Content

1 Introduction

1.1 Preface

Dear DiagRA® S users,

With this manual we want to help the beginner to get to grips with DiagRA® S. For more experienced users, we hope it will provide a detailed reference that will enable you to get even more from the product.

To make the text easy to read, items in menus and submenus in this document are indicated using an arrow. **File** → **New** would mean: choose *New* under the *File* menu. In contrast user keystrokes and actions are shown in bold. So **Ctrl+A** means: Press and hold down the **Ctrl** key and press **A**. Correspondingly ↑, ↓, ← and → stand for the arrow keys **Shift** for the shift key, **Tab** for Tabulation key and **Enter** for the Enter or Carriage-Return Key.

We hope that this manual can enhance your daily experience with DiagRA® S. For further comments and questions we are, of course, at your disposal.

RA Consulting GmbH
Zeiloch 6a
D-76646 Bruchsal

Contact & Product information	
Phone:	+49 7251 9819-500
Fax:	+49 7251 3266882
E-mail:	<info@rac.de>
Web:	https://www.rac.de
Technical support	
Phone:	+49 7251 9819-415
E-mail:	<support@rac.de>
Crash reports	
E-mail:	<bugreport.diagra_s@rac.de>

1.2 Disclaimer

Warning! The application of this product can be dangerous. Therefore, please note the following hints.

With this software it is possible for you to influence or control a networked electronic system. Your actions can therefore lead to severe personal damages or damages to property.

For this reason, only persons, which have understood the possible consequences of their actions with this software or persons, which have been specially trained for the use of this software, may use this software. In case other persons use this software, RA Consulting GmbH hereby expressly gives notice that the warranty shall be limited to the correction of defects, and RA Consulting GmbH hereby expressly disclaims any liability over and above the refunding of the price paid for this software.

Should you integrate this product into a higher-level system by using the product´s automation interface, please note that the warranty of RA Consulting GmbH shall be limited to the correction of defects, and RA Consulting GmbH hereby expressly disclaims any liability over and above the refunding of the price paid for this software, since RA does not have any influence on the implementations of the higher-level system, which may be defective.

Furthermore, RA Consulting shall in no event be held liable for any user-defined scripts, regardless of whether or not you turn off this disclaimer when using this product in the batch mode.

Should you not agree with the above named limitations, please send the software back to us at our costs within one (1) month after delivery. We will then immediately refund the price you have paid.

1.3 Prerequisites to install DiagRA® S

DiagRA® S runs on the following operating systems: Microsoft Windows Windows 7 or higher.

DiagRA® S Basic Version (DiagRA_S.exe) is a 32-bit application.
The main tool for managing simulation files in DiagRA® S Professional Version (DataManager.exe) can run as both 32-bit and 64-bit application (64-bit version is preferred).
NOTE. Independently of the fact that DiagRA_S.exe is a 32-bit application, the simulation can be used from delivered 64-bit PassThru and D-PDU API libraries as well.

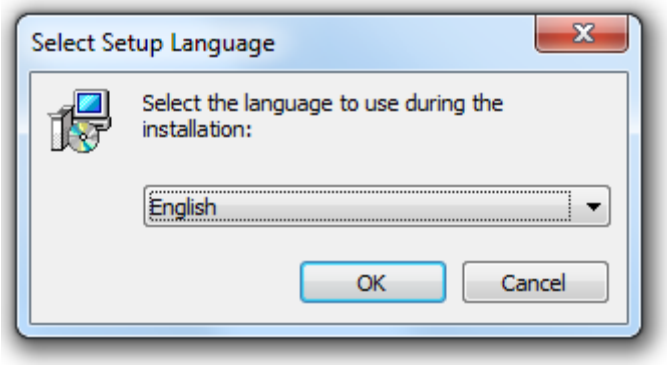
For installation of DiagRA® S you need rights to read and write to the Windows Registry, the folder in which you have installed the application and to the Windows folder (for installation of RP1210 DLL).

DiagRA® S requires the installation of Microsoft's .NET Framework 4.6.1+ and C++ Redistributable Runtime (2022). These packages will be installed during the DiagRA® S installation if they are missing on your computer. An active internet connection is required to install them.

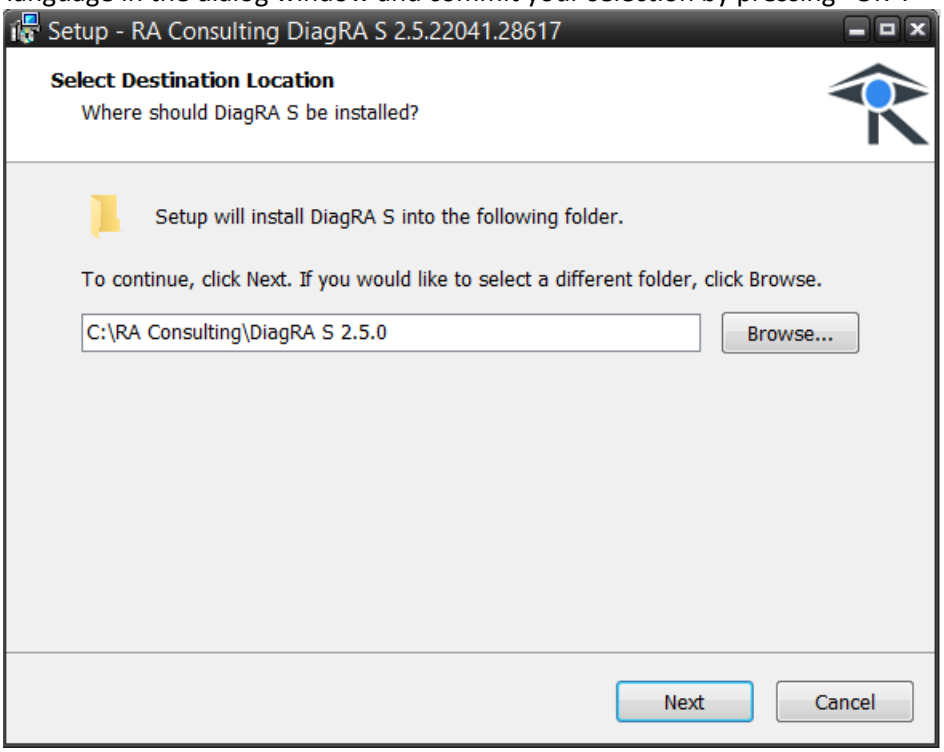
If you don't have rights which are required to install or run DiagRA® S, please contact your local system administrator.

1.4 Installation

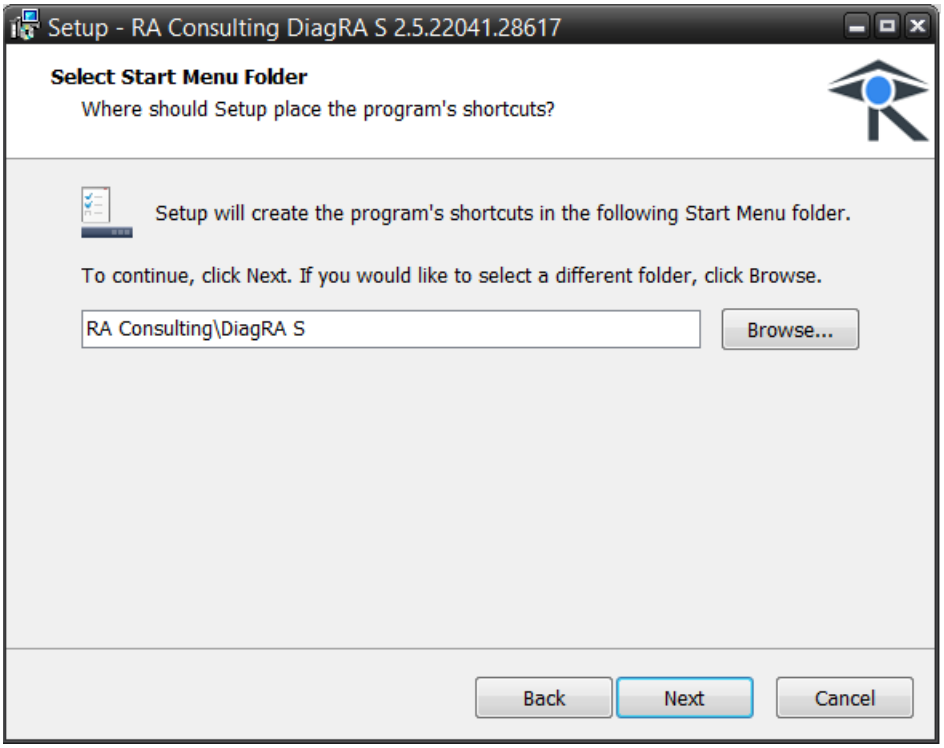
Run the delivered application “DiagRA S XXXX Setup.exe“ („XXXX“ is the version number of your setup, e.g. „3.1.0.34“) to install DiagRA® S.



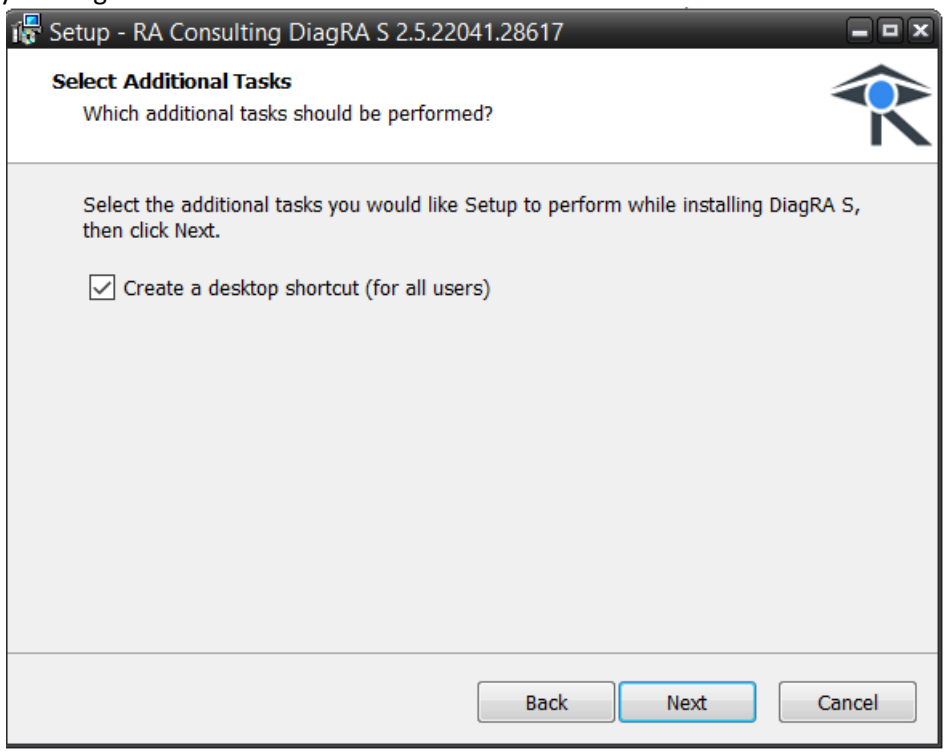
Select your language in the dialog window and commit your selection by pressing “OK”.



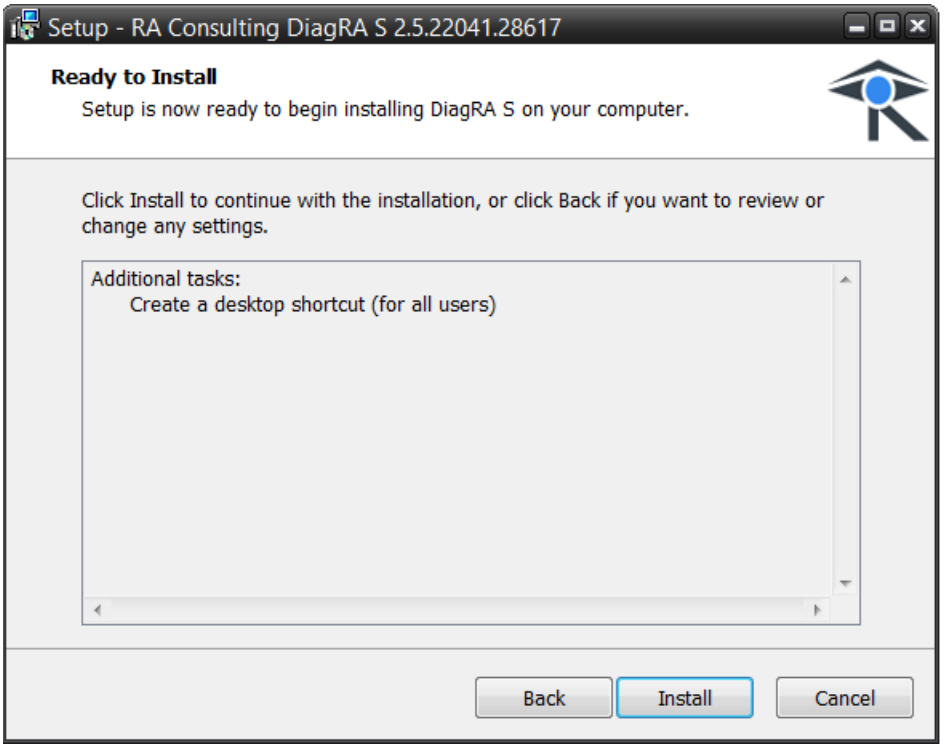
Choose the directory, in which you want to install DiagRA® S. It is recommended to use default value (*C:\Program Files (x86)\RA Consulting\DiagRA S*). Confirm the installation path by clicking on the button “*Next*“. Confirm the following dialog window by clicking on “*Next*“, too.



Select the folder of your start menu in which you want to put the link to DiagRA® S. Commit your selection by clicking on "*Next* ".



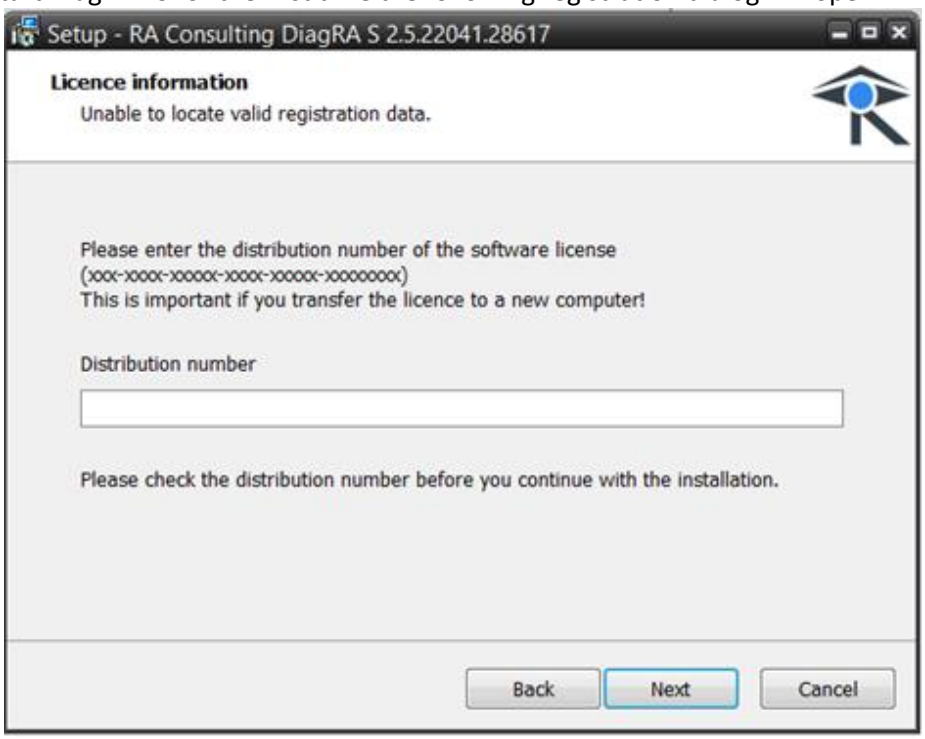
Now, you have the possibility to add a link to DiagRA® S to your desktop and to your task bar. After you made your selection, please click on "*Next*" to continue with the installation.



Finally, you will get a summary of your installation. If one or more options are wrong, feel free to use the button "*Back*" to go back to one of the selections to edit your settings. Confirm your final settings by clicking on "*Install*". You need to input your delivery number and after that, you will be able to install and use DiagRA® S.

1.5 Registration

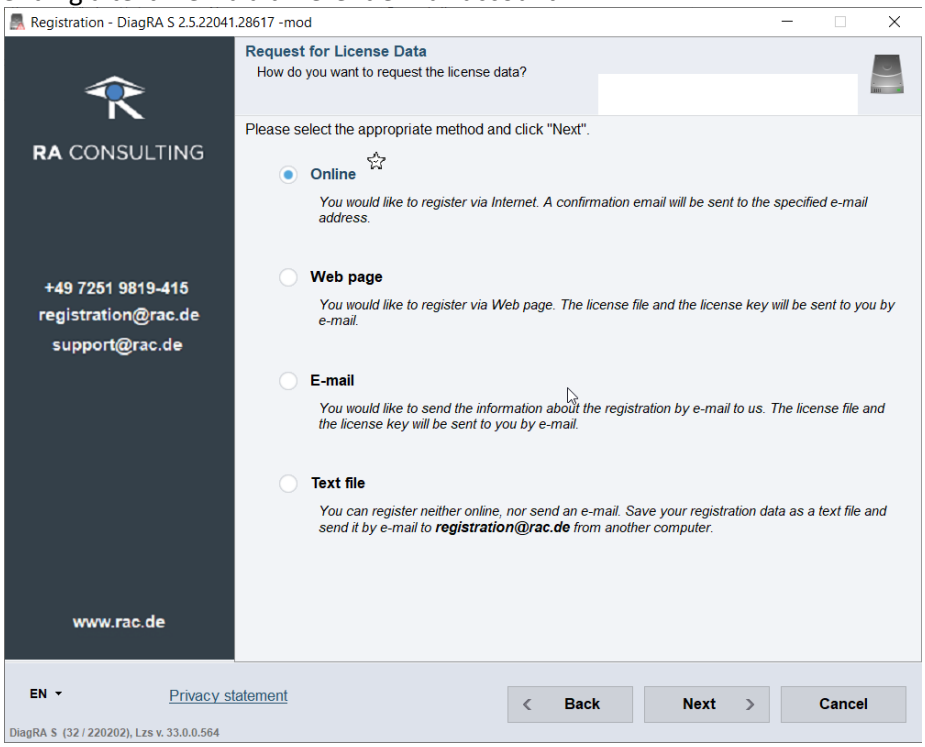
When you start DiagRA® S for the first time the following registration dialog will open:



You must enter the distribution number you've received from RA Consulting.

In the next step you can request your *activation (license) data*. You have the following options for this:

- Online via your web-browser (direct internet connection required)
- Website - by sending registration data via webformular
- By sending an e-mail directly from the registration wizard
- By sending a text file via a different e-mail account



We recommend using the Online Registration, because it's the simplest and fastest method to register your DiagRA® S. After selecting the proper registration method follow the instructions step-by-step to register the application. After the registration has been completed you can communicate with DiagRA® S using its simulated interfaces.

After requesting the activation data you will receive an e-mail from us, which contains

- the corresponding license key.
- In case you use online registration your license data file DiagRA_S.bbb will be copied directly to the corresponding folder. Otherwise, the license data file is included in the e-mail.

Please copy the license file into the folder C:\ProgramData\RA Consulting\DiagRA S\. Afterwards restart DiagRA® S and the registration wizard will show up again. After selecting the menu item Finish registration, your personal information will be shown. Now you have to enter your Company Number and the License Key from the e-mail. After successful completion of the registration you can use DiagRA® S without restrictions.

.

2 Basics

This chapter contains an overview about basic capabilities of DiagRA® S and some of the concepts, that lay behind it.

2.1 Use-cases

DiagRA® S is a software-based simulator for diagnostic communication. Its primary use-case is to simulate ECU / Vehicle behavior for diagnostic communication based on widely used diagnostic standards like:

- UDS (ISO 14229) and any derivatives like WWH-OBD (ISO 27145) and OBDonUDS, ZEVonUDS (SAE J1979-2/3)
- OBD (SAE J1979-1)
- KWP2000 (ISO 14230-3)
- and SAE J1939.

DiagRA® S supports typical transport networks for diagnostic data like

- CAN / CANFD (ISO 11898) as well as corresponding transport protocols like ISO-TP (ISO 15765-2) and SAE J1939-21.
- Ethernet (“DoIP”, ISO 13400)
- K-LINE (ISO 9181, 14230-2)

DiagRA® S delivers its own implementations of PassThru API (SAE J2534), RP1210 API, D-PDU API (ISO 22900-2) and UDP/TCP sockets according to DoIP standard (ISO 13400), allowing the diagnostic tester applications based on these standards to use DiagRA® S as a simulator for their diagnostic test sequences without needing any additional hardware.

NOTE. *There is a SocketCAN implementation for Linux devices is available at RA Consulting GmbH, but it is currently not part of DiagRA® S Basic / Professional Versions. Please, contact us if you are interested in using DiagRA® S simulation over SocketCAN.*

In case you need to test your diagnostic tester application with some specific hardware (VCI), you can connect DiagRA® S to the physical CAN Bus. DiagRA® S will send CAN Rest Bus messages to this CAN Bus and respond to diagnostic requests according to its simulation model. See Chapter Hardware Simulation for more information.

DiagRA® S covers following use-cases:

- Development of diagnostic sequences without hardware available / existing
- Low costs and short time for creation of test setups
    - Reusage of existing monitoring / communication trace data for configuration
    - Fast switching between test setups
- Testing unusual ECU behavior (fail tests)
- Testing different hardware interfaces (VCI)
- Software in the loop (SIL) for diagnostic software
    - Running diagnostic simulation in command line (CL) mode
- Learn usage of diagnostic software
- Learn vehicle diagnostic protocols

Depending on your license, your DiagRA® S will run as *Basic Version* (see DiagRA® S Basic Version) or *Professional Version* (see DiagRA® S Professional Version).

2.2 DiagRA® S Basic Version

DiagRA® S *Basic Version* is a software-based simulator for diagnostic communication.

It covers following main use-cases:

- Create simulation files (“projects”) containing simulation data out of diagnostic log (trace) files, e.g. ASC, PCAP and DiagRA® D log files. See Chapter Supported log (trace) files for more information.
- Select simulation project files for diagnostic communication.
- Setup settings for diagnostic communication.
- Start / stop / pause different simulation servers.
- Change behavior of simulation models at runtime, e.g. by changing ignition sense state.
- Monitor trace of running diagnostic communication on the DiagRA® S side.

The simulation servers can be also started as separate command line (CL) applications:

- PassThruServer.exe: tool for J2534 (PassThru) / RP1210 and D-PDU API simulation like the J2534 simulation “Server” in DiagRA® S. See Chapter Running PassThru simulator PassThruServer.exe for more information.
- DoIPServer.exe: tool for DoIP like the DoIP simulation server integrated in DiagRA® S. See Chapter Running DoIP simulator DoIPServer.exe for more information.
- HardwareSimulation.exe: tool for execution of hardware simulation using physical connection to CAN bus. See Chapter Running hardware simulator HardwareSimulation.exe for more information.

Running simulations in CL mode is primarily used for automated tests and integration to Continuos Integration (CI) / Continuous Development (CD) / Continuous Testing (CT) systems.

It is recommended to start learning DiagRA® S from the understanding simulation models and simulation format (.dsproj) used by the tool, see Chapter Simulation project files.

2.3 DiagRA® S Professional Version

DiagRA® S *Professional Version* includes all features of *Basic Version*.

Additionally, *Professional Version* includes:

- A powerful editor for simulation files (*Data Manager*), see Chapter Data Manager for more information. *Note that Data Manager is still under development and the editing functionalities might be limited, but this is rapidly changing.*
- Testing tool for simulation models (*DsprojTester*), see Chapter Project Tester tool for more information.
- Several command line tools for automatization of processing diagnostic log (trace) and simulation files:
    - DsprojCreator tool for creation of simulation files out of diagnostic log files, see Chapter DsprojCreator.exe for more information.
    - DsprojConverter tool for (batch) operations on simulation data, e.g. convert simulation files to actual schema version or convert data between different simulation models, see Chapter DsprojConverter.exe for more information.
    - DsprojMerger tool for merging multiple simulation files to one file, see Chapter DsprojMerger.exe for more information.

3 Simulation project files

Simulation project files are XML files having a file extension ".dsproj". Every simulation project describes a single ECU or a group of ECUs belonging together, e.g. as part of the vehicle.

From one version to another version of DiagRA® S the simulation model description in XML file might change, because some new features are added or reworked, or deprecated features are removed. However, the DiagRA® S is, unless it is explicitly stated in release notes and documentation, available to read old project files (*backward compatibility*). The information about the model version of simulation project can be found in the attribute "xlmns" in the root node of the XML file, e.g. "3.0.0" in following case:

```
<?xml version="1.0" encoding="utf-8"?>
<Project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="DiagRA_S_Project_3.0.0.xsd"
         xmlns="http://www.rac.de/DiagRA_S/Project/3.0.0"
         xsi:type="Diagnostics">
    <Information>
```

Starting from DiagRA® S version 2.4.0 there is XSD schema file available for .dsproj files. The XSD schema description is delivered in the installation folder (subfolder "schema"), e.g. "schema\DiagRA_S_Project_3.1.0.xsd" for version 3.1.0.

NOTE. In DiagRA® S *Professional Version* the Data Manager tool and other tools can convert simulation project to the newest version. See Chapter Data Manager (Professional Version only) for more information.

3.1 Structure of simulation project files (.dsproj)

Each simulation project file contains the type of the simulation project, which is defined by the xsi:type attribute of the root node.

Currently, there are five project types supported by DiagRA® S:

- Diagnostics
- SAE J1939-73/-84
- SAE J1699-3 (Official SAE Compliance Test for SAE J1979)
- SAE J1699-5 (Official SAE Compliance Test for SAE J1979-2)
- ZEV Compliance Test (RA Consulting Compliance Test for SAE J1979-3)

Each .dsproj file contains some generic and information (XML node "Information") about this project, e.g. short description and the list of document revisions of this project (XML node "DocRevisions"). The main content of this project (the list of ECUs with their data) is dependent on the project type.

In general, all simulation models can be described as the mapping of diagnostic requests to the list of diagnostic responses. During communication, the requests are getting counted and the next response from the list of responses will be returned. This way it is possible to model changing response values (measured data or context / state dependent data). By using communication settings, it is possible to define the behavior of DiagRA® S in case the end of the list is reached. See Chapter Communication Settings for more information.

The general request-response mapping is extended by the messages, which are automatically broadcasted by DiagRA® S. It is needed for broadcasted PGNs (SAE J1939) and for CAN RestBus simulation.

3.2 Projects of type “Diagnostics”

Projects of type “Diagnostics” contain the ECUs supporting CAN / CANFD / K-LINE and (or) DoIP.

For simulation of DoIP communication the diagnostic project requires the presence of DoIP.VIN for the vehicle and DoIP.LA (Logical Address of the DoIP Edge node).

The diagnostic project might also contain data for CAN RestBus simulation. As of current state, the ECU with address “00” and main transport protocol “CAN” is used for this feature.

Diagnostic projects might also contain the references to ODX databases and diagnostic data sets (COD files of DiagRA® D diagnostic tool), which are used during the import of diagnostic log (trace) files or for displaying diagnostic data in Data Manager tool (DiagRA® S *Professional Version* only).

