VECTOR

Diagnostics over Internet Protocol (DoIP) in CANoe

Version 1.2
2020-11-25
Application Note AN-IND-1-026

Author Vector Informatik GmbH
Restrictions Public Document
Abstract Overview over the DoIP features of CANoe, including examples and references

Table of Contents

1 Overview

This document summarizes the features of CANoe regarding the Diagnostics over Internet Protocol (DoIP) for CANoe 14 SP2 and later versions. It will step by step explain how to use CANoe as DoIP tester, gateway and simulated ECU. Additionally, instructions are given how to configure CANoe for different use cases.

This scope of this document is neither to explain the DoIP protocol, nor it explains the diagnostics features of CANoe. For further information on those topics, please refer to [1] and [2].

2 Background and Basics

2.1 Ethernet-specific topics

There is a big topology difference between automotive Ethernet and traditional automotive networks like CAN or LIN: While CAN and LIN use a bus topology, i.e. all network participants communicate over the very same physical electrical lines (the “bus”), Ethernet is typically used as a switched network which only allows point-to-point physical connections. This has some implications on the usage of CANoe. For example, when observing communication in a network, it is no more enough to listen to just one bus line – to observe the whole communication, it is necessary to listen to *every* point-to-point connection.

Furthermore, Ethernet offers the possibility to separate the communication on the very same network by so-called virtual LANs (VLAN) for security and/or performance reasons. This is realized on data link layer by adding so-called VLAN tags, therefore this separation can be performed by HW directly in the switch.

Also, from an addressing point of view, things are becoming more complicated with Ethernet: there is not only a physical address (the MAC address), there additionally are different addressing possibilities on the network layer (IPv4/IPv6). Furthermore, those network layer addresses are not necessarily fix – they might be dynamically assigned in different ways (Auto-IP, DHCP). On top of that, in a switched network, a frame is not automatically broadcasted to every network node (like e.g. on CAN). On Ethernet with IP as network protocol, this is done using special broadcast or multicast addresses.

But IP addresses alone are not enough to allow communication on application layer between two communication participants. The IP stack out-of-the-box supports a connection-oriented protocol (Transmission Control Protocol, TCP) and a fast connection-less protocol (User Datagram Protocol, UDP) on transport layer.

These two protocols introduce a flood of commonly used protocols on top of themselves, as they additionally provide a so-called source and destination “port” which is used to identify the respective session-, presentation- or application layer protocol to be used for a specific communication sequence. Hence, the pair “IP address plus port” is often referred to as “socket”, describing an important aspect of the communication between two participants in an IP network. Many ports are already predefined by commonly used protocols like File Transfer Protocol (FTP), Post Office Protocol (POP3), or Hypertext Transport Protocol (HTTP), to name just a few. Also, for DoIP, a specific port is reserved: 13400, the same number as the number of the ISO standard for DoIP [1].

Based on TCP, even encryption is supported using the Transport Layer Security (TLS) protocol – which is e.g. commonly used by web browsers when accessing web sites via secure HTTP (HTTPS).

Despite the manifold addressing possibilities of the Ethernet data link layer combined with IP, an additional addressing mechanism is needed for automotive purposes: as non-Ethernet ECUs inside the car do not have IP addresses, additionally so-called “DoIP logical addresses” are necessary to reach all in-vehicle ECUs by a diagnostic tester. This is the main reason why the DoIP protocol is needed – it allows to address non-Ethernet ECUs behind a DoIP gateway in the car.

2.2 History of DoIP and its support in CANoe

DoIP support in CANoe started very early (CANoe 7.1 SP4), when no Option Ethernet and no Vector Ethernet network interfaces were available and even the DoIP standard (ISO 13400-2) was only available as a draft version. It was continuously improved and extended, today supporting the latest DoIP standard version (ISO 13400-2:2019) including encryption via TLS. Meanwhile, CANoe.Ethernet in combination with the Vector VN56xx Ethernet HW interface family offers powerful features to analyze DoIP communication and act as a DoIP tester or an ECU simulating diagnostics functionality via DoIP.

It is important to keep in mind that the ISO 13400-2:2012 (first edition) only specified the communication between a diagnostic tester and a DoIP gateway (the so-called DoIP edge node) in a vehicle. It did not specify car-internal diagnostic communication via Ethernet. This led to several different implementations like Diagnostics over AUTOSAR Socket Adaptor (DoSoAd) and OEM-specific derivates of DoIP which were tailored to the OEM's needs for car-internal usage. Meanwhile CANoe supports DoSoAd as well as HSFZ, the OEM-specific predecessor of DoIP. The protocol and its version to be used by CANoe can be selected in the Diagnostics/ISO TP configuration dialog of CANoe (Figure 1).

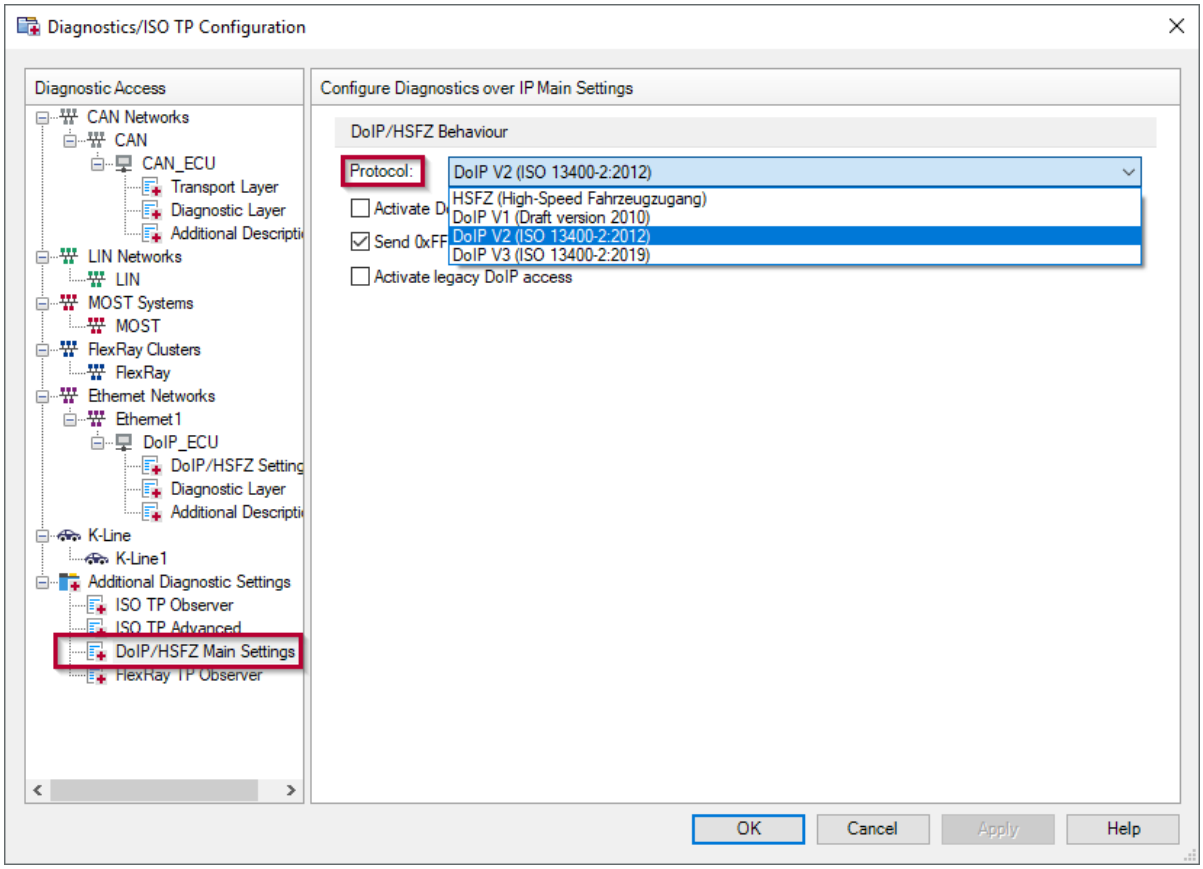


Figure 1: Selecting the DoIP protocol version

2.3 Architectural aspects of CANoe.Ethernet and the operating system

To configure CANoe correctly for DoIP communication, it is helpful to understand CANoe.Ethernet's basic architectural concepts – especially regarding the interfaces to the operating system's TCP/IP stack as well as to the global/shared CANoe TCP/IP Stack and the individual TCP/IP stacks of CANoe (Figure 2).

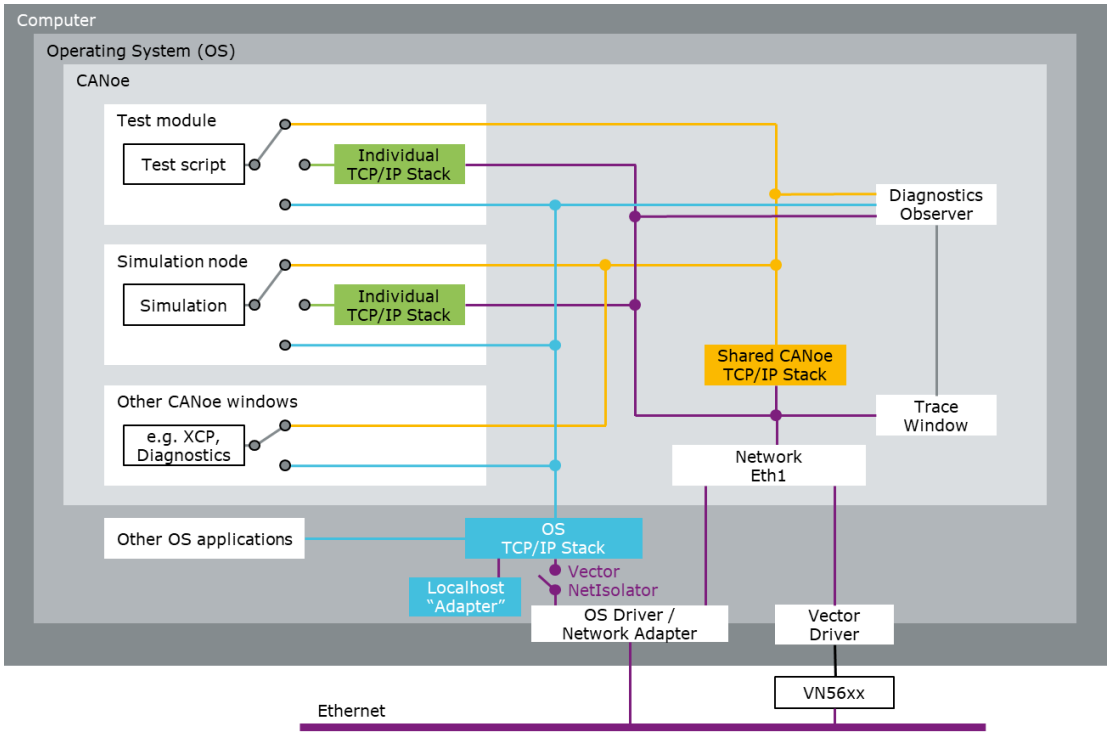


Figure 2: System diagram

This figure also explains some common pitfalls when using the operating system's TCP/IP stack:

- The adapter "Localhost" can only be accessed when using the TCP/IP stack of the operating system (OS). However, the Trace Window will show no Ethernet communication since the physical network adapter of the computer is not used for communication. Therefore, if "simulated bus" is used in CANoe, Ethernet communication cannot be visible in the Trace Window with the OS's TCP/IP stack.
- If the Vector NetIsolator is activated, other OS applications will not be able to communicate on the computer's network adapter, but also CANoe windows, simulation nodes and test modules will not be able to physically communicate over the computer's network adapter if using the OS's TCP/IP stack.
- If using the CANoe TCP/IP stack, messages are sent physically even if the network interface is isolated (only messages of the OS and other OS applications are blocked).
- If using the TCP/IP stack of the operating system, communication problems may be caused by a firewall installed locally on the computer. If the firewall is e.g. blocking TCP packets, this may cause anything from errors to a complete failure of DoIP communication. You or your IT team will need to make sure (e.g. by adjusting the firewall rules) that the firewall is not blocking the required TCP and/or UDP ports.

Therefore, we recommend using the Vector Ethernet network interfaces (VN56xx family) which overcome the limitations of using the operating system's TCP/IP stack, additionally offering precise time stamps and time synchronization to other HW interfaces, e.g. for CAN, LIN, MOST and FlexRay. In the following, we therefore assume you are using a Vector network interface unless mentioned otherwise.

2.4 Basic diagnostics concepts of CANoe

2.4.1 Layer model

Compared to the well-known ISO/OSI layer model, diagnostics in CANoe is implementing a simplified layer model consisting of only 3 layers:

- Physical/Data link layer
- Transport layer
- Diagnostic layer (=Application layer)

This concept is also reflected both in the Diagnostics/ISO TP configuration dialog of CANoe (Figure 3) and in the Trace Window. While the diagnostic layer settings are almost identical for the different networks, the possible settings for the transport layer are highly depending on the respective network. For example for DoIP, the transport layer page is called “DoIP/HSFZ Settings”.

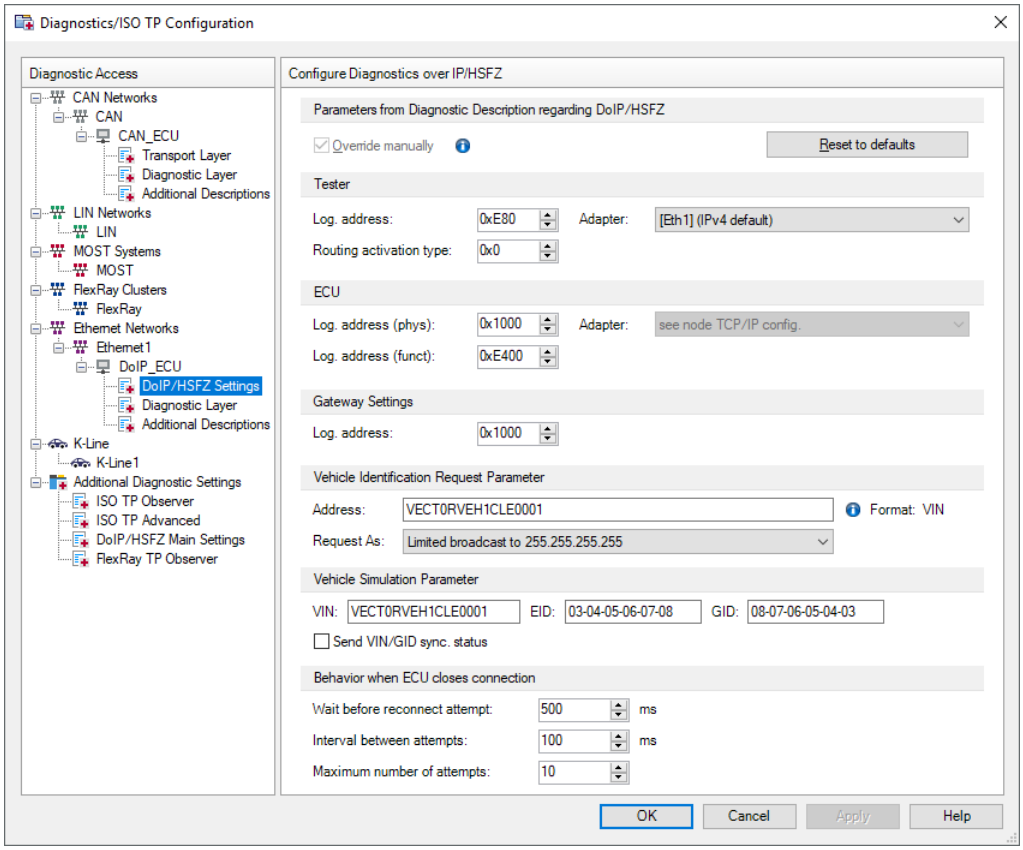


Figure 3: Diagnostics/ISO TP configuration dialog of CANoe

In the Trace Window, the different layers are represented by different event icons while the events carry the exact same time stamp (Figure 4). This helps to distinguish between the raw Ethernet packet (physical/data link layer), the DoIP PDU (transport layer) and diagnostic interpretation (application layer).

