Document Title	Specification of Diagnostic over IP
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	418
Document Classification	Standard
Document Version	1.2.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.03.2014	1.2.0	AUTOSAR Release Management	• Harmonization of identical APIs • Multiplicity of some configuration parameters were updated • Editorial changes
31.10.2013	1.1.0	AUTOSAR Release Management	• Formalization of Service Interfaces • Revised return values of Service Interfaces • Editorial changes
14.02.2013	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

The intent of this document is to specify the functionality, API and the configuration of the AUTOSAR Basic Software module Diagnostic over IP (DoIP).
For detailed introduction and information about DoIP please refer to ISO 13400 documents set.

AUTOSAR as SW standard can provide a standardized solution of the ISO DoIP specification in the already existing Ethernet architecture as depict in Figure 1.

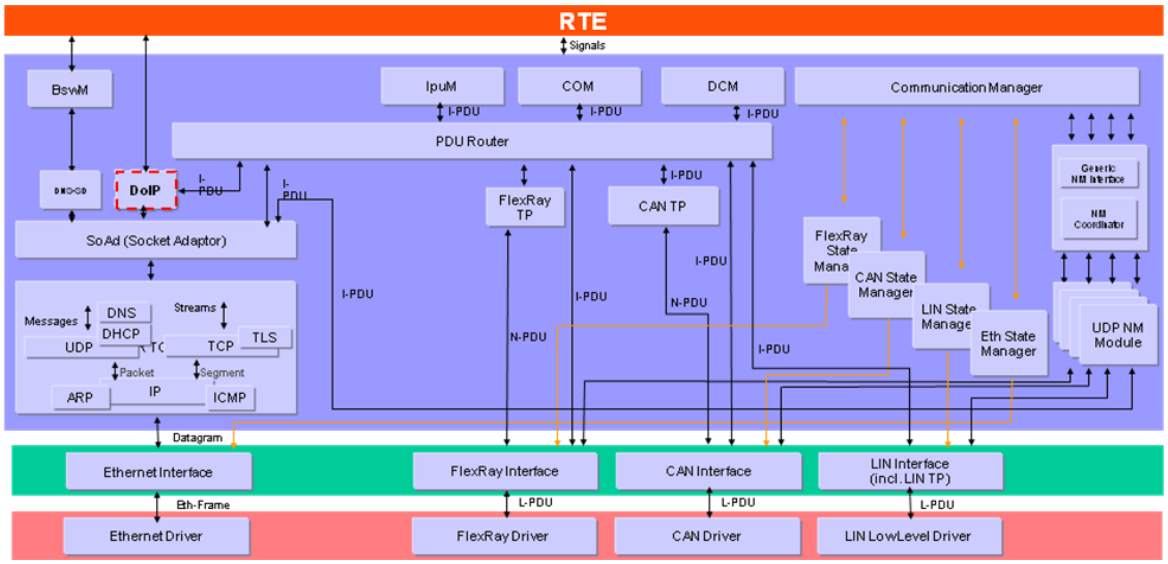


Figure 1: DoIP in the AUTOSAR ComStack Stack Architecture

AUTOSAR as SW standard can provide a standardized solution of the ISO DoIP specification in the already existing Ethernet architecture as depict in Figure 1.

2 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
ARP	Address Resolution Protocol
DHCP	Diagnostic Host Configuration Protocol
EID	Entity identifier
GID	Group identifier
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
TCP	Transmission Control Protocol
TCP/IP	A family of communication protocols used in computer networks
VIN	Vehicle Identification Number
UDP	User Datagram Protocol

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[3] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[4] Specification of Diagnostic Communication Manager
AUTOSAR_SWS_DiagnosticCommunicationManager.pdf

[5] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[6] Specification of RTE
AUTOSAR_SWS_RTE.pdf

[7] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[8] Specification of BSW Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[9] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_SRS_Ethernet.pdf

[10] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[11] Specification of Socket Adaptor
AUTOSAR_SWS_SocketAdaptor.pdf

[12] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[13] Specification of TCP/IP Stack
AUTOSAR_SWS_TCPIP.pdf

[14] AUTOSAR General Specification for Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

[15] ISO 13400-2, Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Transport protocol and network layer services

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [14] (SWS BSW General), which is also valid for the DoIP module.