The simulated diagnostic data for ECU is contained in the (diagnostic) modules of type UDS, OBD, KWP2000 and ExtendedRaw. All the modules support four type of responses:

- Positive (XML node “PosResponse”)
  - Positive response message
- Negative (XML node “NegResponse”)
  - Negative response message
- Multi-response (XML node “MultiResponse”)
  - Multiple responses are sent by the ECU as a reply to one request. Mostly, it is needed for simulation of response pending message (NRC 78) followed by positive / negative response.
- No response (XML node “NoResponse”)
  - No response will be sent

NOTE. KWP2000 module and some UDS services (e.g. ReadDtcInformation) support only positive responses.

In the list of responses all these types can be used in any order. For example, you can define a following response list for request “22 F1 01” of ECU with CAN Response ID “7E8”:

1. First 5 requests will be responded positively.
2. Next request will be responded negatively.
3. Next request will be responded with multi-response (combination of response pending NRC 78 sent several times and positive response at the end).
4. Next request will be not responded.

Each response message might optionally have a enabled state of the message (using XML attribute “e”), timestamp (using XML attribute “t”), delay of the message (using XML attribute “d”)

Enabled attribute (“e” in XML) can have value “true” (default) or “false”. If a response message has enabled attribute set to “false”, it is completely ignored during communication. By default all messages are enabled.

Timestamp is useful for analyzing the source of this response message if it was imported from the diagnostic log (trace) file but has no relevance during communication.

Delay can be defined at two levels: at ECU level (XML node “Delay”) and at response level (XML attribute “d”). If both values are present, the delay is calculated as a sum of both. The delay in the model describes the minimal delay for sending response after latest corresponding matching request. The only exception is MultiResponse: here the delay for first response is the delay regarding

request, the delay values for other responses are relative to the previous response. **NOTE**: Delay value is evaluated only if corresponding communication setting is active!

Consider following MultiResponse for UDS DataIdentifier “01 02”:

```
<MultiResponse n="1">CRLF
    <NegResponse n="1" d="50">78</NegResponse>CR
    <NegResponse n="3" d="500">78</NegResponse>C
    <Response n="1" d="150">00</Response>CRLF
</MultiResponse>
```

The request “22 01 02” will result in several delayed responses.

1. 7F 22 78 after >= 50 ms
2. 3 responses 7F 22 78 after >= 500 ms each (i.e. relative to request: at 550 ms; at 1050 ms and at 1550 ms relative to request).
3. And as last the positive response 62 01 02 00 in about 150 ms after latest negative response, i.e. >=1700 ms after request.

NOTE. In case of Raw CAN (ISO 11898) communication delay is applied to the first frame (FF) of response, all other frames will be sent as fast as possible.

To make the modules process physically address request messages for CAN / CANFD transport protocol (except OBD), you need to add the physical request CAN ID for this ECU inside of the list of module request identifiers. For OBD protocol DiagRA® S uses the addresses predefined by the OBD standard. Functional request identifiers for UDS on CAN / CANFD be defined in the communication settings. For more information see Chapter Communication Settings.

Different modules can be combined in the list of ECU, e.g. you can have OBD, UDS and ExtendedRaw modules for the same ECU. OBD and UDS are responding to different request messages, but ExtendedRaw module might define the requests or request patterns, matching also typical OBD / UDS request messages. In the case of combination of ExtendedRaw module with OBD / UDS / KWP2000 modules following processing order is used by DiagRA® S:

1. ExtendedRaw requests (exact matching to defined request message)
2. OBD / UDS
3. KWP2000
4. ExtendedRaw request patterns

a. ExtendedRaw module

ExtendedRaw module contains pairs of request to response messages in a raw format, without any processing / validation of the data by the simulator. It brings the flexibility to define any possible mapping for any diagnostic protocol, which allows to define a wrong behavior of ECUs. E.g. it is possible to define a response “62 F1 **00**” for the request “22 F1 **01**”, which is wrong for UDS service ReadDataByIdentifier.

Additionally, there is a possibility to define a mapping of request patterns to the responses, which might use some primitive computations.

If request message is not matching to any request / request pattern of this module, it will be not responded.

For more information about ExtendedRaw module, please refer to the example project “DiagRA_S_SimulationProjectFormatDescription_ExtendedRaw.dsproj” in the subfolder “Examples” of the installation folder.

b. UDS module

UDS module implements the handling of UDS (ISO 14229-1) services like TesterPresent ($3E), WriteDataByIdentifier ($2E), WriteMemoryByAddress ($3D), EcuReset ($11), ClearDiagnosticInformation ($14), CommunicationControl ($28), ControlDtcSetting ($85), RequestTransferExit ($37), AccessTimingParameter ($83), SecuredDataTransmission ($84), LinkControl ($87) and ResponseOnEvent ($86). These services are responded positively by DiagRA® S, even if they are not present in the simulation file. It is not possible to define any explicit responses for these services inside of UDS module. If you need to override this behavior, define corresponding request-response mappings in the ExtendedRaw module.

NOTE. The data written by WriteDataByIdentifier ($2E) and WriteMemoryByAddress ($3D) services is made available for the corresponding identifiers of opposite read services ReadDataByIdentifier ($22) and ReadMemoryByAddress ($23).

It is possible to define the services and identifiers (like routines or data identifiers), which are made available only if corresponding DiagnosticSession ($10) or/and SecurityAccess ($27) services were executed successfully. See Chapter ISO 14229 (UDS) for more details.

For services DiagnosticSession ($10), SecurityAccess ($27), ReadDataByIdentifier (§22), ReadScalingDataByIdentifier ($24), ReadDtcInformation ($19), InputOutputControlByIdentifier ($2F), RoutineControl ($31), RequestDownload ($34), RequestUpload ($35) and TransferData ($36) it is possible to define responses in case corresponding identifiers are requested. For each identifier (e.g. Data Identifier (DID) of service ReadDataByIdentifier (§22) it is possible to define the list of responses. In contrast to the ExtendedRaw module, the definition of responses contains only the relevant response data bytes, while the other data bytes of response message are added automatically by DiagRA® S. E.g. for Data Identifier DID=0xF101 of service ReadDataByIdentifier (§22) the positive response message “02 03” corresponds to the full response message “62 F1 01 02 03” and negative response message “31” corresponds to the full response message “7F 22 31”.

NOTE. Data and Routine Identifiers of services ReadDataByIdentifier (§22) and RoutineControl ($31), which are reserved for reading supported Data / Routine Identifiers by WWH-OBD (ISO 27145) and OBDonUDS / ZEVonUDS (SAE J1979-2/3) standards, can be responded automatically by DiagRA® S, see Chapter ISO 14229 (UDS) for more details.
For more information about UDS module, please refer to the example project “DiagRA_S_SimulationProjectFormatDescription_UDS_OBD.dsproj” in the subfolder “Examples” of the installation folder.

c. OBD module

OBD module implements the handling of OBD (SAE J1979-1) ClearEmissionRelatedDiagnosticInformation ($04). This services is responded positively by DiagRA® S, even if it is not present in the simulation file. It is not possible to define any explicit responses for this service inside of OBD module. If you need to override this behavior, define corresponding request-response mappings in the ExtendedRaw module.

For all other services it is possible to define responses in case corresponding identifiers (like PIDs, FreezeFrames or InfoTypes) are requested. Similar to UDS module, the definition of responses contains only the relevant response data bytes, while the other data bytes of response message are added automatically by DiagRA® S. E.g. for PID=0x01 of service RequestCurrentPowertrainDiagnosticData (§01) the positive response message “02 03” corresponds to the full response message “41 01 02 03” and negative response message “31” corresponds to the full response message “7F 01 31”.

For more information about OBD module, please refer to the example project "DiagRA_S_SimulationProjectFormatDescription_UDS_OBD.dsproj" in the subfolder "Examples" of the installation folder.

d. KWP2000 module

KWP2000 is like UDS module, but it allows only definitions of positive responses. Following diagnostic services are implemented directly in the KWP2000 module and it is not possible to define any responses for them: StartDiagnosticSession ($10), ClearDiagnosticInformation ($14), StopDiagnosticSession ($20), SetDataRates ($26), TesterPresent ($3E).

e. RestBus simulation

For diagnostic projects it is possible to simulate CAN Restbus by defining a series of CAN messages sent to the CAN Bus / client application periodically with some raster (repetition rate in milliseconds).

RestBus simulation works only in case the connection established over RawCAN (e.g. Hardware Simulation or PassThru with RawCAN protocol).

NOTE. Currently only CAN is supported for RestBus simulation. Adding CANFD supported is scheduled for an upcoming version 3.2.0.34.

3.3 Projects of types for compliance tests "SAE J1699-3", "SAE J1699-5" and Zev Compliance Test

For the simulation of compliance test a model similar to ExtendedRaw module is used. However, the data for each ECU is additionally structured for each Test separately.

Typically, compliance test projects are created by importing the corresponding log file from SAE J1699-3 or SAE J1699-5 compliance test tool. For SAE J1699-3 only log files of version 14.06.02 or later are supported.

Please note that DiagRA® S is only able to respond to requests which were saved in the log file. If you are running a compliance test to which there is no information in DiagRA® S project file, DiagRA® S won't be able to respond to requests of those tests. To make sure that the compliance test will be successful, you need to run the compliance test tool with an imported log file of the same application version.

You also must note, that aborting test 5.18 (SAE J1699-3) or 5.17 (SAE J1699-5) is not allowed, because DiagRA® S won't be able to synchronize with the compliance test after aborting corresponding test.

3.4 Projects of type "SAE J1939-73/-84"

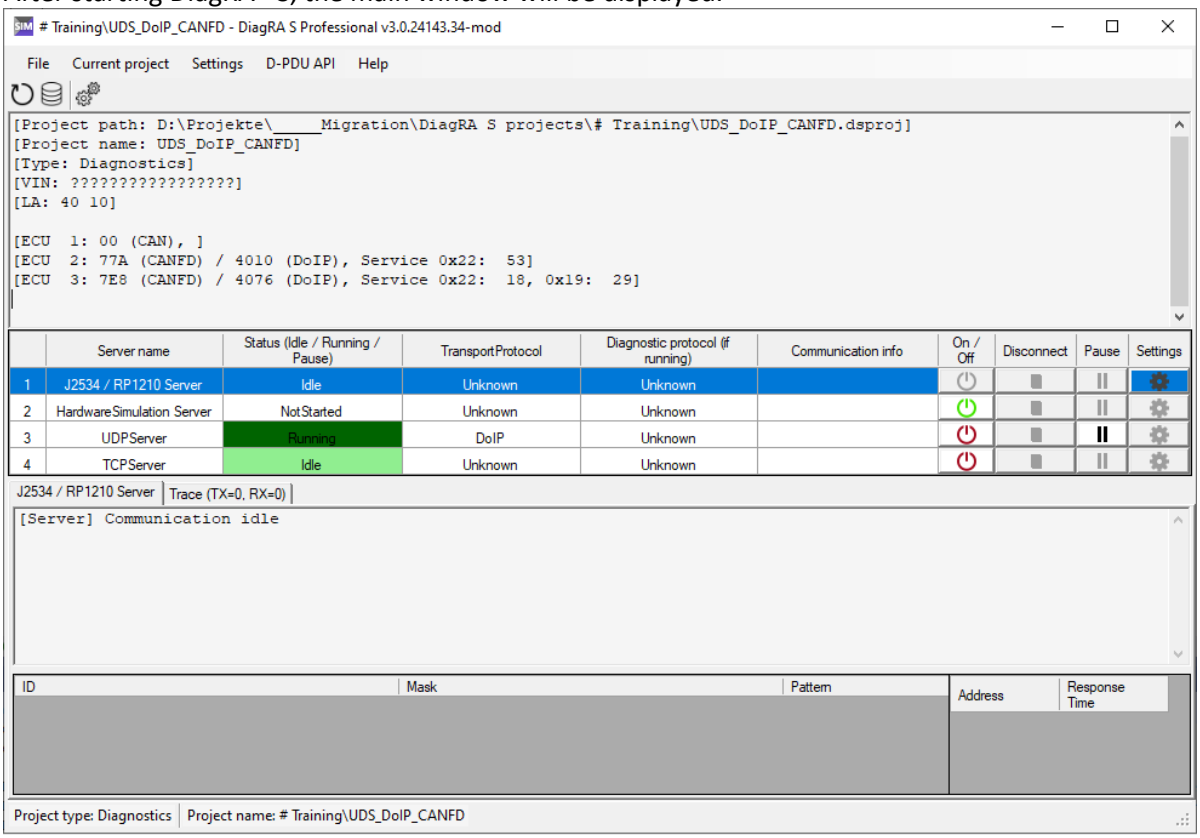
Projects of type SAE J1939-73/-84 contain the ECUs and the corresponding messages grouped by PGN. DiagRA® S uses and internal to distinguish between diagnostic messages (DMs) and broadcast PGNs, which are automatically broadcasted by DiagRA® S at the raster predefined by SAE J1939 standard. The data model for PGN is very similar to the one used by RP1210 for SAE J1939 protocol.

Known Limitations:

- Currently there is no way to define the raster in the model.
- It is not yet possible to define other broadcast PGNs.
- Broadcasting after request is not yet supported and handled in the same way as normal broadcasted PGNs.

4 DiagRA® S Main window

After starting DiagRA® S, the main window will be displayed.



4.1 Quick start

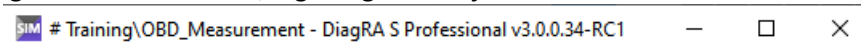
The minimum effort to simulate diagnostic communication with DiagRA® S is:

1. Start the application „*DiagRA_S.exe*“.
2. *Optionally*, select the base project folder for simulation project files (.dsproj) using menu **File** → **Select project folder**.
3. Select in DiagRA® S the existing simulation project you want to work: use menu **File** → **Open project** (any simulation project file) **or** **File** → **Select project** / **Find project** (for simulation files located current project folder) or just drag'n'drop simulation file (.dsproj file) to the main window of DiagRA® S.

   NOTE. In case there is no simulation project file present, you can create a new project by importing diagnostic log (trace) files. See Chapter Create new simulation project for more information.
4. After simulation project is selected, corresponding simulation servers (except *HardwareSimulation Server*) are started automatically and your diagnostic tester application supporting SAE J2534 API, RP1210 API, D-PDU API or DoIP can start communicating with DiagRA® S.
5. *Optionally*, start *HardwareSimulation Server* if you want to connect DiagRA® S to physical CAN Bus (requires at least one CAN ECU to be present in the simulation file).

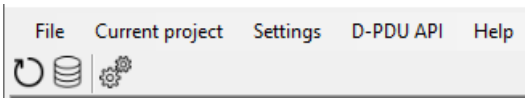
4.2 Overview of main window

The title of the main window contains the name / relative path of currently selected simulation project and DiagRA® S version name, e.g. DiagRA S *Professional* v3.0.0.34-RC1.



The relative path of simulation project is displayed only in case project is located inside the currently selected project folder.

At the top of the window there is a main menu and the toolbar, which repeats some most frequently used functions from the main menu.



Below the toolbar there is a larger text area containing the fast overview over the content of simulation project in a textual form.

```
[Project path: D:\Projekte\_____Migration\DiagRA S projects\# Training\UDS_DoIP_CANFD.dsproj]
[Project name: UDS_DoIP_CANFD]
[Type: Diagnostics]
[VIN: ??????????????????]
[LA: 40 10]

[ECU  1: 00 (CAN), ]
[ECU  2: 77A (CANFD) / 4010 (DoIP), Service 0x22:  53]
[ECU  3: 7E8 (CANFD) / 4076 (DoIP), Service 0x22:  18, 0x19:  29]
```

This overview text contains the generic information about this project file, e.g. path to the simulation project file, project name (equal to file name), type of the project and further information like DoIP.VIN, DoIP.LA or path to the ODX database. Below this generic information, the text area contains the list of ECUs with numbers of defined requests (ExtendedRaw module) or identifiers for some diagnostic services (UDS, OBD).

The list of simulation servers is displayed below the overview of the project content.

	Server name	Status (Idle / Running / Pause)	Transport Protocol	Diagnostic protocol (if running)	Communication info	On / Off	Disconnect	Pause	Settings
1	J2534 / RP1210 Server	Idle	Unknown	Unknown					
2	HardwareSimulation Server	Not Started	Unknown	Unknown					
3	UDPServer	Running	DoIP	Unknown					
4	TCPServer	Idle	Unknown	Unknown					

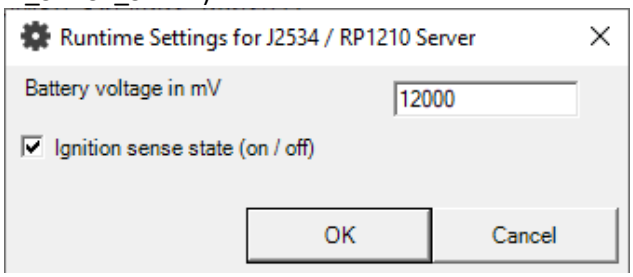
The table contains simulation server name, its status, last used transport and diagnostic protocol and some generic communication info, last the Windows process id (PID) of the last client application connected to the J2534 / RP1210 Server, detailed name of connected hardware interface for the HardwareSimulation Server or IP address and port of the lastly client application connected to UDP- / TCPServer (parts of DoIP Server).

NOTE. J2534 / RP1210 server acts also as simulation server for D-PDU API in case of communication over CAN / CANFD.

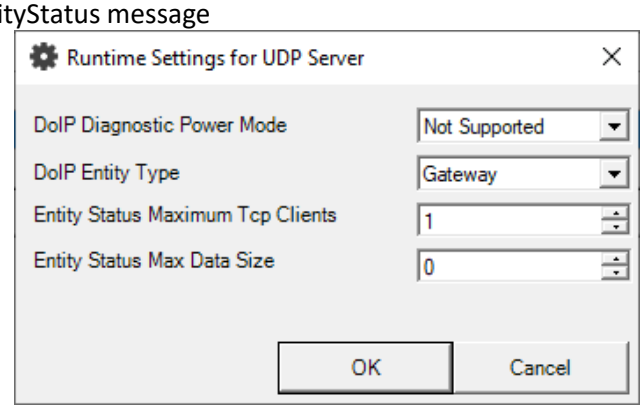
All simulation servers can run in parallel. It is possible to control servers using buttons in columns “On/Off”, “Disconnect” and “Pause”. Note, that not every simulation server supports all these functionalities.

- “On/Off”: start / stop server. If server is not started, no active connection to this simulation server is possible.
    - J2534/RP1210 Server, UDPServer and TCPServer are started automatically.

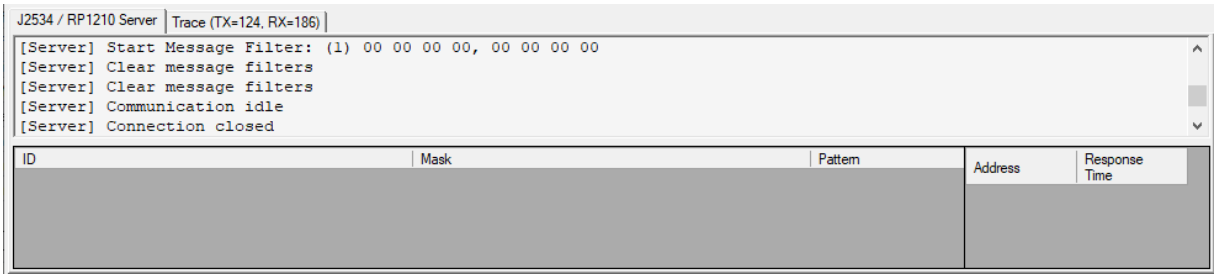
  - J2534/RP1210 Server currently doesn't support manual start and stop. This feature will be added in one of future releases.
  - HardwareSimulation needs to be actively started by the user since it requires physical connection to the CAN bus, which needs to be configured prior to the connection. See documentation for **Fehler! Verweisquelle konnte nicht gefunden werden.** feature.
  - UDPServer and TCPServer are available only if correct DoIP identification data (DoIP.VIN and DoIP.LA) is present in project file
- "Disconnect": disconnect from current client application.
  - J2534/RP1210 Server: disconnect from the tester application and wait for next connection. Tester will have to actively reconnect again. Available only if any diagnostic tester is currently connected.
  - TCPServer: disconnect current TCP socket and wait for next connection. Available only if any TCP connection to tester is active.
  - UDPServer, HardwareSimulation: not available.
- "Pause": requests are accepted, but no request processing takes place and no responses are sent back.
- "Settings": server specific settings which might be changed during communication (if any)
  - For J2534/RP1210 Server two options corresponding to widely used IoCtls are available: battery voltage (READ_VBATT) and ignition sense state (IGNITION_SENSE_STATE).



  - For UDP Server it is possible to configure DoIP Diagnostic Power Mode value, DoIP Entity Type, max number of TCP Clients and Max Data Size reported in DoIP GetEntityStatus message



Below the server table is an area that display several tabs. The number of tabs and the tab content is dependent on the currently selected simulation server. Selection of the server must be done over the server table.

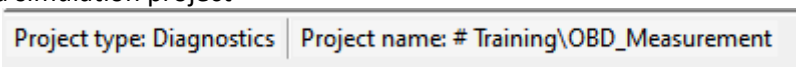


There is at least one tab called "XXXServer" present, e.g. *J2534/RP1210 Server* or *TCPServer* or *HardwareSimulation Server*. This tab displays the connection details and status changes of this simulation server.

The next tab to the server details is *Trace* tab, which contain the last message received and sent by the simulation server. Depending on the setting, the name of the tab might contain the number of TX and RX counters, see Chapter General Settings for more details. The *Trace* tab contains all the messages, which are received and sent by the simulation server, even if they are not diagnostic messages, e.g. Diagnostic ACK message for DoIP server, CAN RestBus simulation messages and SAE J1939 Broadcast PGNs.

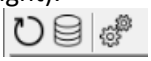
If server can distinguish received diagnostic requests and sent diagnostic responses by type of diagnostic service or ECU identification, the corresponding data will be added in a corresponding tab as well. It depends on current settings, if messages are grouped by diagnostic service type or by ECU ID, see Settings for more information.

At the bottom of the main window, there is a status bar, which contains the type and the name of currently selected simulation project



4.3 Functions of toolbar

The buttons in the toolbar are (from left to right):



- Reload (reset) Project
- Data Manager (only in *Professional Version*)
- Settings

a. Reload (reset) project

Project will be loaded again. Internal communication state will be reverted. This is button is enabled only if project is selected. This button is very useful in case the simulation project file was changed and the new changes are needed to be loaded into DiagRA® S. **NOTE**. In settings you can also activate auto-reload of project files.

b. Data Manager

This button is available only in *Professional Version*.

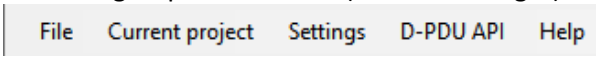
Using this button, you can open the Data Manager tool, see Chapter Data Manager (Professional Version only) for more information. This is button is enabled only if project is selected.

c. Settings

Using this button, you can open corresponding dialog for DiagRA® S settings. This button is enabled only if no communication is running. See Chapter General Settings for more information.

4.4 Functions of main menu

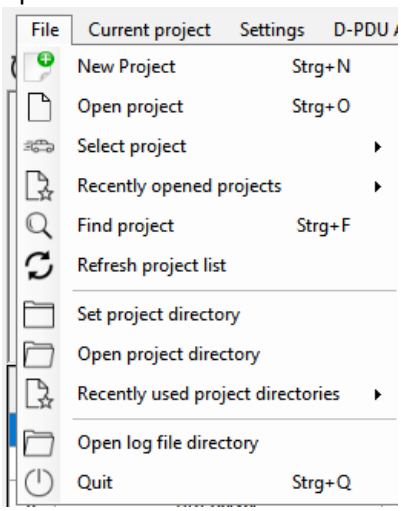
The menu items of main menu are grouped as follows (from left to right):



- **File** (contains file and folder operations)
- **Current project** (contains operations on currently selected simulation project)
- **Settings** (contains setting of DiagRA® S)
- **D-PDU API** (contains documentation and setting of D-PDU API)
- **Help** (contains help and further documentation)

a. File

Menu **File** contains file and folder operations.



File creation and file selection:

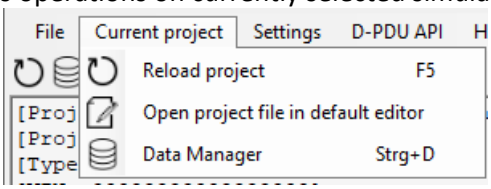
- **New project** starts dialog for creation of new simulation project, see Chapter Create new simulation project
- **Open project** starts dialog for selection of simulation project from any location
- **Select project** contains a submenu, allowing selection of the simulation projects located in the currently selected project folder. Submenu is updated automatically in case files in selected project folder are changed.
- **Recently opened projects** contains a submenu, allowing selection of one of ten recently used simulation projects.
- **Find project** opens a dialog, where it is possible to search projects from the current project folder. Clicking **OK** in the dialog, the project selection will be changed.
- **Refresh project list** allows to explicitly refresh list of the submenu of menu **Select project**

Folder selection and further operations:

- **Set project directory** opens the dialog for selection of current project folder. Default project folder is “C:\Users\Public\Documents\RA Consulting\DiagRA S\Projects”.
- **Open project directory** opens currently selected project folder in Windows Explorer.
- **Recently used project directories** contains a submenu, allowing selection of one of ten recently used project folders.
- **Open log file directory** opens the folder containing internal protocol (log) files of DiagRA® S in Windows Explorer.
- **Quit** closes DiagRA® S.

b. Current project

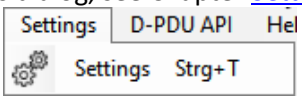
Menu **Current project** contains operations on currently selected simulation project.



- **Reload project** reloads current project, see Chapter Reload (reset) project for more information.
- **Open project file in default editor** opens currently selected project using the default Windows application associated with .dsproj simulation project files. Typically, it will be some XML (Oxygen, XML Spy etc.) or Text Editor (Notepad++ etc.).
- **Data Manager** (only in *Professional Version*) opens Data Manager tool of DiagRA® S for viewing and editing content of simulation project files (.dsproj). See Chapter Data Manager (Professional Version only) for more information.