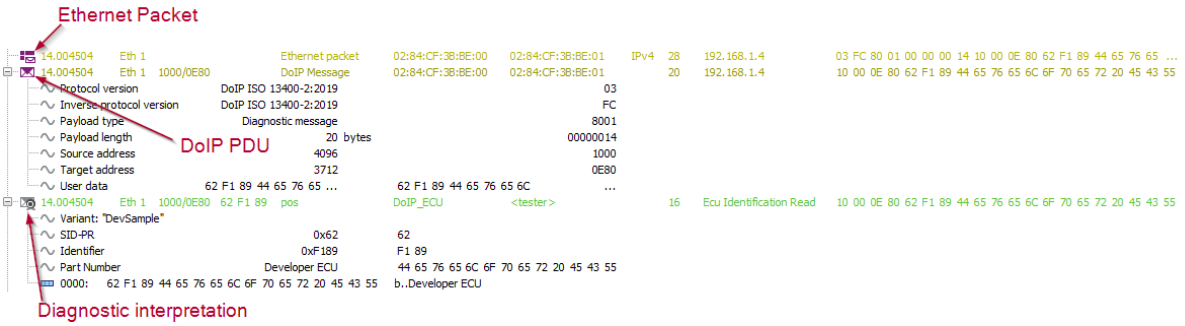


Figure 4: Representation of different layers in the Trace Window

2.4.2 Built-in diagnostic channel

There are two different ways to use diagnostics functionality in CANoe: diagnostics using the built-in diagnostic channel or using the CAPL callback interface.

The built-in diagnostic channel performs diagnostic communication strictly according to the diagnostic specification, given by a diagnostic description. It is used by the diagnostic windows (e.g. Diagnostic

Console, Fault Memory, Session Control etc.), Test Units and .NET test modules. It can be used from CAPL nodes or CAPL test modules as well.

Using the built-in diagnostic channel needs much less effort for configuration and is enough for the typical use cases:

- CANoe observing DoIP communication between diagnostic tester and an ECU
- CANoe implementing a DoIP diagnostic tester
- CANoe implementing a DoIP ECU simulation
- TLS-encrypted DoIP communication between CANoe and an ECU

2.4.3 CAPL callback interface (CCI)

In comparison to the built-in diagnostic channel (see chapter 2.4.2), the CCI offers higher flexibility, but requires more effort to use, resulting from the increased complexity of this topic. Unlike the built-in diagnostic channel, the CCI can only be used in CAPL network nodes, CAPL test modules or test units (i.e. the diagnostic windows and .NET test modules cannot use it). Figure 5 illustrates the usage of the CCI compared to the built-in diagnostic channel in CANoe.

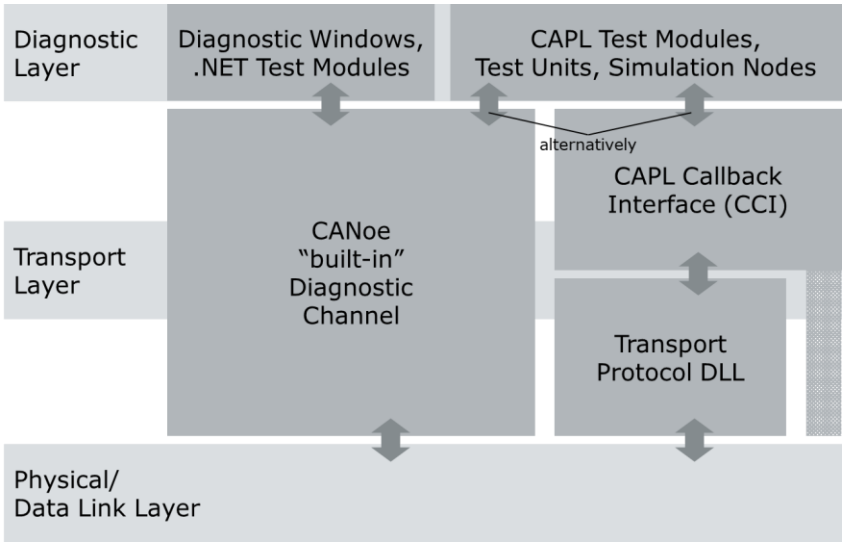


Figure 5: Built-in diagnostic channel vs. CAPL Callback Interface

Out of the box, CANoe offers transport protocol (TP) implementations for the most important automotive networks like CAN, LIN, FlexRay and Ethernet as modeling libraries which can be added as components to a network node or test module. If using the CCI, you need to take care by yourself about adding the necessary TP implementation (you even may implement it by yourself) and its interface (i.e. the CCI implementation) to CANoe's diagnostic layer. For each of the TP implementations, reference CCI implementations exist as CAPL include files (*.cin) which can be used together with the corresponding TP implementation/DLL. For further information on CCI, please refer to the corresponding application note [3].

In the following cases, using the CCI is mandatory:

- CANoe implementing a DoIP gateway
- CANoe implementing fault injection on TP or lower layers for a tester or (simulated) ECU

Nevertheless, the CCI can also be used for the use cases mentioned in chapter 2.4.2.

2.5 Diagnostics and security

2.5.1 Security access / Seed & Key DLL

The traditional way to secure diagnostic access to an ECU, especially as write operations into the ECU's memory are concerned, is implemented using a Seed & Key concept. The benefit of this

security mechanism is its simplicity and its availability for all automotive networks. Nevertheless, it is limited to a specific ECU, i.e. in CANoe you need to configure a dedicated Seed & Key DLL to each diagnostic description you want to secure this way. If the secret changes, you typically need to configure a new Seed & Key DLL. The Seed & Key mechanisms of CANoe are available for DoIP as well. For details, refer to the help (search for “security access”).

2.5.2 Authentication and encryption using the Security Manager

With CANoe 10 SP3, Vector introduced the so-called Security Manager as the central application to handle security mechanisms for Vector products like CANoe, CANalyzer, CANape or vFlash. In comparison to the Seed & Key mechanism (see chapter 2.5.1), it can additionally offer network-wide security with different security mechanisms such as authentication or encryption. These security mechanisms, specified in so-called “security profiles”, can be even OEM-specific by installing an OEM-specific Security Source. The security manager can also be used to configure security for DoIP, details see below (chapter 4.2).

3 Simple use cases and how to configure them in CANoe

3.1 Observe DoIP communication using CANoe

As mentioned above, observing Ethernet communication is a challenging task. The network-based access supported from CANoe 13.0 SP2 helps to reduce the number of hardware connectors needed for this. To enable this feature, make sure that the setting “CANoe Options | Bus Systems / Protocols | Ethernet | Network Access” is set to “Network-based access” (Figure 6).

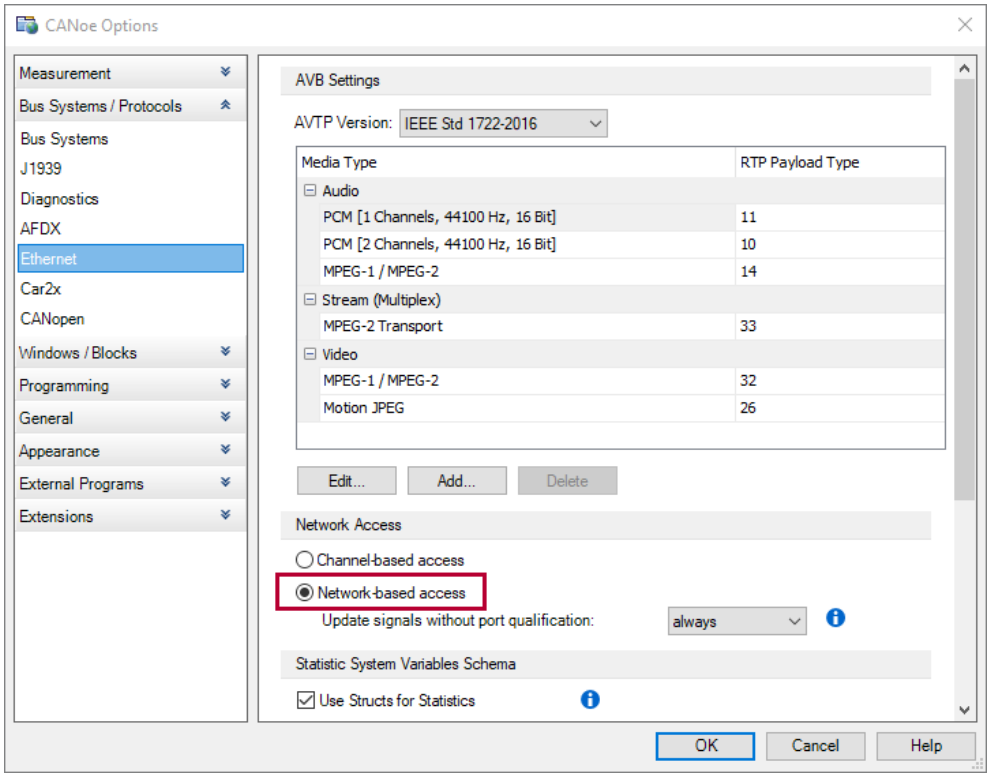


Figure 6: Network-based access setting in the CANoe Options dialog

If not already done, you may need to switch the Ethernet access mode configuration of your VN56xx network interface to network-based access as well. For a setup with two nodes, e.g. a DoIP tester and a DoIP ECU, it is a good idea to define an Ethernet device configuration on your Vector network interface with two physical ports, naming these ports accordingly and connect the real devices to those physical ports. You can define this configuration by opening the Vector Hardware Config dialog, double-clicking on “Ethernet device configuration” for your VN56xx device (Figure 7 and Figure 8).

