disabled, the tool will wait for the final response from the ECU (which could be on the order of seconds if NRC 0x78 is used) prior to proceeding to the next step.

5. Press the "Start" button to begin the periodic transmission.

20.1.1 Non-Diagnostic Example

A non-diagnostic example that illustrates some of the periodic diagnostic tool settings is setting up a normal communication message that is used to turn on and off a turn signal. It is possible to use the periodic diagnostic tool to control the frequency as well as the on time versus the off time. In this example, assume CAN ID 0x123 controls the turn signal and a value of 0x80 in the first byte turns the left turn signal on and a value of 0x00 in the first byte turns the left turn signal off. These two non-diagnostic CAN frames can be defined (see 4.2.31) and added as hot keys. If there was a desire to turn have the turn signal turn on with a period of 200ms, with a duty cycle of 75 percent then the following could be done. The hot key index containing the CAN frame to turn the turn signal off could be added as the first message sent by the periodic diagnostic tool, followed by the hot key index containing the CAN frame to turn the turn signal on. If the overall period was set at 200ms and a "Forced Delay Between Steps" of 50ms was added this would result in the following:

- Time = 0
    - Turn signal off sent
- Time = 50 (delayed due to "Forced Delay Between Steps")
    - Turn signal on sent
- Time = 200 (new loop ready to start after previous response, but delayed due to transmit period timer not expired)
    - Turn signal off sent
- Time = 250 (delayed due to "Forced Delay Between Steps")
    - Turn signal on sent
- The above would repeat until stopped by the user (or until finite transmission duration is satisfied depending upon user options).

20.2 Display

The display options can be modified in Tools > Options > Periodic Diagnostic Tab. For further details on the configurable display information, please refer to section 11.9.

20.3 Save information

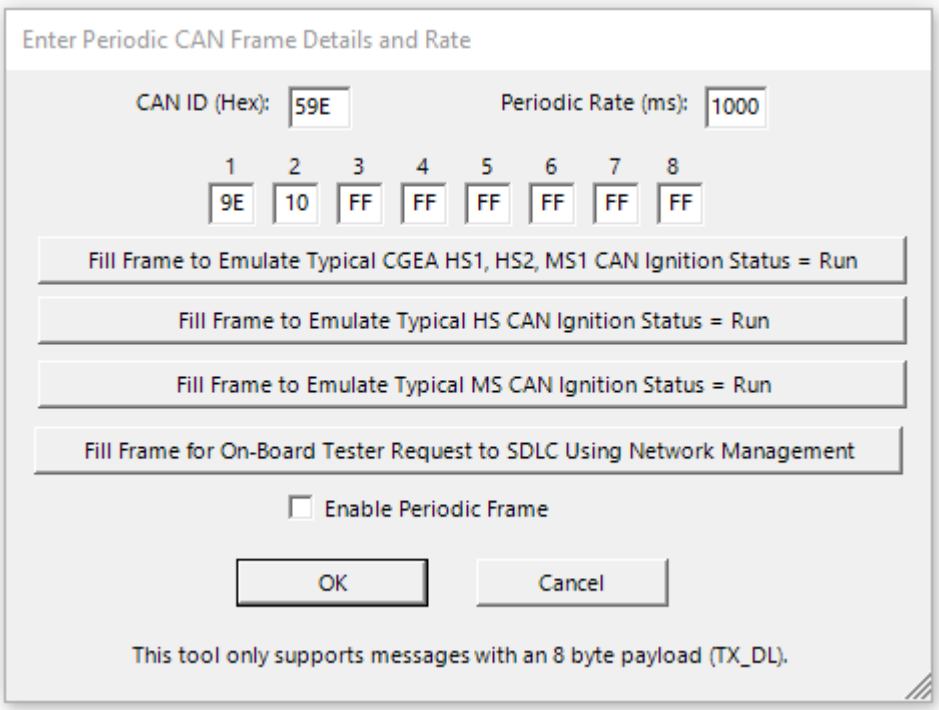
The information that can be saved is:

- The information Log (see section 12.2.11).
- The message frame log (see section 12.4.2).

It can also be saved into disk automatically, under Tools → Options → Running Log Tabs and turning on the desired running log file.

21 Periodic Frame Tool

With this tool, a single CAN frame with a payload (TX_DL) of 8 bytes can be transmitted periodically over the network where the DET is connected. The primary original use case for this was for testing ECUs on a bench where the ECU needed an ignition status signal in order to perform much of its normal functionality with predefined frames available for the ignition status message. However, any CAN frame with a payload (TX_DL) of 8 bytes can be used. The tool is available at: Tools → Periodic Frame Tool.



21.1 Start Periodic Transmission

The following steps should be followed:

- Open Periodic Frame Tool from: Tools → Periodic Frame Tool.
- Fill the data of the message to be transmitted:
  - CAN ID: 11 bit identifier of the frame.
  - Periodic rate (ms): Period time for the message to be transmitted.
  - Frame Data: Fill the 8 bytes of information of the frame.
  - Send the frame periodically: To do so, select the "Enable Periodic Frame" checkbox. The tool will start transmitting the periodic frame on the network it is connected to.

21.2 Fill Frame Data to Emulate Typical HS CAN / MS CAN Ignition Status = Run

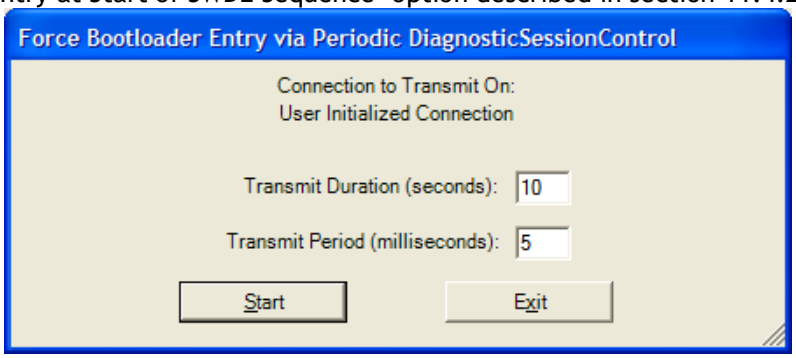
When pressing either of this buttons, the Frame Data and CAN ID fields, are filled with default information emulating Ignition Status = RUN.