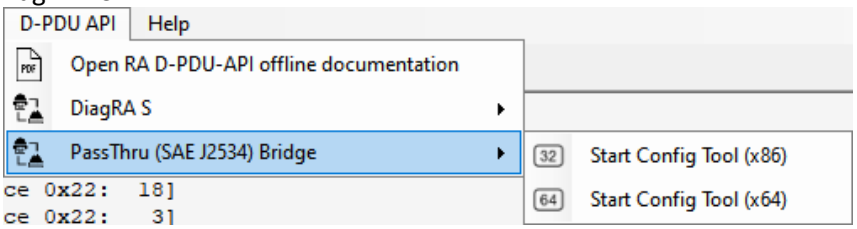
c. Settings

Menu **Settings** calls DiagRA® S **Settings** dialog, see Chapter Settings.



d. D-PDU API

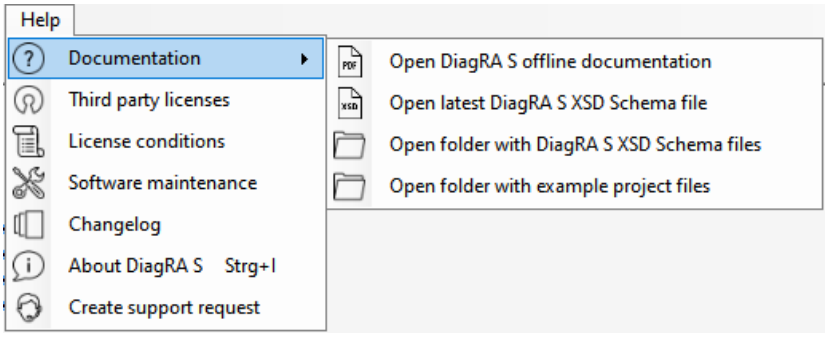
Menu **D-PDU API** contains following items related to D-PDU API by RA Consulting, which is installed together with DiagRA® S:



- **Open RA D-PDU API offline documentation** opens the PDF document describing D-PDU API by RA Consulting.
- **DiagRA S** contains a submenu, which allows to open configuration tool (*DeviceSelection.exe*) of DiagRA® S specific D-PDU API (ISO 22900-2) version, which will be used by x86 or x64 applications. It is possible to configure the name and some further details for corresponding D-PDU API ModuleType and logging settings (file name and log level) of D-PDU API. See Chapter D-PDU-API for more information.
- **PassThru (SAE J2534) Bridge** contains a submenu, which allows to open configuration tool (*DeviceSelection.exe*) of D-PDU API PassThru Bridge version, which will be used by x86 or x64 applications. D-PDU API PassThru Bridge allows the usage of hardware interfaces providing PassThru (SAE J2534 04.04) API using the D-PDU API functions. x86 version of this D-PDU API will be also used by DiagRA® S itself, if you will use *HardwareSimulation* feature. By using configuration tool it is possible to configure the name and some further details for corresponding D-PDU API ModuleTypes, refresh the list of known PassThru interfaces (if some new interfaces were installed), remove or disable PassThru interface and change logging settings (file name and log level) of D-PDU API. See Chapter Hardware Simulation for more information.

e. Help

Menu **Help** contains following menu items:



- **Documentation** → **Open DiagRA S offline documentation** opens this document.
- **Documentation** → **Open latest DiagRA S XSD Schema file** opens the latest XSD schema file (currently DiagRA_S_Project_3.2.0.xsd) in the tool associated in Windows with XSD files (e.g. XML Spy, Oxygen, Notepad++ etc.)
- **Documentation** → **Open folder with DiagRA S XSD Schema files** opens the folder with all XSD schema file of DiagRA® S in Windows Explorer.
- **Documentation** → **Open folder with example project files** opens the folder with example simulation project files in Windows Explorer.
- **Third party licenses** opens the XML file containing all license information of 3rd party libraries used by DiagRA® S.
- **License conditions** opens the PDF document describing license conditions for using software of RA Consulting, e.g. DiagRA® S.
- **Software maintenance** opens the PDF document describing software maintenance conditions for RA Consulting software, e.g. DiagRA® S.
- **Changelog** opens the dialog for viewing protocol of changes made in different DiagRA® S versions.
- **About DiagRA S** opens the dialog with detailed information about your license and version of DiagRA® S.
- **Create support request** starts the tool (cRAshReporter.exe) collecting information about current installation, configuration, log files of DiagRA® S and current process information (threads, stack trace etc.) and creating a zip file with file extension .crashreport. You can add description to your report and send .crashreport file to RA Consulting as suggested by the cRAshReporter tool. This is the most comfortable way to send the information to our support.
  NOTE 1. No simulation project files or diagnostic log (trace) files are collected by the tool. They might be provided separately, if requested by RA Consulting support.
  NOTE 2. The same tool cRAshReporter.exe will be started automatically in case DiagRA® S has unhandled error, causing the crash of the application.

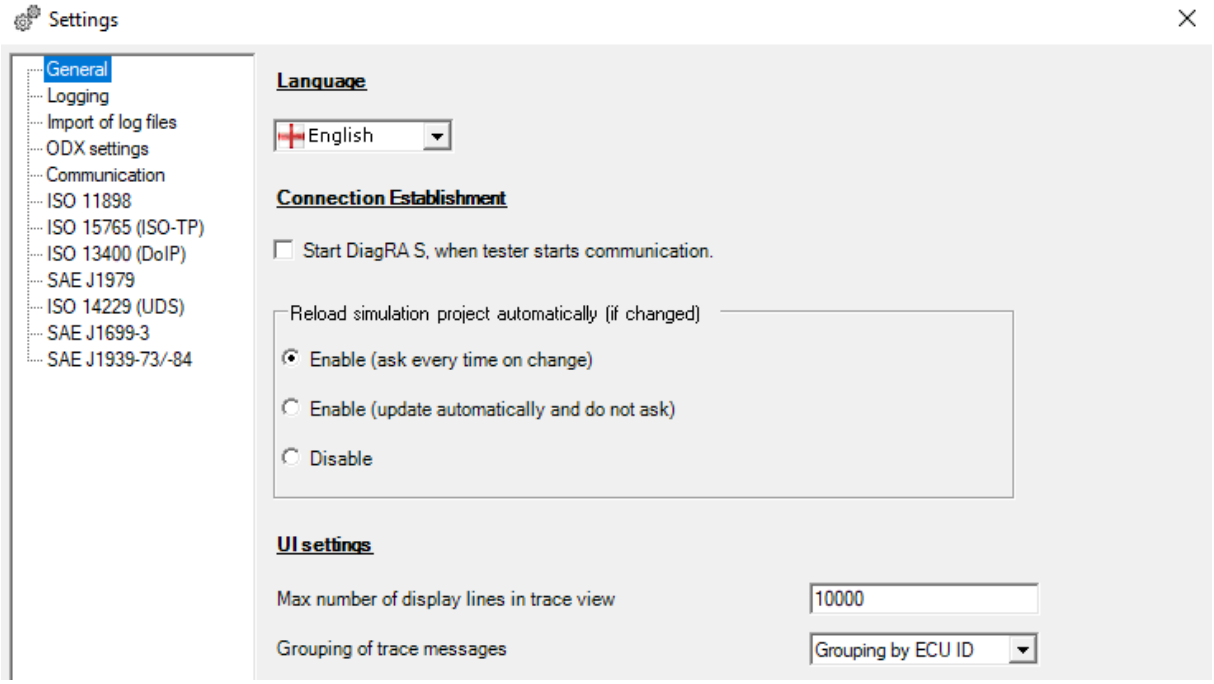
5 DiagRA® S Settings

Settings are grouped in the **Settings** dialog by the corresponding settings type.

NOTE. DiagRA® S settings are not synchronized to Data Manager and DsprojTester, so these tools might have different settings.

5.1 General

In **General** settings it is possible to change following settings:

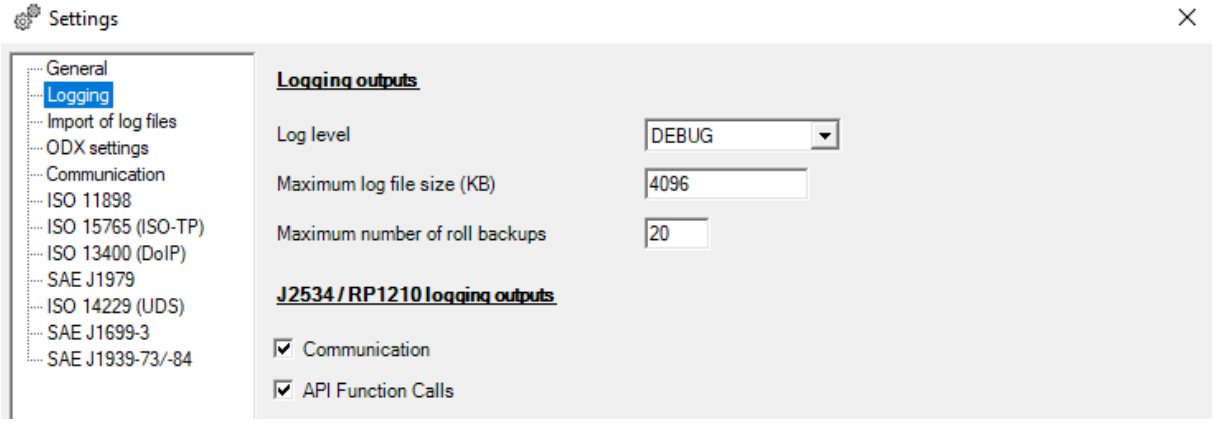


- **Language** selection. Currently, there are three languages supported by DiagRA® S: English, German and Chinese. After selecting a new language, the Settings dialog will automatically be updated. After confirming the settings, *most* of other UI elements will update, too. For complete switching the language, you will need to restart DiagRA® S.
- **In Connection establishment** it is possible:
  - "*Start DiagRA S, when tester starts communication*": this configuration allows to start DiagRA® S automatically if corresponding functions of PassThru API (PassThruOpen), RP1210 API (RP1210_ClientConnect) and D-PDU API (PDUModuleConnect) are called by the client diagnostic application using corresponding DLL of DiagRA® S. Default value is *false*.
  - "*Reload simulation project automatically (if changed)*": this configuration allows to reload simulation project automatically, if the corresponding file is changed. Default value is *"Enabled (ask every time on change)".*
    NOTE. In case of auto-reload DiagRA® S will try to apply new state of the project as fast as possible, however the internal ECU state will be lost!
- **Max number of display lines in trace view** restrict the max number (<=100000) of lines in **Trace** tab. Default value is 10000.
- **Grouping of trace messages** defines if the messages in the trace view are grouped in additional tabs per ECU ID or per Service ID (default). The names of such tabs are dependent on the diagnostic protocol and the number of the tabs depends on the messages received by the simulation server. For example, in case of grouping by Service ID and OBD communication running you'll see the tabs Mode 01 to Mode 0A and each of them will display request and response messages of its Mode.

5.2 Logging

Logging settings are defined in to two subsections:

- **Logging outputs** for settings related to DiagRA® S own log files.
- **J2534 / RP1210 logging outputs** for logging settings related to PassThru API (SAE J2534) and RP1210 API libraries delivered with DiagRA® S.

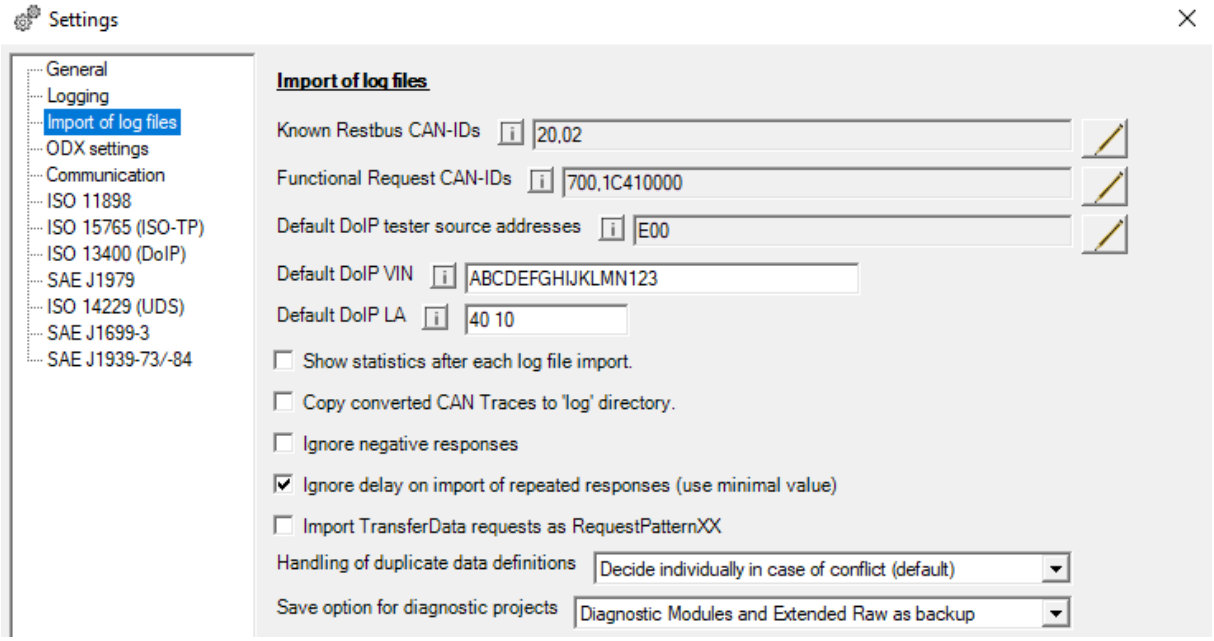


Following logging settings of DiagRA® S itself can be configured: **Log level**, **Maximum log file size** in KB and **Maximum number of roll backups**. Log level “DEBUG” (default) is recommended for most of the use-cases. Use “INFO” / “WARN” / “ERROR” in case higher performance is required.

For SAE J2534 / RP1210 APIs it is possible to configure if **API Function Calls** and transmitted / received messages (**Communication**) are to be protocoled in the corresponding log files. It is recommended to turn on both options unless you have special performance requirements.

5.3 Import of log files

In section **Import of log files** it is possible to configure following settings for importing diagnostic log (trace) files

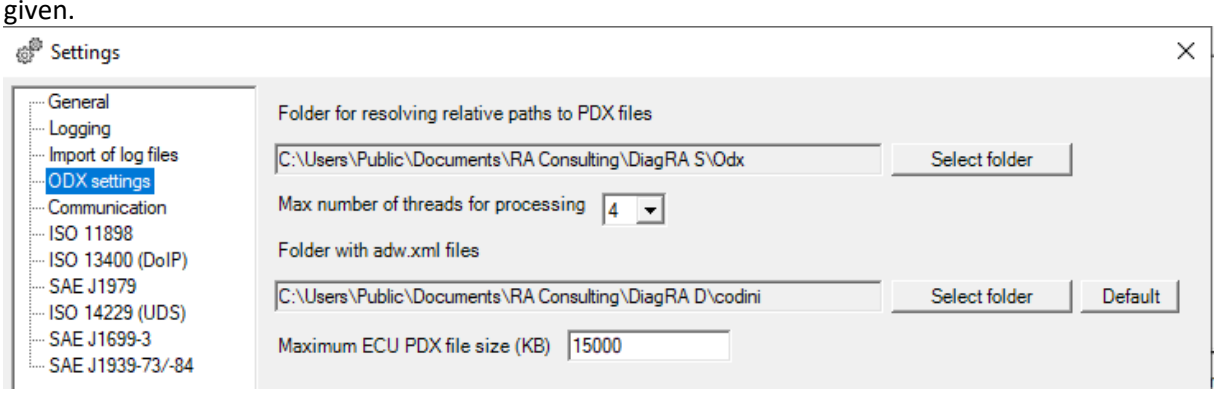


- **Known RestBus CAN IDs** contains the list of CAN IDs automatically assigned to RestBus if importing Raw CAN trace (ASC, TRA) files to project of type “Diagnostics”. Use button with pen symbol to start dialog for managing the list of CAN IDs.

- **Functional Request CAN-IDs** contains the list of CAN IDs assumed as request CAN IDs for functional communication if importing Raw CAN trace (ASC, TRA) files to project of type "Diagnostics". Use button with pen symbol to start dialog for managing the list of CAN IDs.
- **Default DoIP tester source addresses** defines the list of DoIP tester source addresses recognized by default, in case DoIP trace files (PCAP, PCAPNG, ASC etc.) doesn't contain Routing Activation Response message. Default entry is 0E00.
- **Default DoIP VIN** defines the default VIN used on import of DoIP trace files (PCAP, PCAPNG, ASC etc.) in case files don't contain Vehicle Identification Response / Announcement message. Default value is "ABCDEFGHIJKLMN123".
- **Default DoIP LA** defines the default LA used on import of DoIP trace files (PCAP, PCAPNG, ASC etc.) in case files don't contain Vehicle Identification Response / Announcement message. Default value is "40 10".
- **Show statistics after each log file import** defines if dialog with detailed statistics is displayed after each log file is imported. Statistics dialog contains the number of requests and responses stored in the imported file. Default value is *false*.
- **Copy converted CAN traces to log directory** defines if temporary log file, which is created when you import a Raw CAN Trace, is copied as a permanent file into the "log" directory. Default value is *false*.
- **Ignore negative responses** defines if negative responses are ignored during import. Default value is *false*.
- **Ignore delay on import of repeated responses (use minimal value)** defines if for the same request two consequent responses with the same response message, but with different value of delay will be handled as same responses. Activating this setting will be make simulation projects smaller. Default value is *true*.
- **Handling of duplicate ECU definitions** defines how the importer shall handle the case ECU to be imported is already present in the target simulation file. There are four options available: *Keep (existing) ECU data and don't ask*, *Merge ECU data and don't ask, Replace (existing) ECU data and don't ask* and *Device individually per ECU* (default).
- **Save option for diagnostic projects** defines for project files of type 'Diagnostics', which diagnostic module (simulation model) is created: as diagnostic modules (UDS, OBD etc.), as ExtendedRaw or both at the same time (ExtendedRaw is stored in the second (backup) file), which is a default setting.

5.4 ODX settings

In section **ODX Settings** it is possible to define the base folder for resolving relative paths to PDX files ("C:\Users\Public\Documents\RA Consulting\DiagRA S\Odx" is default), maximum number of threads for processing PDX files and maximum ECU PDX file size for the case folder with ECU PDX files is given.

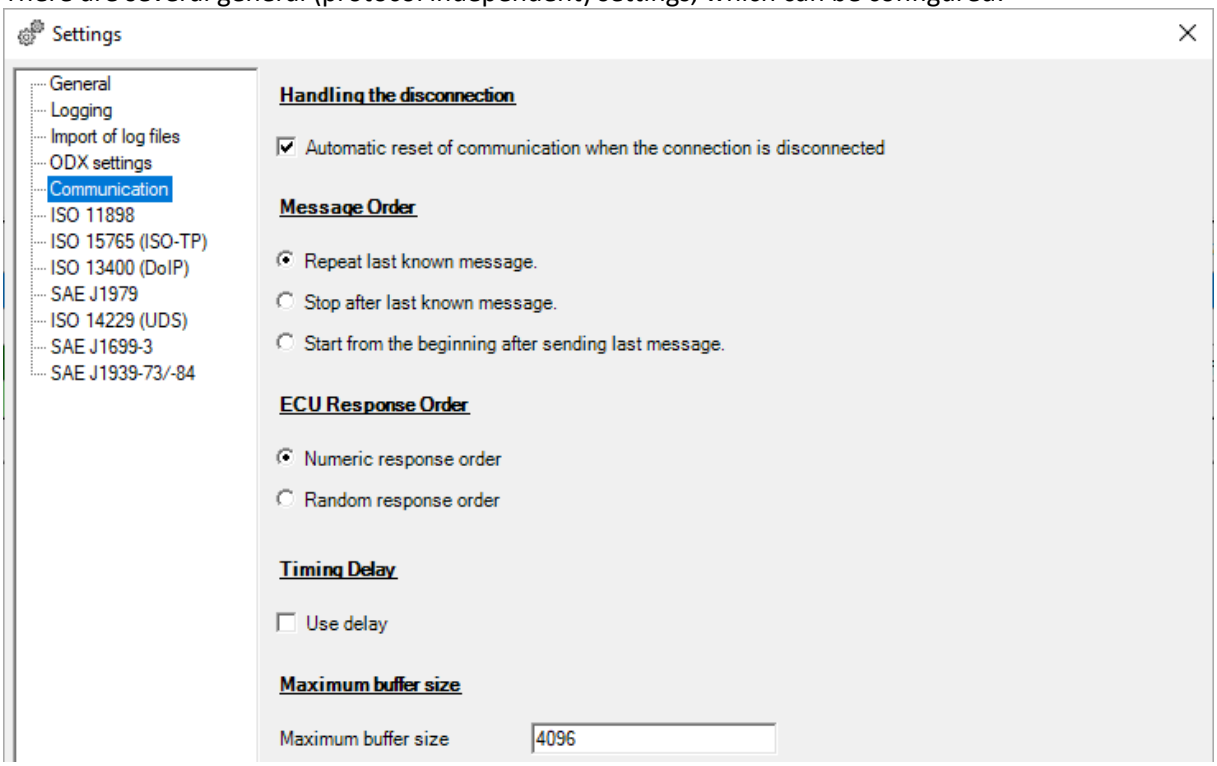


Additional setting „Folder with adw.xml files" defines the location of the folder with adw.xml files. If you have diagnostic tool DiagRA® D installed, its folder with adw.xml files ("C:\Users\Public\Documents\RA Consulting\DiagRA D\codini") will be used by default.

NOTE 1. adw.xml are specific to DiagRA® D diagnostic software and contain information about ECU identifiers (CAN / DoIP) and supported protocols.

5.5 Communication

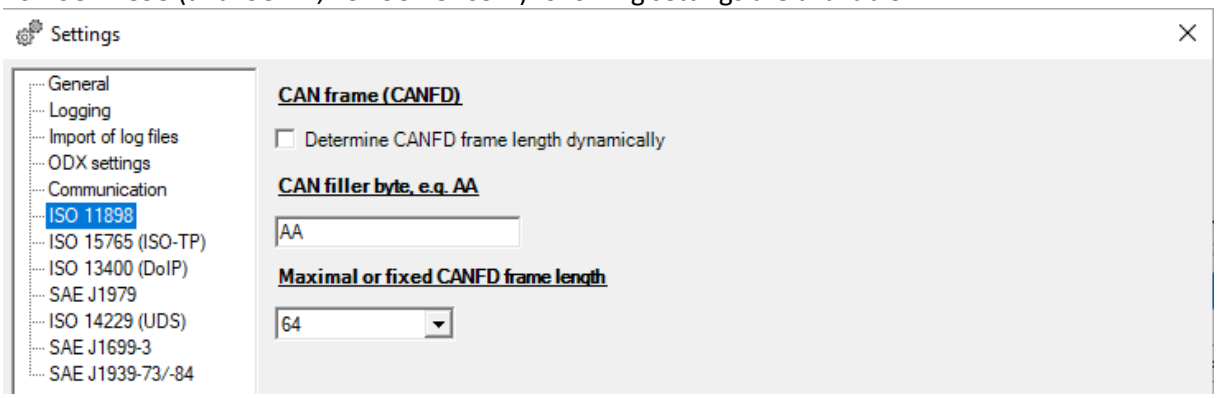
There are several general (protocol independent) settings, which can be configured:



- **Automatic reset** defines, if DiagRA® S internal has to be reset to initial state after connection for simulation server is disconnected (i.e. diagnostic tester was disconnected). Resetting internal state will reset the internal positions in response lists (see **Message Order**). Default value is “true”, i.e. the internal state will be reset.
- **Message Order** defines behavior of DiagRA® S if the last message in response list is reached. The three options are (from top to bottom):
    a. The last known message is repeated (default).
    b. The communication stops after the last known message has been sent.
    c. After the last known message has been sent, the communication restarts at the first known message.
- **ECU Response Order.** The default option, *Numeric response order*, states that all ECUs respond in a numerical order to a request. For example, if you have two ECUs with CAN identifiers 7E8 and 7E9 and you are using the OBD protocol, the ECU 7E8 will always respond before 7E9. The *Random response order* randomizes the responses, i.e. you can’t say which ECU responds first and which responds last. This option can be used only in case response delay option is turned off (see below).
- **Timing Delay** defines handling of the response delay. This feature, which is explained in detail in Chapter Projects of type “Diagnostics”, enables the user to change the response timings of ECUs and single responses during communication. Default value is *false*.
- **Maximum buffer size** sets the maximal number of Bytes DiagRA® S accepts in an incoming message (ISO 15765). Any messages that are longer will result in an Overflow-FlowControl message. Default is *4096*.

5.6 ISO 11898

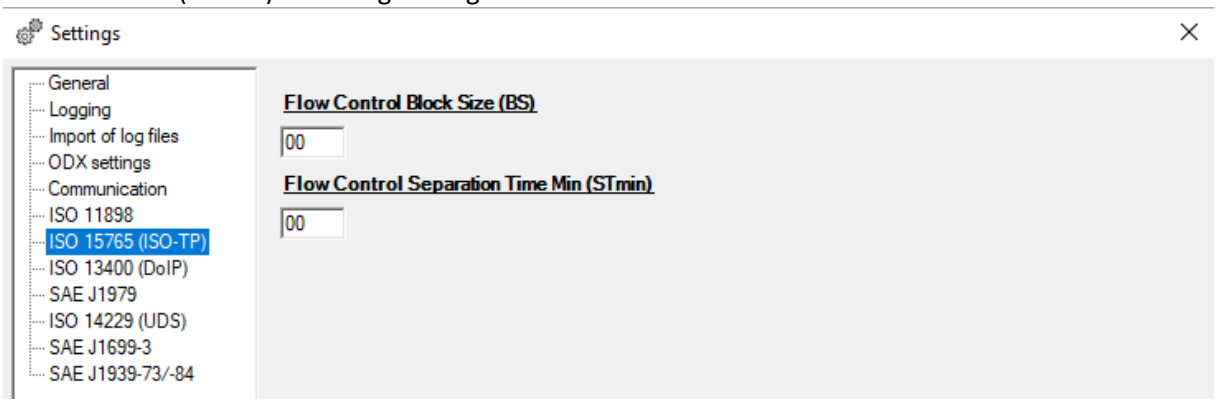
For ISO 11898 (and ISO-TP, i.e. ISO 15765-2) following settings are available:



- **Determine CANFD frame length dynamically**. If option is set, CANFD frame length is determined dynamically. Otherwise, fixed CANFD frame length will be used. Default is *true*.
- **CAN filler byte**. Default is *0xAA*.
- **Maximal (or fixed) CANFD frame length**. Default is *64*.

5.7 ISO 15765 (ISO-TP)

For ISO 15765 (ISO-TP) following settings are available:

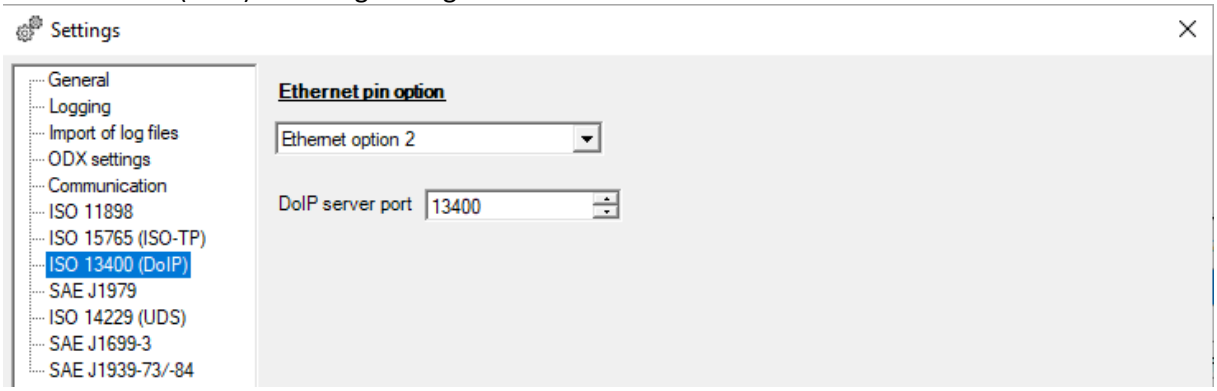


- **Flow Control Block Size (BS)**. The value of Block Size (hex) reported by DiagRA® S to the tester in the Flow Control message. Default value is *00*.
- **Flow Control Separation Time Min (STmin)**. The value of STmin (hex) reported by DiagRA® S to the tester in the Flow Control message. Default value is *00*.

NOTE. If ECU has its own settings for FlowControl messages, they have a higher priority as the global setting.