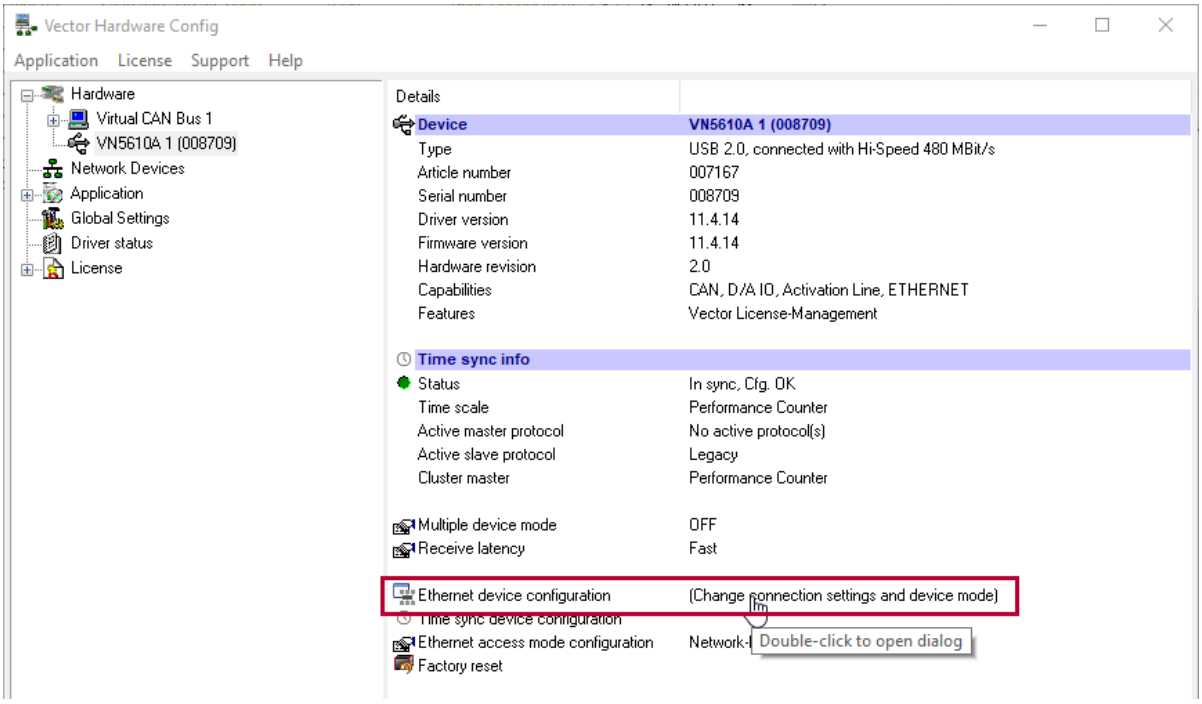


Figure 7: Ethernet device configuration in Vector Hardware Config dialog

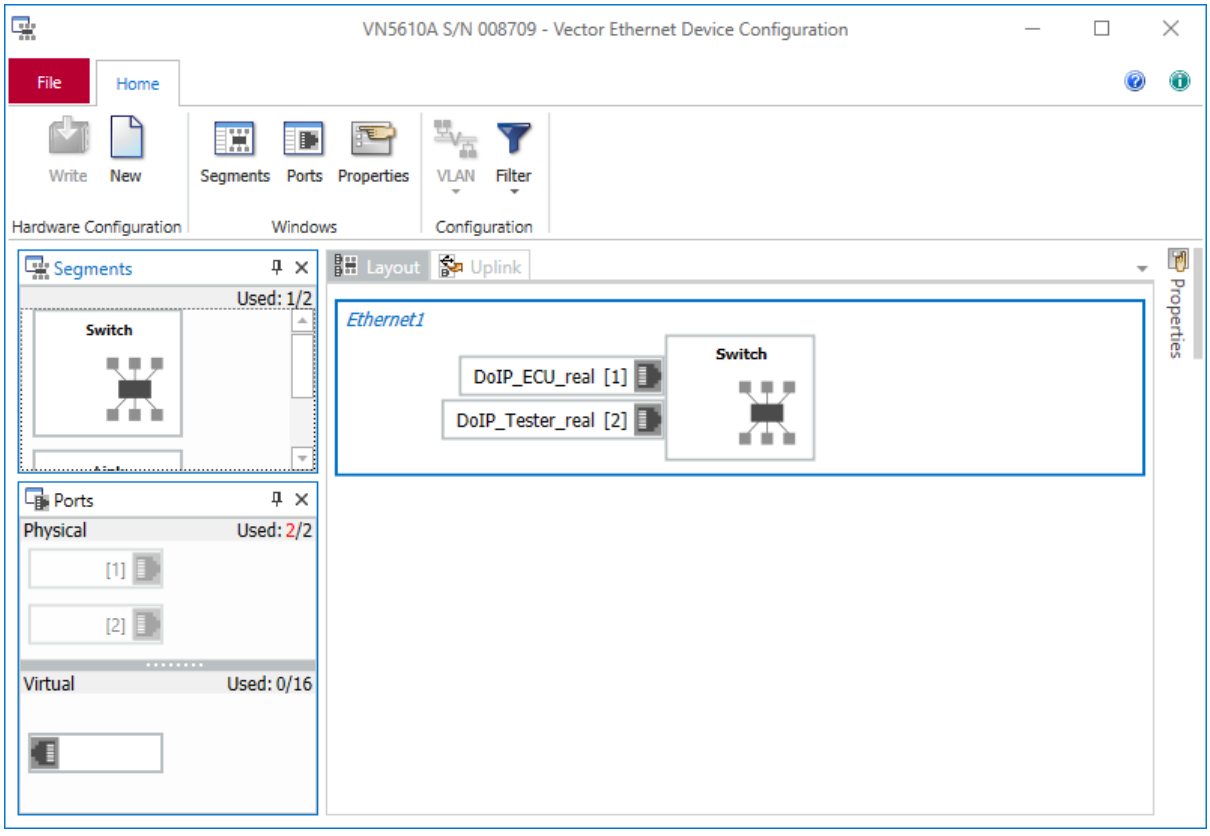


Figure 8: Ethernet Device Configuration with a real DoIP tester and a real DoIP ECU

As mentioned above, the IP addresses of the communication participants are essential for a working setup. As a first step to set up a CANoe configuration, add a network node both for the tester and the ECU (even if you intend to deactivate them because you just want to observe the communication between two real nodes), choose an individual TCP/IP stack for each network node and assign IP addresses to them in the TCP/IP Stack dialog (Figure 9).

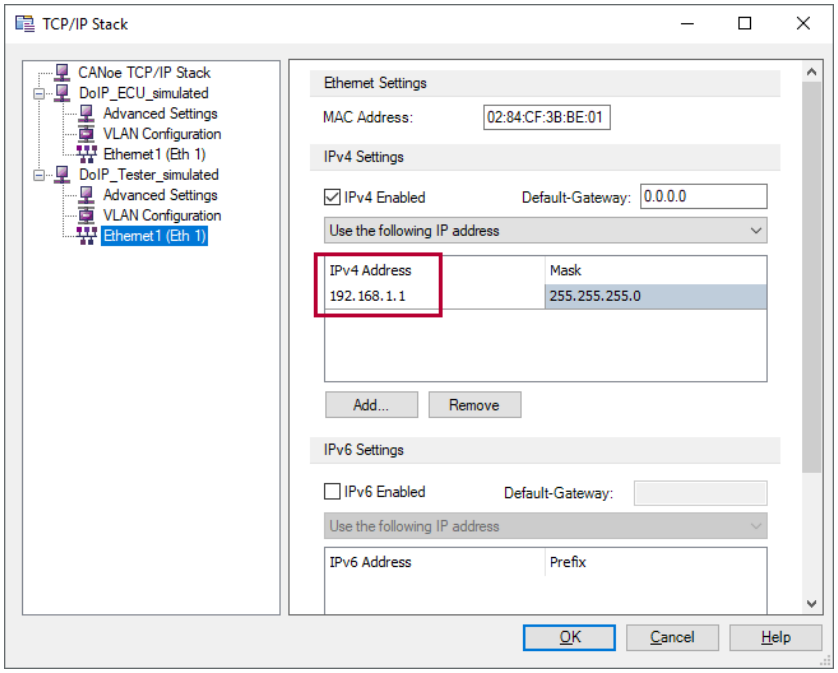


Figure 9: Defining the IP addresses in the TCP/IP Stack dialog

In order to see not only DoIP PDUs but also the diagnostic interpretation of DoIP Diagnostic Message PDUs (payload type 0x8001) in the trace, it is necessary to add a diagnostic description of the corresponding ECU to the network and configure it correctly. As the UDP and TCP listen port for DoIP is fixed to 13400, only three things remain for configuration:

1. select the correct TCP/IP stack to use when simulating the respective node
2. set the Vehicle Identification Number (VIN) which the simulated ECU simulation should use and
3. set the DoIP logical addresses of tester, ECU and Gateway (Figure 10).

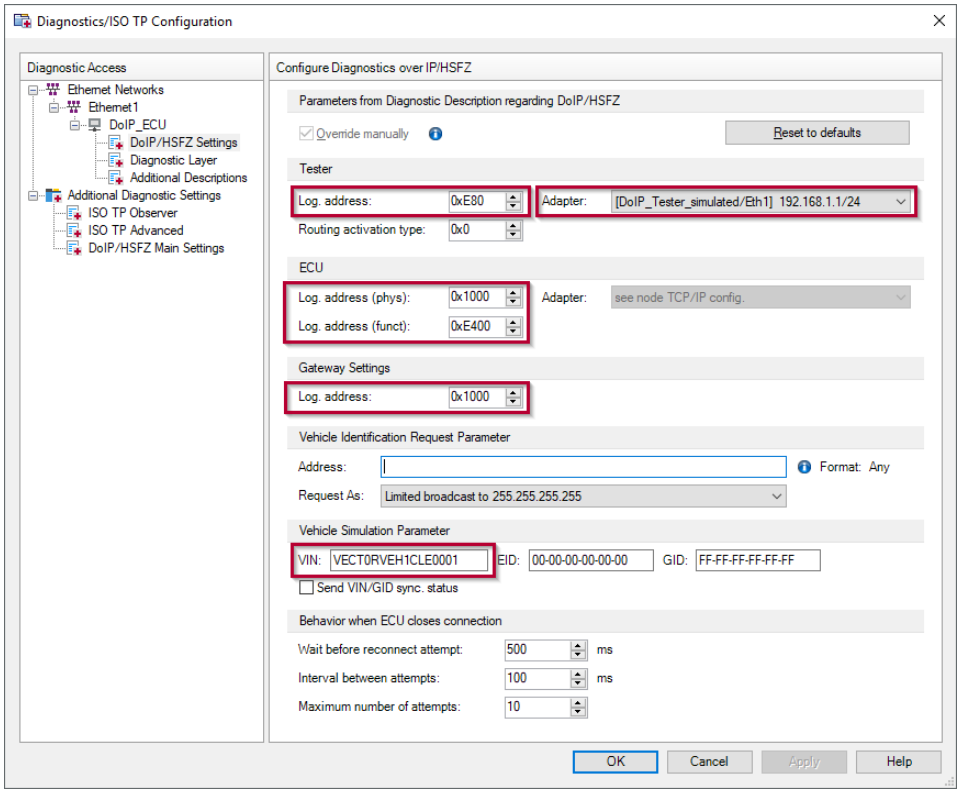


Figure 10: Important settings in the Diagnostics/ISO TP configuration dialog

Note that some of these settings might be already configured by the diagnostic description, especially the settings on the “Diagnostic Layer” page. If these are not correct, you can activate the “Override manually” check box to override them with the correct values until the diagnostic description was corrected.

Next, you should define the measurement ports for your setup. To observe the communication between DoIP tester and ECU connected over a switch on your Vector Ethernet hardware, in most cases it is enough to select just one of the real ports for measurement to avoid duplicated events when measuring both at the sender and the receiver side (Figure 11). If you have simulation nodes defined in your CANoe configuration which are intended to replace the HW device if necessary, you should deactivate them in the Simulation Setup if you are using real hardware devices instead.

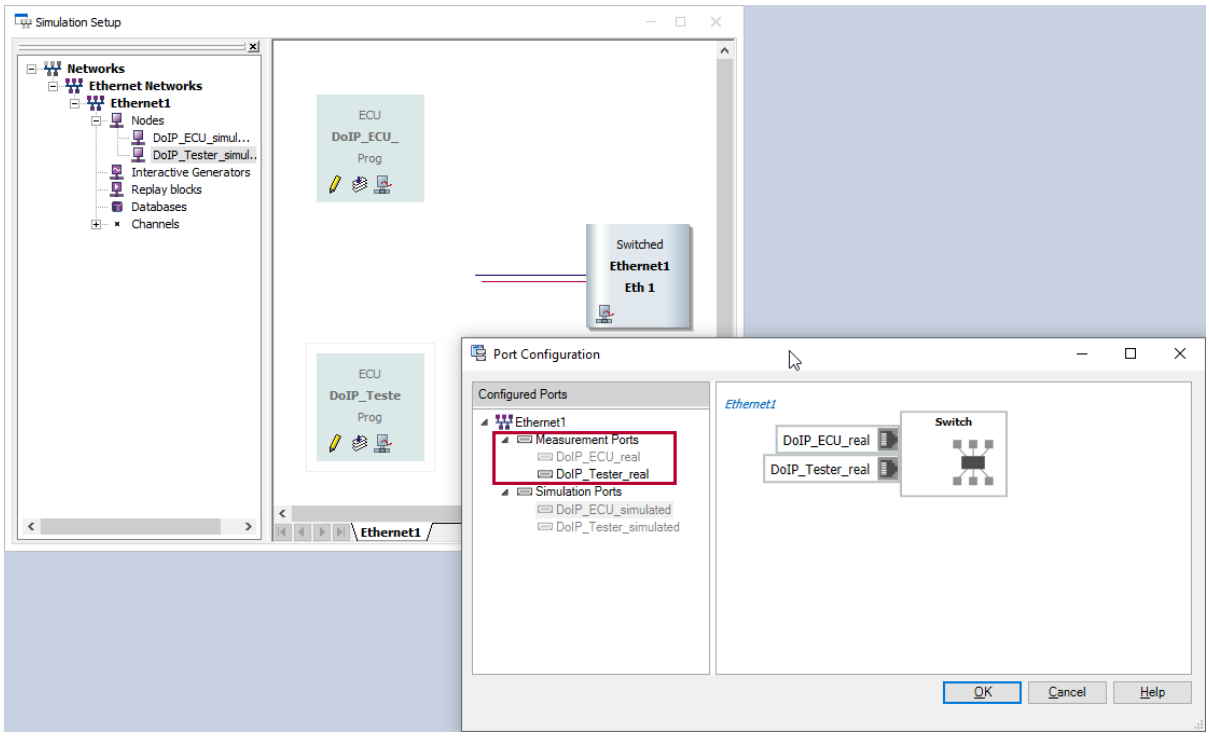


Figure 11: Simulation Setup and Port Configuration (port “DoIP_Tester_real” selected for measurement)

What remains to do, is to set sensible filter settings in order to filter out unnecessary information from the trace. The following settings have been helpful in practice:

1. Add a Channel Filter for the Ethernet network you want to observe.
2. After the Channel Filter, add an Event Filter (IP) and add the settings shown in Figure 12. This will limit the trace content to MAC Address information about the participants both for IPv4 and IPv6 (ARP, NDP), address assignment information if IP addresses are assigned via an DHCP server (DHCPv4 and DHCPv6) and the IP basic protocols for the transport layer (TCP and UDP).

After starting the measurement and sending a request from the tester, you now should be able to see all three levels of interpretation in the trace as shown in Figure 4 (i.e. Ethernet packets, DoIP PDUs and Diagnostic Interpretation).

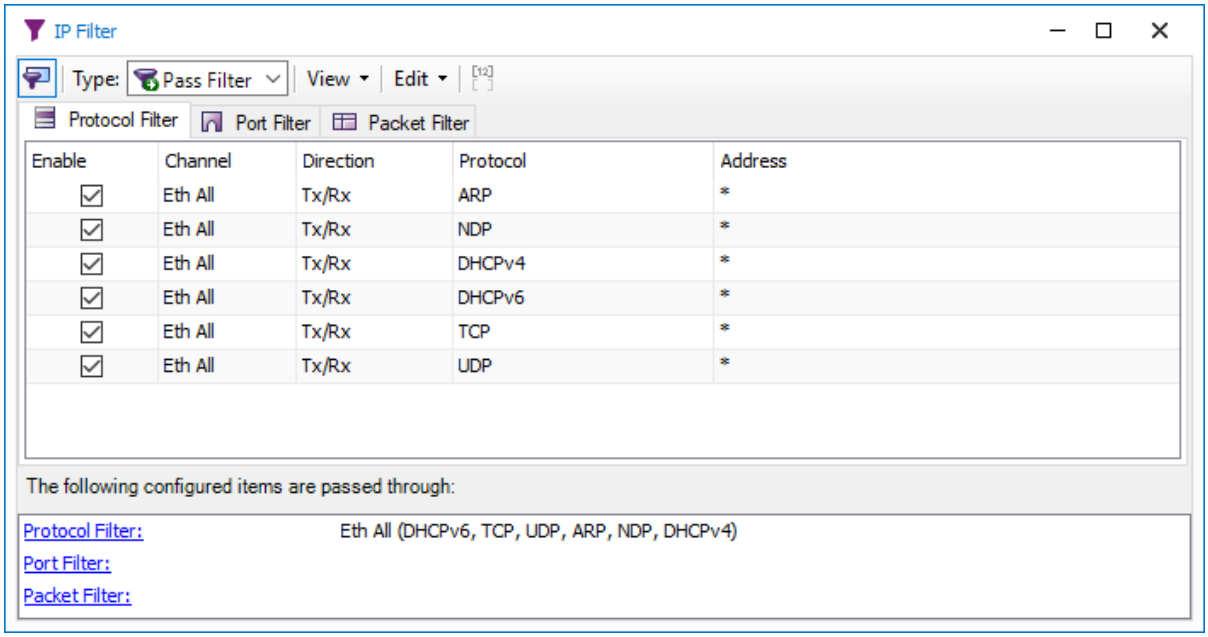


Figure 12: Helpful IP Filter settings for DoIP

3.2 CANoe as DoIP tester

For this use case, in most cases the built-in diagnostic channel of CANoe is enough, i.e. using the CCI is not necessary (see chapters 2.4.2 and 2.4.3). Typically, you have the diagnostic description of the ECU to communicate with, which describes the communication parameters like logical address of ECU, gateway and tester.