Thus, the specification SWS BSW General [14] shall be considered as additional and required specification for the DoIP module.

4 Constraints and assumptions

4.1 Applicability to car domains

The DoIP basic software module may be used for all car domains.

5 Dependencies to other modules

This section describes the relations and dependencies between the DoIP module and other AUTOSAR Basic Software modules. It describes briefly the services and interfaces required from other modules and how they call the DoIP module and how they are called by the DoIP module.

5.1 Socket Adaptor (SoAd)

The Socket Adaptor [11] is the lower layer module of the DoIP module. It provides:

- Interfaces and callbacks for Socket connection establishment and notification
- Transmission of Data via multiple socket connection
- Reception of Data via multiple socket connection
- Notification on Socket status changes
- Notification on IP Address status changes

The Socket Adaptor is the interfacing module for the TCP/IP Stack [13] that supports IP, TCP, UDP,IPv4, Pv6 and address assignment mechanisms like AutoIP and DHCP.

5.2 Pdu Router (PduR)

The Pdu Router [12] is the module used by the DoIP module to connect to the rest of the communication stack. It provides:

- Forward diagnostic messages from the DoIP module to other modules (i.e. internal Dcm or other TP module)
- Forward diagnostic messages from Dcm or other TP modules to the DoIP module.

The PduR is the module to route the diagnostic message from the DoIP module to their according destination and back.

5.3 Diagnostic Communication Manager (Dcm)

The Diagnostic Communication Manager [4] is the module providing the VIN to the DoIP module. Additionally the Dcm will execute the ECU local diagnostic routed via PduR.

5.4 Development Error Tracer (Det)

If the configuration parameter DoIPDevelopmentErrorDetect is set to true and a DoIP API is called with incorrect parameters, the Development Error Tracer [7] is called with an error ID.

5.5 File structure

5.5.1 Code file structure

For details refer to chapter 5.1.6 “Code file structure” in SWS_BSWGeneral [14].

5.5.2 Header file structure

[SWS_DoIP_00158]⌈
The DoIP module shall provide the following H-files:

- `DoIP.h` (for declaration of provided interface functions)
- `DoIP_Types.h` (for public types defined by SoAd)

⌋()

[SWS_DoIP_00157]⌈
The DoIP module shall include the following H-files of other modules:

- `SoAd.h` – header file of the AUTOSAR SoAd module
- `ComStack_Types.h` [3]
- `PduR_DoIP.h` (for callback functions of the DoIP upper layer module PduR)

⌋()