5.8 ISO 13400 (DoIP)

For ISO 13400 (DoIP) following settings are available:

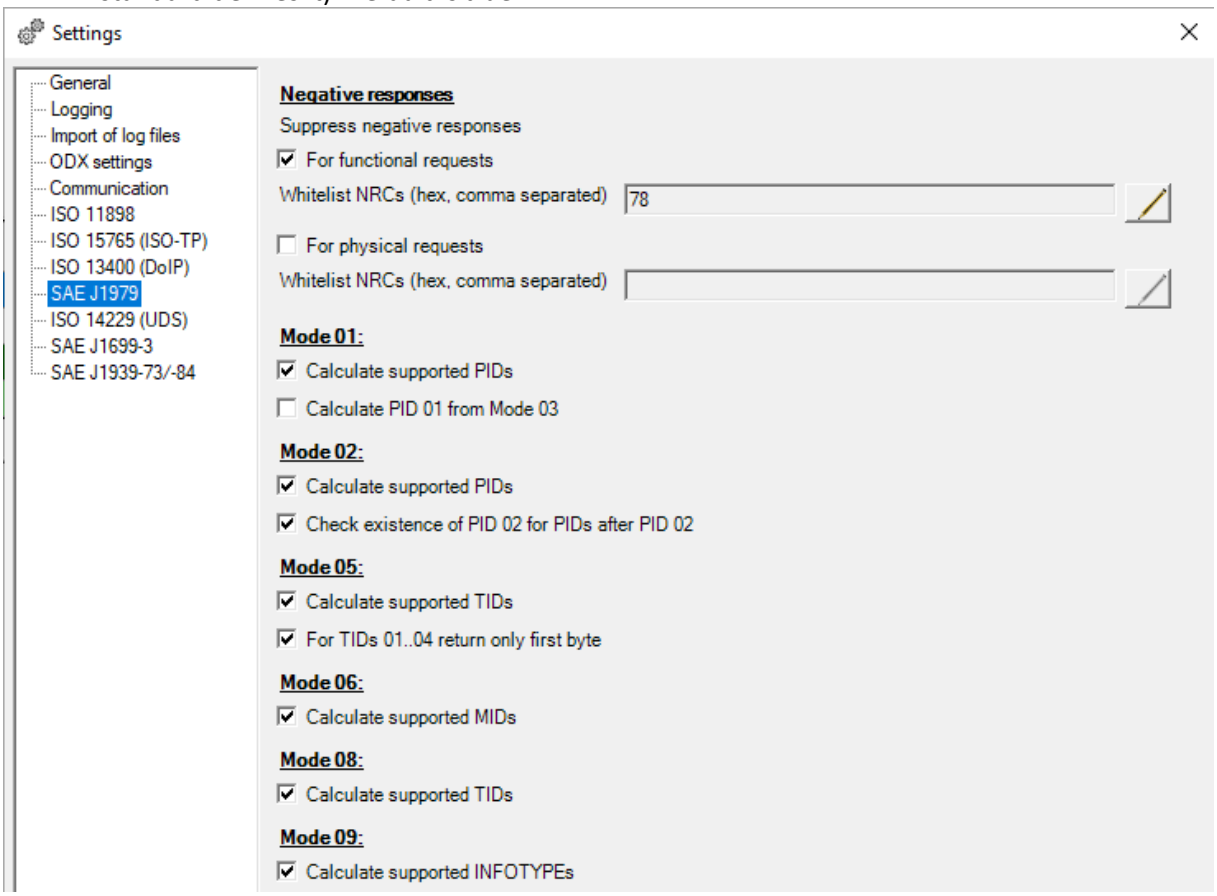


- **Ethernet pin option** 1 or 2. Used for D-PDU API IO Controls "IOCTL_GET_ETH_SWITCH_MODE" and "IOCTL_SET_ETH_SWITCH_MODE". Default is *Ethernet option 1*.
- **DoIP server port**. The port used for monitoring TCP and UDP connections for DoIP. Default is *13400*.

5.9 SAE J1979

For OBD classic (SAE J1979-1) following settings are available:

- **Suppress negative responses for functional and physical**. If enabled, corresponding negative responses are not sent by DiagRA® S. In case responses are suppressed it is possible to exclude some Negative Response Codes (NRCs) over corresponding whitelists. Default is to suppress negative responses for functional requests, except of NRC 78 (Response pending) and not suppress negative responses for physical requests.
- **Calculate supported PIDs / TIDs / MIDs / INFOTYPEs**. If enabled, corresponding values are calculated by DiagRA® S itself and responses defined for this PID / TID / MID / INFOTYPE are ignored. Default is *true*.
- **Mode 2: Check existence of PID 02 for PIDs after PID 02**. If PID 02 is not defined in project file, all other PIDs after PID 02 (e.g. PID 03) will be not responded. Default is *true*.
- **Mode 5**: for TIDs 01..04 return only first byte. If set, the response is limited to first byte (as standard defines it). Default is *true*.

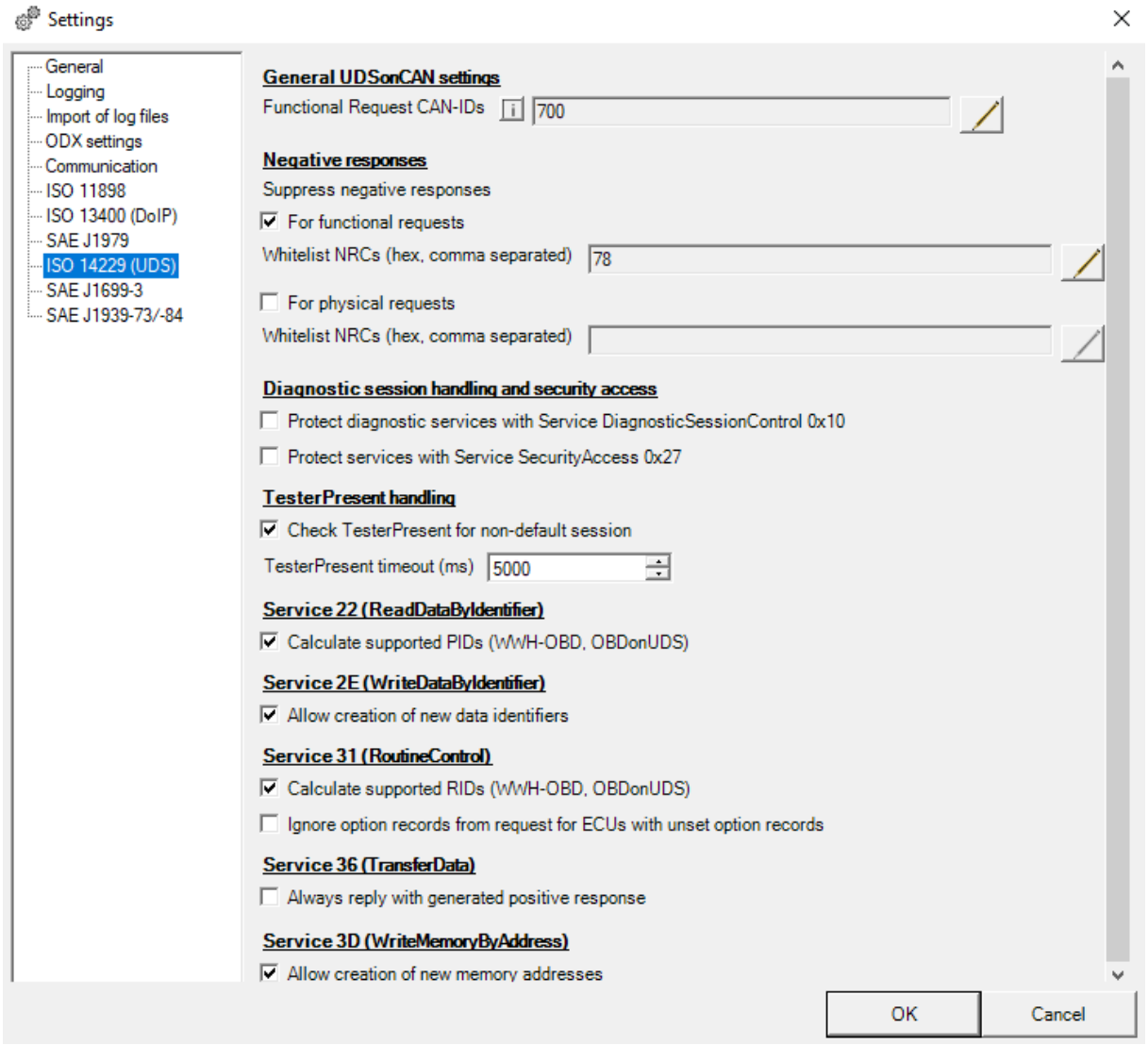


5.10 ISO 14229 (UDS)

For UDS (ISO 14229) there are following settings available:

- **Functional Request CAN-IDs** contains the list of CAN IDs accepted as request CAN IDs for functional communication. Use button with pen symbol to start dialog for managing the list of CAN IDs.

- **Suppress negative responses for functional and physical**. If enabled, corresponding negative responses are not sent by DiagRA® S. In case responses are suppressed it is possible to exclude some Negative Response Codes (NRCs) over corresponding whitelists. Default is to suppress negative responses for functional requests, except of NRC 78 (Response pending) and not suppress negative responses for physical requests.
- **Diagnostic session handling and security access**: **Protect diagnostic services using DiagnosticSessionControl** (according to UDS standard). By default, this option is deactivated.
- **Diagnostic session handling and security access**: **Protect diagnostic services using SecurityAccess service** (according to UDS standard). By default, this option is deactivated.
- **TesterPresent handling** defines if for non-default UDS Session the diagnostic tester must send TesterPresent message (or any other diagnostic request message) periodically in order to keep current non-default UDS Session. It is possible to configure the time interval for checking TesterPresent message sent by diagnostic tester. If there is no request from tester in the given time interval, the UDS Session is switched to default session. By default, this option is activated with default period of 5000 ms.
- **Calculate supported PIDs / RIDs** for WWH-OBD (ISO 27145) and OBD on UDS (SAE J1979-2). If enabled, corresponding values are calculated by DiagRA® S itself and responses defined for this PID / RID are ignored. Default is *true*.
- **Service 2E (WriteDataByIdentifier): Allow creation of new data identifiers** and **Service 3D (WriteMemoryByAddress): Allow creation of new memory addresses**. If option is activated (default), DiagRA® S will create new data identifier or memory address and allow reading it using corresponding UDS read service. If option is deactivated, DiagRA® S will assume such data identifiers / memory addresses as not known and will reply such requests with negative response (NRC 31).
- **Service31 RoutineControl: Ignore option records from request for ECUs with unset option records**. If this option is activated (by default it is *false*) and for ECU no optionRecord values are defined at all, it is assumed that all optionRecords are equally valid and the response is calculated without matching of optionRecords (behavior of DiagRA® S before version 2.3.0).
- **Service 36 (TransferData): Always reply with generated positive response**. If this option is activated (by default it is *false*), all formally valid requests for these service will be responded with generated positive response, independent of the fact that corresponding request is stored in the project file or not.

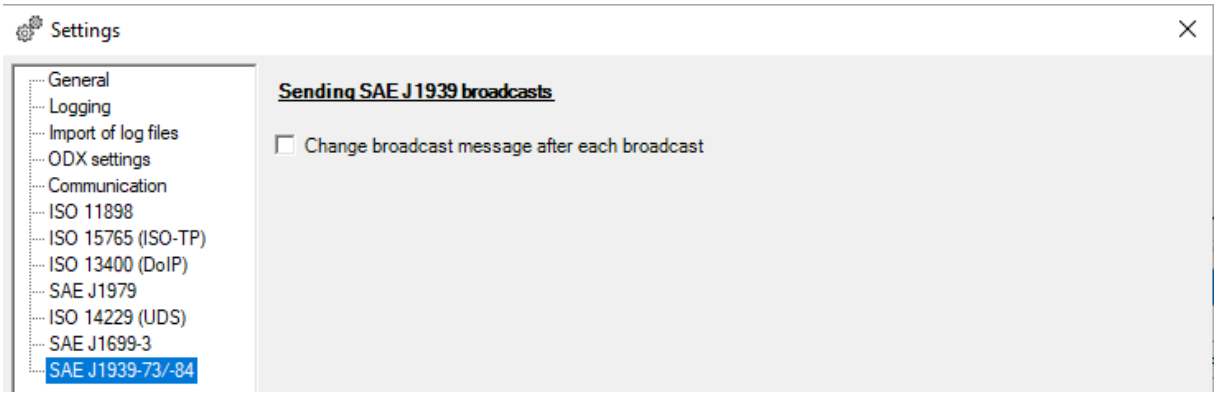


5.11 SAE J1699-3

DiagRA® D offers a comfortable GUI to run the SAE J1699-3 Compliance Test and DiagRA® S offers an easy way to configure it. If this option is selected, DiagRA® S will change the settings of DiagRA® D's remote control of SAE J1699-3, so that you don't need to change any settings in DiagRA® D manually.

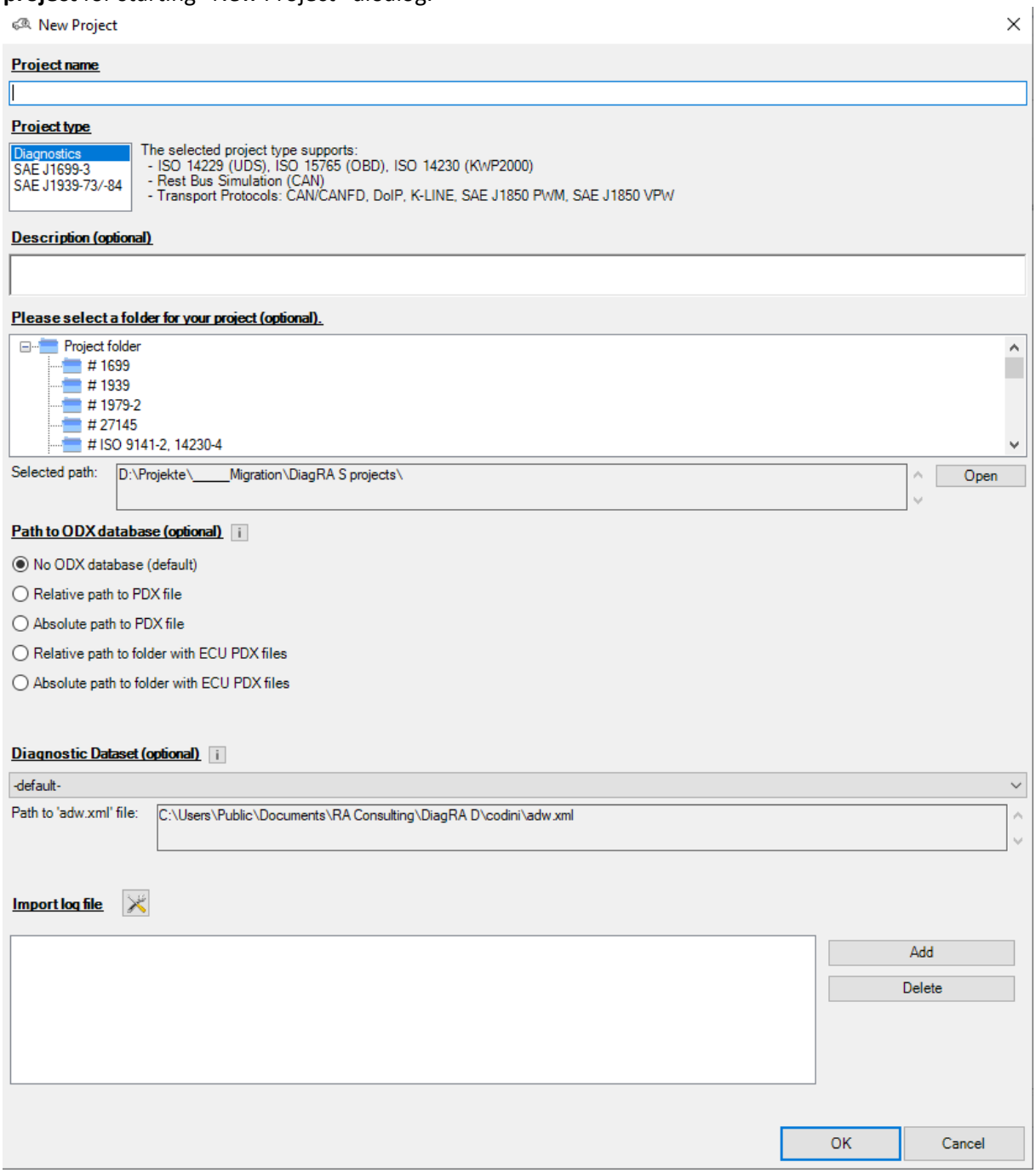
5.12 SAE J1939-73/-84

Many Parameter Groups will be sent using a broadcast message. The message will change each time if this option is selected. If it's not selected, the message will only change after a request was received.



6 Create new simulation project

You can create a new project from diagnostic log (trace) files. Use main menu entry **File** → **New project** for starting “New Project” dioalog.



1. Select simulation project type (“Diagnostics” is default)
2. Input the unique name of the simulation project (for the currently selected project folder / subfolder)
3. *Optionally*, input the description of the simulation project
4. *Optionally*, select the folder, where the project file will be located. The folders displayed are the subfolders of the currently selected project folder. You can create new subfolders using Windows Explorer, the tree view will be automatically updated. Use **Open** button to start Windows Explorer for currently selected subfolder.
5. *Optionally*, select diagnostic data set from the folder with adw.xml files (see ODX settings for more information).

NOTE. The selection of diagnostic data set makes sense only for projects of type “Diagnostics”.

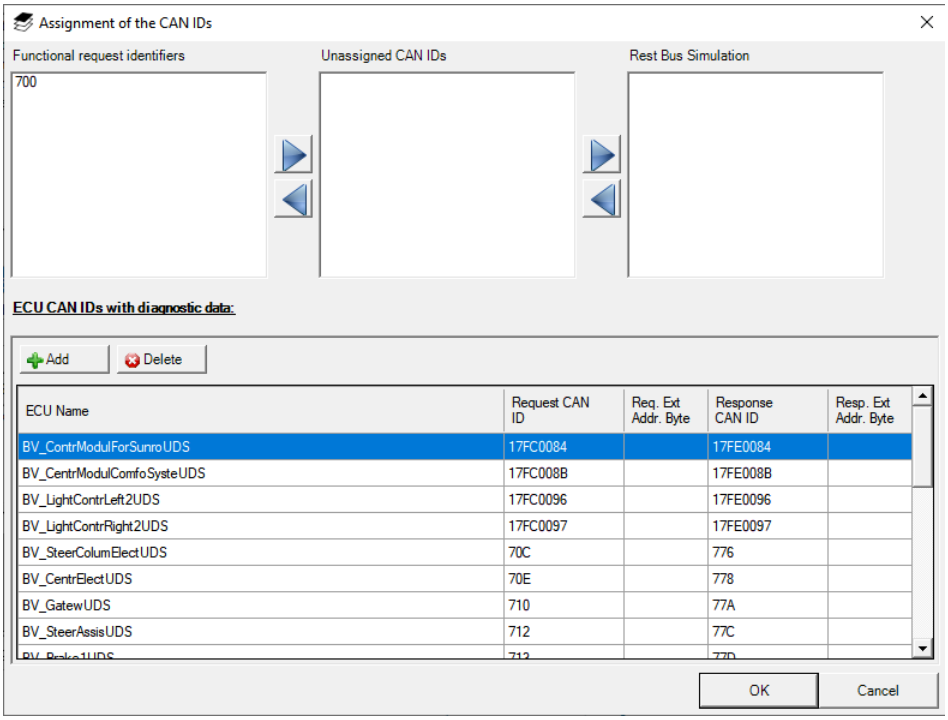
6. *Optionally*, select ODX database for the vehicle.
   NOTE. Only for projects of type “Diagnostics”.
7. Select one or more file diagnostic log (trace) files to be imported by using “Add” button or by Drag’n’Drop of log (trace) files to the corresponding area.
   NOTE. You shall select files allowed to be imported for this project type. See Chapter Import of diagnostic log (trace) files for more information.
8. Click on **OK** to start creation of the new project
9. *(Conditional)* If you’ve selected a project type “Diagnostics” and one or more CAN trace files with raw CAN messages, e.g. ASC (Vector) or TRA (IXXAT), for each such file you will see the dialog **Assigning of the CAN IDs**, for more information see Chapter Assigning of the CAN IDs for raw CAN trace files. For other diagnostic log (trace) file formats, this step is not needed.
10. *(Conditional)* After the import of all log files, the dialog displaying import statistics might be displayed if corresponding setting is activated. See Chapter Import of log files for more information.
11. After creation of new simulation project, it gets automatically selected in DiagRA® S.

6.1 Assigning of the CAN IDs for Raw CAN trace files

For projects of type “Diagnostics” both diagnostic response messages and CAN RestBus messages can be simulated. On importing raw CAN trace messages from ASC (Vector) or TRA (IXXAT) files, it is impossible to deterministically distinguish between diagnostic messages and CAN RestBus messages, because the format of the CAN frames might be nearly identical.

However, in case diagnostic data set or / and ODX database is selected for this project, the of DiagRA® S will use the request and response CAN IDs from these databases for distinguishing between diagnostic messages on CAN and other RestBus CAN messages. Also, the known functional Request CAN IDs and Known RestBus CAN IDs configured in settings will be used for automatic assigning of CAN IDs to the correct target. Based on this information DiagRA® S will assign all CAN IDs to the diagnostic ECU CAN IDs, functional request identifiers and RestBus CAN IDs. The remaining CAN IDs will be moved to the group of “Unassigned CAN IDs”. Unassigned CAN IDs will be ignored by DiagRA® S and not imported to the simulation project file.

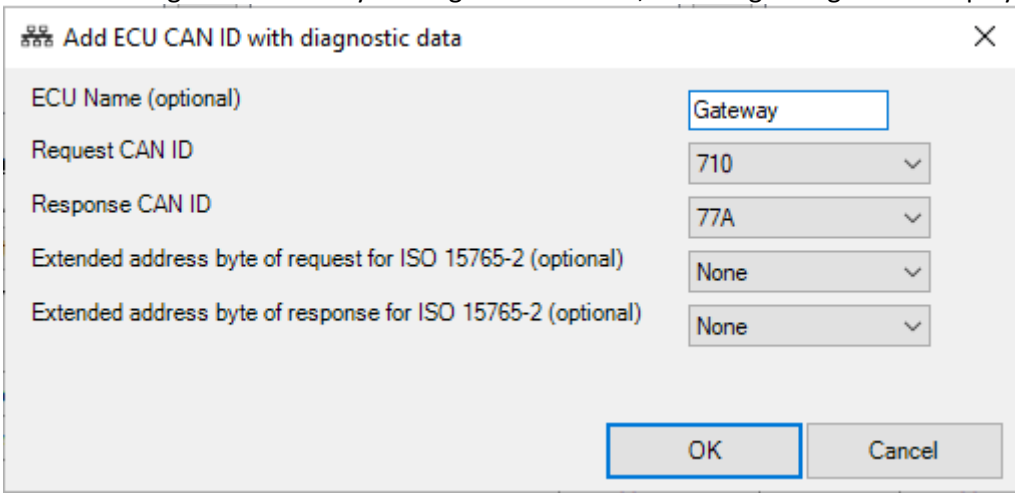
It is possible to manually adapt assignments made by DiagRA® S. Following dialog will be displayed by DiagRA® S automatically:



All unassigned CAN Identifiers are listed in the corresponding list in the top center part of the dialog. These CAN Identifiers will be ignored. To use CAN Identifiers for the RestBus simulation, select them in the central top list and click on the button with the right arrow located to the right from this list. As you can see, the selected CAN Identifiers are moved to the right column and will be used for rest bus simulation after dialog is finished by clicking **OK** button.

To mark CAN identifiers as functional request identifiers, select them in “unassigned” list and move them to the list on the left.

To define diagnostic request and response CAN identifiers of ECUs, use button **Add** after the line “ECU CAN Ids with diagnostic data”. By clicking on this button, following dialog will be displayed:

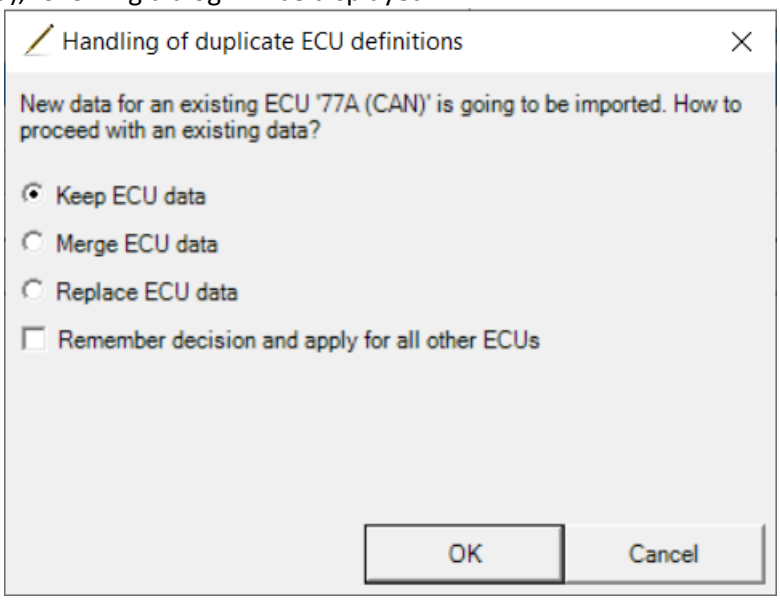


In the dialog **Add ECU CAN ID with diagnostic data** you can assign available CAN IDs to values of request / Response CAN ID. For ECUs using extended address byte according to ISO 15765-2, it is possible to select one of possible determined values for these bytes. Optionally, it is possible to give a name for the ECU (the name will be currently not imported!). Click on **OK** button if all assignments are correct.

6.2 Import of multiple files / conflict with already existing data

In case multiple files imported to the simulation project or in general, ECU to be imported is already present in the simulation project, this case is considered by importer as a conflict and needs to be resolved. Resolving of the conflict is done using corresponding import setting, see Chapter Import of log files for more information.

In case the import setting requires that the decision is made individually for each ECU (and for each file to be imported), following dialog will be displayed:



It is possible to keep existing data, replace existing data or merge data. Individual decision can be remembered and applied for further ECUs.

6.3 Import of diagnostic log (trace) files

There are several diagnostic trace file types, which can be imported to DiagRA® S simulation projects. Please, contact us if you require the import of data from other file type.

a. Diagnostic projects

DiagRA® D / Silver Scan-Tool log files (.log):

- ISO 14229 (UDS, WWH-OBD, OBDonUDS, ZEVonUDS) on CAN / CANFD or IP (DoIP)
- ISO 15765 / SAE J1850 PWM / SAE J1850 VPW / ISO 9141 (OBD classic)
- ISO 14230 (KWP2000) on CAN

DiagRA® F files (.log):

- ISO 14229 (UDS) on CAN / CANFD

DiagLogging trace files from **emotive** or VWG OTX Runtime (DiagLogging_*XXX*.log):

- ISO 14229 (UDS, WWH-OBD, OBDonUDS, ZEVonUDS) on CAN / CANFD or IP (DoIP)
- ISO 15765 (OBD classic)
- ISO 14230 (KWP2000) on CAN

CAN Trace files with OBD classic, ISO 14229, KWP2000 on CAN data and CAN RestBus data (only CAN and no CANFD yet):

- Vector ASCII Traces (.asc)
- IXXAT CAN Traces (.tra)

- Mercedes (XENTRY) CAN Traces (CanMonitoring_*XXX*.log)

PCAP and PCAPNG DoIP trace files (.pcap, .pcapng):

- ISO 14229 (UDS, WWH-OBD, OBDonUDS, ZEVonUDS) on IP (DoIP)

Trace files with raw Ethernet frames (DoIP):

- Vector ASCII Traces (.asc)

SR Files (.sr):

- ISO 14229 (UDS, WWH-OBD, OBDonUDS, ZEVonUDS) on CAN
- ISO 15765 / SAE J1850 PWM / SAE J1850 VPW / ISO 9141 (OBD classic)
- ISO 14230 (KWP2000)

SimFrog simulation files (.sim, .par):

- ISO 14229 (UDS, WWH-OBD, OBDonUDS, ZEVonUDS) on CAN
- ISO 15765 / SAE J1850 PWM / SAE J1850 VPW / ISO 9141 (OBD classic)
- ISO 14230 (KWP2000) CAN or K-LINE

DST-i simulator files (.ini):

- ISO 14229 (UDS, WWH-OBD, OBDonUDS, ZEVonUDS) on CAN
- ISO 15765 / SAE J1850 PWM / SAE J1850 VPW / ISO 9141 (OBD classic)
- ISO 14230 (KWP2000) on CAN or K-LINE

b. SAE J1939 projects

DiagRA® D / Silver Scan-Tool log files (.log):

- SAE 1939 with and without RawCAN messages (SAE J1939-73/-84 or SAE J1939-21)

CAN Trace files:

- Vector CAN Traces (.asc)

c. SAE J1699-3, SAE J1699-5, Zev Compliance Test projects

Supported is import of SAE J1699-3 compliance test files starting from version 14.06.02.