What is often missing in the diagnostic description are typical tester parameters and important communication parameters which may be subject to change such as the VIN (because this depends on the vehicle) or the IP addresses of both, ECU and DoIP tester (which might change due to Auto-IP or DHCP mechanisms). Therefore, typically some “manual” configuration is needed additionally. In most cases, also the diagnostic description of the DoIP gateway is missing – but the tester needs to know some of these parameters for communication with ECUs behind the gateway.

The essential parameters for a working DoIP communication are:

- IP address of the ECU
- IP address of the tester
- DoIP logical address of the tester
- DoIP logical address of the DoIP gateway
- DoIP logical address of the ECU
- VIN or Entity ID (EID) of the DoIP gateway

However, there are two major use cases when CANoe is used as a DoIP tester which should also be configured differently in CANoe:

1. Direct DoIP communication with an (Ethernet-) DoIP ECU or with the DoIP gateway itself
2. Indirect DoIP communication with an ECU behind the DoIP gateway

Direct DoIP communication with an (Ethernet-) DoIP ECU

The first thing to configure is the TCP/IP stack to be used by the diagnostic tester. By default, the diagnostic windows (Diagnostic Console, Fault Memory and Session Control window, Variant Coding window, …) are using the (shared) CANoe TCP/IP stack. To configure this correctly, you need to know the following details of your system:

1. From which Ethernet network/segment should the tester communicate?
2. Shall the tester use a VLAN tag? If yes:

   a. VLAN ID
   b. VLAN priority

3. Shall the IP address of the tester be fix or dynamic?
   a. If fix: The IP address of the tester
   b. If dynamic: The way how the IP address is determined:
      i. Automatically via DHCP (using a preferred address or not)
      ii. Automatically select a link-local address according to RFC3927 (“Auto-IP”)

As the CANoe TCP/IP Stack is connected to all configured networks/segments, it is important either to only define one IP address at this stack or - if this is not possible due to the system setup - select the stack and IP address for the tester later in the diagnostic configuration dialog. Figure 13 shows a typical setup of the CANoe TCP/IP Stack where the DoIP tester should use the fix IP address “192.168.1.1” on network/segment “Eth1” without VLAN (for VLAN usage, see chapter 3.4).

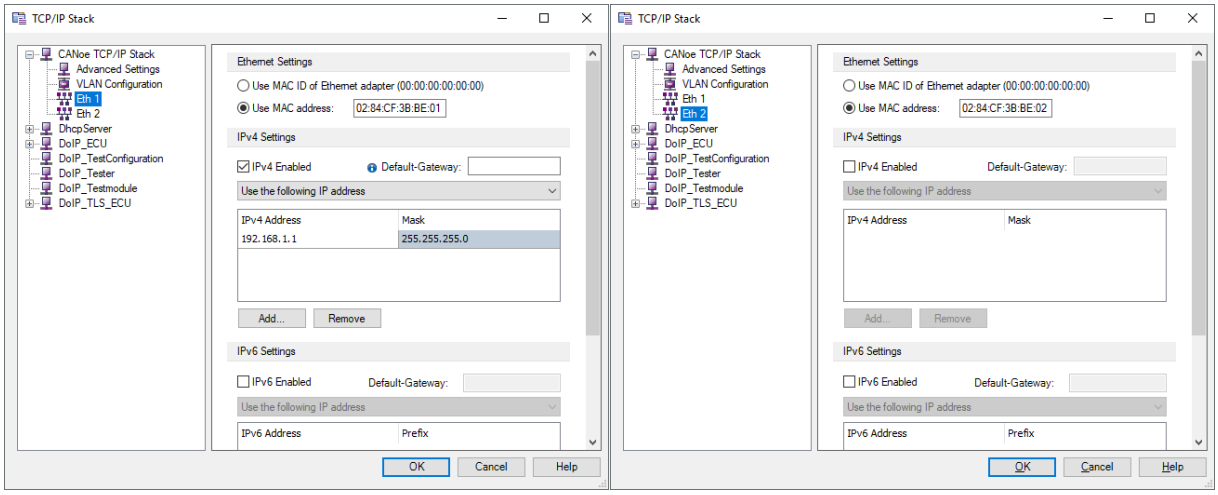


Figure 13: Typical settings for the stack of the DoIP tester

In many cases, the DoIP ECU can only accept one TCP connection for DoIP at the same time. Therefore, it is important that all nodes in CANoe which act as DoIP tester (in the example of Figure 13 the network node “DoIP_Tester”, the CAPL test module “DoIP_TestModule” and the test unit “DoIP_TestConfiguration”) are using the same TCP/IP stack and IP address. This can be achieved by telling CANoe to use the shared CANoe TCP/IP stack for these nodes (Figure 14).

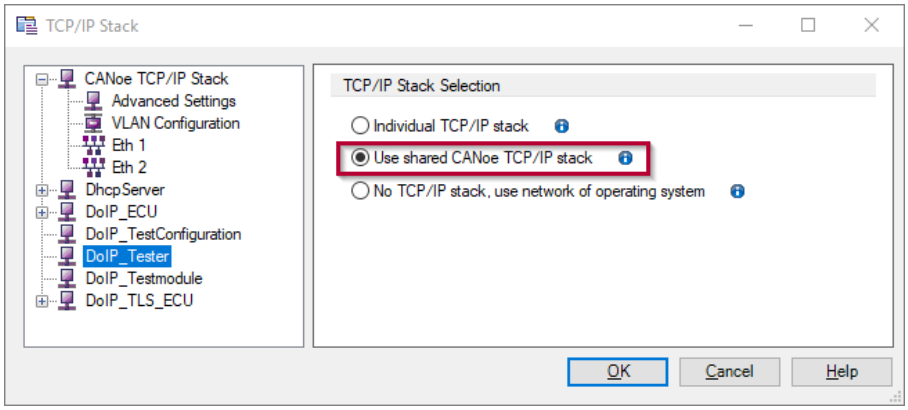


Figure 14: Test node using the shared CANoe TCP/IP stack

Next, you need to configure the diagnostic description of the ECU (CDD, ODX/PDX, MDX) at the network/segment of the DoIP tester. Make sure that on the main page of the diagnostic description, the check box “Diagnostic Tester” under “Usage of the Diagnostic Description” is activated (Figure 15).

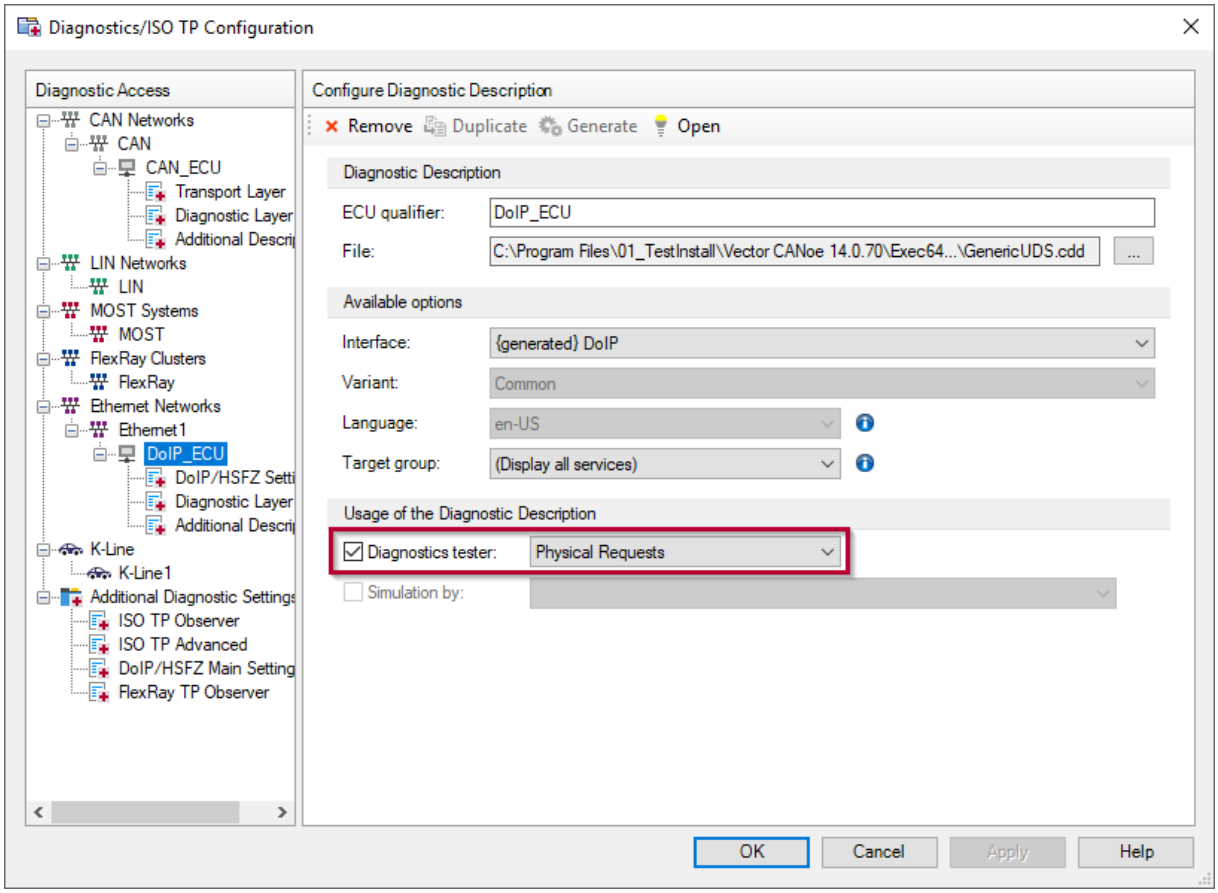


Figure 15: Usage of the diagnostic description as tester for physical requests

The built-in diagnostic channel is defined by the following settings, i.e. if one of these settings is different for another ECU, another diagnostic channel is used:

- Adapter and IP address of the tester
- DoIP logical address of the tester
- DoIP Logical address of the DoIP gateway
- DoIP Logical address of the ECU
- Vehicle Identification Request Parameter (empty, VIN, EID or given IP address in case of unicasts)

However, this does not imply that another TCP connection is used for a different diagnostic channel, because all ECUs of the same vehicle can (and should!) be accessed via the same TCP connection if this vehicle has only one DoIP gateway (“edge node”).

For the adapter, the “Automatic” setting is only a fallback for CANoe in case the adapter previously selected is not available anymore (e.g. if an operating system adapter is used and the CANoe configuration is operated on a different computer). Therefore, this setting should be avoided as it will only rarely work by chance. Figure 16 shows typical settings[1] for a DoIP tester, configuring the logical addresses 0xE80 for the tester, 0x1000 for the ECU and the gateway (physical) and 0xE400 for the ECU’s functional logical address. In the example, the “Address” at “Vehicle Identification Request Parameter” is empty, therefore available vehicles are determined during a vehicle identification phase. To speed up this phase, it is helpful to enter the correct VIN or EID of the ECU here.

[1] The settings configured in the Diagnostics/ISO TP configuration dialog are used by the built-in diagnostic channel of CANoe. If you are using the CCI instead, you can also configure them in this dialog, but these settings are not automatically used by CANoe in this case. You need to explicitly read these settings using the diagGetCommParameter() CAPL function and use them in your CAPL code or for parameterization of used modeling libraries (e.g. the DoIP.dll). The CCI reference implementation for DoIP (CCI_DoIP.cin, see [3] for details) can serve as an example how to do this.

The prefix “[Eth1]” at the adapter selection indicates that the CANoe TCP/IP Stack (i.e. the “shared CANoe TCP/IP stack”) of network “Eth1” should be used for the diagnostic tester of the ECU “DoIP_ECU”. Stacks of individual nodes or test modules/units additionally carry their Identifier in the prefix, e.g. “[DoIP_ECU/Eth1]”. The settings with “(IPv4 default)” or “(IPv6 default)” should especially be used if the IP address assignment is dynamic and the IP address is not known at the start of a measurement (because it is e.g. assigned by a DHCP server).

Finally, the reconnect settings may be adapted under “Behavior when ECU closes connection” as different ECUs have different boot-up times e.g. when they are re-started using a HW Reset service.

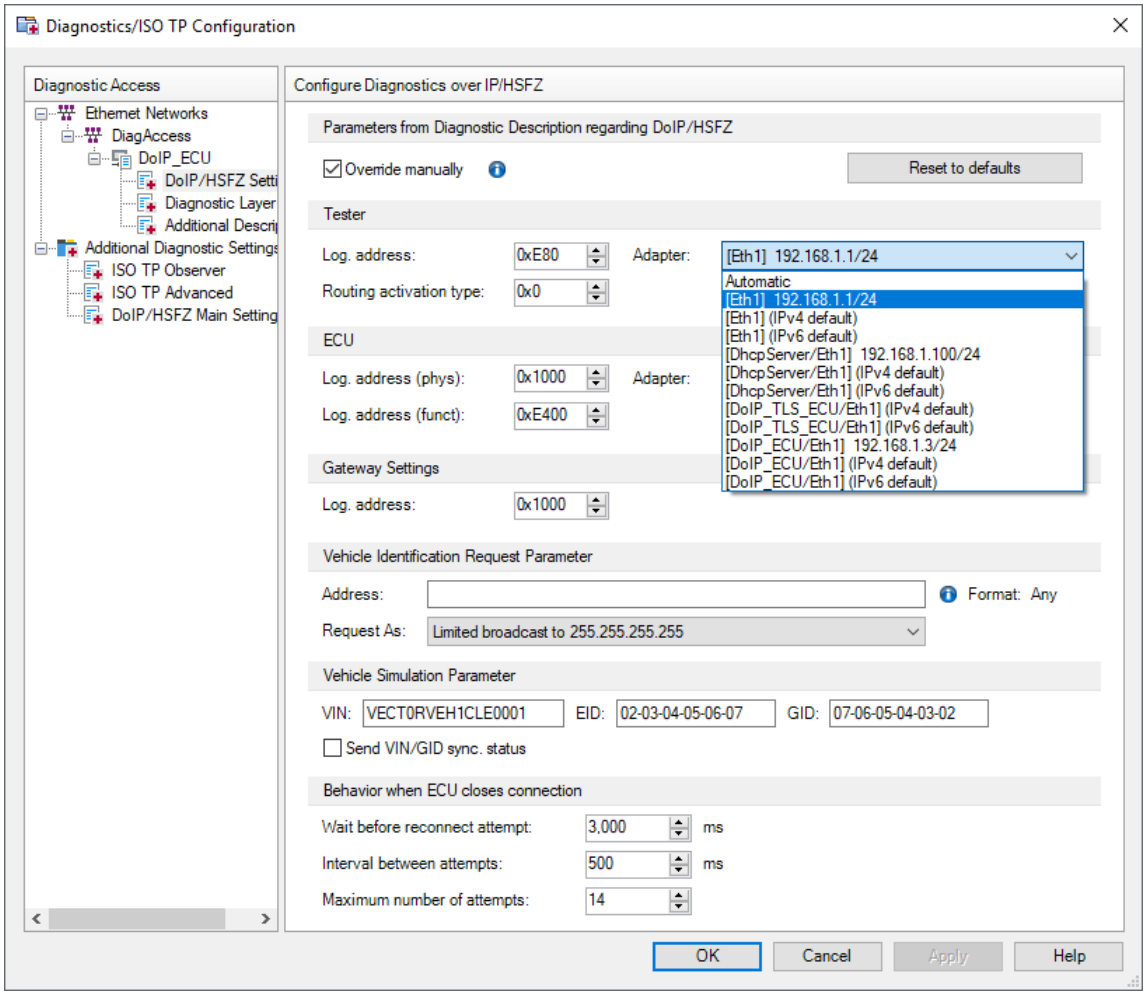


Figure 16: Typical settings for the Diagnostics/ISO TP configuration of the tester