On HS CAN the message is: 0x3B3 0x41 0x04 0x00 0x00 0x00 0x00 0x00 0x00, which means Ignition Status = Run and Key = In.

On MS CAN the message is: 0x3A0 0x4A 0x00 0x00 0x00 0x00 0x00 0x00 0x00, which means Ignition Status = Run, Key = In, and Ignition Switch stable.

22 Force Bootloader Entry

This tool allows for a simple mechanism to force entry to the bootloader using the backdoor method by sending a periodic functionally addressed diagnosticSessionControl with session equal to the programmingSession. The default values are to transmit at a period of 5ms for a duration of 10s. When the "Start" button is selected, the user will be prompted to cycle power to the ECU. If this approach is used, the normal periodic testerPresent should be enabled to prevent the ECU from timing out. Note that this approach can be done automatically as part of the programming process by selecting the "Attempt Backdoor ProgrammingSession Entry at Start of SWDL Sequence" option described in section 11.4.2.



23 MGM Image Flash Tool

This tool is utilized to perform a USB based update of the MGM / APIM / Sync module. Further details will be added if necessary in a future release of this user manual.

24 Command Line Interface

24.1 Summary

The Diagnostic Engineering Tool has a command line interface that can be used in conjunction with the Quick Connects feature. The intent of this functionality is to allow people to utilize batch files or an external program to launch the Diagnostic Engineering Tool, perform specific functionality such as a software download, etc. and then determine the success or failure of the action based upon the returned error code. Note that the batch file or external program can utilize the "-f" command line argument in order to utilize specific configuration settings, etc.

24.2 Command Line Format

The format of command line arguments (where parameters in [ ] are optional) is as follows:

[-f dcsfile.dcs] [–NN [X]]

- The command line argument of "-f" (if used) must be the first command line argument and must be followed by the desired DCS file to load. Without this command line argument, the DET will always load the DefaultSettings.dcs

The –NN command line argument may be used with or without the "-f" argument. If both are used, the "-f" argument including the filename must proceed the –NN argument. The following are valid values for the –NN command line argument:

- A value of "-01" will automatically connect quick connects, initiate an automated SWDL (see section 9) based upon the .DVS associated with the loaded DCS file, and then exit
- A value of "-02" will automatically connect quick connects, initiate a normal SWDL sequence based upon the 1st sequence in the .VBS associated with the loaded DCS file, and then exit
- A value of "-03" will automatically connect quick connects, send all user defined hot keys, and then exit.
- A value of "-04" will automatically connect quick connects, perform standard bus query, and then exit.
- A value of "-05" will automatically connect quick connects, initiate a normal ECU Configuration transfer sequence based upon the 1st sequence in the .ECD associated with the loaded DCS file, and then exit
- A value of "-06" will automatically connect quick connects, and send all user defined messages in the 1st user defined message sequence associated with the loaded DCS file, and then exit
- The –NN values of "-02", "-05", and "-06" have the ability to have an optional additional command line argument immediately following it. "-02 X" will automatically connect quick connects, initiate a normal SWDL sequence based upon the Xth sequence in the .VBS associated with the loaded DCS file, and then exit. For example "-02 1" performs the same functionality as "-02". "-05 X" will automatically connect quick connects, initiate a normal ECU Configuration transfer sequence based upon the Xth sequence in the .ECD associated with the loaded DCS file, and then exit. Similarly, "-06 X" will automatically connect quick connects, and send all messages in the Xth user message sequence in the loaded DCS file, and then exit.

24.3 File Locations

It is strongly recommended that the .dcs file be placed in the same directory as the .bat file, as well as any other files such as a .key, .ecd, .mdx, etc., or that the file paths be fully qualified with the complete paths.

If these files are located on network drives and are shared with other people, errors may result if users do not have access to the network drives.

24.4 Error Codes

The Diagnostic Engineering Tool will return an error code when executing from the command line. The returned error codes are decoded as follows:

- 0 = Success
- 1 = Invalid Command Line Format
- 2 = Invalid Parameters (e.g., index of VBS sequence to download out of range)
- 3 = DVS (Auto SWDL) Attempted and Failed
- 4 = VBS (Normal SWDL) Attempted and Failed
- 5 = Transmission of All Hot Keys Failed
- 6 = Transmission of a Bus Query Request Failed
- 7 = ECD (Normal ECU Configuration Transfer) Attempted and Failed
- 8 = DCS File To Load Not Found
- 9 = Transmission of User Message Sequence Failed
- 10 = .key file was not found
- 11 = .key file is invalid
- 12 = .key file could not be opened
- 13 = .vbs file was not found
- 14 = .vbs file is invalid
- 15 = .vbs file could not be opened
- 16 = .ecd file was not found
- 17 = .ecd file is invalid
- 18 = .ecd file could not be opened
- 19 = .udd file was not found
- 20 = .udd file is invalid
- 21 = .udd file could not be opened

24.5 Example Batch File Using Command Line Arguments

The example batch file in section 24.5.1 is a good template to use when creating your own batch file. The example illustrates how the various command line arguments and error codes can be used to along with specific .dcs files to drive functionality. With this example, an automated SWDL is first performed. If this is not performed successfully, then an error is displayed. If it is performed successfully, then certain DID values on the ECU also need to be updated. However, the values to update to are dependent upon the existing values. For example, if DID $1234 = a and DID $4567 = k, then they need to be updated to b and k respectively (this is done by copying a .dcs file with hot keys that write data b and k to each), and verified. However, if DID $1234 = c and DID $4567 = m, then they need to be updated to d and n respectively (by first copying a different .dcs file with hot keys that write data d and n), and verified.