7 Supported standardized APIs for communication

DiagRA® S installs three standardized APIs for diagnostic communication: PassThru (SAE J2534-1), RP1210-B API and D-PDU API (ISO 22900-2). All three APIs are “virtual”, in other words they don’t require any additional hardware to be installed and they communicate to DiagRA® S (or PassThruServer.exe) over inter-process communication (IPC). These APIs can be used by any diagnostic tester application supporting these standards.

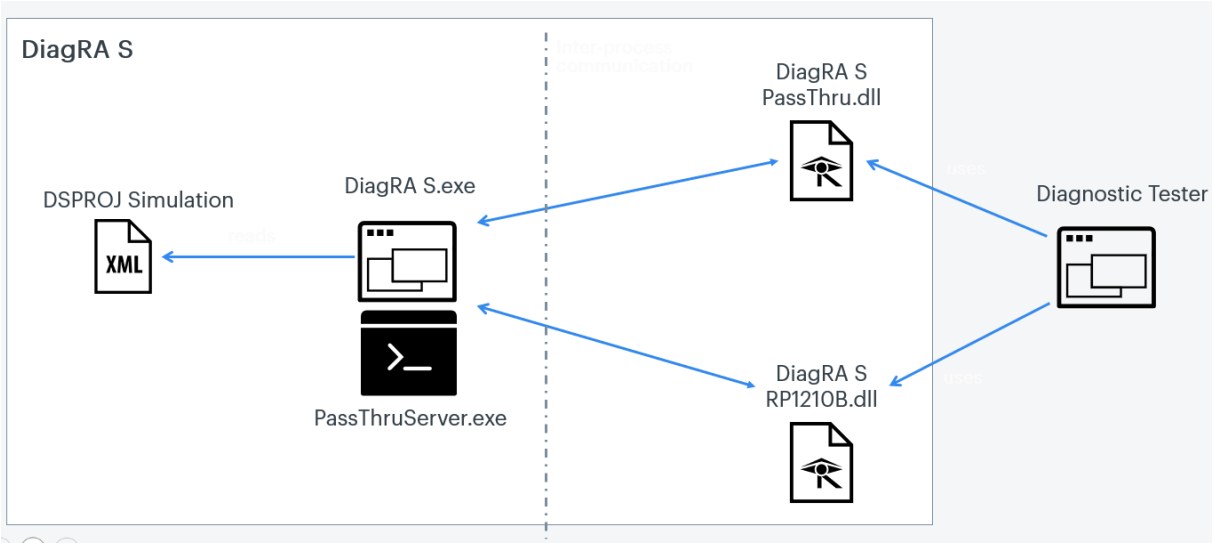
The following table shows all installed virtual diagnostic interfaces (APIs) and the supported protocols.

Standard	PassThru	RP1210 B	D-PDU API
ISO 9141 (OBD on K-LINE)	✔	✖	✖
ISO 11898 (Raw CAN)	✔	✔	✔
ISO 15765-3 / ISO 14229-3 (UDS on CAN / CANFD)	✔	✖	✔
ISO 14229-5 (UDS on IP)	✖	✖	✔
ISO 14230 (KWP2000 on CAN)	✔	✖	✔
ISO 15031-5 / SAE J1979-1 (OBD on CAN)	✔	✖	✔
SAE J1939 (SAE J1939-73/-84)	✖	✔	✖
SAE J1850 PWM	✔	✖	✖
SAE J1850 VPW	✔	✖	✖

NOTE. If application communicates over Raw CAN (ISO 11898), all CAN based protocols are supported, including SAE J1939.
NOTE. All timestamps, which are returned by implementations of standardized APIs, are in milliseconds.

7.1 PassThru API

PassThru API of DiagRA® S (PassThru.dll) is installed in the installation folder of DiagRA® S in x86 and x64 versions. It is registered in Windows Registry under PassThruSupport.04.04 node with the name “RAC – EcuSim”.



Supported version of PassThru SAE J2534-1:2004 (PassThru 04.04), however some features from SAE J2534-2:2019 (e.g. CANFD) are supported as well. Also, some proprietary extensions are supported. Please contact us if you need more detailed information about our PassThru API v04.04 implementation.

Supported protocols:

- ISO 9141 (OBD on K-LINE)
- ISO 11898 (Raw CAN)
- ISO 14230 (KWP2000 on K-LINE)
- ISO 15765 (for OBD / UDS on CAN or CANFD)
- SAE J1850 PWM
- SAE J1850 VPW

Support of ISO 11898 (Raw CAN) means that any supported CAN based protocol (UDS, OBD, SAE J1939 etc.) is also supported in case diagnostic application uses Raw CAN for communication.

7.2 TMC RP1210 B API

RP1210-B API of DiagRA® S (RARP1210B.dll) is installed to C:\Windows folder only in x86 version as required by the RP1210 standard and listed in RP121032.ini file under the name RARP1210B.

Supported protocols:

- ISO 11898 (RawCAN)
- SAE J1939 (SAE J1939-73/-84)

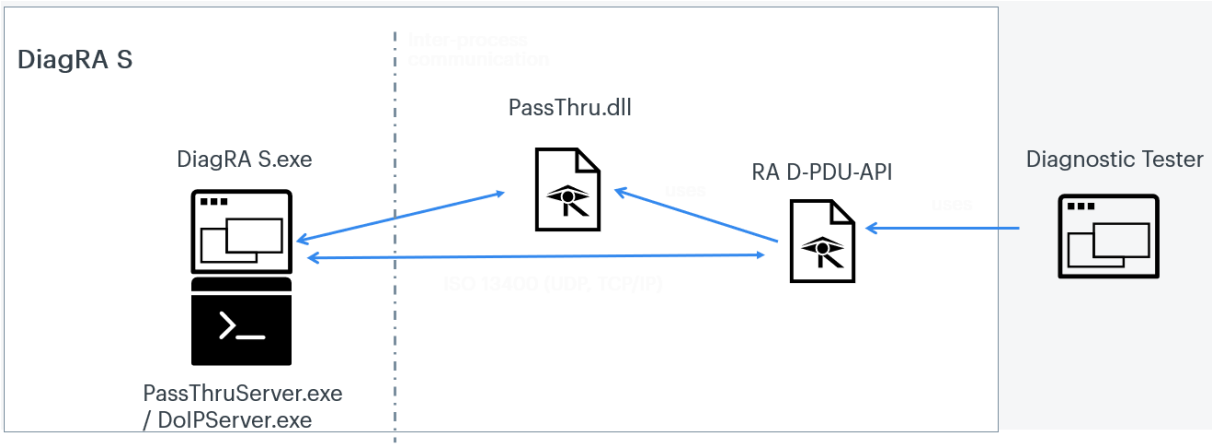
Again, support of ISO 11898 (Raw CAN) means that any supported CAN based protocol (UDS, OBD, SAE J1939 etc.) is also supported in case diagnostic application uses Raw CAN for communication.

7.3 D-PDU-API

DiagRA® S installs and configures RA D-PDU API (Bridge) on your computer during the installation process. The D-PDU API is installed two times: one *DiagRA® S version* (with SHORT_NAME "D_PDU_API_BY_RA_CONSULTING_DIAGRA_S") and one *"Normal" version* (with SHORT_NAME "D_PDU_API_BY_RA_CONSULTING"). Both D-PDU APIs are installed and usable by x86 and x64 applications. The x64 version of D-PDU API has suffix "_64_BIT" in the SHORT_NAME, e.g. "D_PDU_API_BY_RA_CONSULTING_64_BIT".

DiagRA® S version of D-PDU API communicates over software way to DiagRA® S and uses PassThru.dll of DiagRA® S.

"Normal" version of D-PDU API allows to use hardware devices providing PassThru API (SAE J2534) over D-PDU API (ISO 22900-2). The main use case is the Hardware Simulation, see Hardware Simulation for more information.



Supported protocols:

- ISO 11898 (Raw CAN)
- ISO 15765-3 / ISO 14229-3 (UDS on CAN / CANFD and any derivatives)
- ISO 14229-5 (UDSonIP and any derivatives)

- ISO 15031-5 / SAE J1979-1 (OBD on CAN)
- ISO 14230 (KWP2000 on CAN)

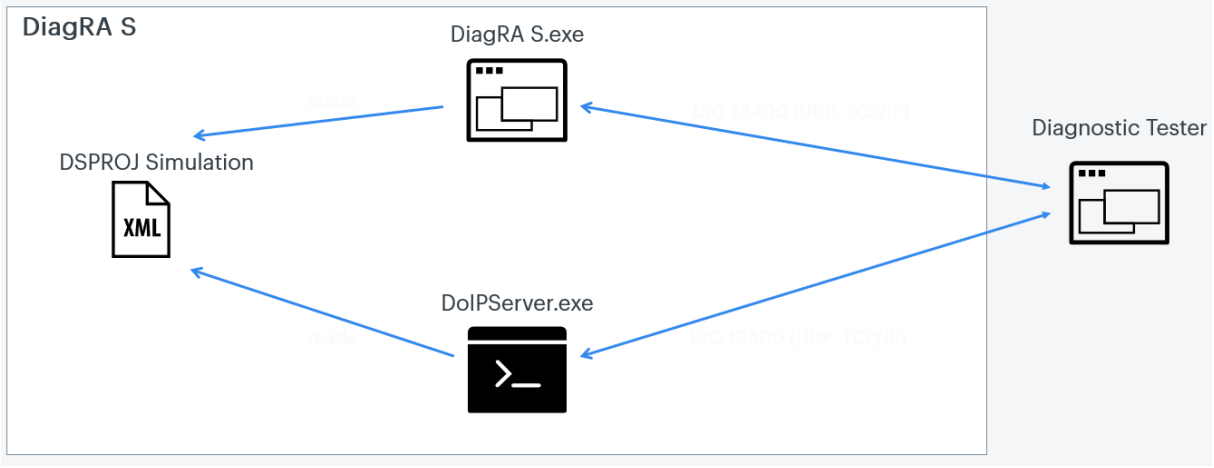
Again, support of ISO 11898 (Raw CAN) means that any supported CAN based protocol (UDS, OBD, SAE J1939 etc.) is also supported in case diagnostic application uses Raw CAN for communication.

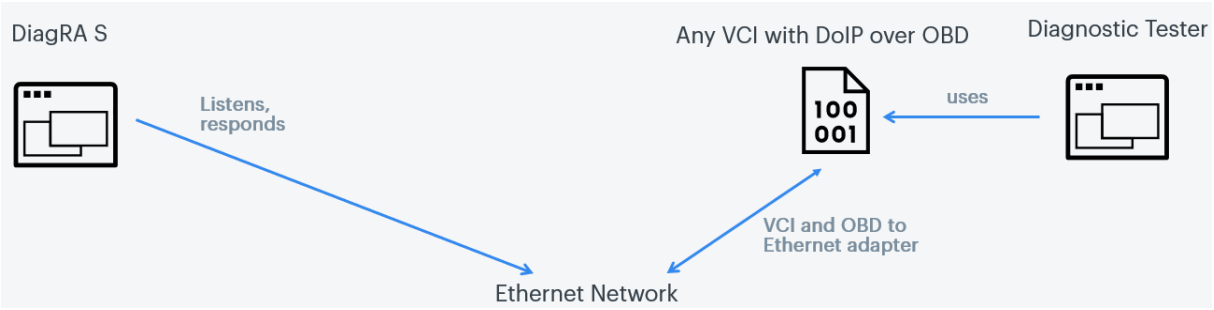
DiagRA® S version of D-PDU API supports both DoIP use cases: with or without VCI. It is also configured to use loopback Ethernet adapter for communication by default (broadcast to 127.255.255.255).
"Normal" version D-PDU API supports only DoIP without VCI use-case.

8 DoIP (ISO 13400 over Ethernet)

DiagRA® S can act as a DoIP ECUs / DoIP Gateway.
In case DoIP identifications are present (DoIP.VIN and DoIP.LA) and DoIP capable ECUs are present (with DoIPLogicalEcuAddress set), DiagRA® S will start monitoring port 13400 for TCP and UDP messages on all available networks of your PC. Monitoring is started automatically upon the loading simulation project file. UDP and TCP Sockets according to ISO 13400 will run locally and the diagnostic requests (ISO 14229, ISO 15765, …) will be processed by DiagRA® S Core.

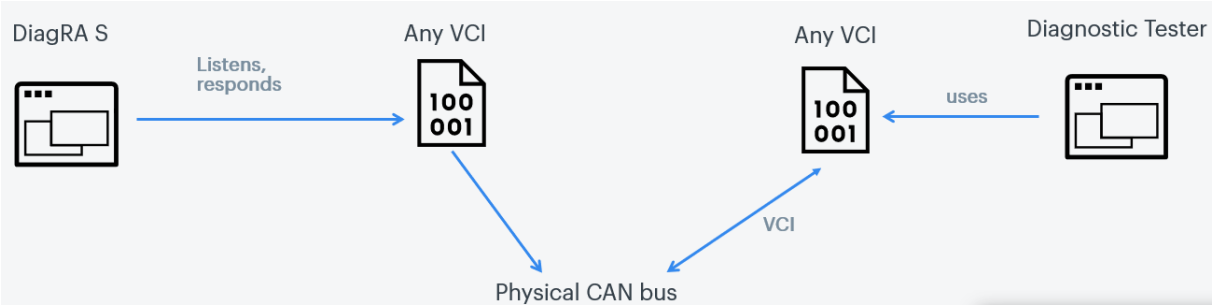
NOTE 1. It depends on the Firewall settings, which networks can be monitored by DiagRA® S. You might need to change your Firewall settings to provide DiagRA® S an access to the required network.
NOTE 2. Diagnostic tools can also use DiagRA® S over Loopback network adapter in the network (e.g. at the network address 127.0.0.1:13400 for IPv4).
NOTE 3. DoIP Simulation can be started without UI using DoIPServer.exe –dsproj "<path-to-dsproj-file>" call.





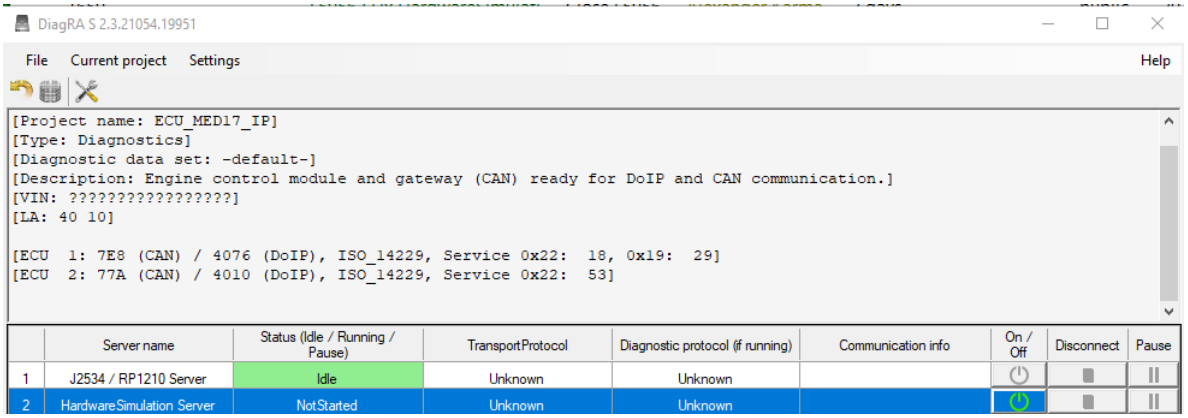
9 Hardware Simulation

You can use DiagRA® S for simulating a vehicle / ECUs on the physical CAN bus.

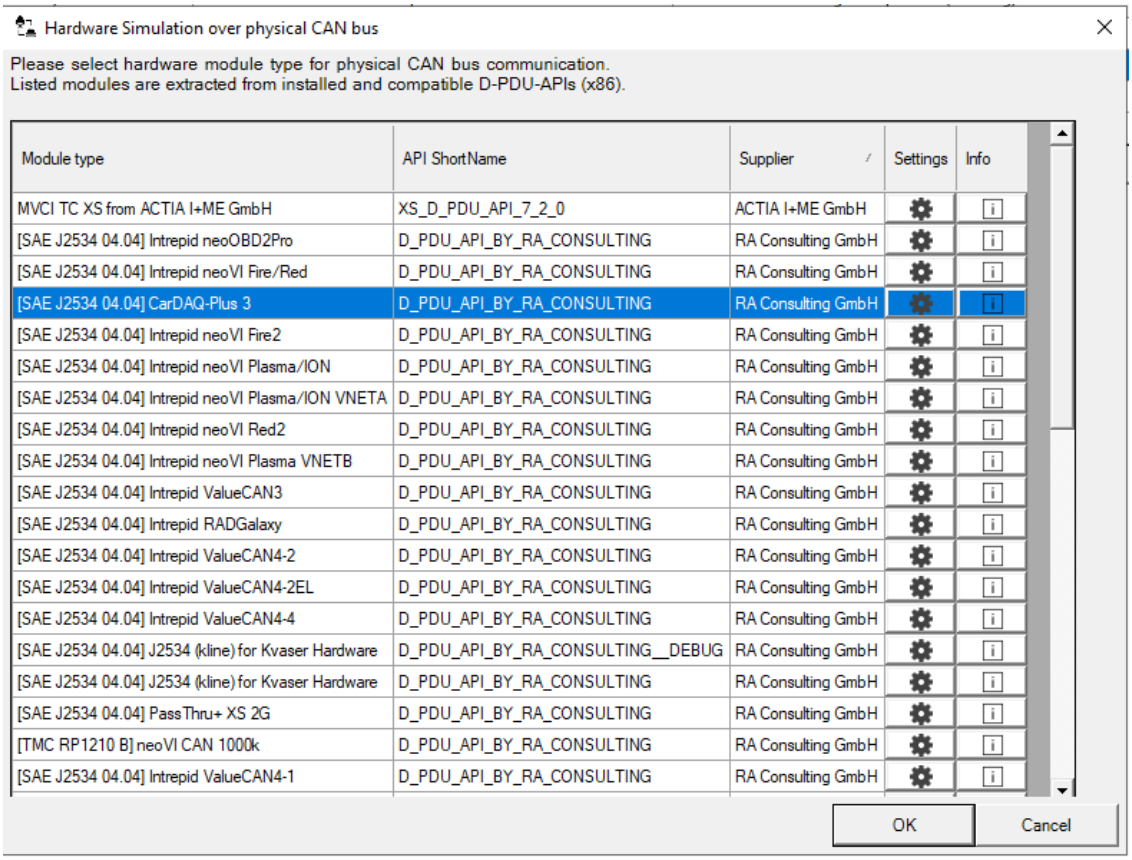


DiagRA® S will monitor the physical CAN bus using a D-PDU-API device and all recognized Requests are processed sending their resulting Responses back to the CAN bus. By loading proper log files, DiagRA® S enables you to simulate a complete vehicle network. **This feature is currently available only for projects of type "Diagnostics".**

To start the hardware simulation, you need to click on the "On/Off" button of HardwareSimulation server:



The following dialog will open:

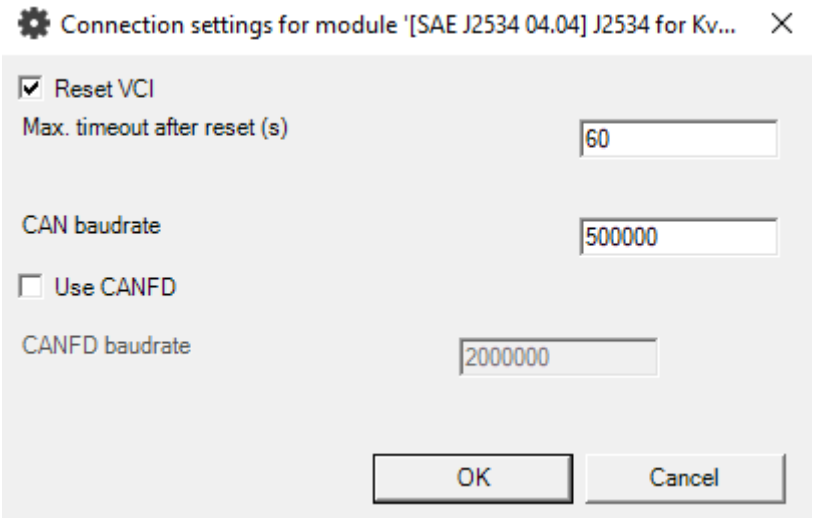


The entries appearing in the table are the module types (MODULE_TYPE) of D-PDU-APIs (ISO 22900-2) installed on your PC and compatible with hardware simulation. Module type is compatible if it supports ISO 11898 protocol (RAW CAN). Currently only x86 version of D-PDU-API devices is supported. The number of entries in the table is highly dependent on D-PDU-APIs (x86) and PassThru 04.04 (SAE J2534 04.04) devices locally installed at your PC. It might happen that you don't have any module type available at all.

NOTE. In case SAE J2534 driver was installed after the installation of DiagRA® S, the corresponding module type will be not available in DiagRA® S. It is recommended to run config tool for the D-PDU API over main menu and refresh list of devices.

9.1 Settings

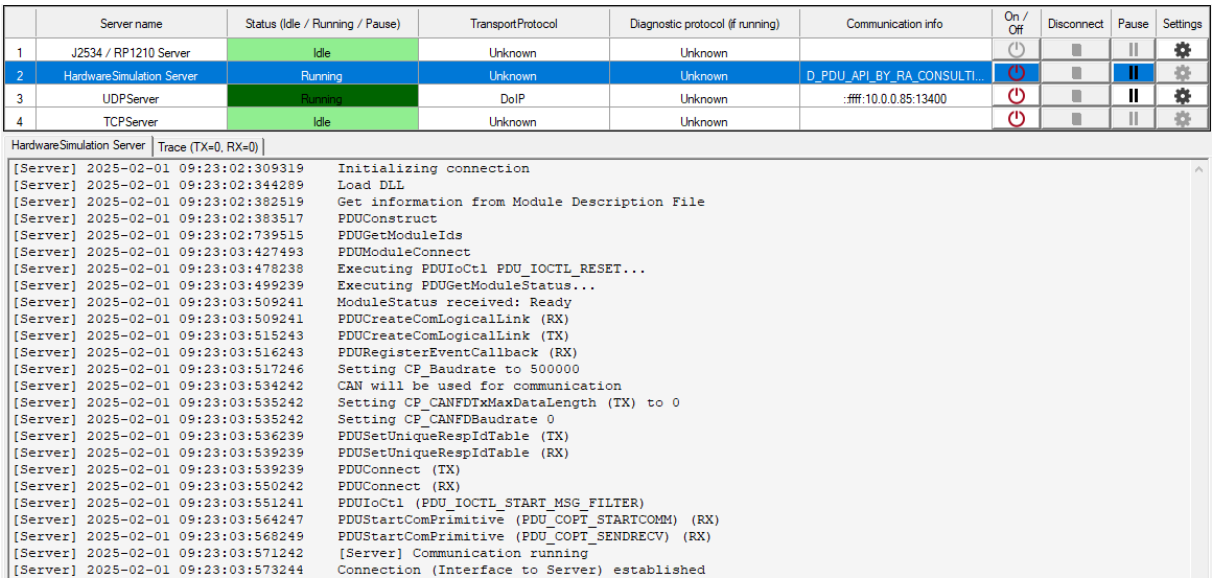
Using the corresponding buttons in the "Settings" column in the dialog it is possible to setup some settings of concrete module types, e.g. CAN Baudrate, CANFD baudrate and also resetting VCI before starting communication. Some VCIs might need long time after reset and will be available again only after significant amount of time. If the default timeout (60 seconds) is not enough, you might need to increase this value.



NOTE: CANFD option is available only in case corresponding CANFD communication parameters are defined in the MDF file of D-PDU-API.

9.2 Star / Stop hardware simulation

By double-clicking on the entry in the table or selecting the entry in the table and clicking “OK” button you will start the monitoring of the CAN bus. In case of successfully connection to the CAN Bus, the HardwareSimulation server will switch to the state “Running” and you will see some output below the servers table:



	Server name	Status (Idle / Running / Pause)	TransportProtocol	Diagnostic protocol (if running)	Communication info	On / Off	Disconnect	Pause	Settings
1	J2534 / RP1210 Server	Idle	Unknown	Unknown					
2	HardwareSimulation Server	Running	Unknown	Unknown	D_PDU_API_BY_RA_CONSULTI...				
3	UDPServer	Running	DoIP	Unknown	:ffff:10.0.0.85:13400				
4	TCPServer	Idle	Unknown	Unknown					

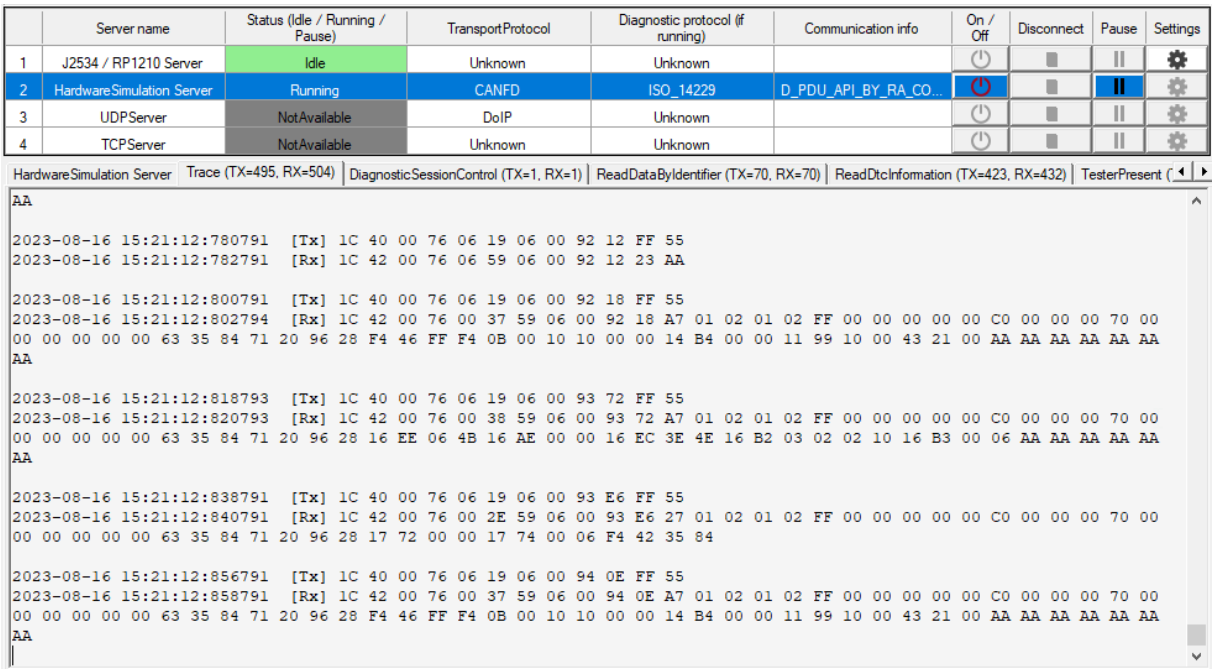
HardwareSimulation Server | Trace (TX=0, RX=0)

```
[Server] 2025-02-01 09:23:02:309319    Initializing connection
[Server] 2025-02-01 09:23:02:344289    Load DLL
[Server] 2025-02-01 09:23:02:382519    Get information from Module Description File
[Server] 2025-02-01 09:23:02:383517    PDUConstruct
[Server] 2025-02-01 09:23:02:739515    PDUGetModuleIds
[Server] 2025-02-01 09:23:03:427493    PDUModuleConnect
[Server] 2025-02-01 09:23:03:478238    Executing PDUIoCtl PDU_IOCTL_RESET...
[Server] 2025-02-01 09:23:03:499239    Executing PDUGetModuleStatus...
[Server] 2025-02-01 09:23:03:509241    ModuleStatus received: Ready
[Server] 2025-02-01 09:23:03:509241    PDUCreateComLogicalLink (RX)
[Server] 2025-02-01 09:23:03:515243    PDUCreateComLogicalLink (TX)
[Server] 2025-02-01 09:23:03:516243    PDURegisterEventCallback (RX)
[Server] 2025-02-01 09:23:03:517246    Setting CP_Baudrate to 500000
[Server] 2025-02-01 09:23:03:534242    CAN will be used for communication
[Server] 2025-02-01 09:23:03:535242    Setting CP_CANFDTxMaxDataLength (TX) to 0
[Server] 2025-02-01 09:23:03:535242    Setting CP_CANFDBaudrate 0
[Server] 2025-02-01 09:23:03:536239    PDUSetUniqueRespIdTable (TX)
[Server] 2025-02-01 09:23:03:539239    PDUSetUniqueRespIdTable (RX)
[Server] 2025-02-01 09:23:03:539239    PDUConnect (TX)
[Server] 2025-02-01 09:23:03:550242    PDUConnect (RX)
[Server] 2025-02-01 09:23:03:551241    PDUIoCtl (PDU_IOCTL_START_MSG_FILTER)
[Server] 2025-02-01 09:23:03:564247    PDUStartComPrimitive (PDU_COPT_STARTCOMM) (RX)
[Server] 2025-02-01 09:23:03:568249    PDUStartComPrimitive (PDU_COPT_SENDRECV) (RX)
[Server] 2025-02-01 09:23:03:571242    [Server] Communication running
[Server] 2025-02-01 09:23:03:573244    Connection (Interface to Server) established
```

In the column “Communication info” the currently selected module type with more specific details will be displayed (use mouse over to display tooltip).