If this is configured correctly, you should be able to use the diagnostic windows of CANoe to communicate with your ECU via DoIP. If you are using CAPL test modules or test units, you can use the same diagnostic channel by using the ECU Qualifier assigned to its diagnostic description. The following CAPL code for a network node sends out a diagnostic request for service “SerialNumber_Read” to the ECU with the ECU qualifier “DoIP_ECU”, when the key ‘r’ is pressed:

```
on key 'r'
{
  diagRequest DoIP_ECU.SerialNumber_Read reqSerialNoRead;

  reqSerialNoRead.SendRequest();
}
```

When pressing the ‘r’ key, the built-in diagnostic channel of CANoe will automatically take care of the following before sending out the DoIP Diagnostic Message (payload type 0x8001) with the UDS “Serial Number Read” request (if this diagnostic channel was not already opened before):

- Send a vehicle identification request without parameter (as the “Address” field in the “Vehicle Identification Request Parameter” section is empty).
- If one vehicle (i.e. the corresponding DoIP edge node/gateway) replies, establish a TCP connection to this vehicle[2].
- If the TCP connection was successfully set up, send a routing activation request to the DoIP edge node.

Only if the routing activation response was received with response code 0x10 (), CANoe can send the “Serial Number Read” request as a DoIP Diagnostic Message.

Afterwards, the TCP connection to the DoIP gateway typically will remain open until either

- measurement is stopped
- *all* diagnostic channels using this TCP connection have been closed using the diagCloseChannel(<EcuQualifier>) CAPL function. Note that if a request was sent functionally[3], also this functional diagnostic channel needs to be closed separately.

The correct setting for “Request As” in the section “Vehicle Identification Request Parameter” depends on the protocol (IPv4 or IPv6) to be used on the configured tester adapter (TCP/IP stack). A limited broadcast to 255.255.255.255 is only possible for IPv4, whereas multicasts to FF02::1 are only possible for adapters with IPv6 addresses configured. If you choose “Use given address, omit Vehicle Identification phase”, you need to configure the ECU’s IP address (IPv4 or IPv6, depending on the tester adapter’s IP address) in the “Address” field (Figure 17). Note that a DoIP ECU may not accept omitting the Vehicle Identification Phase.

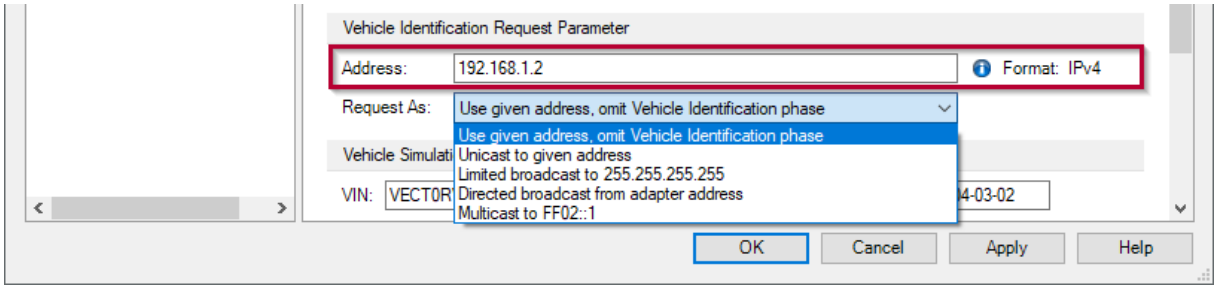


Figure 17: Configuring the Vehicle Identification Request

Indirect DoIP communication with an ECU behind the DoIP gateway

In that case the configuration of the TCP/IP Stack for the tester must be done in the same way like in the direct communication case. The main difference is that the target ECU is not necessarily an Ethernet ECU, but the communication to it from tester side must be performed using Ethernet / DoIP.

[2] If multiple vehicles may reply, you need to configure the VIN or the EID of this vehicle/edge node or a dedicated IP address as “Address” in the “Vehicle Identification Request Parameter” section. Otherwise CANoe does not know, to which of the responding vehicles it should connect. Additionally, this will speed up the connection setup process because CANoe does not need to wait until all potential vehicles have responded.

[3] To send a request functionally, you may additionally configure the same diagnostic description as for the physical requests and set in the section “Usage of the Diagnostic Description” the usage for the “Diagnostics tester” to “Functional Group Requests”. The ECU qualifier of this diagnostic description can be used as parameter for the diagCloseChannel() CAPL function. Of course, the logical DoIP functional addresses (“Log. Address (funct)”) need to be the same for all diagnostic descriptions in the same functional group.

In order to configure this, you need to add the diagnostic description to the network where the ECU is located. Let's say your ECU is a CAN ECU with ECU qualifier "ECU", located at the CAN network "CAN1", then the diagnostic description needs to be added to this network (CAN1). Without any change, CANoe would try to send out diagnostic requests addressed to this ECU as CAN frames on CAN1. Therefore, it is necessary to "redirect" the diagnostic communication over Ethernet / DoIP by activating the check box "Enable diagnostics via common DoIP/HSFZ gateway for this network" (Figure 18). This will add two new pages at the selected Ethernet network, one for the gateway ("Gateway to CAN1") and one for each CAN ECU available on the CAN network for which the "redirect" mechanism was activated ("ECU" under "Gateway to CAN1").

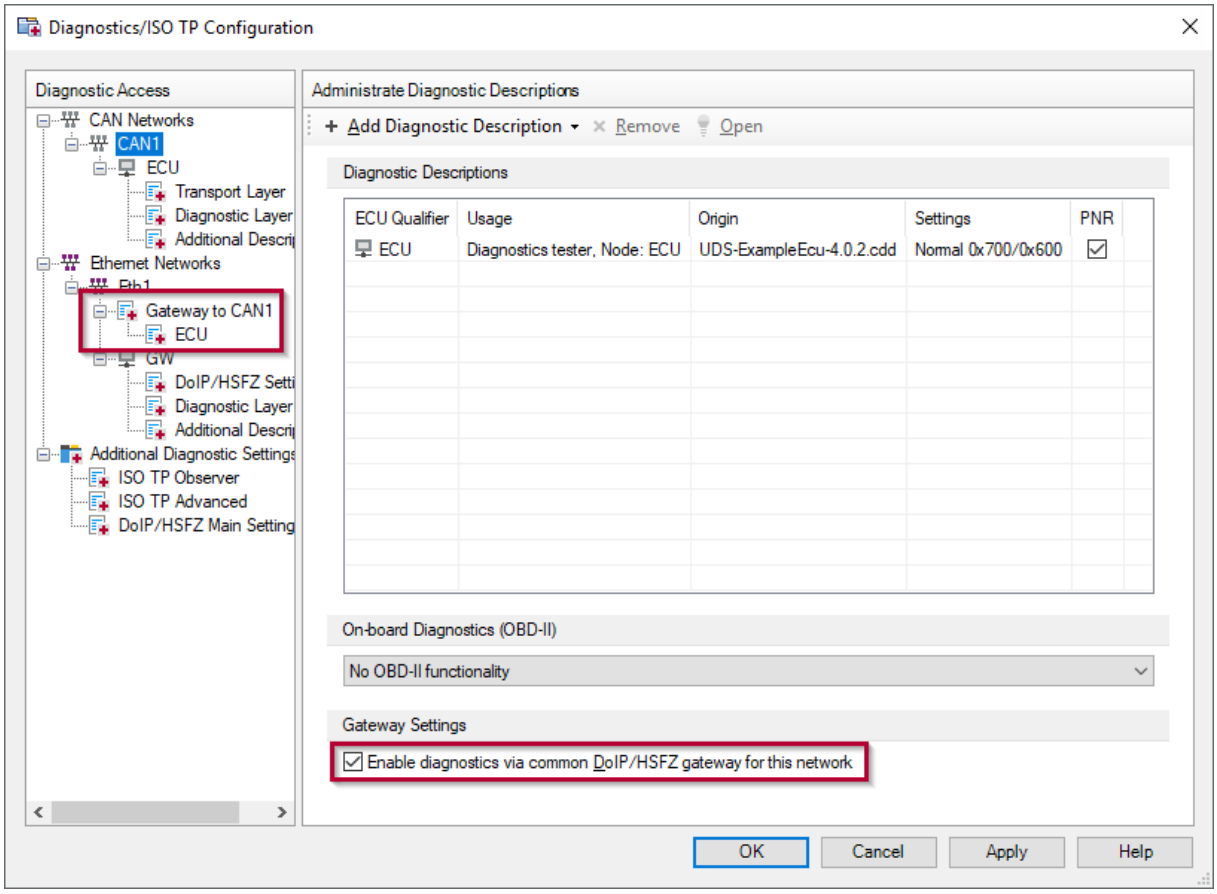


Figure 18: "Redirecting" CAN communication to CAN ECU over Ethernet / DoIP

In those new pages you can now configure the DoIP-specific communication parameters which are not available in the diagnostic description for CAN. On top of that, no diagnostic description is necessary for the gateway itself – you just need to set the DoIP parameters like in the direct DoIP communication case (Figure 19).

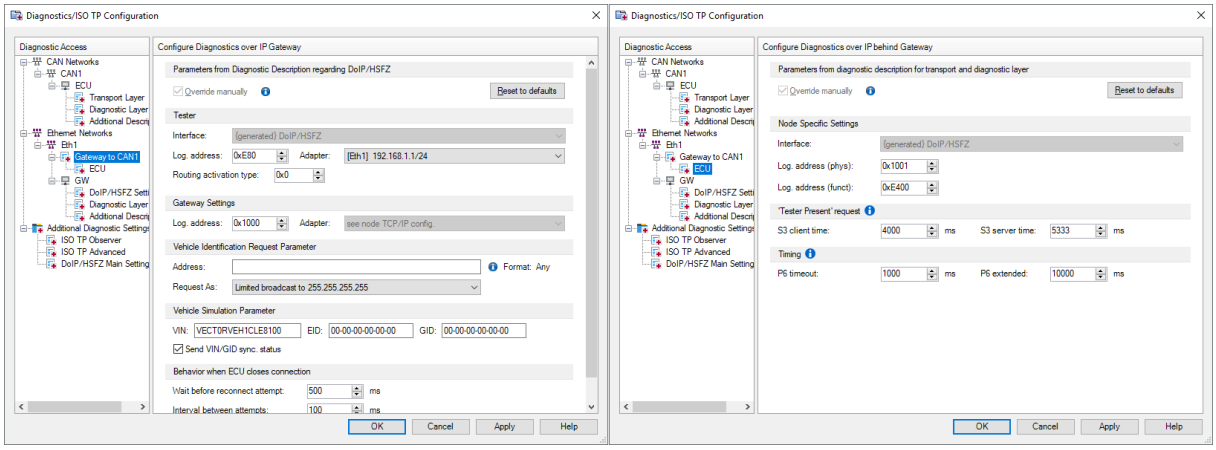


Figure 19: Configuring settings for “Gateway to CAN1” and node-specific settings for “ECU”

3.3 CANoe simulating DoIP ECUs

For the „test the tester” use case, it is very helpful to be able to simulate an ECU on diagnostics layer easily. In order to do this, you need to add a network node to the Ethernet network serving as ECU simulation to which the diagnostic tester will connect. It is helpful to assign a CAPL file and save it even if you do not intend to edit it in the first step in order to avoid that the name of the network node changes when adding the CAPL file later.

Next, you need to configure the TCP/IP stack of this simulation node in the TCP/IP Stack dialog. Like described in chapter 3.1, configure the VLAN settings (if necessary, see chapter 3.4 for details) and IP address of this simulated ECU.

If CANoe is used for both, the diagnostic tester and the ECU simulation, you can use the same diagnostic description both for the tester and the ECU. If you did not already configure the diagnostic description, you need to add it first. Activate the check box “Simulation by”, select the network node implementing the diagnostics simulation for this ECU (Figure 20) and enter the correct logical address for this ECU (Figure 10). Note that the network node name changes if you assign a CAPL file to a network node without CAPL file or if you re-assign a different CAPL file to it. In this case, you also need to re-assign the (renamed) network node here.