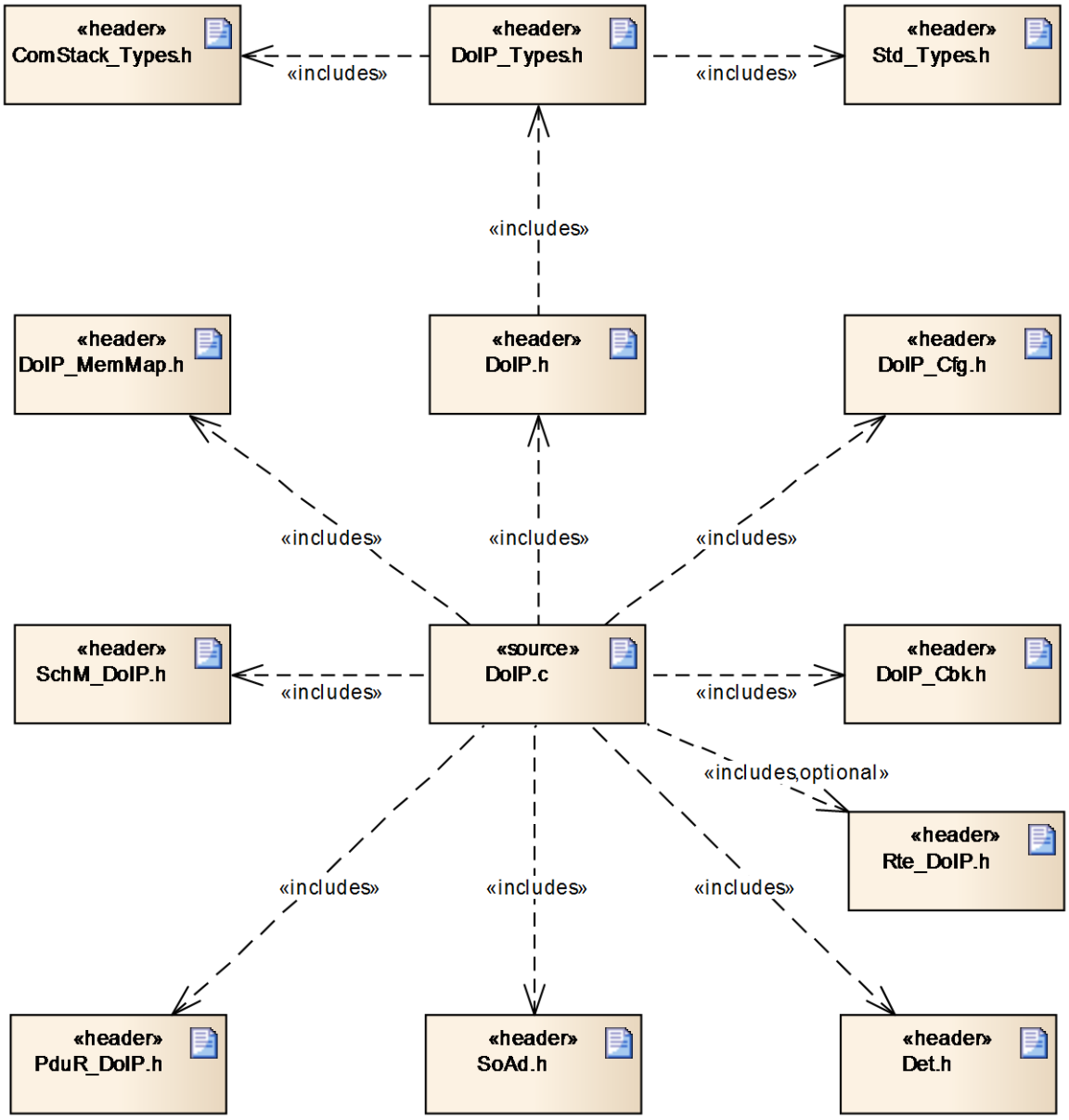


Figure 2: AUTOSAR DoIP header file structure

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_DoIP_00020
-	-	SWS_DoIP_00025
-	-	SWS_DoIP_00030
-	-	SWS_DoIP_00036
-	-	SWS_DoIP_00042
-	-	SWS_DoIP_00043
-	-	SWS_DoIP_00044
-	-	SWS_DoIP_00045
-	-	SWS_DoIP_00052
-	-	SWS_DoIP_00148
-	-	SWS_DoIP_00157
-	-	SWS_DoIP_00158
-	-	SWS_DoIP_00162
-	-	SWS_DoIP_00163
-	-	SWS_DoIP_00164
-	-	SWS_DoIP_00166
-	-	SWS_DoIP_00167
-	-	SWS_DoIP_00169
-	-	SWS_DoIP_00170
-	-	SWS_DoIP_00171
-	-	SWS_DoIP_00175
-	-	SWS_DoIP_00176
-	-	SWS_DoIP_00177
-	-	SWS_DoIP_00178
-	-	SWS_DoIP_00180
-	-	SWS_DoIP_00181
-	-	SWS_DoIP_00182
-	-	SWS_DoIP_00183
-	-	SWS_DoIP_00184
-	-	SWS_DoIP_00186
-	-	SWS_DoIP_00187
-	-	SWS_DoIP_00188
-	-	SWS_DoIP_00189
-	-	SWS_DoIP_00190
-	-	SWS_DoIP_00191