24.5.1 Example Batch File

```
::Comment - Example file for how to use DET in command mode
::
::This example reflects the need of an ECU to first do an Automated SWDL.
::After performing the automated SWDL, the tool then needs to write new values to certain DIDs based upon existing values.
::  The tool first checks to see if DID 1234 = a and DID 4567 = k. If so, it updates the values in each to b and l.
::  If the first DID checks don't match, the tool then checks to see if DID 1234 = c and DID 4567 = m, and updates to d and n if so.
::
::The batch file loads a specific DCS file so that the DET will use the associated hot keys, DVS file, etc.
::
::Note that when checking error level (i.e., IF ERRORLEVEL X), this evaluates to true if the ERRORLEVEL is >= X.
::
@echo off
@echo.
@echo Running DET in command line mode
@echo.

:: Using -01 runs the DVS file (referenced in new DefaultSettings.dcs) to actually do the download
diagnosticengineeringtool –f DCSFileThatAutomaticallyUpdatesSoftware.dcs -01
IF ERRORLEVEL 1 GOTO PROBLEM

:: Then load file that runs a DVS that checks for Type 1 (DID 1234 = a, DID 4567 = k),
::    and has hotkeys to write desired replacement values (e.g., DID 1234 = b, DID 4567 = l)
::    and has VBS ACTION of SENDALLHOTKEYS
::    and has post SWDL verification of (e.g., DID 1234 = b, DID 4567 = l)
:: Using -01 runs the DVS file
diagnosticengineeringtool –f DCSFileThatDeterminesType1.dcs -01
IF ERRORLEVEL 1 GOTO CHECKTYPE2
:: It will go to CHECKTYPE2 if it did not find type 1, or if a problem writing
echo Successfully update to Type 1
GOTO END

:CHECKTYPE2
:: Then copy file that runs a DVS that checks for Type 2 (DID 1234 = c, DID 4567 = m),
::    and has hotkeys to write desired replacement values (e.g., DID 1234 = d, DID 4567 = n)
::    and has VBS ACTION of SENDALLHOTKEYS
::    and has post SWDL verification of (e.g., DID 1234 = d, DID 4567 = n)
IF ERRORLEVEL 1 GOTO FILECOPYPROBLEM
:: Using -01 runs the DVS file
```

```
diagnosticengineeringtool –f DCSFileThatDeterminesType2.dcs -01
IF ERRORLEVEL 1 GOTO UNKNOWNTYPE
echo Successfully update to Type 2
GOTO END

:: This is reached if not Type 1 or Type 2 match
:UNKNOWNTYPE
echo Was not type 1 or type 2
GOTO END

:FILECOPYPROBLEM
echo Error copying file
GOTO END

:PROBLEM
IF ERRORLEVEL 9 GOTO UMSFAIL
IF ERRORLEVEL 8 GOTO DCSNOTFOUND
IF ERRORLEVEL 7 GOTO ECDFAIL
IF ERRORLEVEL 6 GOTO BUSQUERYFAIL
IF ERRORLEVEL 5 GOTO HOTKEYFAIL
IF ERRORLEVEL 4 GOTO VBSFAIL
IF ERRORLEVEL 3 GOTO DVSFAIL
IF ERRORLEVEL 2 GOTO INVPARAM
IF ERRORLEVEL 1 GOTO INVFORMAT

:UMSFAIL
echo User Message sequence Transmittal Failed
GOTO END

:DCSNOTFOUND
echo DCS File Not Found
GOTO END

:ECDFAIL
echo ECD Transmittal Failed
GOTO END

:BUSQUERYFAIL
echo Bus Query Transmittal Failed
GOTO END

:HOTKEYFAIL
echo Hot Key Transmittal Failed
GOTO END

:VBSFAIL
echo VBS Failed
GOTO END

:DVSFAIL
echo DVS Failed
GOTO END

:INVPARAM
echo Invalid Parameters
```

```
GOTO END

:INVFORMAT
echo Invalid format
GOTO END

:END
pause
```

Annex A - File Types Associated with DET

The Diagnostic Engineering Tool interfaces with a wide variety of different files. Many of the files used by the DET are XML files but purposely use different extension to make it easier for the open file dialog, etc. to filter on the correct ones. A summary of the various file types and how they are used with the DET are summarized in this Annex.

A.1 DCS Files

This is an XML file that is used exclusively with the DET that stores almost all of the user personal settings in the DET (e.g., those set under the Tools → Options form) as well as other information such as hot keys, user messages, quick connections, associated MDX files, VBF sequences, ECU Configuration Sequences, etc. This allows individual users to easily customize the DET to their own preferences and keep this configuration at start-up. Alternatively, it allows a user to save their setup so that another user can load the same setup (or a portion of the setup such as only the hot key messages).

A.2 KEY Files

This is an XML file that is used exclusively with the DET that stores the fixed byte security level information (either encrypted or unencrypted) for 1 to n ECUs.

A.3 VBF Files

VBF files are the required Ford format for official storing programmable ECU data (for ISO 14229 ECUs) that is released with a WERS software part number. This is a binary file format that is detailed in the Versatile File Format specification. This format officially encapsulates ECU's secondary bootloaders, strategy files, and calibration files.

A.4 VBS Files

This is an XML file that is used exclusively with the DET that stores 1 to n named sequences of VBF files to download. Each sequence within the VBS file can consist of 1 to m different VBF files.

A.5 ECD Files

This is an XML file that is used exclusively with the DET that stores 1 to n named sequences of DIDs to read or write. Each sequence within the ECD file can consist of 1 to m different DIDs for 1 to p ECUs.

A.6 GDX Files

This is an XML file that is used with numerous engineering tools and conformance tools within product development, but also within service and vehicle operations. This file contains a snapshot of the diagnostic GMRDB. In other words, it contains the Ford main list of ECU IDs, DTCs, DIDs, and routines. There is an XML DTD that describes the structure.