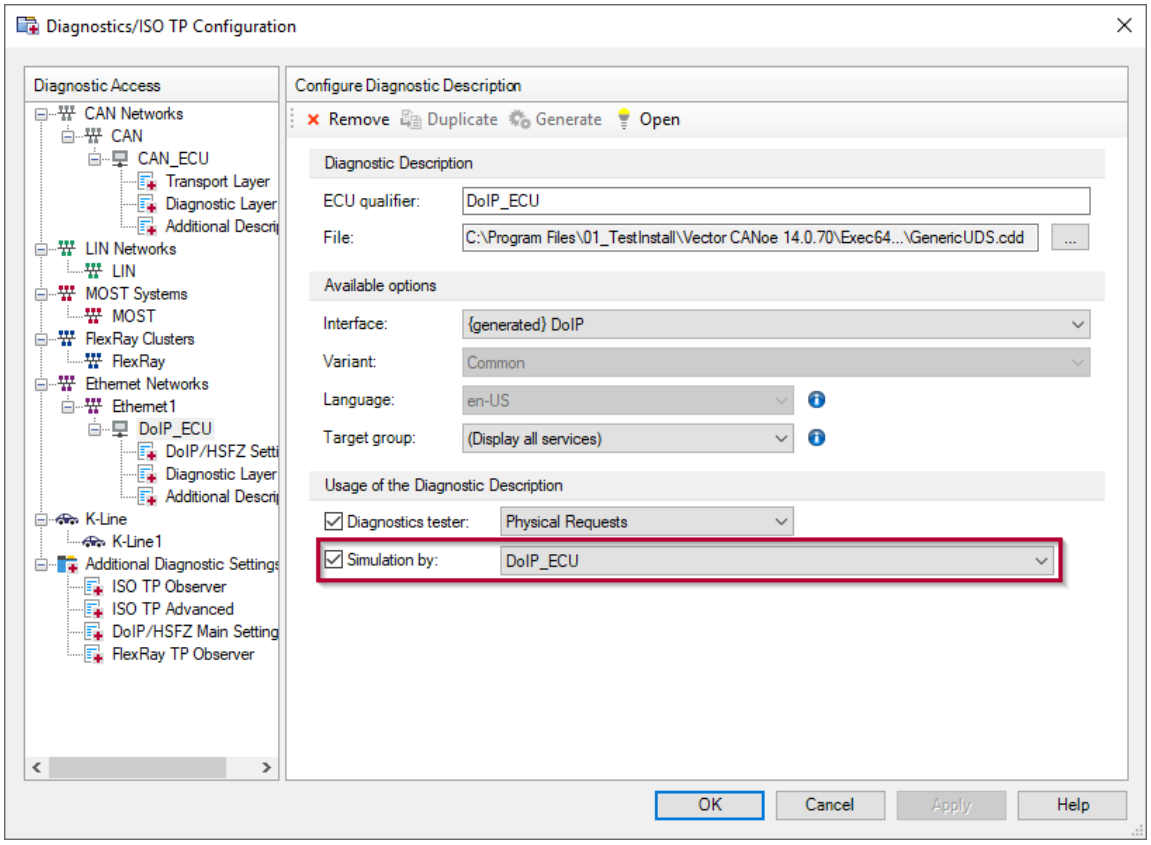


Figure 20: Simulating a DoIP ECU

Additionally, you need to configure the Vehicle simulation parameters, such as VIN, EID and Group Identification (GID) so that the built-in diagnostic channel of the simulated ECU can send correct Vehicle Announcements and Vehicle Identification Responses.

At last, you may need to add some CAPL code for the simulated ECU's behavior when receiving diagnostic requests – otherwise, the built-in diagnostic channel will only use default values for diagnostic parameters in the responses, which might lead to incorrect diagnostic responses. For each possible diagnostic request, you may add an "on diagRequest" handler in the following manner:

```
on diagRequest DoIP_ECU.DiagnosticSessionControl_Process
{
  diagResponse this resp;
  double diagSessionType;
  byte sessionParameterRecord[4]={0x00, 0x96, 0x00, 0xC8};

  diagSessionType=this.GetParameter("diagnosticSessionType");

  resp.SetParameter("diagnosticSessionType", diagSessionType);
  resp.SetParameterRaw("sessionParameterRecord",
          sessionParameterRecord, elcount(sessionParameterRecord));
  resp.SendPositiveResponse();

}
```

In this example, all requests to the service "`DiagnosticSessionControl_Process`" will be answered with a positive response by the network node when using this code. In the response, the same session type (response parameter "`diagnosticSessionType`") will be used like given in

the request and four raw data bytes for the response parameter “`sessionParameterRecord`” will be added.

3.4 Using VLAN tagging

For this use case, the built-in diagnostic channel of CANoe is enough as well. If the DoIP communication should be performed using a virtual LAN (VLAN), the configuration steps are the same as for a network without VLAN tagging.

Note
In case of a simulated ECU, we recommend removing the IP addresses defined directly at the node (i.e. the IP addresses used without VLAN tag), if possible. Otherwise, it may happen that the built-in diagnostic channel of the ECU simulation will use the wrong IP address as it will choose the first address found at the adapter.

Figure 21 shows how to add and configure an IP address using a VLAN.

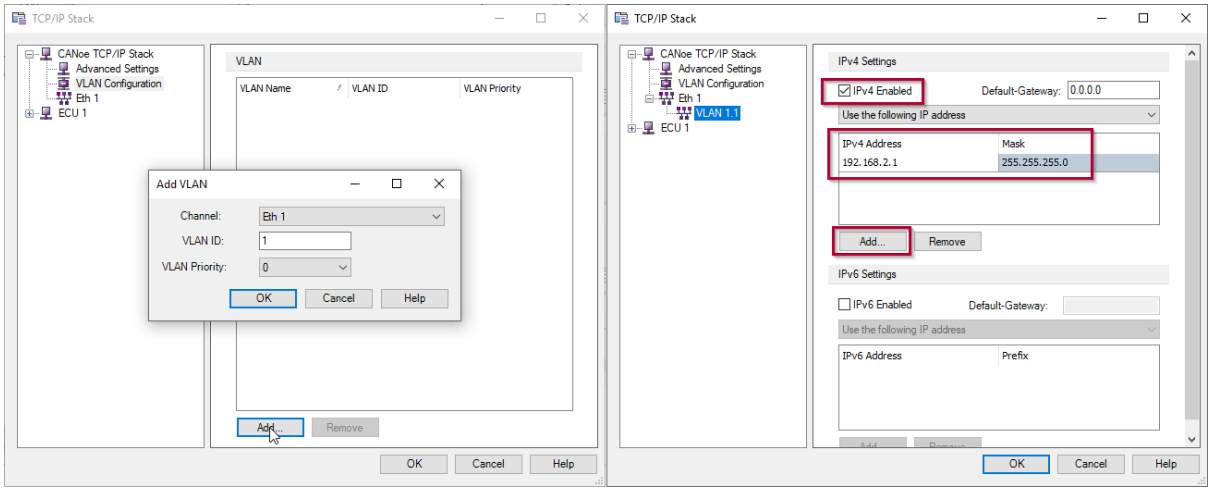


Figure 21: Adding and configuring the IP address for a VLAN

Next, you need to select the correct adapter, i.e. the TCP/IP Stack for the diagnostic tester so that it will use the IP address defined for the VLAN. Just choose an IP address with the prefix [Ethx/VLAN x.y], where “x” is a placeholder for the network index and y for the VLAN ID (Figure 22).

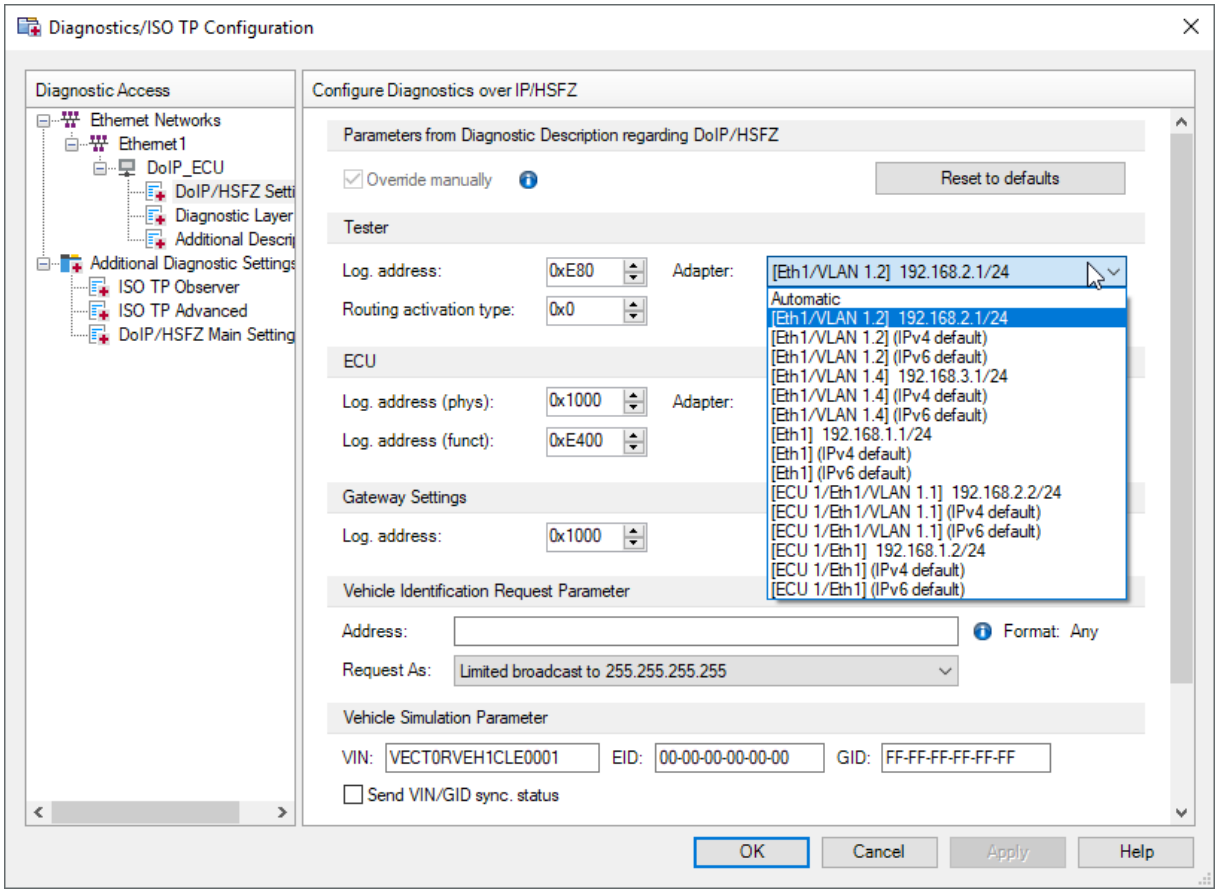
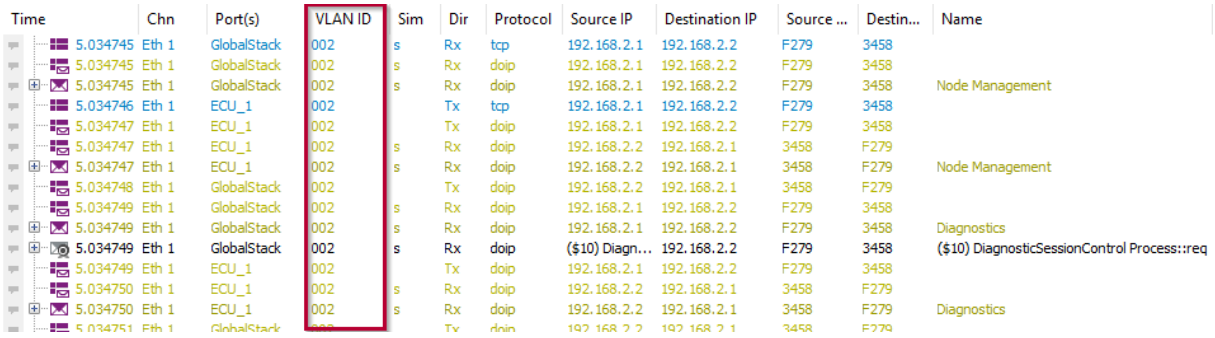


Figure 22: Choosing an adapter with VLAN tagged IP address

In the trace, the column “VLAN ID” – available e.g. in one of the “Ethernet IP/UDP/TCP” column layouts – should show the intended VLAN ID (Figure 23) when sending a diagnostic request e.g. via Diagnostic Console.



Time	Chn	Port(s)	VLAN ID	Sim	Dir	Protocol	Source IP	Destination IP	Source ...	Destin...	Name
5.034745	Eth 1	GlobalStack	002	s	Rx	tcp	192.168.2.1	192.168.2.2	F279	3458	
5.034745	Eth 1	GlobalStack	002	s	Rx	doip	192.168.2.1	192.168.2.2	F279	3458	
5.034745	Eth 1	GlobalStack	002	s	Rx	doip	192.168.2.1	192.168.2.2	F279	3458	Node Management
5.034746	Eth 1	ECU_1	002		Tx	tcp	192.168.2.1	192.168.2.2	F279	3458	
5.034747	Eth 1	ECU_1	002		Tx	doip	192.168.2.1	192.168.2.2	F279	3458	
5.034747	Eth 1	ECU_1	002	s	Rx	doip	192.168.2.2	192.168.2.1	3458	F279	
5.034747	Eth 1	ECU_1	002	s	Rx	doip	192.168.2.2	192.168.2.1	3458	F279	Node Management
5.034748	Eth 1	GlobalStack	002		Tx	doip	192.168.2.2	192.168.2.1	3458	F279	
5.034749	Eth 1	GlobalStack	002	s	Rx	doip	192.168.2.1	192.168.2.2	F279	3458	
5.034749	Eth 1	GlobalStack	002	s	Rx	doip	192.168.2.1	192.168.2.2	F279	3458	Diagnostics
5.034749	Eth 1	GlobalStack	002	s	Rx	doip	($10) Diagn...	192.168.2.2	F279	3458	($10) DiagnosticSessionControl Process::req
5.034749	Eth 1	ECU_1	002		Tx	doip	192.168.2.1	192.168.2.2	F279	3458	
5.034750	Eth 1	ECU_1	002	s	Rx	doip	192.168.2.2	192.168.2.1	3458	F279	
5.034750	Eth 1	ECU_1	002	s	Rx	doip	192.168.2.2	192.168.2.1	3458	F279	Diagnostics

Figure 23: Trace with VLAN tagged DoIP communication

3.5 Using the DoIP activation line

When CANoe as tester communicates to a DoIP edge node (i.e. typically a DoIP gateway), the DoIP standard requires to use an activation line to indicate to the DoIP edge node that a tester is connected. You can use any IO port of Vector HW interfaces which fulfills the electrical requirements for a DoIP activation line. The latest Vector network interfaces for Ethernet (VN56xx family) offer an IO port which can be used for this purpose. If you are using the built-in diagnostic channel of CANoe for the tester side, you simply need to define the predefined system variable which controls this IO port and - if the DoIP edge node reacts too slowly – define a delay after which CANoe should start DoIP communication after

activating the activation line. The rest, i.e. setting and resetting the system variable for the activation line, is done automatically by the built-in diagnostic channel of CANoe.