In case any automatically broadcasted messages (e.g. CAN RestBus for diagnostics or broadcast PGNs for SAE J1939) in currently selected simulation project, DiagRA® S immediately starts the transmission to the CAN Bus.

In case some CAN ID and request message on the CAN Bus can be responded by DiagRA® S, DiagRA® S will respond back to the CAN bus. As for other simulation servers, the communication trace will be displayed in the corresponding trace tab below the servers' table:



	Server name	Status (Idle / Running / Pause)	TransportProtocol	Diagnostic protocol (if running)	Communication info	On / Off	Disconnect	Pause	Settings
1	J2534 / RP1210 Server	Idle	Unknown	Unknown					
2	HardwareSimulation Server	Running	CANFD	ISO_14229	D_PDU_API_BY_RA_CO...				
3	UDPServer	NotAvailable	DoIP	Unknown					
4	TCPServer	NotAvailable	Unknown	Unknown					

By clicking the "On/Off" button again, the communication will be stopped.

10 Running simulations from command line

Besides running simulation over DiagRA® S user interface it is possible to run simulation from command line.

10.1 Starting DiagRA S from command line

DiagRA® S User Interface can be either started remotely in case of client trying to establish connection to DiagRA_S.exe over PassThru DLL / RP1210 DLL / RA D-PDU API DLL (see corresponding setting in Chapter General) or from command line by using the call

DiagRA_S.exe *"<path-to-dsproj-file>"*

10.2 Running DoIP simulator DoIPServer.exe

In the installation folder of DiagRA® S you can find DoIPServer.exe, which is a console application for running DoIP diagnostic simulation. As soon as DoIPServer.exe is started, it tries to listen for UDP port 13400 and accepts incoming TCP connections over the same port. However, since the port can be bound only to one application at the same time, you can't run DoIPServer.exe on the same machine in parallel with DiagRA® S or other instances of DoIPServer.exe.

Starting of DoIPServer.exe requires only one argument to be passed:

DoIPServer.exe --dsproj *"<path-to-dsproj-file>"*

There are no further command line arguments supported by DoIPServer.exe yet. This executable doesn't require an installation and can run on any computer. The only pre-requirement is the valid DiagRA® S license installed on this computer and that dependencies of DoIPServer.exe can be resolved (all needed files are located in DiagRA® S installation folder).

NOTE 1. Stopping the simulation can be done by killing the process.
NOTE 2. DoIPServer.exe in delivery is currently compiled to be run in .NET Framework 4.6.1 (i.e. only for Windows). However, there is a .NET 6.0 version of this executable obtainable, which can run also under Linux. Please, contact us if you need to run DoIP simulation on Linux.
NOTE 3: Tool can't be run simultaneously with DiagRA® S, because both tools are trying to acquire the same TCP/IP port).

10.3 Running PassThru simulator PassThruServer.exe

In the installation folder of DiagRA® S you can find PassThruServer.exe, which is a console application for running simulation server part in case your diagnostic application under test ("diagnostic tester") uses PassThru.dll (J2534 / PassThru driver, either x86 or x64) or RA D-PDU API for diagnostic communication to DiagRA® S. As soon as PassThruServer.exe is started, it waits for the connection from PassThru.dll (or RA D-PDU API for DiagRA® S). The connection will be established in case client application ("diagnostic tester") calls PassThruOpen / PDUModuleConnect function. It is usually happened automatically, since PassThru.dll (RA D-PDU API) iterates over all processes and searches for either PassThruServer.exe (first choice) or DiagRA_S.exe (second choice). In case multiple PassThruServer.exe instances are running simultaneously, the choice of PassThruServer.exe (simulator process) will be not deterministic. To overcome this limitation, it is possible to configure PassThru.dll to connect to the concrete process given by process id. In this case the file "PassThruConfig.ini" needs to be present in the same folder as PassThru.dll. This INI file can have following content:

[Connection]
ServerProcessId=<process-id>
[Logging]

LogFilesFolder=<path-to-log-files-folder>

Starting of PassThruServer.exe requires only one argument to be passed:
--dsproj “<path-to-dsproj-file>”

Optionally, following arguments might be additionally provided:
--useDelay <true/false> defines if delay of responses is to be used (default is false)
--randomizeEcuOrder <true/false>. This option makes sense only if **useDelay** is false. If false (default), on processing functional request ECUs are processed in natural order of ECUs (sorted by response address). If true, order of ECUs responding to functional request is randomized.
--msgHandlingOrder <0/1/2> handling order on reaching last response if the list. 0 - repeat last, 1 - stop (no response), 2 - start from the beginning. “0” is default.
--maxCanFrameLength this option influences max CAN FD frame length used for CAN FD communication. Allowed values are: 8, 12, 16, 20, 24, 32, 48 or 64.
--bLogPassThruApiCallsAndLogPassThruComm <0,1,2,3>
“bLogPassThruApiCallsAndLogPassThruComm” defines logging level inside PassThru.dll: 3 – highest level (most logging output), 0 – lowest level (few logging output)
--ignitionSenseState, state of ignition (true / false), true is default
--batteryVoltage – value of battery voltage (READ_VBATT) in mV. 12000 mV is default.
--logConfigFile Path to logging configuration file (XML)

This executable doesn’t require an installation and can run on any computer. The only pre-requirement is the valid DiagRA® S license installed on this computer and that dependencies of PassThruServer.exe and corresponding PassThru.dll can be resolved (all needed files are located in DiagRA® S installation folder).

NOTE 1. Stopping the simulation can be done by killing the process.
NOTE 2. PassThruServer.exe acts as a server part for PassThru DLL (or RA D-PDU API DLL). Both make sense only for Windows. If you are interested to run DiagRA® S based ECU Simulation over SocketCAN on Linux machine, the DiagRA® S CanPlayer is obtainable. In case of interest, please contact us.

10.4 Running hardware simulator HardwareSimulation.exe

In the installation folder of DiagRA® S you can find HardwareSimulation.exe, which is a console application for running hardware simulation server without UI. See chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**
for more details about it.

As soon as HardwareSimulation.exe is started, it connects to the physical CAN bus over physical CAN interface (VCI). For the connection the D-PDU-API or alternatively PassThru API (SAE J2534) using the delivered D-PDU-API wrapper will be used.

Starting of HardwareSimulation.exe requires following arguments to be passed:
--dsproj “<path-to-dsproj-file>”
--pduApiShortName “<PDU_API_SHORT_NAME>” (SHORT_NAME of D-PDU-API to be used)
--pduApiModuleTypeShortName “<PDU_API_MODULE_TYPE_SHORT_NAME>” (SHORT_NAME of D-PDU-API MODULE_TYPE to be used)

Optionally, following arguments might be additionally provided:
--rdfFilePath “<path-to-RDF-file>” Path to RDF file with the list of installed D-PDU-APIs. By default, the path is obtained from Windows registry according to specification in ISO 22900-2.
--useDelay <true/false> defines if delay of responses is to be used (default is false)

--randomizeEcuOrder <true/false>. This option makes sense only if **useDelay** is false. If false (default), on processing functional request ECUs are processed in natural order of ECUs (sorted by response address). If true, order of ECUs responding to functional request is randomized.
--msgHandlingOrder <0/1/2> handling order on reaching last response if the list. 0 - repeat last, 1 - stop (no response), 2 - start from the beginning. “0” is default.
--maxCanFrameLength this option influences max CAN FD frame length used for CAN FD communication. Allowed values are: 8, 12, 16, 20, 24, 32, 48 or 64.
--canBaudrate <CP_CanBaudrate> CAN baudrate. Default is 500000 (i.e. 500 Kbps).
--useCanFd <true/false> If true, CANFD will be used for the communication. Default value is false.
--canFdBaudrate <CP_CANFDBaudrate> CANFD data phase rate. Default is 2000000 (i.e. 2 Mbps).
--logConfigFile Path to logging configuration file (XML)

This executable doesn’t require an installation and can run on any computer. The only pre-requirements are

- the valid DiagRA® S license installed on this computer.
- dependencies of PassThruServer.exe can be resolved (all needed files are located in DiagRA® S installation folder)
- he installation of at least one compatible D-PDU-API / PassThru (x86) for physical CAN bus connection on your PC.

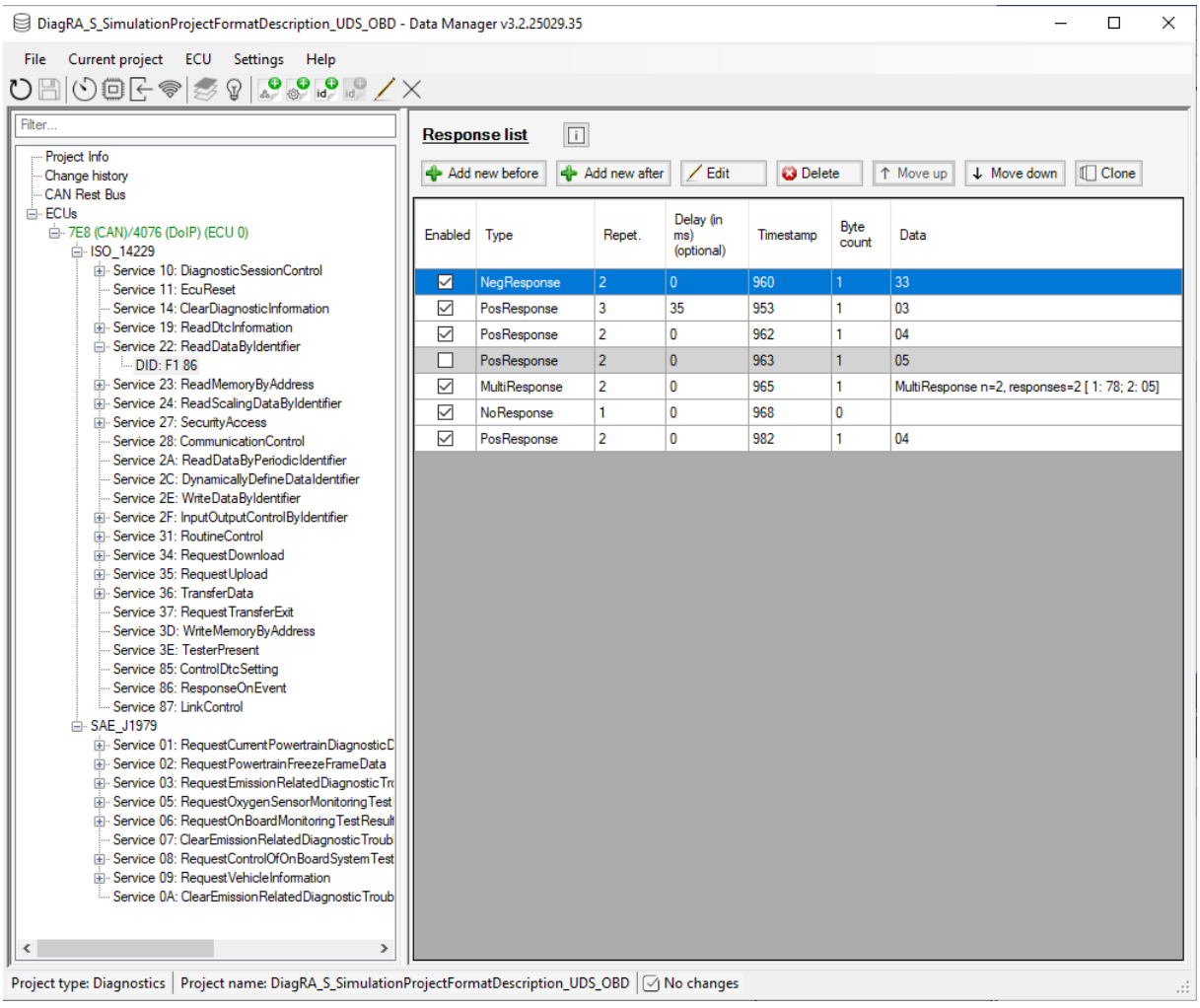
NOTE 1. Stopping the simulation can be done by killing the process.
NOTE 2. HardwareSimulation.exe is currently available only for Windows. Please, contact us if you need to run hardware simulation on Linux.

11 Data Manager (*Professional Version* only)

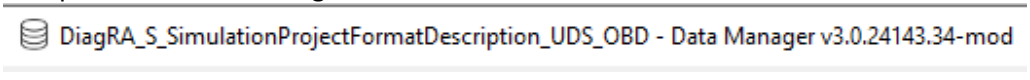
Data Manager is authoring tool for DiagRA® S simulation project files (.dsproj). Data Manager is available only in case you've purchased Professional Version of DiagRA® S.

NOTE 1. Data Manager provides different features for editing .dsproj files of DiagRA® S, but it is still under active development and not all features are available. So, it might happen that for some use-cases some typical actions like editing (response) messages or identifiers (e.g. UDS Data Identifiers) are not yet supported.
NOTE 2. Data Manager and all other DiagRA® S tools can load simulation project files in old format but it can only write simulation project file in the newest format. So, in case the simulation project file is changed and saved, the format is changed automatically to the newest. It is also possible to migrate multiple simulation project to a newest version in a batch operation using command line tool delivered with DiagRA® S, called *DsprojConverter.exe*. See chapter DsprojConverter.exe for more details.

11.1 Overview of main window

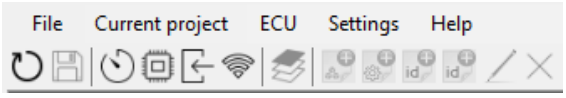


The title of the main window contains the name / relative path of currently selected simulation project and Data Manager version name, e.g. DiagRA S *Professional* v3.0.0.34-RC1, the version number is equal to number of DiagRA® S



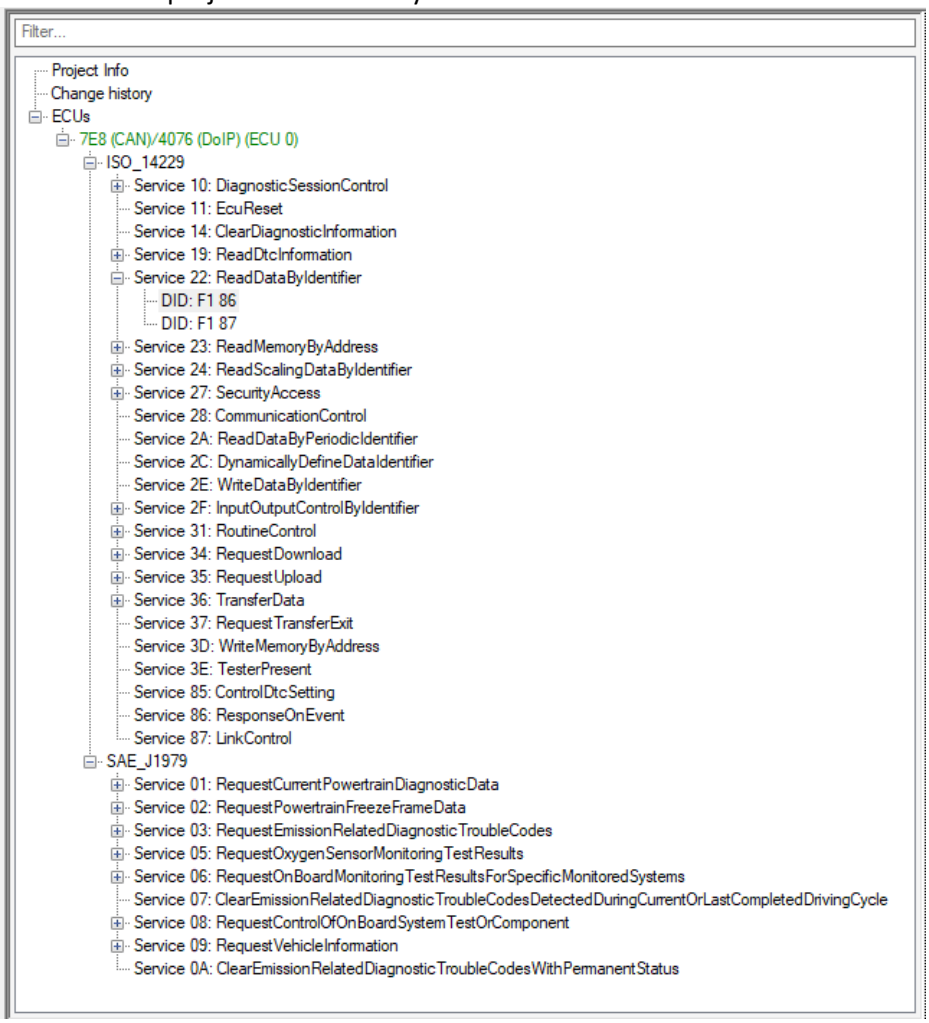
The relative path of simulation project is displayed only in case project is located inside the currently selected project folder.

At the top of the window there is a main menu and the toolbar, which repeats some most frequently used functions from the main menu and provides some additional functions for editing, which might be not available from the main menu.



NOTE. The editing of some node types (for example for KWP2000 and SAE J1939) is currently not supported and will be added in future releases.

In the left middle part of the window (below the toolbar) there is a tree view, which displays the content of the simulation project as a hierarchy:



Above the tree view, there is text box for filtering the nodes of the tree view. Filtering is done based on matching node names and node content.

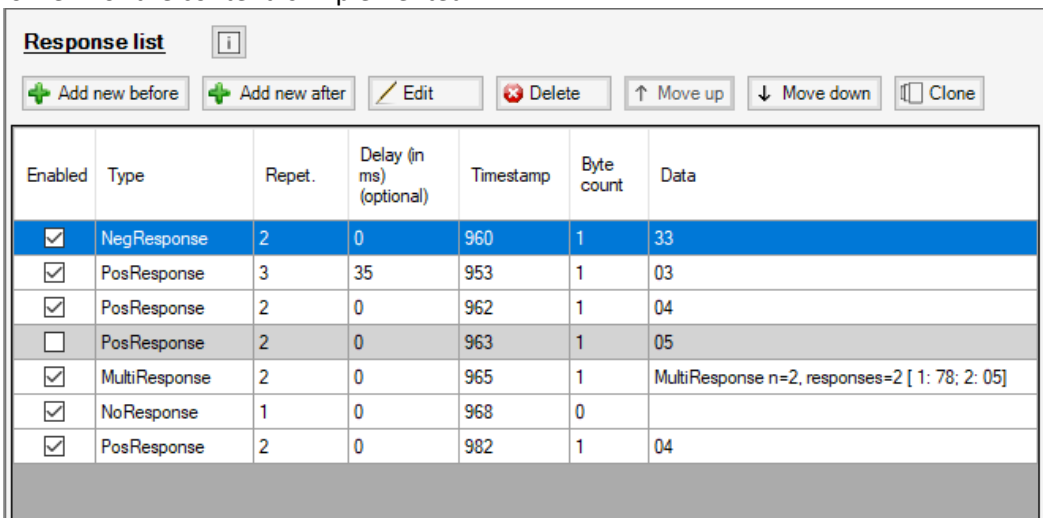
The image above shows the opened Data Manager for a project of type “Diagnostics”. For all project types, the left tree view will contain following nodes:

- “*Project Info*” contains brief overview over project content (the same as in DiagRA® S main window).

- "*CAN Bus*" contains CAN messages to be sent during rest bus simulation (optional, only for projects of type "Diagnostics")
- "*Change history*" contains history of this document.
- "*ECUs*": parent node for all ECU instances.

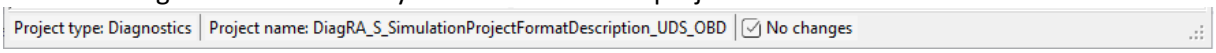
NOTE 1. Change history is not editable over Data Manager.
NOTE 2. Editing CAN Bus is currently not supported. Please, contact us if you are interested in editing of CAN Bus messages in Data Manager UI.

In the right middle part of the window, below the toolbar and to the left from tree view, the content of currently selected node in the tree view is displayed. Some nodes might have no content or there is just no view for the content is implemented.



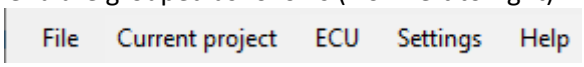
Enabled	Type	Repet.	Delay (in ms) (optional)	Timestamp	Byte count	Data
☑	NegResponse	2	0	960	1	33
☑	PosResponse	3	35	953	1	03
☑	PosResponse	2	0	962	1	04
☐	PosResponse	2	0	963	1	05
☑	MultiResponse	2	0	965	1	MultiResponse n=2, responses=2 [1: 78; 2: 05]
☑	NoResponse	1	0	968	0	
☑	PosResponse	2	0	982	1	04

At the bottom of the main window, there is a status bar, which contains the type, the name and current change status of currently selected simulation project:



11.2 Functions of main menu

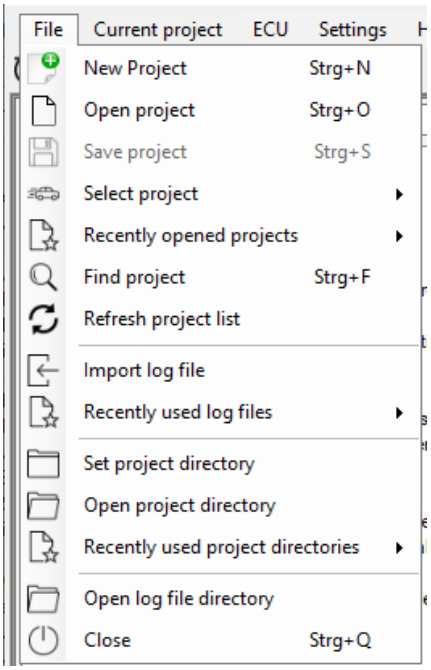
The menu items of main menu are grouped as follows (from left to right):



- **File** (contains file and folder operations)
- **Current project** (contains operations on currently selected simulation project)
- **ECU** (contains operations for selected ECU or creation of new ECU)
- **Settings** (contains settings of Data Manager)
- **Help** (contains help and further documentation)

a. File

Menu **File** contains file and folder operations.

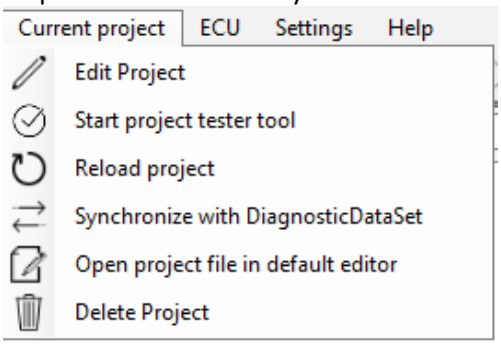


This menu is nearly identical to the menu **File** for corresponding menu of DiagRA® S (see Chapter File for more information), except of following additional functions:

- **Save project** saves current project to the disk.
- **Import log file** starts dialog for import log (trace) files.
- **Recently used log files provides** the overview of recently imported log (trace) files. By clicking the file, the tool "LogViewer" will be started for this file. **WARNING**. Tool LogViewer will be removed in one of the next releases.

b. Current project

Menu **Current project** contains operations on currently selected simulation project.

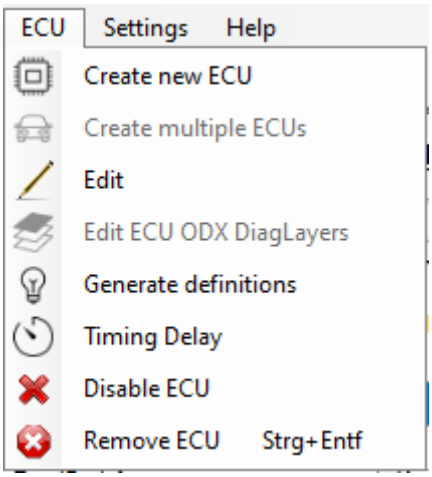


Some functions are the same as for DiagRA® S, except of following additional functions:

- **Edit project** allows editing project description, renaming of project and assigning different diagnostic data set and ODX database
- **Start project tester tool** starts DsprojTester tool for current project, see see Chapter Project Tester tool (Professional Version only) for more information.
- **Synchronize with DiagnosticDataSet** starts synchronization of ECU IDs (CAN / CANFD / DoIP) from current project with selected ODX database or with configured adw.xml file of DiagRA® D diagnostic tool. It is useful in case simulation project is created for DoIP data, but this project needs to be used for CAN communication as well (or vice versa, simulation project is created for CAN data, but it is needed to be used for DoIP communication as well)
- **Delete Project** will delete current project and select previously selected project.

c. ECU

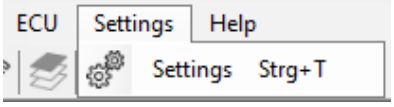
Menu **ECU** provides editing functions for ECUs:



- **Create new ECU** displays dialog for creation of new ECU.
  NOTE. If ODX database is configured for project, it is possible to select ECU from the list of ECUs defined in ODX database.
- **Create multiples ECUs** displays dialog for creation of multiples ECUs in one action.
  NOTE. This option is available only if ODX database is configured for project.
- **Edit ECU properties** displays dialog for editing main ECU properties, e.g. CAN Response CAN ID, DoIP LogicalEcuAddress, Main Transport Protocol etc.
- **Edit ECU ODX DiagLayers** displays dialog for editing handling of ECU DiagLayer detection based on ODX Database: it can be chosen between automatic detection and manually created list of ODX DiagLayers.
- **Generate definitions** allows generation of UDS and OBD data for diagnostic projects based on configured ODX DiagLayers or OBD ODX databases delivered with DiagRA® S, see Generation of data for more information.
- **Timing delay** displays dialog for setting up ECU-wide delay of response messages (added to each individual delay of response message).
- **Enable / Disable** allows to enable / disable ECU. If ECU is disabled, it will not reply to requests, even if it is present in the model.
- **Remove ECU** removes currently selected ECU.