A.7 MDX Files

This is an XML file that is used with numerous engineering tools and conformance tools within product development, but also within service and vehicle operations. This file is the official format of the Diagnostic Part 2 Specification for an ECU and therefore captures the details for all supported diagnostic services, data content, etc. for a particular ECU. A separate detailed specification (MDX Specification) describes the details for this. There are two MDX versions current (002 and 003) and an XML schema for each.

A.8 DVS Files

This is an XML file that is used exclusively with the DET and contains information used to drive the automated software download. The DIDs to read, the data to compare, and the files to download are completely specified by the creator of the DVS file. Refer to section 9 for more details on the automated software download and DVS examples and refer to Annex B for the DVS schema.

A.9 RTF Files

Most of the information displayed to the user is saved in this Rich Text Format in order to preserve the colored text and be capable of being opened in a wide variety of software programs.

A.10 TXT Files

The running log files used with the tool (see section 11.11) are saved in this format which is ASCII text.

A.11 ASC Files

This is an ASCII text format that files saved from the Generic Bus Monitor are saved in. The format of the file is compatible with Vector CANalyzer so that traffic can be imported and replayed using that tool.

A.12 INI Files

This is the standard Windows ASCII INI format, and is only used when loading a legacy Central Car Configuration INI file used when decoding DID 0xF106 (see section 12.2.10 for more details).

Annex B - DVS Schema

B.1 DVS Schema for DVS File Validation

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!--
**********************************************************************************************
**********************************************************************************************
**          FORD MOTOR COMPANY CONFIDENTIAL AND PROPRIETARY
**
** This document contains Ford Motor Company confidential and proprietary information.
** Disclosure of the information contained in any portion of this document is not
** permitted without the expressed, written consent of a duly authorized representative
** of Ford Motor Company, Dearborn, Michigan, USA
**
**********************************************************************************************
**********************************************************************************************

**********************************************************************************************
**********************************************************************************************
**
**  Schema / Content Author:   Jason Miller
**  Version:                2.0
**  Date:                   31-July-2012
**   - Added XML elements of ALWAYS_ANALYZE_ALL and VERIFY_NOMATCH
**
**********************************************************************************************
**********************************************************************************************
-->

  <xsd:element name="DVS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ALWAYS_ANALYZE_ALL" type="CONSTRAINT_TRUE_FALSE" minOccurs="0"
maxOccurs="1"/>
        <xsd:element name="SEQUENCE" type="ANY_SEQUENCE" minOccurs = "0"
maxOccurs="unbounded"/>
        <xsd:element name="DEFAULT_SEQUENCE" type="ANY_SEQUENCE" minOccurs = "0"
maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="VERSION" type="xsd:string" use="required" fixed="2.0" />
    </xsd:complexType>
  </xsd:element>



  <!-- *** Begin Core Elements *** -->

   <xsd:complexType name="ANY_SEQUENCE">
      <xsd:sequence>
      <xsd:element name="NAME" type="xsd:string" />
```

```
      <xsd:element ref="PRE_SWDL" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="ACTION" type="CONSTRAINT_ACTION" />
      <xsd:element name="VBS_INDEX" type="CONSTRAINT_1_255" minOccurs="0"/>
      <xsd:element ref="POST_SWDL" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="FINAL_TEXT_SUCCESS" type="xsd:string" minOccurs="0"/>
      <xsd:element name="FINAL_TEXT_FAIL" type="xsd:string" minOccurs="0"/>
      <xsd:element name="POPUP_SUCCESS" type="CONSTRAINT_TRUE_FALSE" minOccurs="0"/>
      <xsd:element name="POPUP_FAIL" type="CONSTRAINT_TRUE_FALSE" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>

  <xsd:element name="PRE_SWDL">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="DID_DATA" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="POST_SWDL">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="DID_DATA" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DID_DATA">
    <xsd:complexType>
      <xsd:sequence>
      <xsd:element name="ID" type="CONSTRAINT_FFFF" />
      <xsd:element name="DATA_LENGTH" type="CONSTRAINT_4095" />
      <xsd:element name="DATA" type="RAW_HEX_4095" />
      <xsd:element name="MASK" type="RAW_HEX_4095" minOccurs="0" />
      <xsd:element name="ECU_ID" type="CONSTRAINT_7FF" />
    <xsd:element name="VERIFY_NOMATCH" type="CONSTRAINT_TRUE_FALSE" minOccurs="0" maxOccurs="1"
/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>


  <!-- RAW_HEX_4095: new datatype
       1) Constraints to only 8190 hex characters
  -->
  <xsd:simpleType name="RAW_HEX_4095">
    <xsd:union>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="[0-9a-fA-F]{1,8190}" />
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>
```

```
  <!-- CONSTRAINT_TRUE_FALSE: new datatype
       1) Constraints element values to {true, false}, case insensitive
   -->
  <xsd:simpleType name="CONSTRAINT_TRUE_FALSE">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[Tt][Rr][Uu][Ee]"/>
      <xsd:pattern value="[Ff][Aa][Ll][Ss][Ee]"/>
      <xsd:whiteSpace value="collapse"/>
    </xsd:restriction>
  </xsd:simpleType>


  <!-- CONSTRAINT_ACTION: new datatype
       1) Constraints element values to {SWDL, None, ForcedFail, SendAllHotKeys
                                SWDLAndSendAllHotKeys}, case insensitive
   -->
  <xsd:simpleType name="CONSTRAINT_ACTION">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[Ss][Ww][Dd][Ll]"/>
      <xsd:pattern value="[Nn][Oo][Nn][Ee]"/>
      <xsd:pattern value="[Ff][Oo][Rr][Cc][Ee][Dd][Ff][Aa][Ii][Ll]"/>
      <xsd:pattern value="[Ss][Ee][Nn][Dd][Aa][Ll][Ll][Hh][Oo][Tt][Kk][Ee][Yy][Ss]"/>
      <xsd:pattern
value="[Ss][Ww][Dd][Ll][Aa][Nn][Dd][Ss][Ee][Nn][Dd][Aa][Ll][Ll][Hh][Oo][Tt][Kk][Ee][Yy][Ss]"/>
      <xsd:whiteSpace value="collapse"/>
    </xsd:restriction>
  </xsd:simpleType>