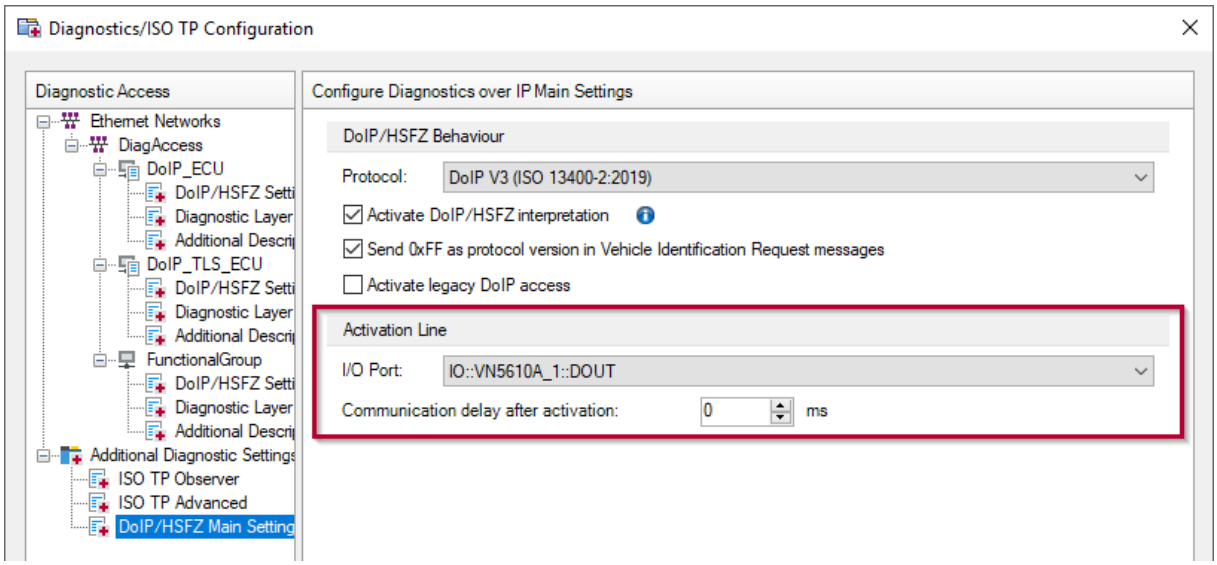


Figure 24: Define the system variable for the DoIP activation line IO port

If you are using the CCI for the tester in CANoe, you may want to set and reset the activation line by yourself in your CAPL code by setting and resetting the same system variable – however, in this case you should set the IO port in the Diagnostics/ISO TP configuration dialog to “No activation line used” to avoid conflicts between the built-in diagnostic channel and your CCI implementation. The following example shows the CAPL code for the IO port of the VN5610A network interface configured above in the Diagnostics/ISO TP configuration dialog which could be used in a CCI implementation.

```
// value = 0: Set activation line to 0V
// value = 1: Set activation line to U_batt (battery voltage)
void SetActivationLine(byte value)
{
  if (value>1) return; // invalid value

  @sysvar::IO::VN5610A_1::DOUT=value;
}
```

4 Advanced use cases

4.1 CANoe simulating a DoIP gateway

Simulating a DoIP gateway is a quite complex task, as behind the gateway, different types of networks may need to be addressed. The routing mechanisms typically are very OEM- and vehicle-specific. Therefore, it is always necessary to use the CCI when implementing such gateways in CANoe.

A detailed description of how to implement a DoIP gateway is beyond the scope of this document. For further information on this topic, please refer to the Vector application notes [4] and [3]. You will find an example implementation of such a gateway in the sample configuration “HSFZSystem.cfg”, which is using the HSFZ protocol but can be switched to DoIP by simply changing the protocol version (Figure 1).

4.2 Encrypted communication via Transport Layer Security (TLS)

Starting with CANoe 14, it is possible to encrypt DoIP communication via TLS. In order to do this, you need a Security Source with a Security Profile providing the necessary information for TLS encryption, like certificates and supported cipher suites. As soon as a Security Source providing a security profile for TLS is installed, you can use this feature by just configuring the appropriate Security Profile to the TCP/IP Stacks which are used for tester and (simulated) ECU. In the sample configuration “DoIPSystem.cfg”, one of the simulated ECUs is communicating with the DoIP tester via TLS, using the security profile “DoIP over TLS Demo” both for tester and ECU (Figure 25).

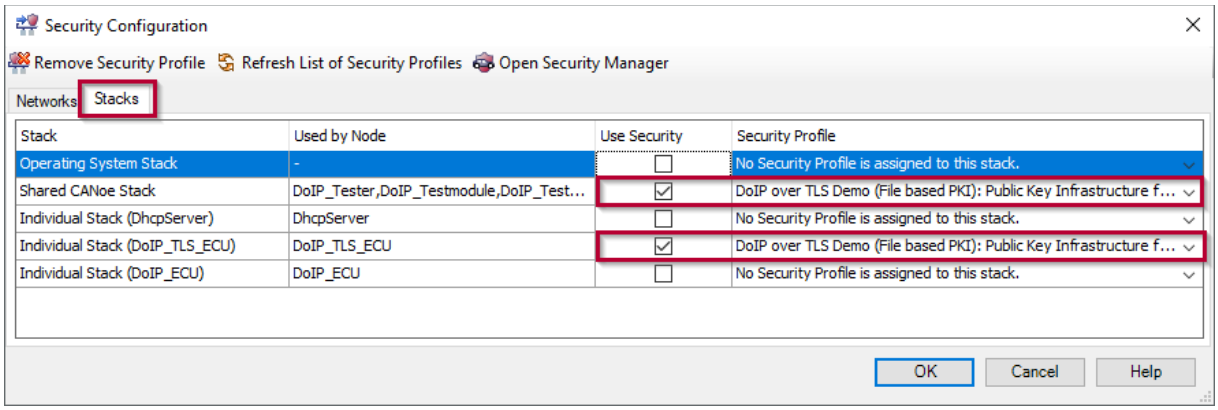


Figure 25: Selection of a TLS Security Profile at the TCP/IP Stack of the tester and the ECU

As soon as this is configured, the Diagnostics/ISO TP configuration dialog provides some additional controls for ECUs where a tester or ECU simulation uses one of the TCP/IP stacks (set as “Adapter”) with “Use Security” activated (Figure 26).

For the tester, you need to select the mode how the tester should operate. The following settings are available:

Setting	Explanation
Disable TLS	The tester does not use TLS even if a security profile for TLS is configured at the TCP/IP stack of the tester. No connection is established, if the ECU is responding with routing activation response code (RARC) 0x07 (“Routing activation denied because the specified activation type requires a secure TLS TCP_DATA socket”).
Enable TLS by routing activation (NACK 0x07)	The tester at first tries to set up an unencrypted TCP connection. If the ECU responds with RARC 0x10 (“Routing successfully activated”), the tester will continue with an unencrypted TCP connection. If the ECU responds with RARC 0x07, the tester tries to connect via TLS.
Enable TLS, allow secure connection only	The tester at first tries to set up an unencrypted TCP connection. If the ECU responds with RARC 0x10, the tester stops the communication. If the ECU responds with RARC 0x07, the tester tries to connect via TLS.
Enable TLS, direct connection to TLS port for DoIP	The tester does not try to set up an unencrypted TCP connection. Instead, the tester directly tries to connect to the ECU via TLS. This can speed up the connection setup but needs to be supported by the ECU.

In case of an ECU simulation, the check box “Enable TLS” activates TLS communication for the respective TCP/IP stack. Additionally, you need to configure the name of the certificate to be used by this ECU simulation. You can find the certificate name in the corresponding security source when selecting the TLS security profile (Figure 27).

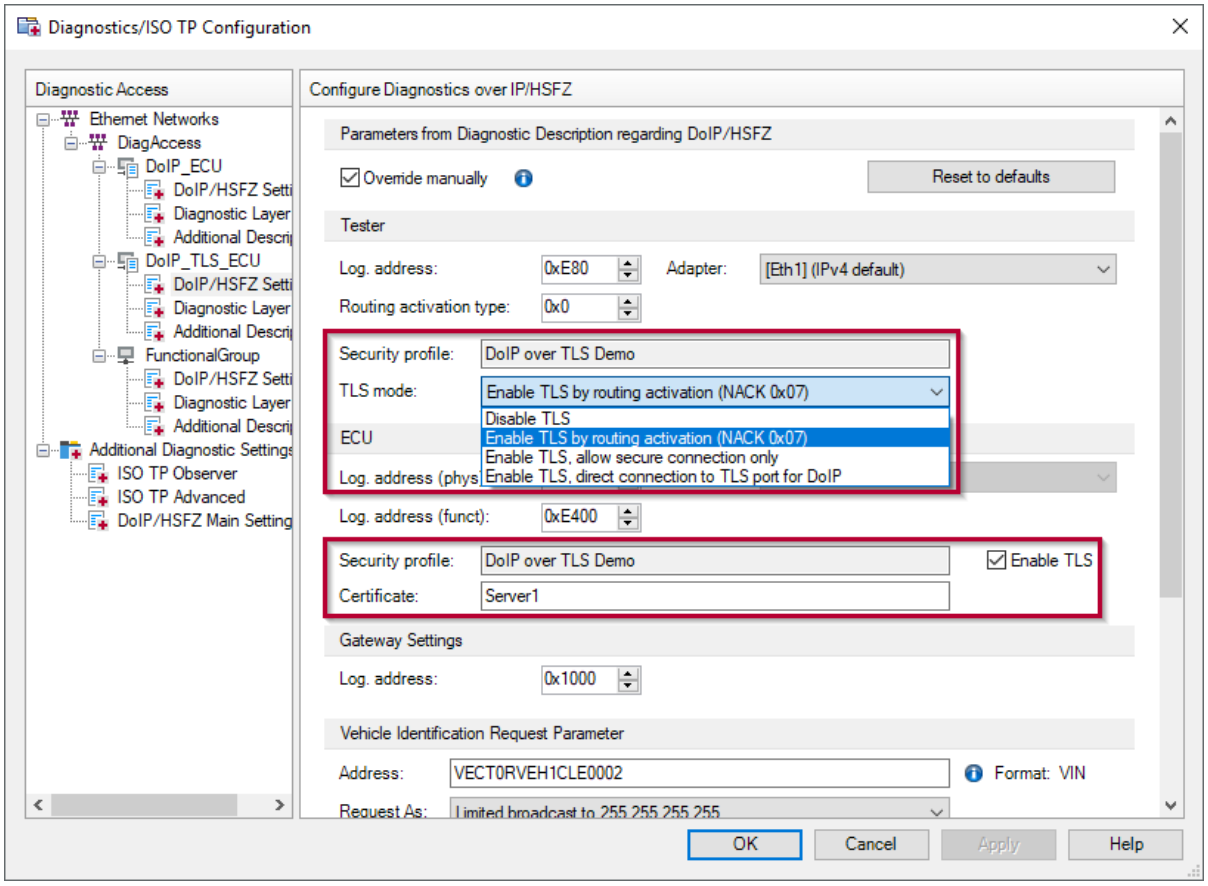


Figure 26: TLS settings in the Diagnostics/ISO TP configuration dialog after configuring a TLS security profile at the stack

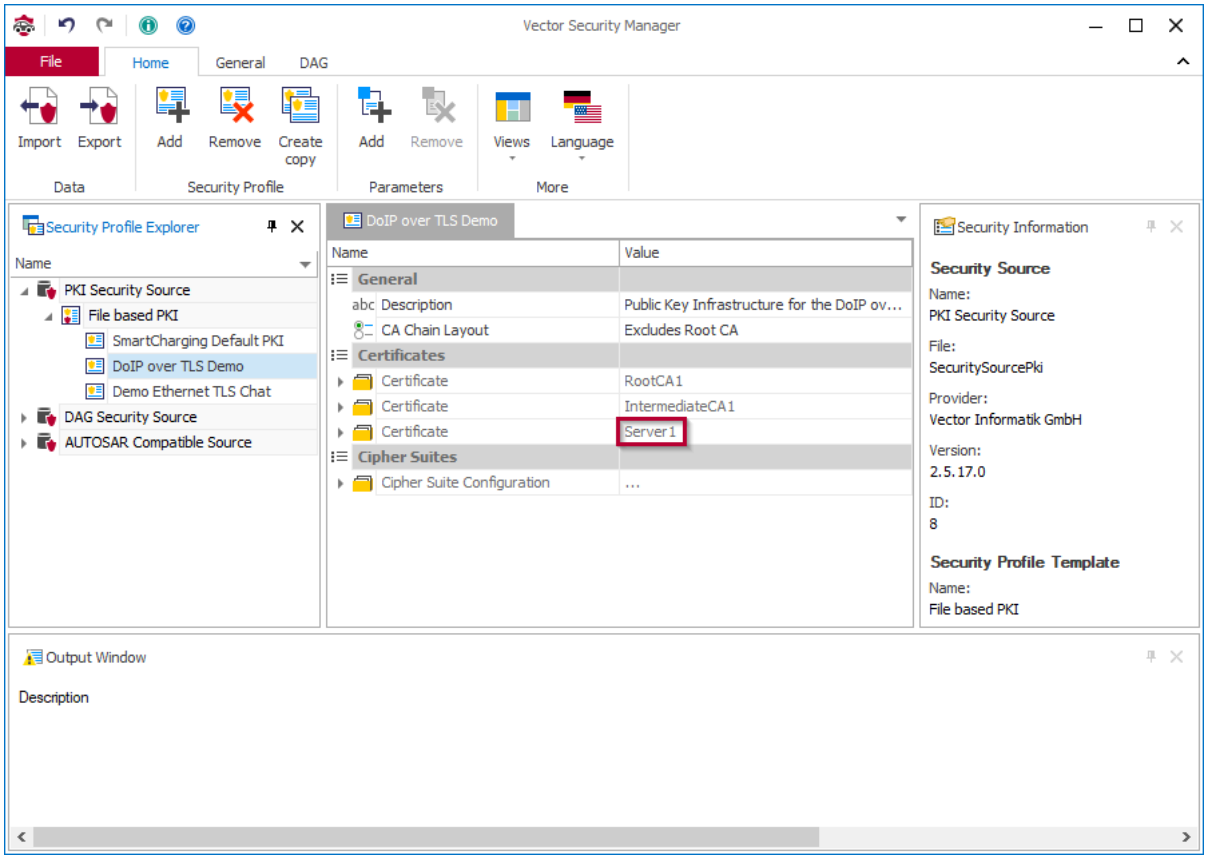


Figure 27: Certificate name in the security manager

As a result, the DoIP Diagnostic Messages will be encrypted when sent by CANoe. As CANoe knows the master secret for this session if it simulates one of the communication endpoints, CANoe can identify DoIP PDUs within this encrypted communication and from these DoIP PDUs interpret the UDS payload, i.e. the diagnostic services and the transmitted parameter values (Figure 28).