-	-	SWS_DoIP_00192
-	-	SWS_DoIP_00193
-	-	SWS_DoIP_00194
-	-	SWS_DoIP_00195
-	-	SWS_DoIP_00196
-	-	SWS_DoIP_00199
-	-	SWS_DoIP_00242
-	-	SWS_DoIP_00246
-	-	SWS_DoIP_00247
-	-	SWS_DoIP_00248
-	-	SWS_DoIP_00249
-	-	SWS_DoIP_00250
-	-	SWS_DoIP_00251
-	-	SWS_DoIP_00252
-	-	SWS_DoIP_00258
-	-	SWS_DoIP_00265
-	-	SWS_DoIP_00266
-	-	SWS_DoIP_00267
-	-	SWS_DoIP_00268
-	-	SWS_DoIP_00269
-	-	SWS_DoIP_00270
-	-	SWS_DoIP_00271
-	-	SWS_DoIP_00272
-	-	SWS_DoIP_00273
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_DoIP_00172
SRS_BSW_00376	-	SWS_DoIP_00041
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_DoIP_00172
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_DoIP_00027
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_DoIP_00027
SRS_Eth_00024	DoIP messages shall be bi-directionally routed	SWS_DoIP_00022, SWS_DoIP_00023, SWS_DoIP_00024, SWS_DoIP_00026, SWS_DoIP_00031, SWS_DoIP_00032,

		SWS_DoIP_00033, SWS_DoIP_00037, SWS_DoIP_00038, SWS_DoIP_00197, SWS_DoIP_00198, SWS_DoIP_00200, SWS_DoIP_00207, SWS_DoIP_00208, SWS_DoIP_00209, SWS_DoIP_00210, SWS_DoIP_00212, SWS_DoIP_00214, SWS_DoIP_00216, SWS_DoIP_00217, SWS_DoIP_00218, SWS_DoIP_00219, SWS_DoIP_00220, SWS_DoIP_00221, SWS_DoIP_00223, SWS_DoIP_00224, SWS_DoIP_00225, SWS_DoIP_00226, SWS_DoIP_00228, SWS_DoIP_00229, SWS_DoIP_00230, SWS_DoIP_00231, SWS_DoIP_00232, SWS_DoIP_00233, SWS_DoIP_00244, SWS_DoIP_00245, SWS_DoIP_00253, SWS_DoIP_00254, SWS_DoIP_00257, SWS_DoIP_00259, SWS_DoIP_00260
SRS_Eth_00025	-	SWS_DoIP_00004, SWS_DoIP_00005, SWS_DoIP_00006, SWS_DoIP_00007, SWS_DoIP_00008, SWS_DoIP_00009, SWS_DoIP_00010, SWS_DoIP_00012, SWS_DoIP_00013, SWS_DoIP_00014, SWS_DoIP_00016, SWS_DoIP_00017, SWS_DoIP_00018, SWS_DoIP_00019
SRS_Eth_00026	DoIP Vehicle Identification shall be provided	SWS_DoIP_00003, SWS_DoIP_00015, SWS_DoIP_00050, SWS_DoIP_00051, SWS_DoIP_00056, SWS_DoIP_00057, SWS_DoIP_00059, SWS_DoIP_00060, SWS_DoIP_00061, SWS_DoIP_00062, SWS_DoIP_00063, SWS_DoIP_00064, SWS_DoIP_00065, SWS_DoIP_00066, SWS_DoIP_00067, SWS_DoIP_00068, SWS_DoIP_00069, SWS_DoIP_00070, SWS_DoIP_00071, SWS_DoIP_00072, SWS_DoIP_00073, SWS_DoIP_00074, SWS_DoIP_00075, SWS_DoIP_00076, SWS_DoIP_00077, SWS_DoIP_00078, SWS_DoIP_00079, SWS_DoIP_00080, SWS_DoIP_00081, SWS_DoIP_00082, SWS_DoIP_00083, SWS_DoIP_00084, SWS_DoIP_00086, SWS_DoIP_00087, SWS_DoIP_00088, SWS_DoIP_00089, SWS_DoIP_00205, SWS_DoIP_00263, SWS_DoIP_00264
SRS_Eth_00027	DoIP diagnostic message shall have a format	SWS_DoIP_00121, SWS_DoIP_00122, SWS_DoIP_00123, SWS_DoIP_00124, SWS_DoIP_00125, SWS_DoIP_00126, SWS_DoIP_00127, SWS_DoIP_00128, SWS_DoIP_00129, SWS_DoIP_00130, SWS_DoIP_00131, SWS_DoIP_00132, SWS_DoIP_00133, SWS_DoIP_00134, SWS_DoIP_00135, SWS_DoIP_00136, SWS_DoIP_00137, SWS_DoIP_00138, SWS_DoIP_00173, SWS_DoIP_00174
SRS_Eth_00028	Multiple DoIP sockets shall be allowed on a single port	SWS_DoIP_00002, SWS_DoIP_00039, SWS_DoIP_00040, SWS_DoIP_00058, SWS_DoIP_00085, SWS_DoIP_00115,