  <!-- CONSTRAINT_4095: new datatype
       1) Constraints integer values to [0..4095]
       2) Constraints hexadecimal vlues to [0x0..0xFFF]
   -->
  <xsd:simpleType name="CONSTRAINT_4095">
    <xsd:union>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="(\+)?[0][xX]0*[0-9a-fA-F]{1,3}" />
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:restriction base="xsd:nonNegativeInteger">
          <xsd:maxInclusive value="4095"/>
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>


  <!-- CONSTRAINT_7FF: new datatype
       1) Constraints integer values to [0..4095]
       2) Constraints hexadecimal vlues to [0x0..0xFFF]
   -->
  <xsd:simpleType name="CONSTRAINT_7FF">
    <xsd:union>
```

```
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="(\+)?[0][xX]0*[0-9a-fA-F]{1,3}" />
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:restriction base="xsd:nonNegativeInteger">
          <xsd:maxInclusive value="2047"/>
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>


  <!-- CONSTRAINT_FFFF: new datatype
       1) Constraints integer values to [0..65535]
       2) Constraints hexadecimal vlues to [0x0..0xFFFF]
  -->
  <xsd:simpleType name="CONSTRAINT_FFFF">
    <xsd:union>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="(\+)?[0][xX]0*[0-9a-fA-F]{1,4}" />
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:restriction base="xsd:nonNegativeInteger">
          <xsd:maxInclusive value="65535"/>
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>


  <!-- CONSTRAINT_1_255: new datatype
       1) Constraints integer values to [1..255]
       2) Constraints hexadecimal vlues to [0x1..0xFF]
  -->
  <xsd:simpleType name="CONSTRAINT_1_255">
    <xsd:union>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="(\+)?[0][xX]0*[1-9a-fA-F][0-9a-fA-F]?" />
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:restriction base="xsd:positiveInteger">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="255"/>
          <xsd:whiteSpace value="collapse" />
        </xsd:restriction>
      </xsd:simpleType>
```

```
    </xsd:union>
  </xsd:simpleType>

</xsd:schema>
```

Annex C – Network Hardware Support with DET

There are currently 3 Application Programming Interfaces (APIs) supported by the Diagnostic Engineering Tool. Those are:

- Vector XL Hardware (to be used for any of the Vector XL hardware such as CANcardXL)
  - Note: This uses the vxlapi.dll interface to the hardware.
- Vector CANcardX (to be used for any of the pre Vector XL hardware such as CANcardX)
  - Note: This uses the vcand32.dll interface to the hardware.
- SAE J2534 04.04 API
  - Note: The DET will scan the Windows Registry at HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\PassThruSupport.04.04 for all installed SAE J2534 04.04 hardware devices and will dynamically populate the list of available hardware based upon the SAE J2534 04.04 hardware devices installed on the computer. If your particular device is not appearing, then it is not correctly installed on the laptop as a registered SAE J2534 device, or is not a Windows 32 bit J2534 04.04 device J2534 04.04 does not currently support 64 bit Windows, and DET will not be able to detect 64 bit devices.

Note that all network hardware should always be installed based upon the device drivers and instructions provided by the hardware's manufacturer.

C.1 Tested Hardware

Any Vector hardware which supports the Vector DLLs mentioned above should work with the DET. In addition, any compliant SAE J2534 hardware should also work without issue with the DET. The hardware needs to be installed per the hardware manufacturer's directions. However, it is known that certain versions of various manufacturers' SAE J2534 DLLs have had issues in the past. It is recommended to always use the latest manufacturer's DLLs. This may be necessary in order for certain functionality to be supported through the DET (e.g., generic support of all CAN channels). The following sections describe the hardware devices that have been tested locally with the DET along with support information for any issues. Contact the hardware manufacturer directly for hardware suppliers not on this list. Section A.2 contains further details on programming time comparisons as well as the manufacturer DLL versions used for testing.

C.1.1 Vector Hardware

The Vector hardware DLLs are included with the default normal install of the tool. The hardware only needs to be installed on the computer per the hardware manufacturer's directions.

For support issues installing or using this hardware, please refer to the following:

- Company: Vector CANtech
- Website: http://www.vector.com/vi_downloadcenter_en.html
- Phone: (248)449-9290 Option 2
- Email: mailto:support@us.vector.com
- Known hardware supported: CANcardX, CANcardXL, CANcardXLe, CANCaseXL, VN1600 Family

C.1.2 Intrepid Hardware

Intrepid hardware is supported both through the SAE J2534 API as well as through the Vector API. It is recommended to use the Vector API as this provides more functionality and flexibility. This requires that the DLL swap procedure described in section 12.3.1 of the DET user manual is performed. For issues swapping the DLL, contact Intrepid.

For support issues installing or using this hardware, please refer to the following:

- Company: Intrepid Control Systems
- Website: http://intrepidcs.com/
- Direct Link to Drivers:
  - https://cdn.intrepidcs.net/updates/files/RP1210KitInstall.zip
- Phone: (586)731-7950 x713
- Email: mailto:moreinfo@intrepidcs.com
- Known hardware supported: ValueCAN4, NeoVI Red, NeoVI Fire
- ValueCAN3 has reached the end of life. Upgrade your hardware.

C.1.3 Movimento Hardware

Movimento hardware is supported only through the SAE J2534 API,and supports the ability to connect to multiple CAN channels.

For support issues installing or using this hardware, please refer to the following:

- Company: Movimento
- Website: http://www.movimentogroup.com/
- Phone: (734)272-4590
- Email: mailto:vodu@movimentogroup.com
- Known hardware supported: Puma2, Puma PT

C.1.4 EEPod Hardware

EEPod hardware is supported only through the SAE J2534 API, and supports the ability to connect to multiple CAN channels.

For support issues installing or using this hardware, please refer to the following:

- Company: EEPod LLC
- Website: http://www.eepod.com/
- Specific Driver / J2534 Install help: http://www.eepod.com/page7.html
  - Be sure to install USB drivers **AND** run the mcs1install.exe or mycanicinstall.exe
- Phone: (480)288-4905
- Email: mailto:support@eepod.com
- Known hardware supported: MyCANic, MCS1

C.1.5 Kvaser Hardware

Kvaser hardware such as LapCAN is supported both through the SAE J2534 API and also through the Vector API.

For support issues installing Kvaser hardware, please refer to the following:

- Company: Kvaser
- Website: http://www.kvaser.com/

- Specific Driver / J2534 Install help: http://www.kvaser.com/en/support/faq/274-installing-driver-with-j2534-support.html
- Phone: (866)317-2455
- Email: mailto:support@kvaser.com
- Known hardware supported: LAPcan, Leaf Light, Leaf SemiPro, Leaf Professional, USBcan Professional, USBcan II

Driver Installation Procedure:
1. Download the latest drivers from www.kvaser.com.
2. Make sure you have admin privileges on the computer before attempting to run the driver installation.
3. Run the driver installation package
4. Check the optional J2534 support and press Next.
5. Follow remaining dialogs until driver installation is complete.
6. Attach Kvaser hardware to the PC and complete the found new hardware wizard.

Note that some users have had some issues due to incomplete or incorrect installation of the Kvaser SAE J2534 driver, related to the paths in the Windows registry. The symptom of this can be failure of the Kvaser J2534 device to appear in the hardware list or closure of the DET when the Kvaser SAE J2534 hardware is selected through the Initialize Hardware form. Kvaser has provided the following steps to address this:

```
1. Open a Command Prompt.

2. In the command prompt, change your current directory to directory containing
the J2534 support files.  A command like:
   cd \Program Files\Kvaser\Canlib\Bin\j2534

3. Once you confirm you are in the correct directory location, perform the
following command in the command prompt:
   j2534_RegUpdate -samedir
```

C.1.6 Dearborn Group (dba DG Technologies)

Dearborn Group Technology hardware is supported only through the SAE J2534 API, and supports the ability to connect to multiple CAN channels.

For support issues installing or using this hardware, please refer to the following:
- Company: Dearborn Group
- Website: http://www.dgtech.com/
- Phone: (248)888-2000
- Specific Driver and User Manual Download: http://www.dgtech.com/product/d-briDGe/d-briDGe.php
- Email: mailto:techsupp@dgtech.com
- Known hardware supported: Soft Bridge, d-briDGe

C.1.7 Controller Technologies Hardware

Controller Technologies hardware is supported only through the SAE J2534 API

For support issues installing or using this hardware, please refer to the following:
- Company: Controller Technologies Corp.
- Website: http://www.networkvehicle.com/
- Phone: (586)532-8400
- Email: sales@networkvehicle.com
- Known hardware supported: U-CAN 2

C.1.8 Other SAE J2534 Hardware

Any compliant SAE J2534 hardware should work with the DET. If the hardware you are using is not on this list, it is because it has likely not been reported as tested and verified to work. Please report any SAE J2534 hardware that you are using on a regular basis without issue.

C.2 Programming Times and DLL Versions

As a comparison of programming times, the following available hardware devices were tested by programming a simulated ECU at a bench. Two files were programmed on 500k CAN including an approximately 14k bootloader and a 1MB application using Diagnostic Engineering Tool v5.10.3 on Windows 7 (unless otherwise noted). The files were programmed 8 times with each piece of hardware by changing parameters in the simulated ECU and the DET. The varied parameters included a STmin of 0 versus an STmin of 1, and also a maxNumberOfBlockLength of 512 versus a maxNumberOfBlockLength of 4095. Each of these scenarios were performed with the DET in normal download mode and fast download mode (see section 11.4.1). The results of each programming time along with the hardware vendor's DLL versions used are listed in the table below.

Tool Supplier	Hardware	API	DET D/L Method	STmin = 0, MaxBL = 4095 Time (s)	STmin = 0, MaxBL = 512 Time (s)	STmin = 1, MaxBL = 4095 Time (s)	STmin = 1, MaxBL = 512 Time (s)	DLL Info
Dearborn Group	Soft Bridge	J2534	Normal	321.7	341.43	473.5	492.24	SoftBridge.dll, Firmware Version= 3.106, DLL Version = 1.00.23
			Fast	140.22	159.79	212.81	219.05	
Dearborn Group	d_briDGe	J2534	Normal	155.7	212.5	318.12	334.72	dbriDGe.dll, Firmware Version= 3.109 , DLL Version = 1.00.1.9. Date: 4/7/2014, Size 179 KB
			Fast	132.47	145.47	164.05	176.87	
EEPOD	MYCANIC	J2534	Normal	2567.36			2913.38	McS2v232.dll, Date = 8/22/2012, Size = 287 kb, Zip File Version = MyCANICv228. Firmware & Dll Version = 02.28
			Fast	78.86	246.47	220.87	384.13	
	McS1	J2534	Normal	2,480.76				McS1v232.dll, Date = 3/15/2012, Size = 273 kb, Zip File Version McS1v234. Firmware & Dll Version = 02.34.
			Fast	2,631.55				
Movimento	Puma 2	J2534	Normal	109.11	125.28	273.84	289.37	Mov04P32.dll, Firmware Version:2.4.9.4, DLL Version 1.3.5.2, Driver Version: 1.6.21
			Fast	46.51	60.42	165.1	181.46	
Network Vehicle	UCAN2	J2534	Normal	168.52	183.74	324.14	337.64	UCAN2 A Dll = C:\Program Files (x86)\Controller Technologies Corp\UCAN2\ctcuc232.dll Firmware Version = 1.0 DLL Version = 1.1 API Version = 04.04
Intrepid	ValueCAN3	vxlapi	Normal	477.77	494.58	476.73	496.09	vxlapi.dll File Version: 3.5.1.47, 212k,

Tool Supplier	Hardware	API	DET D/L Method	STmin = 0, MaxBL = 4095 Time (s)	STmin = 0, MaxBL = 512 Time (s)	STmin = 1, MaxBL = 4095 Time (s)	STmin = 1, MaxBL = 512 Time (s)	DLL Info