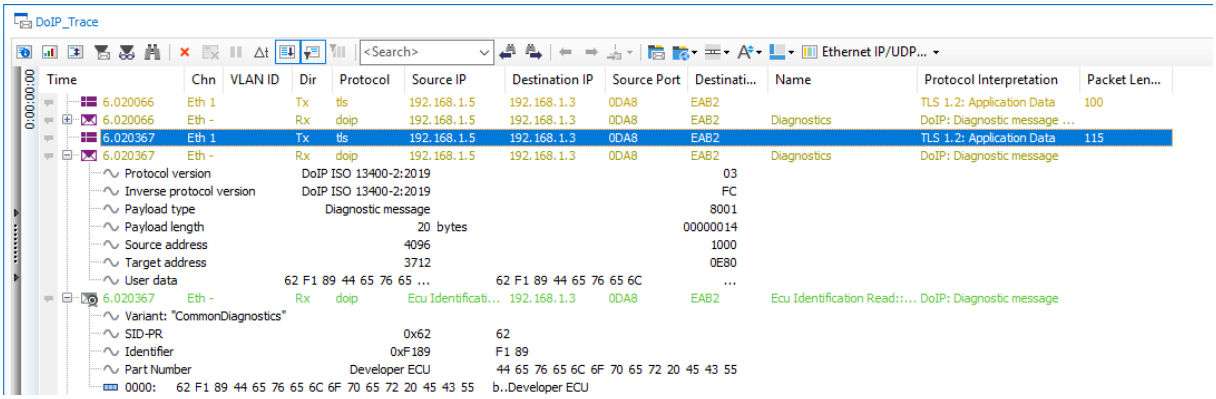


Figure 28: TLS encrypted DoIP communication with identified DoIP PDU including diagnostic interpretation of DoIP PDU

4.3 Fault injection

Whether you can use the built-in diagnostic channel of CANoe for fault injection depends on the layer into which the faults should be injected. For injecting faults on diagnostic layer, the built-in diagnostic channel is enough. You may simply send raw data using the CAPL functions diagResize() and diagSetPrimitiveData() like in the following code snippet:

```
testcase TC_FaultInjection()
{

  // unmodified request is three bytes long
  diagRequest DoIP_TLS_ECU.EcuIdentification_Read req;
  // but request should be sent with following 5 bytes
  byte rawRequestData[5] = {0x22, 0xF1, 0x89, 0x00, 0x00};

  // resize request to length of rawRequestData[]
  diagResize (req, elcount(rawRequestData));

  // overwrite request with raw request data
  diagSetPrimitiveData(req, rawRequestData, elcount(rawRequestData));

  req.SendRequest(); // send request
}
```

However, if you need to manipulate communication on lower levels, you need to implement the CCI in order to be able to use all CAPL functions the TP DLL (in this case the DoIP DLL) provides. This gives you access to a wide variety of functions, some of them violating even the DoIP protocol itself.

As this document cannot cover all CAPL functions available for DoIP, we just can explain the basic principle as an example here.

Let's assume you want to test a diagnostic tester and want your simulated ECU to respond on DoIP level using a wrong payload type. Stimulating this would not be possible on diagnostic layer and can therefore not be achieved with the built-in diagnostic channel.

The solution for this is using the CCI [3] for the network node simulating the DoIP ECU. As a preparation, you need to add the DoIP.dll as component to the network node simulating the DoIP ECU by right-clicking on the network node, selecting the context menu "Configuration…" and then adding the DLL on the tab "Components" (Figure 29). You can find the DLL in the EXEC32-folder of your

CANoe installation. This provides access to a variety of new DoIP_* CAPL functions available for this network node. For details, refer to the technical reference for the DoIP CAPL functions.

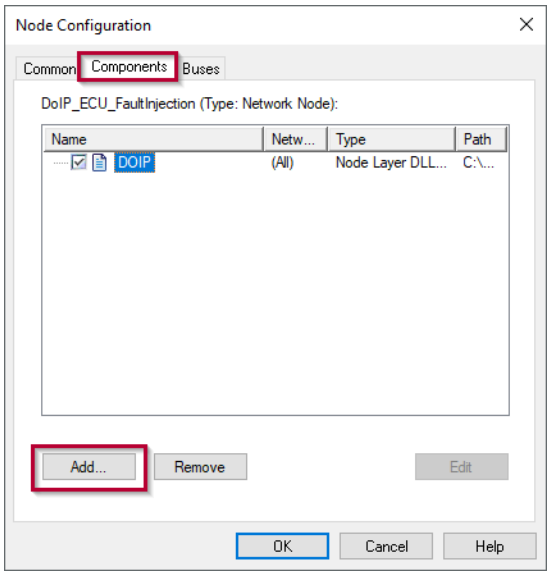


Figure 29: Adding the DoIP DLL to a network node

Note
Additionally, you should deactivate the check box “Simulation by:” for this network node in the Diagnostics/ISO TP configuration dialog, as you want to use the CCI instead of the built-in diagnostic channel for the ECU simulation in this case. Activating the check box would enable the built-in diagnostic channel for the ECU simulation.

The CAPL code for modifying the payload type is quite simple:

```
includes
{
  #include "Diagnostics\CCI_DoIP.cin" // include CCI reference implementation
}

variables
{
  char gECU[20]="DoIP_ECU";
  const cIsTester=0;
  const injectFault=1;
}

on start
{
  diagInitEcuSimulation(gECU);
}

// simple handler sending positive responses for all diagnostic requests
on diagRequest DoIP_ECU.*
{
  diagResponse this resp;

  resp.SendPositiveResponse();
}

// Callback function called by diagnostic layer to send DoIP PDUs via TCP
```

```
long _DoIP_TCPPreSend( BYTE DoIPVersion[], WORD payloadType[], DWORD payloadLen[], BYTE
payload[])
{
  write( "TCPPreSend( version=%02x, payloadtype=%04x, payloadLen=%d, [%d]<%02x ...>)"
  , DoIPVersion[0], payloadType[0], payloadLen[0], elcount( payload)
  , elcount( payload) > 0 ? payload[0] : 0);

  // If PDU to be sent is a DoIP Diagnostic message, use wrong payload type
  if (injectFault && payloadType[0]==0x8001) payloadType[0] += 0x100;

  return 0;
}
```

4.4 Multiple identical testers

Sometimes it is necessary to perform tests in parallel with multiple identical devices in the exact same way (e.g. tests in a climatic chamber with multiple ECUs). Typically, such devices for testing have the very same (default) IP address and therefore need to be operated on multiple different network segments which are not connected with each other to avoid IP address collisions. Furthermore, it is helpful if the diagnostic tester is behaving exactly in the same way in order to rule out differences in diagnostic testing in case an error occurs.

On the other hand, the CANoe TCP/IP stack, which is typically used by default for the interactive diagnostic windows, is always connected to all available Eth networks. This would lead to IP address conflicts if trying to assign the same IP address at the CANoe TCP/IP stack for each Eth network.

In this case, you need to create one “dummy” network node for each diagnostic tester instance to have an independent TCP/IP stack for every DoIP tester on every Eth network/segment. Additionally, make sure that no IP addresses at the CANoe TCP/IP stack collide with the previously configured IP addresses.

Finally, you need to configure the TCP/IP Stacks of the “dummy” network nodes at the “Adapter” setting for the testers in the Diagnostics/ISO TP configuration dialog (Figure 30). Now, you can use the very same IP address for each DoIP tester instance, testing identically configured ECUs, while every ECU is using an identical IP address as well and each ECU’s network segment is separated from the other network segments.

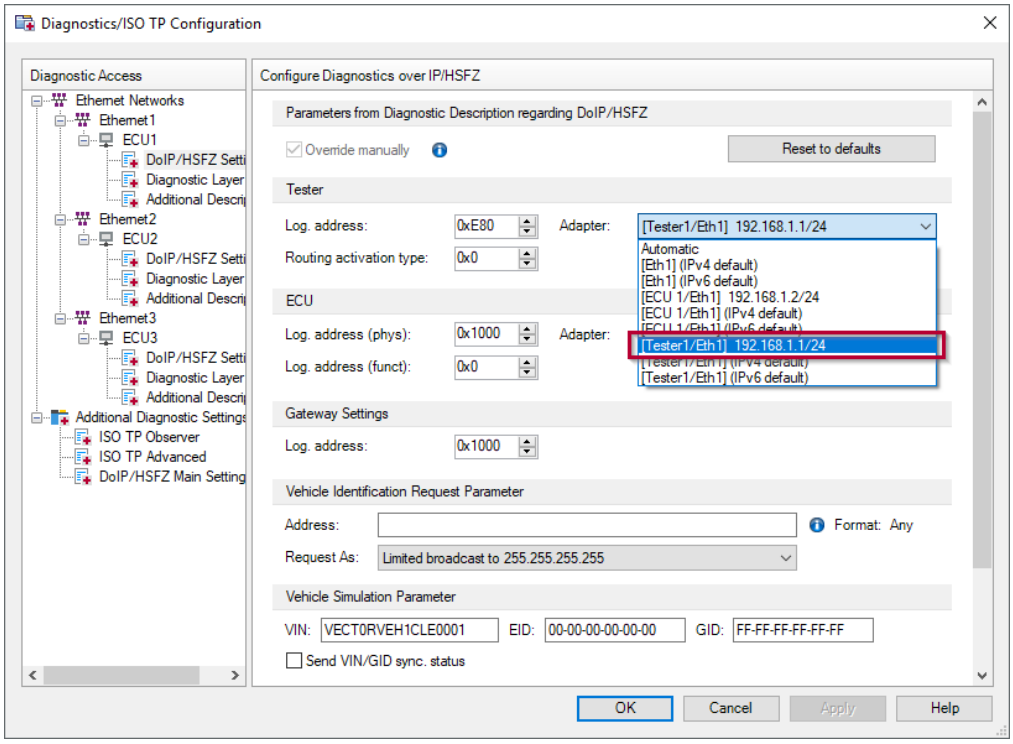


Figure 30: Configuring individual tester network nodes for each tester adapter

4.5 Car-internal communication: Reverse TCP connection setup

According to the ISO 13400 standard, the tester (diagnostics client) is responsible for initiating the TCP connection setup to the ECU (diagnostics server). However, as mentioned above, the ISO 13400 only covered the communication between tester and DoIP edge node, which led to several OEM-specific implementations for car-internal communication.

Some OEM-specific implementations require for car-internal communication, that the ECU initiates the TCP connection to the DoIP gateway. In other words, the diagnostics client (i.e. the DoIP gateway, representing the tester side) should listen to the diagnostics server for TCP connection setup. Typically, such car-internal ECUs will also not respond to vehicle identification or routing activation requests and do not even implement the DoIP Diagnostic Message Positive or Negative Acknowledgment messages as these are not necessary for direct car-internal communication.

As the implementation of such a car-internal ECU is not conform to the DoIP standard, the built-in diagnostic channel of CANoe cannot be used to connect to this ECU directly. In this case, you additionally need to implement a network node using the CCI, simulating the gateway functionality to the respective ECU.

Figure 31 shows the recommended setup for this use case.

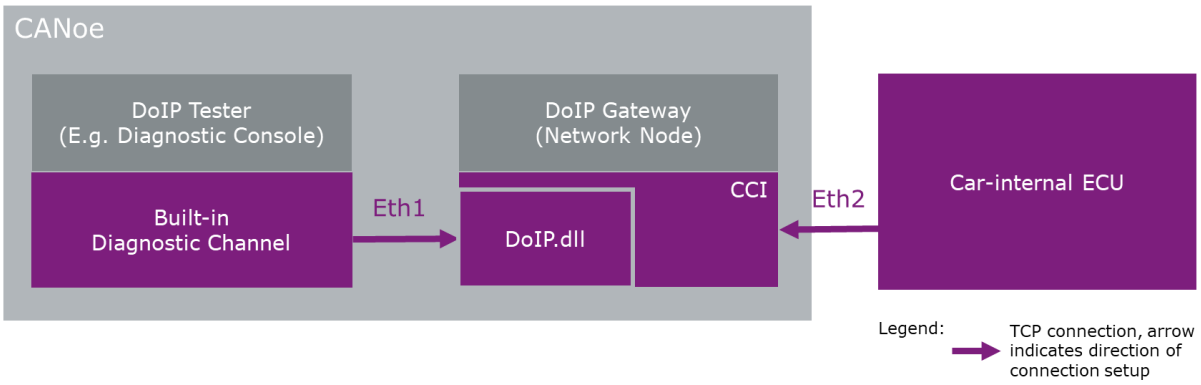


Figure 31: CANoe configuration to communicate to an ECU with reverse TCP connection setup

The CCI implementation of the DoIP Gateway network node would in this case act as a TCP Server in direction to the car-internal ECU, listening to incoming TCP connection requests on port 13400. On the DoIP Tester side, this network node uses the DoIP.dll to satisfy the DoIP protocol towards the DoIP tester. After the TCP connection to the car-internal ECU is established, the DoIP Gateway network node can forward the UDS payload of DoIP Diagnostic Messages from the DoIP tester to the car-internal ECU and vice versa. The CCI in this case needs to add the DoIP header of a DoIP Diagnostic Message to the UDS payload indicated by the DoIP.dll before forwarding it to the DoIP ECU and remove it for responses from the car-internal ECU before sending it to the tester via the DoIP.dll. You can find an example of a TCP server implementation in the help of CANoe and in the “TCP Chat” sample configuration (simulation node “ChatServer”). You will find details regarding CCI and gateway implementation in [3] and [4].

5 Abbreviations

CAPL	CAN Access Programming Language
CCI	CAPL Callback Interface
CDD	CANdela Diagnostic Description
DHCP	Dynamic Host Configuration Protocol
DoIP	Diagnostics over Internet Protocol
ECU	Electronic Control Unit
EID	Entity ID
DLL	Dynamic Link Library
GID	Group ID
HSFZ	High Speed Fahrzeugzugang (OEM specific diagnostic protocol, predecessor of DoIP)
ID	Identifier
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISO	International Standardization Organization
LAN	Local Area Network
LIN	Local Interconnect Network
MAC	Media Access Control
MDX	OEM-specific diagnostic description format

ODX	Open Diagnostic Data Exchange
OEM	Original Equipment Manufacturer
OS	Operating System
PDU	Protocol Data Unit
PDX	Packed ODX
RARC	Routing Activation Response Code
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VIN	Vehicle Identification Number
VLAN	Virtual LAN

6 References

[1] *ISO 13400-2:2019 Road vehicles — Diagnostic communication over Internet Protocol (DoIP), Part 2.*

[2] *Vector Application Note AN-IND-1-001: CANoe and CANalyzer as Diagnostic Tools.*

[3] *Vector Application Note AN-IND-1-012: CAPL Callback Interface in CANoe.*

[4] *Vector Application Note AN-IND-1-004: Diagnostics via CANoe Gateways.*

The Vector application notes mentioned above are part of the documentation that is included with every CANoe installation. You can open them from the help of CANoe under [Start Page | Quick Access | Delivered Documents | Application Areas | Diagnostics].

7 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.