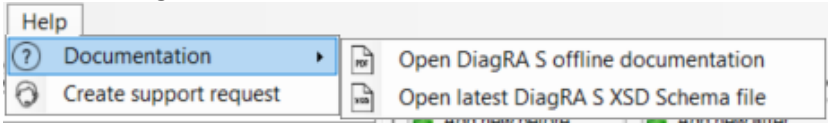
d. Settings

Menu **Settings** calls Data Manager **Settings** dialog, see Chapter Settings for more information.



e. Help

Menu **Help** contains following menu items:



- **Documentation → Open DiagRA S offline documentation** opens this document.
- **Documentation → Open latest DiagRA S XSD Schema file** opens the latest XSD schema file (currently DiagRA_S_Project_3.0.0.xsd) in the tool associated in Windows with XSD files (e.g. XML Spy, Oxygen, Notepad++ etc.)
- **Create support request** starts the tool (cRAshReporter.exe) collecting information about current installation, configuration, log files of Data Manager and current process information

(threads, stack trace etc.) and creating a zip file with file extension .crashreport. You can add description to your report and send .crashreport file to RA Consulting as suggested by the cRAshReporter tool. This is the most comfortable way to send the information to our support.
NOTE 1. No simulation project files or diagnostic log (trace) files are collected by the tool. They might be provided separately, if requested by RA Consulting support.
NOTE 2. The same tool cRAshReporter.exe will be started automatically in case Data Manager has unhandled error, causing the crash of the application.

11.3 Functions of toolbar

The buttons in the toolbar are (from left to right):

- **Reload (reset) Project**
- **Save project**
- **Timing delay** for ECUs
- **Create new ECU**
- **Import log files**
- **Edit DoIP identification data** (VIN and LA of DoIP Edge node)
- **Edit ECU ODX DiagLayers**
- **Add Diagnostic module to ECU**
- **Add Service / Request (for ExtendedRaw) to ECU**
- **Add Identifier to current Service (**e.g. Data Identifier, DTC, etc.)
- **Add Sub-Identifier to current Identifier** (e.g. RoutineOption for Routine)
- **Edit currently selected node**
- **Delete currently selected node**

11.4 Settings

Settings represent the subset of corresponding settings from DiagRA® S.

NOTE. Data Manager uses own settings and never does a synchronization of settings with DiagRA® S.

11.5 ECU node in tree view

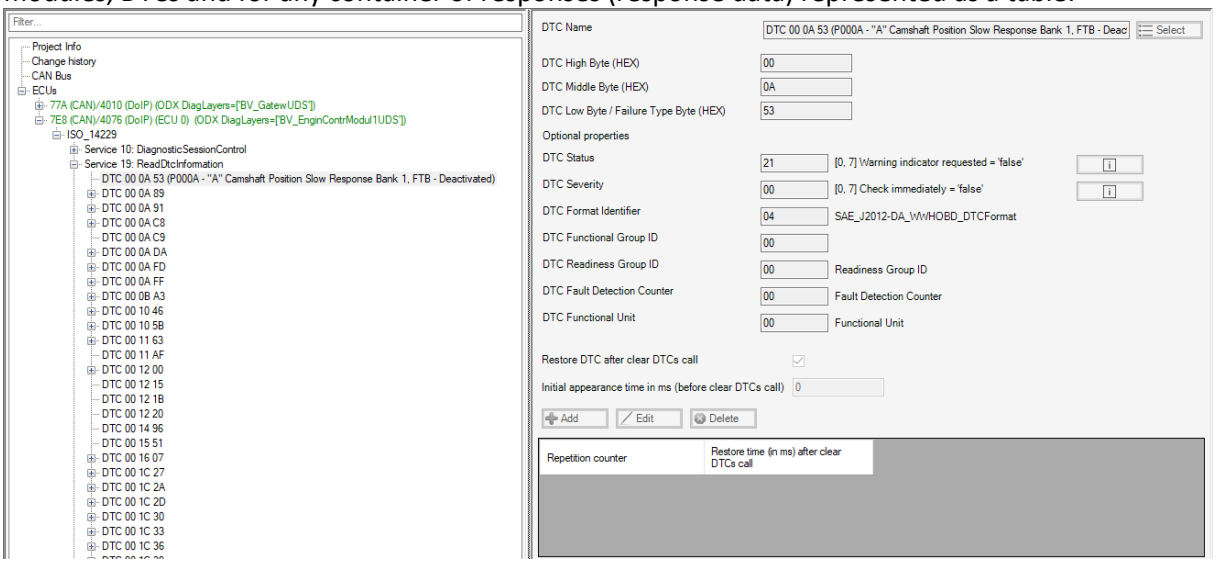
Each ECU node contains identification data for the ECU (CAN ID, DoIP Logical Ecu Address). In case more information is available for this ECU (e.g. matching diagnostic data set or ODX database is selected), the node will also contain corresponding ECU name and list of corresponding DIAG-LAYERs from ODX. For information about using ODX see Chapter Configuration of ODX database.

If you fold the ECU node in the tree view, you can see all content for the selected ECU. The content in a tree view depends on the type of the selected project. For *diagnostic projects*, all *services* will be displayed as nodes and all DTCs and values (e.g. DID) will be child nodes of the proper service nodes.

For *SAE J1939* projects, all Parameter Groups of the selected ECU are listed in the tree view. If there are broadcast or additional SPN information about a PGN, they will be displayed as child nodes of the corresponding PGN.

For *SAE J1699-3, SAE J1699-5 and Zev Compliance Test* projects, all tests are listed, which were made with the selected ECU. If a test failed, its text will be displayed in red, otherwise it will be displayed in green.

For some of selected modes the details can be displayed on the right side, e.g. for diagnostic modules, DTCs and for any container of responses (response data) represented as a table.



11.6 Configuration of ODX database

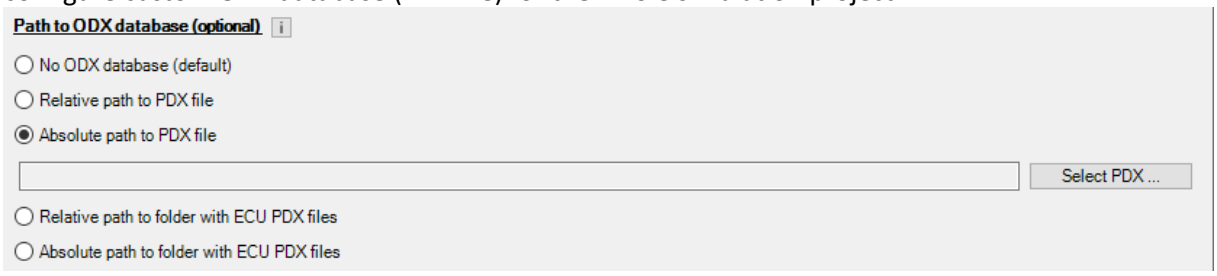
Data Manager can display and interpret diagnostic messages by using custom ODX database(s) (ISO 22901-1:2008). This feature is available is available only for simulation projects of type “Diagnostics”. Data Manager supports ODX databases in commonly used ODX versions: 2.0.1, 2.1.0 and 2.2.0.

It is possible to configure one ODX database for the whole simulation project at project level (vehicle / platform ODX) and/or separate ODX databases for each ECU (ECU specific ODX database). Custom ODX database(s) (PDX file(s)) will be loaded immediately after the simulation project is loaded into Data Manager.
NOTE. Data Manager doesn’t import / store ODX data at any location, it just loads the PDX file(s) into memory and unloads this data if it is unneeded.

If ODX database is configured at both levels (ECU and Project), the one configured for ECU will have a higher priority.

a. Configuring path to ODX database

On creation of new simulation project (**New project**) or over menu **Edit project** it is possible to configure custom ODX database (PDX file) for the whole simulation project.



If *Relative path to PDX file* is selected, Data Manager will resolve the path to PDX file based on the folder configured in *ODX Settings* of Data Manager. This is a preferred option if you want to share your simulation projects with colleagues.
If *Absolute path to PDX file* is selected, the absolute path local to current PC will be stored in the simulation project. Most likely this path will be not resolvable on other PC.
Option *Relative path to folder with ECU PDX files* is needed if ECUs of vehicle are described in separate ECU PDX files, which are located in some folder. Data Manager will assign PDX file to ECU

automatically. The folder path is resolved by Data Manager automatically based on the folder configured in *ODX Settings* of Data Manager. This is a preferred option if you want to share your simulation projects with colleagues.
Option *Absolute path to folder with ECU PDX files* is similar to the option *Relative path to folder with ECU PDX files*, but the path to the folder with ECU PDX files is given as an absolute path. Most likely this path will be not resolvable on other PC.

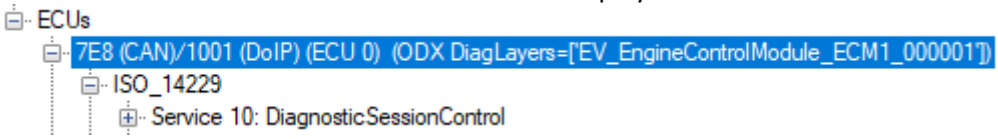
The configuration of *Relative path to PDX file* and *Absolute path to PDX file* is also possible for each diagnostic ECU separately over **Create new ECU** and **Edit ECU properties**.

b. Configuring DiagLayers for ECU

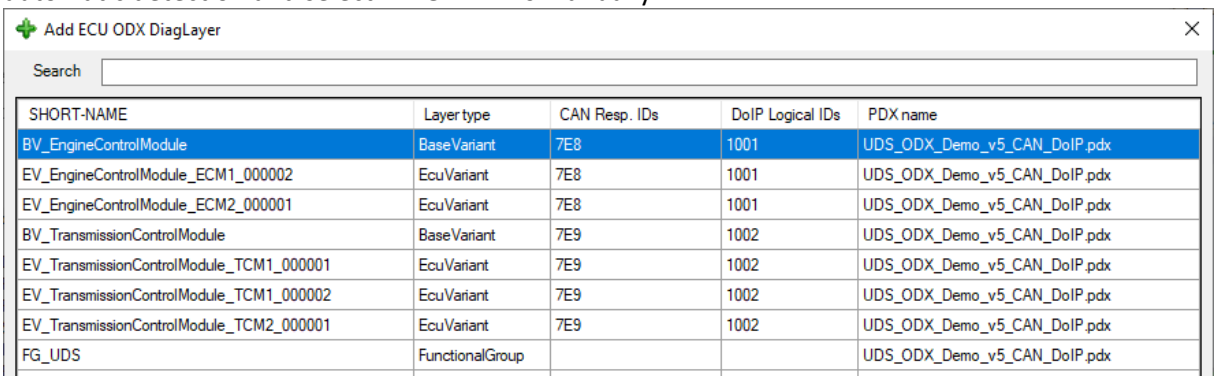
If ODX database(s) are configured, Data Manager will try to automatically find best matching ODX DIAG-LAYER for each ECU. It will be either ECU-VARIANT or BASE-VARIANT.

Matching of BASE-VARIANT is done by matching corresponding COMPARAM values from ODX with ECU identification data (Response CAN ID, DoIP Logical Ecu Address).

ECU-VARIANT is selected based on VariantIdentificationAndSelection algorithm described in ODX standard. Identified DIAG-LAYER's SHORT-NAME will be displayed in the node name in tree view:



By using **ECU → Edit ECU ODX DiagLayers** or corresponding toolbar button, it is possible to turn off automatic detection and select DIAG-LAYERs manually:



SHORT-NAME	Layer type	CAN Resp. IDs	DoIP Logical IDs	PDX name
BV_EngineControlModule	BaseVariant	7E8	1001	UDS_ODX_Demo_v5_CAN_DoIP.pdx
EV_EngineControlModule_ECM1_000002	EcuVariant	7E8	1001	UDS_ODX_Demo_v5_CAN_DoIP.pdx
EV_EngineControlModule_ECM2_000001	EcuVariant	7E8	1001	UDS_ODX_Demo_v5_CAN_DoIP.pdx
BV_TransmissionControlModule	BaseVariant	7E9	1002	UDS_ODX_Demo_v5_CAN_DoIP.pdx
EV_TransmissionControlModule_TCM1_000001	EcuVariant	7E9	1002	UDS_ODX_Demo_v5_CAN_DoIP.pdx
EV_TransmissionControlModule_TCM1_000002	EcuVariant	7E9	1002	UDS_ODX_Demo_v5_CAN_DoIP.pdx
EV_TransmissionControlModule_TCM2_000001	EcuVariant	7E9	1002	UDS_ODX_Demo_v5_CAN_DoIP.pdx
FG_UDS	FunctionalGroup			UDS_ODX_Demo_v5_CAN_DoIP.pdx

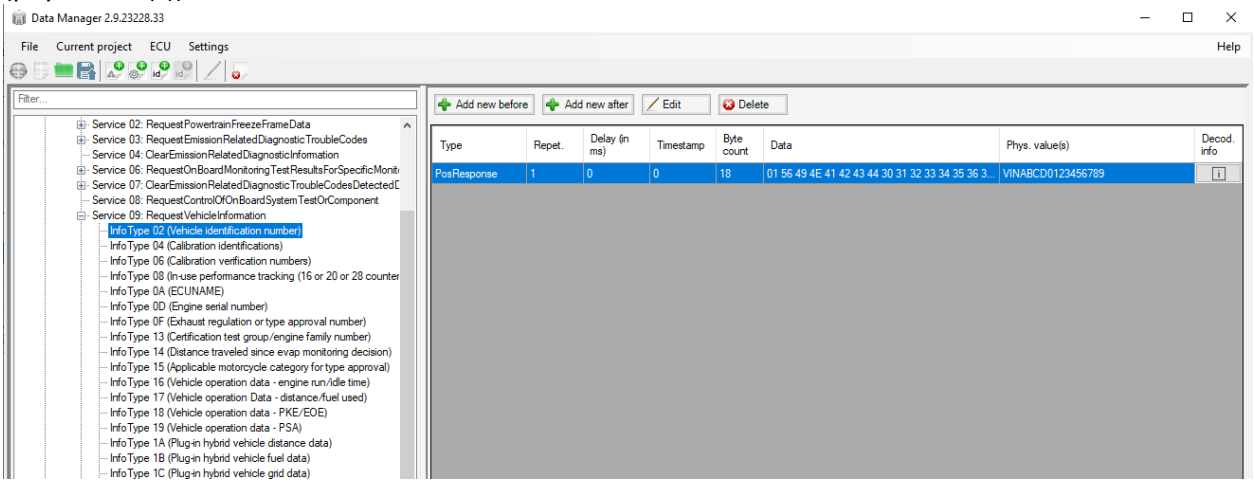
c. OBD classic and OBDonUDS data

Most of OBD (SAE J1979-1 and SAE J1979-2) requests, responses, "identifiers" (like PIDs and DTCs) inside of OBD and UDS diagnostic modules are interpreted (decoded) by Data Manager automatically and don't require any configuration of the custom ODX databases(s). The data definitions for OBD are installed together with DataManager in RA Consulting specific ODX Runtime format. See folder "OdxGraphDatabases" of installation folder. These databases can be also opened and displayed by RA Consulting tool RA® ODX Viewer.

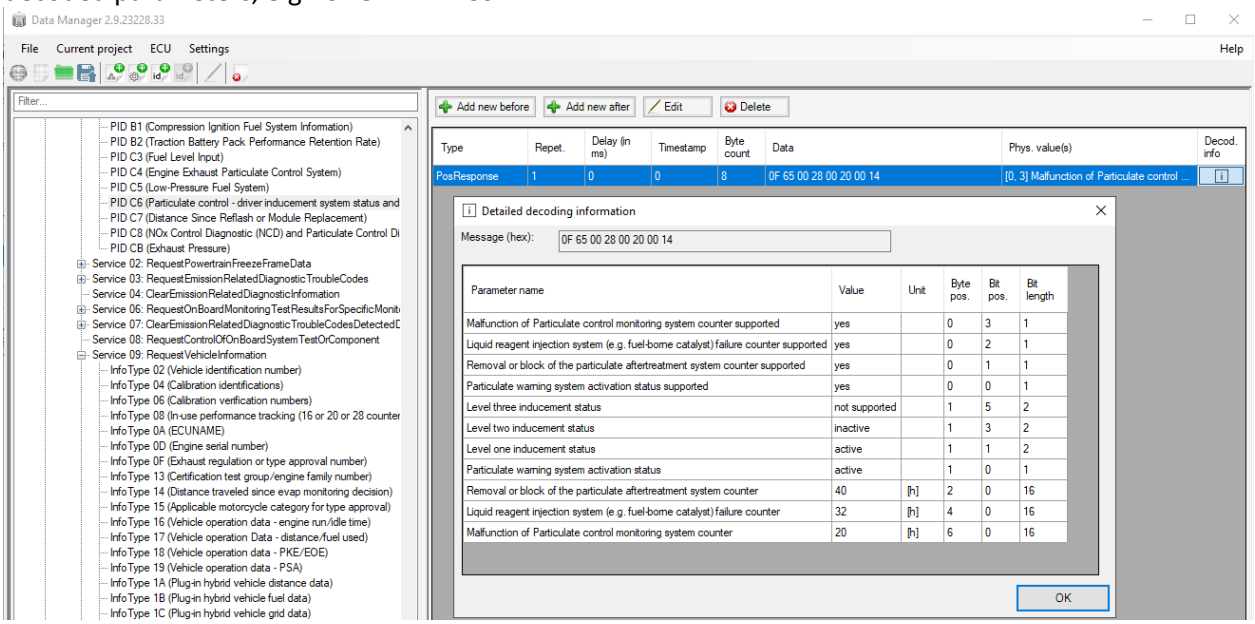
NOTE. Custom ODX databases have higher priority than default OBD databases.

11.7 Interpretation (decoding) of data

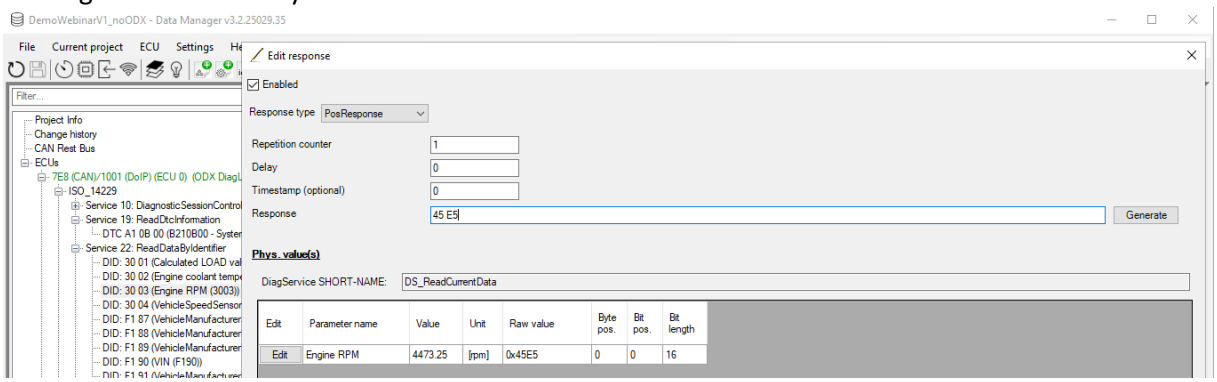
Corresponding physical values (e.g. PID name and values) are displayed in the corresponding column (phys. value(s)).



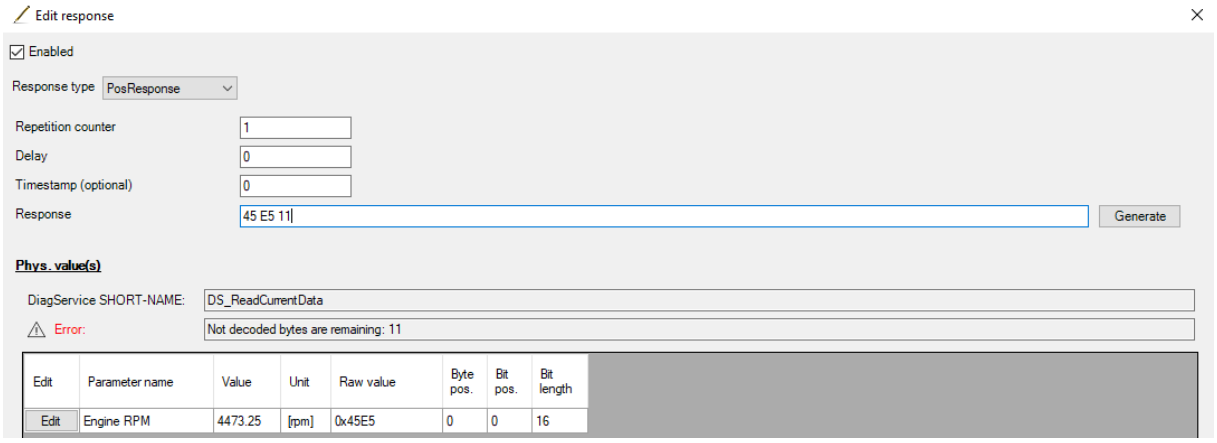
By clicking on the button in the column “Decod. Info”, you will receive detailed information about the decoded parameters, e.g. for OBD PID C6.



In case decoding has some errors, corresponding response will have different color and the button in the Decod. Info” will change to “Warning”. By clicking on corresponding button, more detailed information about decoding errors will be displayed. E.g. in this case for PID 0D the positive response message contains one byte too much:

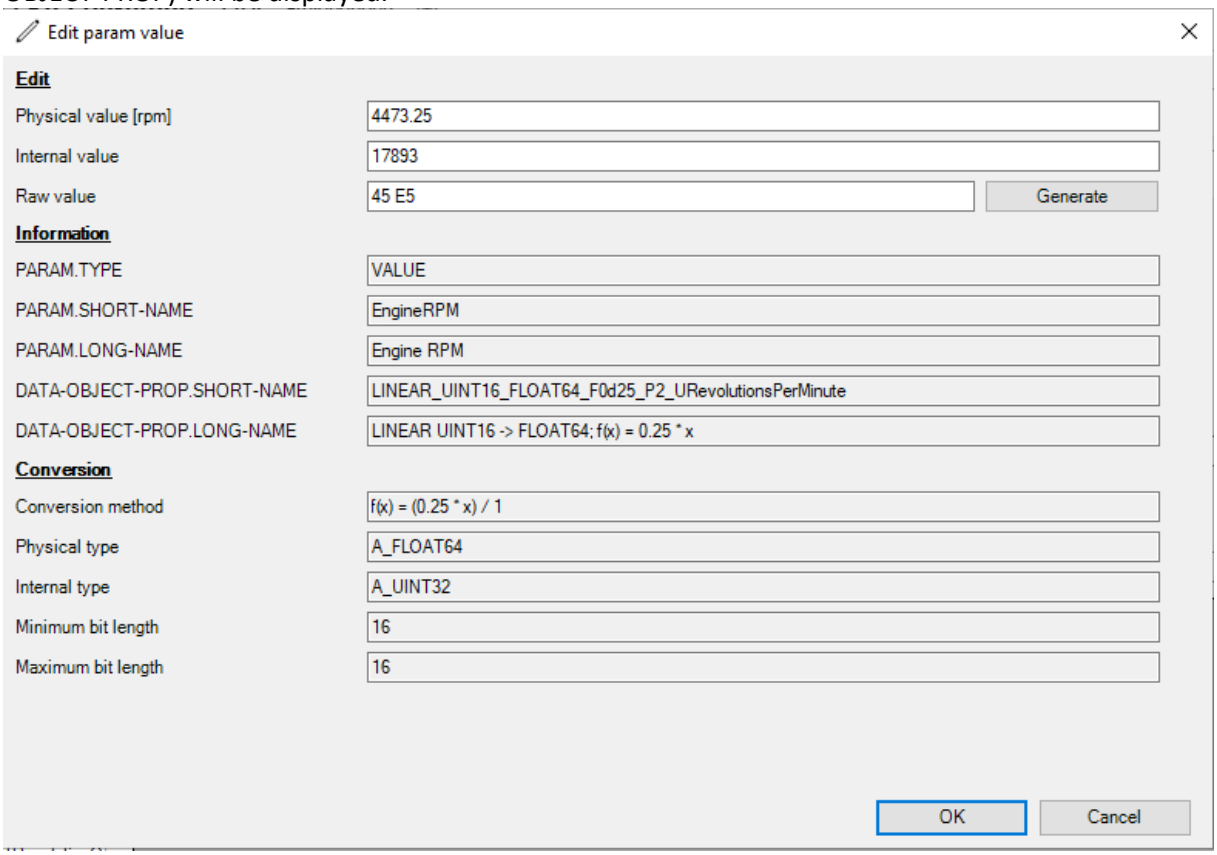


On editing such a response in corresponding “Add/Edit response” dialog, the response will be decoded automatically, while you are typing response message in hex. E.g.



If interpretation is available, it is possible to generate corresponding response message using the “Generate” button on the right side.
NOTE. Generation of response message works in most of cases, but it might fail in certain situations.

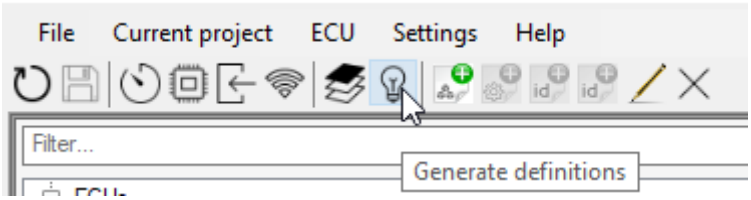
By clicking on “Edit” button in the list of decoded parameters, it is possible to edit the physical value of the parameter. In the edit dialog the information about the conversion method (ODX DATA-OBJECT-PROP) will be displayed.



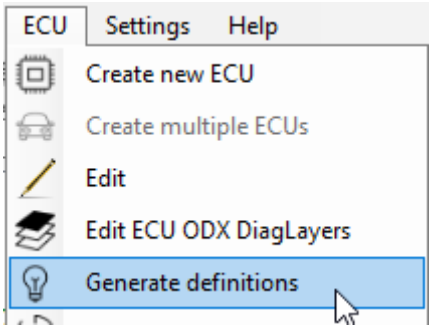
It is possible to edit parameter value as raw value, internal value or physical value. It is also possible to generate the value (button “Generate”). After the “Edit param value” dialog is finished, the corresponding part of full message (several bits / bytes) will be replaced by the new value.