Control Systems			Fast	47.35	74.66	165.2	183.96	4/18/2011
	ValueCAN3	J2534	Normal	474.88	491.72	473.28	490.92	Intrepid ValuCAN3(J2534): icsJ2534VCAN3.dll, File Version 3.503.1.0, 80k, 3/14/2012
			Fast	46.89	71.18	166.23	184.56	
	NeoVI Fire	vxlapi	Normal	245.19	259.48	382.33	400.26	vxlapi.dll File Version: 3.5.1.47, 212k, 4/18/2011
			Fast	46	68.51	165.2	180.95	
	NeoVI Fire	J2534	Normal	163.92	180.95	316.52	329.87	Interpid neoVI Fire(J2534): icsJ2534Fire.dll, File Version 3.503.1.0, 80k, 3/14/2012. Firmware Version: 3 App 04.15
			Fast	45.84	63.85	166.68	181.27	
Kvaser	Kvaser LapCAN	J2534	Normal	110.33	131.01	237.36	258.02	kvj2534.dll, File Version:7.8.8331, 349k, 5/23/2012
			Fast	105.24	128.3	165.21	181.12	
	Kvaser USBCan II (HS/HS)	J2534	Normal	109.56	131.05	240.02	255.24	kvj2534.dll, File Version: 7.9.8500, Product Version 4.9, 349k, 9/6/2012. Dll Version: 7.09.8500
			Fast	105.8	125.61	165.28	180.91	
	Kvaser USBCan II (HS/HS)	J2534	Normal	166.58	184.11	325.64	336.26	kvJ2534.dllDll Version: 8.00.111(BETA), 359k, 2/27/2013
			Fast	49.78	69.14	164.96	180.76	
Vector	CANCardXL (Windows XP)	vxlapi	Normal	93.34	123.2	245.88	265.95	vxlapi.dll, File version = 7.4.50.0, Size = 174 kb, Date = 7/14/2010
			Fast	54.39	80.41	174.59	195.41	
	CANCaseXL (Windows XP)	vxlapi	Normal	127.33	155.02	280.46	299.17	
			Fast	52.18	81.68	178.7	200.83	
	CANCaseXL	vxlapi	Normal	99.26	117.62	254.8	270.29	
			Fast	50.38	69.2	166.66	181.21	
	VN1610	vxlapi	Normal	84.01	101.61	238.5	255.79	
			Fast	45.58	59.01	165.34	181	
Bosch	VCM – II	J2534	Normal	364.95	376.91	355.62	382.29	BVTX4J32.dll, Firmware Version= 2.1.119, DLL Version: 2.1.119.3, Size 811 kb, Date 2/13/2014
				340.07	341.43	336.83	355.34	

Annex D - Frequently Asked Questions

This section is intended to contain a list of frequently asked questions by users of the Diagnostic Engineering Tool. If there are questions and answers you feel would be useful to be added, please contact the author.

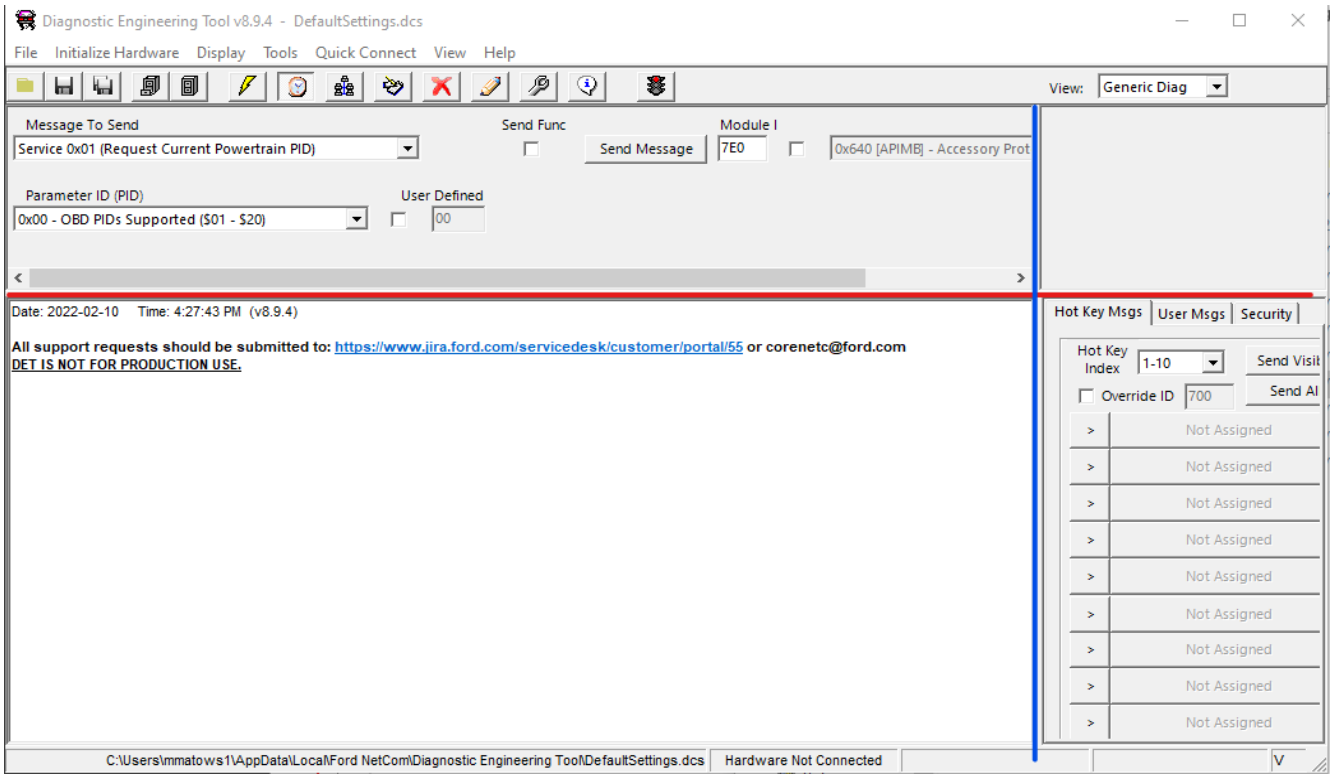
D.1 Where Do I Get .NET Framework 4?

.NET Framework 4 is a Microsoft provided software framework which is intended to be used by most new applications created for the Windows platform. During the install of the DET, a direct link is provided. This link is duplicated here: http://www.microsoft.com/net/download

D.2 Why Can't I See the Hot Key Messages, User Messages, or Security Tab?

Depending on the display settings of the computer, the control in the bottom right of the application consisting of Hot Key Messages, User Messages, and Security information may not be visible. To address this, one needs to change the DPI settings under the Display control within the Control Panel. Set the DPI setting to "Normal Size (96 DPI)". This normally addresses the display issue on any computer.

In the picture below, the edges of the windows may be adjusted at the red and blue lines. Note that the lines are not actually red and blue in DET. Hover your mouse over the border, and the mouse icon will change to an up/down arrow or to a left/right arrow. Click and you can move the edge and resize as needed.



D.3 What Does the Error About Encrypted Security Bytes Mean?

Depending on the version of DET you are using, when attempting to perform security access and/or enter encrypted security bytes you may see error text of either "Encrypted bytes are secured with a level that is not authorized by your specific instance of the installed tool." OR "The encrypted bytes are secured with

production encryption, which is not authorized by this tool. Contact ECU supplier or Ford D&R for unencrypted fixed bytes or with non-production encryption."