		SWS_DoIP_00201, SWS_DoIP_00202, SWS_DoIP_00203, SWS_DoIP_00204, SWS_DoIP_00234, SWS_DoIP_00235, SWS_DoIP_00241, SWS_DoIP_00243
SRS_Eth_00047	DoIP shall be able to access the DHCP host name option.	SWS_DoIP_00154, SWS_DoIP_00155, SWS_DoIP_00156
SRS_Eth_00080	DoIP shall implement a mechanism to retrieve diagnostic power mode	SWS_DoIP_00047, SWS_DoIP_00054, SWS_DoIP_00090, SWS_DoIP_00091, SWS_DoIP_00092, SWS_DoIP_00093, SWS_DoIP_00261
SRS_Eth_00081	DoIP shall be able to dynamically maintain connection to different testers	SWS_DoIP_00001, SWS_DoIP_00002, SWS_DoIP_00039, SWS_DoIP_00040, SWS_DoIP_00058, SWS_DoIP_00085, SWS_DoIP_00115, SWS_DoIP_00201, SWS_DoIP_00202, SWS_DoIP_00203, SWS_DoIP_00204, SWS_DoIP_00234, SWS_DoIP_00235, SWS_DoIP_00241, SWS_DoIP_00243
SRS_Eth_00082	-	SWS_DoIP_00094, SWS_DoIP_00095, SWS_DoIP_00096, SWS_DoIP_00097, SWS_DoIP_00098, SWS_DoIP_00099, SWS_DoIP_00100
SRS_Eth_00083	-	SWS_DoIP_00058, SWS_DoIP_00105, SWS_DoIP_00107, SWS_DoIP_00115, SWS_DoIP_00139, SWS_DoIP_00140, SWS_DoIP_00141, SWS_DoIP_00142, SWS_DoIP_00143, SWS_DoIP_00144, SWS_DoIP_00145, SWS_DoIP_00146, SWS_DoIP_00159
SRS_Eth_00084	-	SWS_DoIP_00048, SWS_DoIP_00049, SWS_DoIP_00055, SWS_DoIP_00101, SWS_DoIP_00102, SWS_DoIP_00103, SWS_DoIP_00104, SWS_DoIP_00105, SWS_DoIP_00106, SWS_DoIP_00107, SWS_DoIP_00108, SWS_DoIP_00109, SWS_DoIP_00110, SWS_DoIP_00111, SWS_DoIP_00112, SWS_DoIP_00113, SWS_DoIP_00116, SWS_DoIP_00117, SWS_DoIP_00118, SWS_DoIP_00119, SWS_DoIP_00120, SWS_DoIP_00160, SWS_DoIP_00161, SWS_DoIP_00262, SWS_DoIP_00274

7 Functional specification

This specification provides the AUTOSAR representation of ISO 13400-2 as specified in the following chapters.