11.8 Generation of data definitions

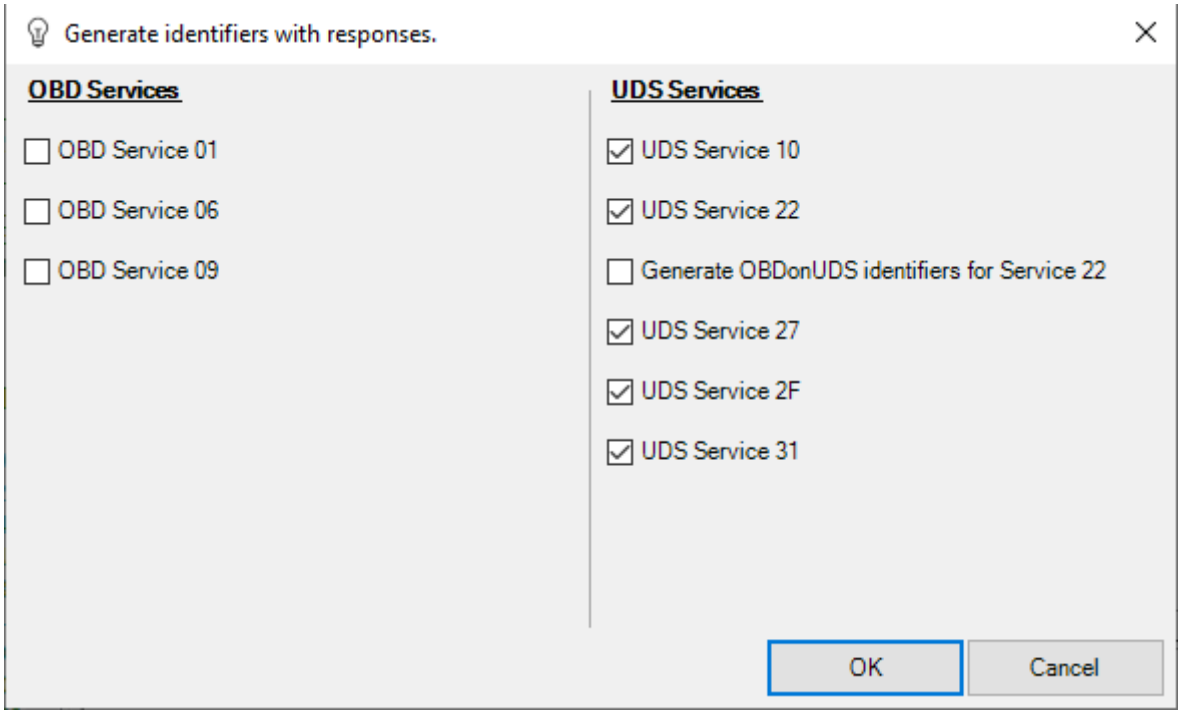
In case ODX Database is configured for an ECU and ODX DiagLayer is selected, it is possible to generate UDS DataIdentifiers, DiagnosticSessions, Routine Identifiers, SecurityAccessTypes and DataInputOutputIdentifiers by using the response generator functionality. This function is accessible either over the toolbar:



or over the main menu **“ECU”**



In the corresponding dialog it is possible to select, for which UDS and OBD Services the data will be generated



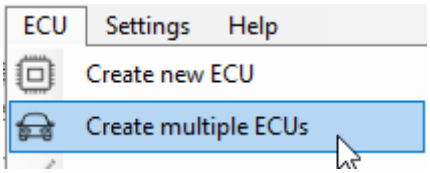
NOTE 1. Generation of OBD and OBDonUDS data doesn’t require configuration of ODX database and ODX DiagLayer for ECU

NOTE 2. Generation of response message works in most of cases, but it might fail in certain situations. At the end of the dialog, you will get the report of generation failures and partly generated messages.

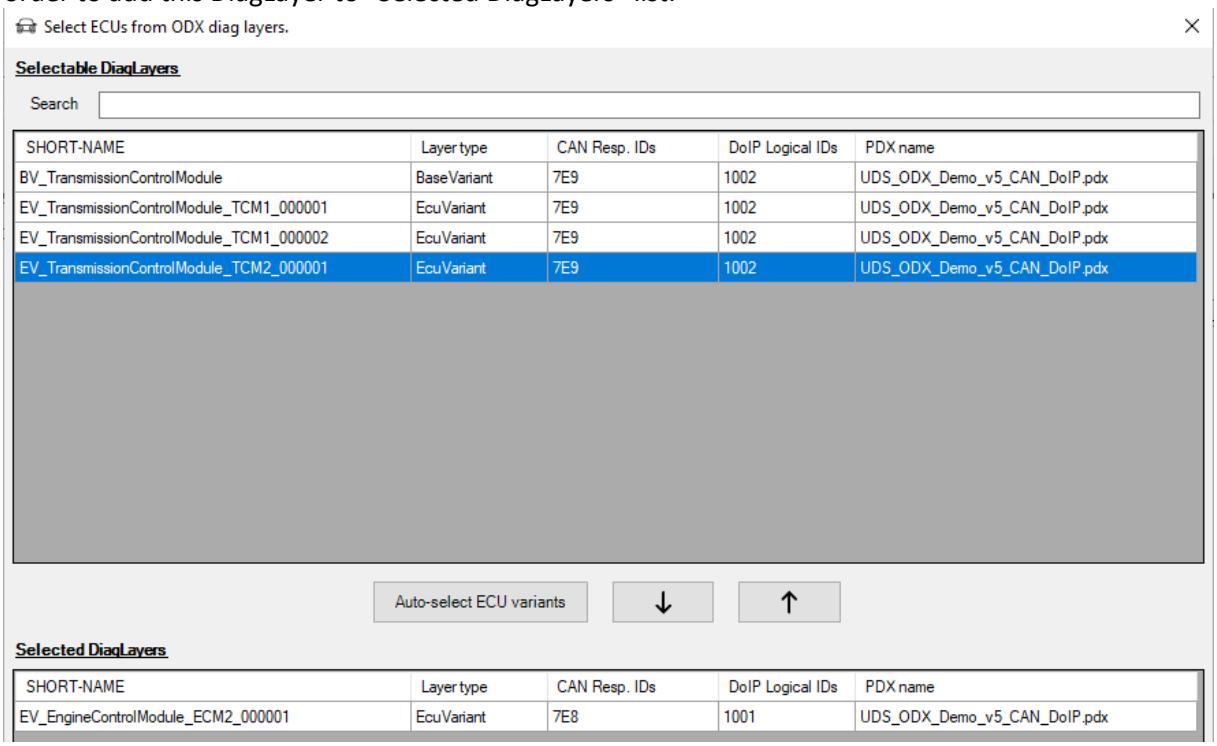
NOTE 3. If ECU-VARIANT is selected, for DIAG-SERVICEs in ODX used for Variant Identification and Selection (VIS) the generation of messages will use the first ECU-VARIANT-PATTERN for generating corresponding responses.

11.9 Creation of multiple ECUs

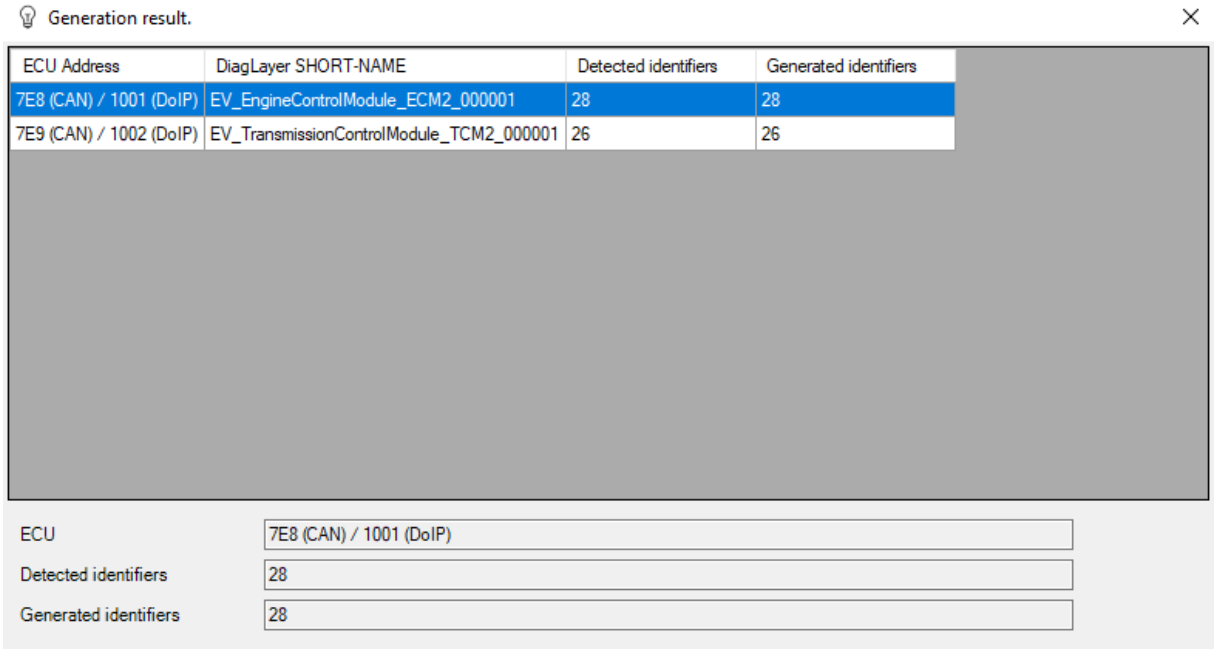
In case ODX Database is configured and in ODX database there are any ECUs available, which might be added to the current simulation project, you can add multiple ECUs to the project in one action. This function is accessible either over the main menu **"ECU"**:



In the corresponding dialog it is possible to select DiagLayers, which will be used for creation of ECUs. Use double-click on the row in the list of "Selectable DiagLayers" or corresponding "Move" button in order to add this DiagLayer to "Selected DiagLayers" list.



After pressing "OK" button in the dialog, you should add generation of data definitions. This is the same action and configuration as in chapter Generation of data definitions, but applied for all selected ECUs (DiagLayers). You have select **at least one of** the options for generation, otherwise the creation of ECUs will be cancelled. After this setup you will see the result of the generation:



11.10 Project editing

If you decide to edit project files manually i.e. without using import functionality, Data Manager brings several helpful functions:

- Creation, editing and deleting of ECUs
- Creation, editing and deleting of diagnostic modules
- Creation, editing and deleting of data specific for diagnostic modules, e.g. requests, diagnostic services, data identifiers, DTCs and etc.
- Creation, editing and deleting response definitions

What is a diagnostic module?
Diagnostic module contains data definitions used for simulation of diagnostic protocol.

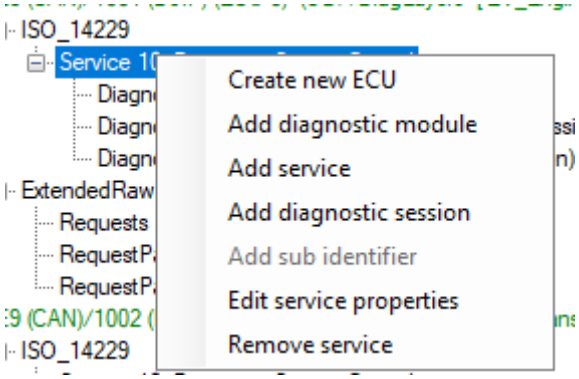
Extended Raw diagnostic module is a most primitive one and can be used for simulation of any diagnostic protocol. It defines full request-response pairs for diagnostic communication. It is quite simple to define, but it requires that all pairs of request-response (or request pattern with response) are defined in the diagnostic module. Every request not matching definitions of **Extended Raw** diagnostic module will be not responded by this module. It is important to know that "*Requests*" from this diagnostic module has a priority over other diagnostic modules since it is requested first on incoming diagnostic request. "*RequestPatterns*" defined in **Extended Raw** diagnostic module are processed as last after all diagnostic modules if no exact match for request was found.

Other diagnostic modules provide more complex data definitions based on corresponding diagnostic standards. I.e. **ISO 14229** diagnostic module contains data definitions for UDS communication, **SAE J1979** diagnostic module contains data definitions for emission-related diagnostics (OBD communication) and **KWP 2000** diagnostic module contains data definitions for KWP 2000 communication. The data definitions are structured by diagnostic services and their subfunctions / data identifiers. Definition of diagnostic data in these diagnostic modules allows to reduce data amount in a project file and allows to simulate some protocol specific business logic but DiagRA® S itself without requiring defining all request-response pairs in **Extended Raw** module. For example, for UDS following business logic is supported: diagnostic session handling (some services can be executed only after diagnostic session is set to something else as "default"), security access handling (successful SecurityAccess (0x27) execution is pre-requirement for execution of services 0x34, 0x35,

0x36), multi-DID request handling (for ReadDataByIdentifier (0x22) or support for all ReadDtcInformation (0x19) subfunctions based on the unified model for DTC.

Creation of diagnostic modules and corresponding data definitions is available over tool bar (see buttons with “+” symbol)

or other context menu for the element in the tree view on the left:

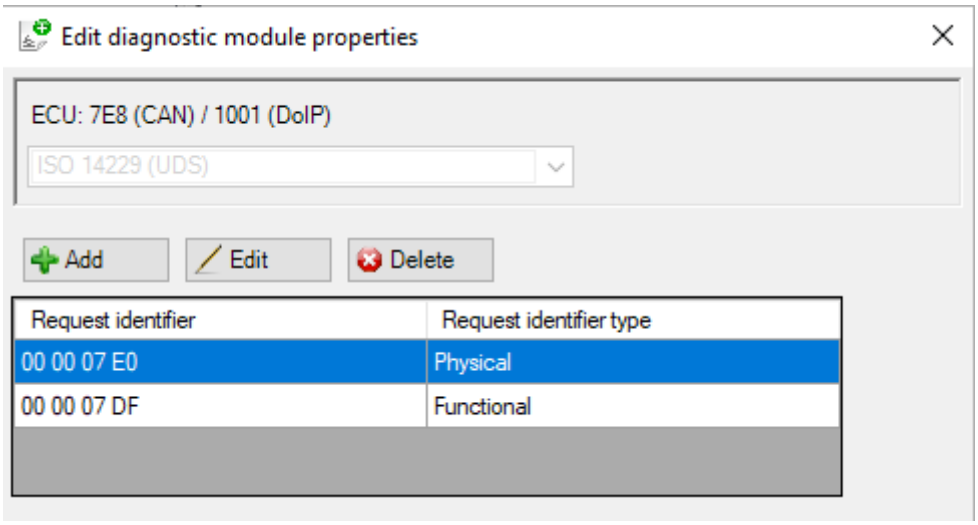


Toolbar buttons and context menu items are dependent on currently selected node in a tree on the left side. Buttons are allowing (from left to right):

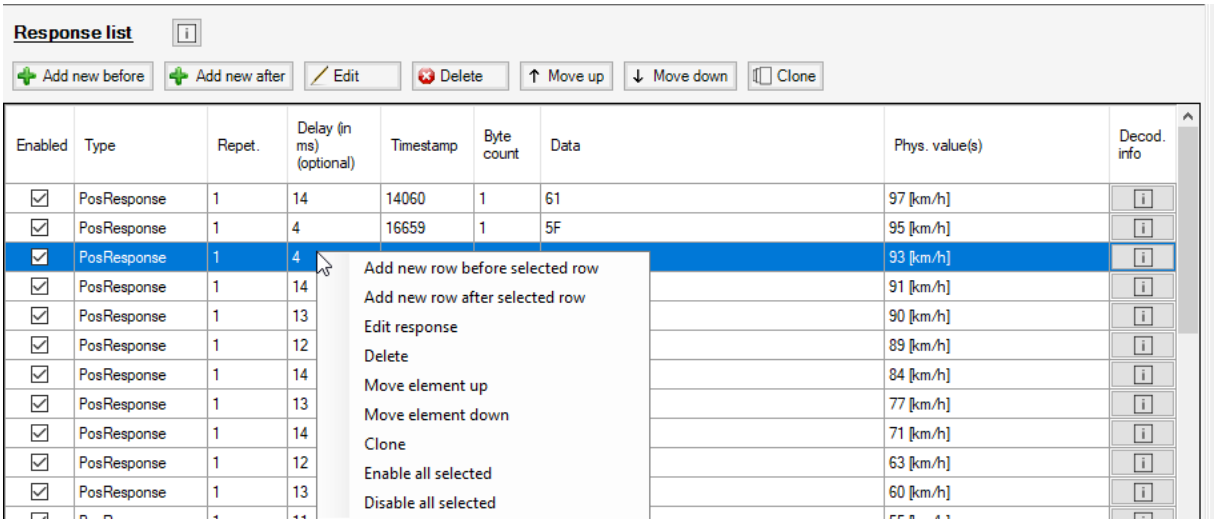
- add diagnostic module
- add diagnostic service / request / request pattern
- add identifier (UDS data identifier, UDS DTC etc.)
- add “sub”-identifier (option record for UDS routine identifier, extended record number for UDS DTC etc.)
- edit selected node
- delete selected node (ECU, diagnostic module, diagnostic service, identifier, “sub”-identifier)

NOTE 1. Creation of all types of diagnostic modules (**ISO 14229**, **SAE J1979**, **KWP 2000**, **Extended Raw**), Editing of data definitions (services, requests, responses) is available for all diagnostic modules except of **KWP 2000. KWP 2000** editing support in Data Manager will be implemented only in case it is explicitly requested by customers. To fill the data in such diagnostic modules, use log file import of manual editing of dsproj files (based on corresponding XSD schema definition and examples delivered together with DiagRA® S).

NOTE 2. For diagnostic modules **UDS** and **ExtendedRaw** there is one information required for communication over CAN / CANFD: CAN request identifiers (to be defined on creation of diagnostic modules).



Editing of response data can be done in Data Manager using buttons located above the response data table and context menu for table entries.



If multiple rows in the table are selected, they can be deleted or enabled / disabled in one batch action.

For following cells in the table above it is possible to edit data just in the cell itself (“inline edit”): data bytes (column “Data”), number of repetitions (column “Repet.”), delay (column “Delay”), timestamp (column “Timestamp”).

It is important to note that only the “data record” part needs to be specified in response data table for **ISO 14229 / OBD / KWP2000** diagnostic module excluding Service-ID, data identifier etc. On the other side, the response data table for **Extended Raw** diagnostic module contains full response definitions (including service-ID, data identifier etc.).

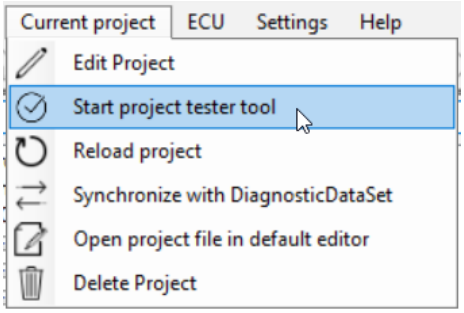
Following four types of responses are supported:

- PosResponse (positive response)
- NegResponse (negative response)
- NoResponse (response is suppressed)
- MultiResponse (multi-part response consisting of multiple response messages sent for one request)

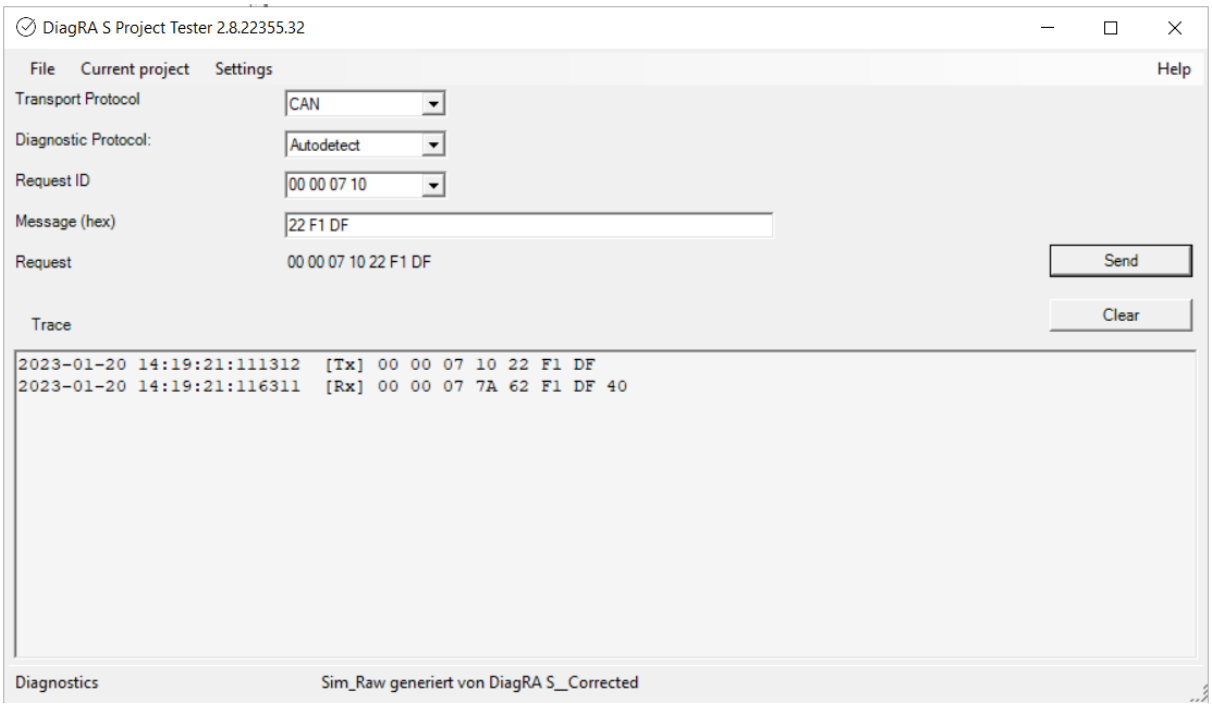
NOTE: UDS Service 0x19 ReadDtcInformation supports only positive responses.

12 Project Tester tool (*Professional Version* only)

Data Manager is delivered with a tool called Project Tester ("DsprojTester.exe"), which can be used for testing diagnostic communication based on this project. This tool can be started from Data Manager over menu "Current project":



If Project Tester is started, you will be able to select transport and diagnostic protocol, to select or type your own request ID (Request CAN ID for CAN / CANFD or DoIP Logical ECU Address for DoIP) and type request message. Clicking on "Send" button will send the request to DiagRA® S communication core. Corresponding communication trace (request and responses if any) will appear in the output text area below input fields. Button "Clear" removes the whole content from the trace.



Over menu *Settings* it is possible to set the communication settings of Project Tester tool. Settings are almost identical to the corresponding settings of DiagRA® S.

Known limitations of "Project tester":

- Only diagnostic projects are supported. Loading and trying to test communication for SAE J1939, SAE J1699-3, SAE J1699-5 and Zev Compliance Test project files will have no effect.
- RawCAN protocol is not supported, i.e. it is not possible to test processing of raw can messages in "Project Tester".
- K-LINE based protocols are not supported.

13 Adapt simulation files using CL tools (*Professional Version* only)

Several command line tools for automated processing of .dsproj simulation files are delivered together with DiagRA® S and can be used if you have license for *Professional Version*.

13.1 DsprojCreator.exe

Starting from v2.7.0.42 there is a possibility to create project files using command line interface of DsprojCreator.exe. This tool can create simulation file (.dsproj) using communication trace files (.log, .asc and etc.).

For obtaining supported command line parameters start executable without any arguments.

13.2 DsprojConverter.exe

This tool can convert single or multiple simulation files (.dsproj) to a current schema version. It can be also used for converting OBD classic simulation data to OBDonUDS data. For obtaining supported command line parameters start executable without any arguments.

13.3 DsprojMerger.exe

This tool can merge multiple simulation files in one file. For obtaining supported command line parameters start executable without any arguments.

NOTE: currently only merging of OBD or OBDonUDS data is supported.

14 Appendix: Diagnostic Devices

14.1 SAE J2534 (PassThru)

a. Supported Functions

PassThru Functions	Status
PassThruOpen	✔
PassThruClose	✔
PassThruConnect	✔
PassThruDisconnect	✔
PassThruReadMsgs	✔
PassThruWriteMsgs	✔
PassThruStartPeriodicMsg	✔
PassThruStopPeriodicMsg	✔
PassThruStartMsgFilter	✔
PassThruStopMsgFilter	✔
PassThruSetProgrammingVoltage	✔
PassThruReadVersion	✔
PassThruGetLastError	✔
PassThruIoctl	✔

b. Supported Configuration

PassThru IOCTLs	Status
GET_CONFIG	✔
SET_CONFIG	✔
GET_DEVICE_INFO (SAE J2534-2)	✔
FIVE_BAUD_INIT	✔
FAST_INIT	✔
CLEAR_TX_BUFFER	✔
CLEAR_RX_BUFFER	✔
CLEAR_PERIODIC_MSGS	✖
CLEAR_MSG_FILTERS	✔
CLEAR_FUNCT_MSG_LOOKUP_TABLE	✖
ADD_TO_FUNCT_MSG_LOOKUP_TABLE	✖
DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE	✖
READ_PROG_VOLTAGE	✔
READ_VBATT (can be configured in UI)	✔
INGITION_SENSE_STATE (custom, can be configured in UI)	✔

PassThru SET_CONFIG / GET_CONFIG Parameters	Status
DATA_RATE	✔
LOOPBACK	✔
NODE_ADDRESS	✔
NETWORK_LINE	✖
P1_MIN	✔

P1_MAX	✗
P2_MIN	✓
P2_MAX	✓
P3_MIN	✗
P3_MAX	✓
P4_MIN	✗
P4_MAX	✓
W0	✗
W1	✗
W2	✗
W3	✗
W4	✗
W5	✗
TIDLE	✗
TINIL	✗
TWUP	✗
PARITY	✗
BIT_SAMPLE_POINT	✗
SYNC_JUMP_WIDTH	✗
T1_MAX	✗
T2_MAX	✗
T3_MAX	✗
T4_MAX	✗
T5_MAX	✗
ISO15765_BS	✗
ISO15765_STMIN	✗
BS_TX	✗
STMIN_TX	✗
DATA_BITS	✗
FIVE_BAUD_MOD	✗
ISO15765_WFT_MAX	✗

14.2 TMC RP1210 B

a. Supported Functions

RP1210 B Functions	Status
RP1210_ClientConnect	✓
RP1210_ClientDisconnect	✓
RP1210_SendMessage	✓
RP1210_ReadMessage	✓
RP1210_ReadVersion	✓
RP1210_ReadDetailedVersion	✓
RP1210_GetErrorMsg	✓
RP1210_GetLastErrorMsg	✓
RP1210_GetHardwareStatus	✗

RP1210_SendCommand	✔

b. Supported Configuration

RP1210 B Parameters	Status
RP1210_Reset_Device	✘
RP1210_Set_All_Filters_States_to_Pass	✘
RP1210_Set_Message_Filtering_For_J1939	✔
RP1210_Set_Message_Filtering_For_CAN	✔
RP1210_Set_Message_Filtering_For_J1708	✘
RP1210_Set_Message_Filtering_For_J1850	✘
RP1210_Set_Message_Filtering_For_ISO15765	✘
RP1210_Generic_Driver_Command	✘
RP1210_Set_J1708_Mode	✘
RP1210_Echo_Transmitted_Messages	✔
RP1210_Set_All_Filters_States_to_Discard	✔
RP1210_Set_Message_Receive	✘
RP1210_Protect_J1939_Address	✘
RP1210_Set_Broadcast_For_J1708	✘
RP1210_Set_Broadcast_For_CAN	✘
RP1210_Set_Broadcast_For_J1939	✘
RP1210_Set_Broadcast_For_J1850	✘
RP1210_Set_J1708_Filter_Type	✘
RP1210_Set_J1939_Filter_Type	✘
RP1210_Set_CAN_Filter_Type	✘
RP1210_Set_J1939_Interpacket_Time	✘
RP1210_SetMaxErrorMsgSize	✘
RP1210_Disallow_Further_Connections	✘
RP1210_Set_J1850_Filter_Type	✘
RP1210_Release_J1939_Address	✘
RP1210_Set_ISO15765_Filter_Type	✘
RP1210_Set_Broadcast_For_ISO15765	✘
RP1210_Set_ISO15765_Flow_Control	✘
RP1210_Clear_ISO15765_Flow_Control	✘
RP1210_Set_ISO15765_Link_Type	✘
RP1210_Set_J1939_Baud	✘
RP1210_Set_BlockTimeout	✘
RP1210_Set_J1708_Baud	✘