The ECU specific fixed bytes used as an input to the Ford security algorithms may be encrypted with one of 3 encryption levels. Two of these levels are intended for engineering usage and are supported by the DET. Note that the DET provides the ability to store the user entered unencrypted fixed bytes in this encrypted format. The other level is intended for production usage and is only supported in the DET on a very limited, user specific basis. If you are seeing the previous error message, this means the encrypted fixed bytes have been encrypted using the production level encryption. You will either need to obtain the unencrypted fixed bytes or the fixed bytes encrypted with the non-production level from the D&R or ECU supplier.

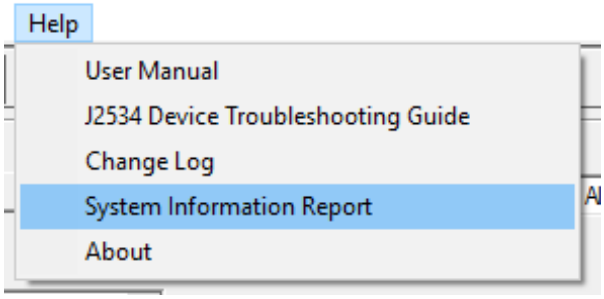
D.4 What Network Hardware Is Supported By the DET?

See Annex C for details and support information on the network interface hardware.

D.5 Why Isn't My Hardware Being Detected or Working Correctly?

The device drivers and SAE J2534 DLLs may not be installed correctly according to the hardware manufacturer's directions. Contact the hardware manufacturer directly for more information and/or assistance. Refer to Annex C for details and support information on the tested network interface hardware. If using Intrepid hardware with the Vector API, in addition to the device driver installation, be sure the DLL swap procedure was performed correctly (see section 12.3.1).

A J2534 Device Troubleshooting Guide is also available under the Help Menu. If contacting Net Com for support, please also send the System Information Report from the Help Menu.



D.6 How Do I Get My Personalized Unlock Key?

The DET will walk each user through the process of how to obtain a personalized unlock key once the program is installed and executed. The process essentially consists of emailing an encrypted string to a set of people at Ford who will then process this string and email back the personalized unlock key. If Outlook is used on the computer, then this process can be automated. Otherwise, the encrypted string displayed must be copied and pasted, and then emailed to the contacts displayed by the tool. When submitting a personalized code or entering the Ford provided unlock key, it is critical that the **entire** string is copied and pasted. Screen shots are not acceptable and hand-typed requests often result in errors (e.g., capital i's, lower case L's, and number 1's all look alike).

D.7 Does the Unlock Key Expire and Is It Version Specific?

The unlock key is personalized for each user and computer combination. This personalized unlock key does not expire and is not version specific. Therefore, new versions of the DET can be installed with no impact. However, each DET version is set up to expire and no longer run one year after the release date. Therefore, it is important to always use the latest version. Note that attempts to roll back the system date will prevent the DET (both previous and future versions) from running on the computer, and will necessitate the development of a specialized application by Ford to allow the DET to run again.

D.8 Where Is the Unlock Key Stored?

Once entered, the unlock key information is stored in a file called "Engineering Unlock Key.xml". The file is located within the special folder designated as the "My Documents" or "Documents" folder by the Windows operating system. Under this folder, a sub folder of "Ford NetCom" is created, with a sub folder of "Diagnostic Engineering Tool" under that. The "Engineering Unlock Key.xml" is stored here. When the DET is launched, it will first look for the XML key file within this folder. If the file is not found, the DET will look in the directory from where the DET application itself is being executed from. Manually copying the unlock key into this folder may be necessary for scenarios where the Windows designated "My Documents" folder is stored on a network drive and the user is working offline.

D.9 How do I use DET with S19 and hex files?

S19 and hex files are not supported by DET. NetCom does not have any access to them.