7.1 DoIP usage scenarios

This chapter gives only a brief overview of some use cases. For detailed information about DoIP usage scenarios please refer to ISO 13400-1.

The use cases for usage of DoIP differ from the single connection of external test equipment (see Figure 3) to a brought interconnectivity of the car or single ECUs with the environment (see Figure 4).

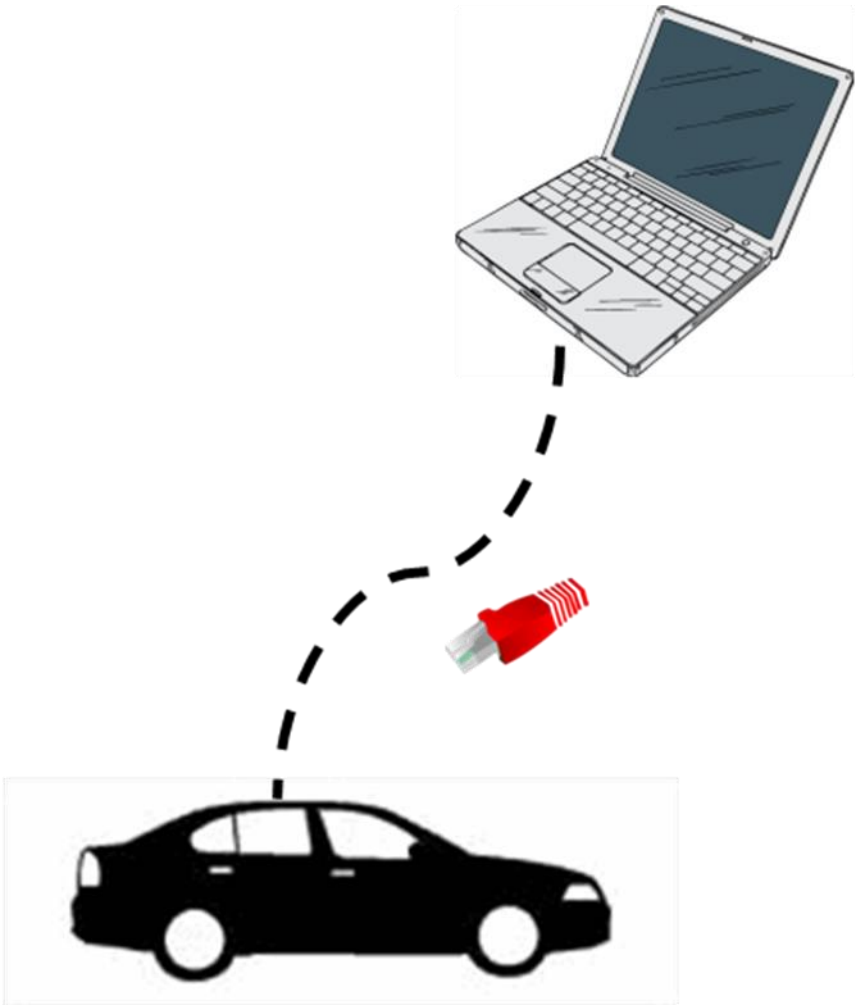


Figure 3: Connection of an external test equipment directly to the car (see ISO 13400-1 [15])

The DoIP is using for this interaction a protocol that executes several services within the single DoIP entities to fulfil the service related requirements of the DoIP ISO 13400 [15]:

Some of the DoIP services are exemplarily:

- Vehicle identification and announcement: Is necessary to detect who is participating in the DoIP communication
- Routing Activation: Allows that single Diagnostic Message pathes are activated or not to treat different protocols different (like UDS and OBD) and to also treat single testers different
- Node information: Provides general information of the single DoIP entity. Usually used by the testers to get the current DoIP protocol relevant information from the single DoIPEntities
- Alive mechanism: Is used to maintain different tester connections

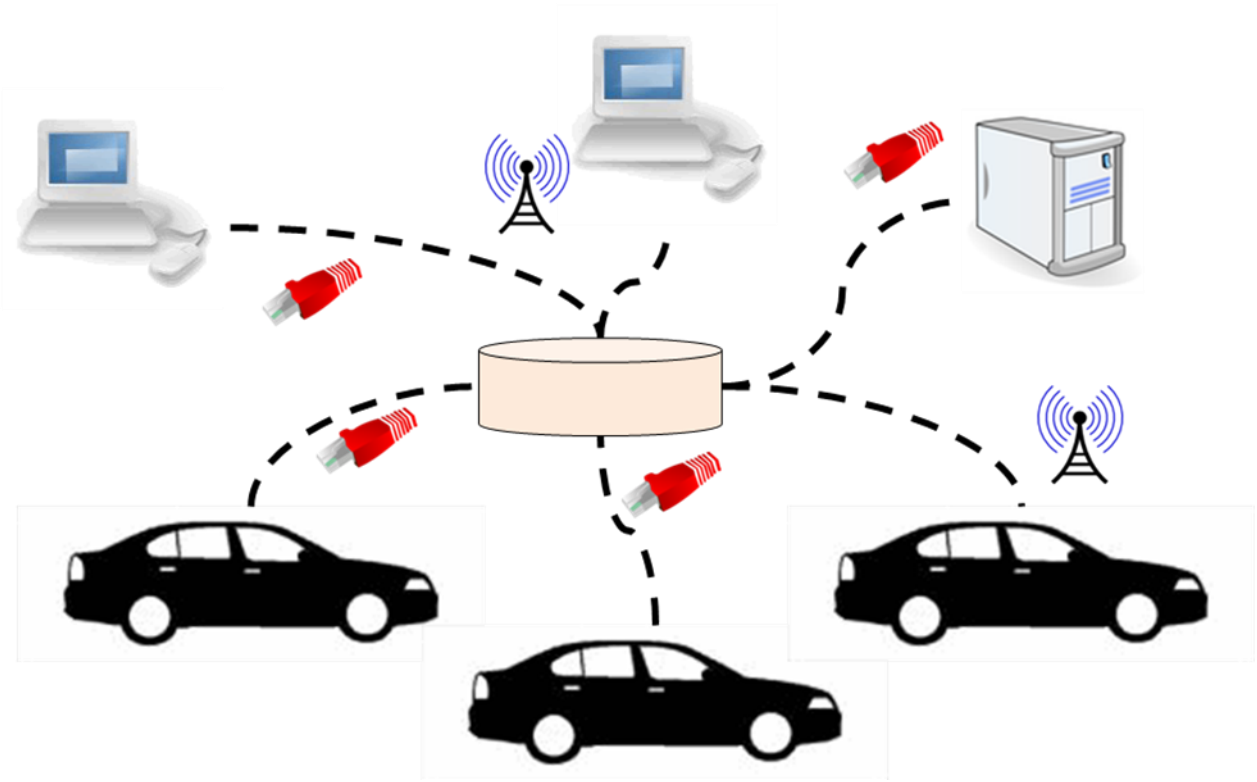


Figure 4: Highly interconnected system of several Cars via the DoIP protocol (see ISO 13400-1 [15])

7.2 Connection establishment

This chapter describes the maintenance of the socket connections of the DoIp module

[SWS_DoIP_00201]⌈ The DoIP module shall maintain the DoIP Activation Line status by the calls to DoIP_ActivationLineSwitch.
⌋(SRS_Eth_00081, SRS_Eth_00028)

Note: The API is called by the Rte or the SchM based on the Mode Switch Listening Port as described in the Chapter 8.6.4.

[SWS_DoIP_00202]⌈ If data is received from SoAd or PduR (i.e. communication related interfaces are called) as long as the DoIP Activation Line status is DOIP_ACTIVATION_LINE_INACTIVE the DoIP module shall ignore all these requests and return a negative return value as return value
⌋(SRS_Eth_00081, SRS_Eth_00028)

Note: The return value depends on the API that is called. If it is Std_ReturnType it shall return E_NOT_OK, if it is BufReq_ReturnType it shall return BUFREQ_NOT_OK.

[SWS_DoIP_00203]⌈ If the function DoIP_ActivationLineSwitch is called, the DoIP module shall call the function Rte_Mode_DoIPActivationLineSwitchNotification_CurrentDoIPActivationLineStatus to retrieve the current DoIP Activation Line status.
⌋(SRS_Eth_00081, SRS_Eth_00028)

Note: The Name of the function is derived from the DoIP Service Component that is described in the Chapter 8.6.4.

[SWS_DoIP_00204]⌈ If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_INACTIVE to DOIP_ACTIVATION_LINE_ACTIVE, the DoIP module shall retrieve the SoConId of the first configured UDPConnection, via call to the SoAd_GetSoConId and trigger the IP Address assignment via 2 subsequent calls to SoAd_RequestIpAddrAssignment with the retrieved SoConId, LocalIpAddrPtr set to NULL_PTR and in the first call type set to TCPIP_IPADDR_ASSIGNMENT_AUTOIP and in the second call type set to TCPIP_IPADDR_ASSIGNMENT_DHCP.
⌋(SRS_Eth_00081, SRS_Eth_00028)

Note: It is only necessary to trigger the IP Address assignment for one SocketId, as a valid DoIP configuration is related to exactly one Ethernet Interface, that has one IP Address but can have several valid socket connections.

[SWS_DoIP_00234]⌈ If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_ACTIVE to DOIP_ACTIVATION_LINE_INACTIVE, the DoIP module shall retrieve all the SoConId of all the configured UDPConnection, via call to the SoAd_GetSoConId and close all the UDP sockets by calls to the SoAd_CloseSoCon with the all the retrieved SoConId.
⌋(SRS_Eth_00081, SRS_Eth_00028)

[SWS_DoIP_00235]⌈ When all UDP sockets are closed (i.e for all the Sockets the function DoIP_SoConModeChg was called with something else than SOAD_SOCON_ONLINE), the DoIP module shall retrieve the SoConId of the first configured UDPConnection, via call to the SoAd_GetSoConId and release the IP

Address assignment via the call to SoAd_ReleaseIpAddrAssignment with the retrieved SoConId.

⌋(SRS_Eth_00081, SRS_Eth_00028)

Note: It is only necessary to release the IP Address assignment for one SocketId, as a valid DoIP configuration is related to exactly one Ethernet Interface, that has one IP Address but can have several valid socket connections.

[SWS_DoIP_00001]⌈

The DoIP module shall maintain the following information of the configured DoIPUDPConnection (for UDP communication):

(a) State of the SocketConnection

⌋(SRS_Eth_00081)

[SWS_DoIP_00002]⌈

The DoIP module shall be able to maintain DoIPMaxTesterConnections configured connections with the following information:

(a) DoIPSoAdRxPduId, describes the connection to the SocketConnection
(b) Source Address (SA) as soon as the information is available for the DoIP module
(c) All Routing activation status of this socket connection
(d) Status of the SocketConnection
(f) Time since last TCP communication (Rx or Tx)
(g) Information if the connection is active or not

⌋(SRS_Eth_00081, SRS_Eth_00028)

[SWS_DoIP_00241]⌈

If the DoIP module is called with DoIP_SoConModeChg and the Mode set to SOAD_SOCON_ONLINE the state of the socket connection shall be considered as online and the DoIP module shall behave as described in SWS_DoIP_00143.

⌋(SRS_Eth_00081, SRS_Eth_00028)

[SWS_DoIP_00243]⌈

If the DoIP module is called with DoIP_SoConModeChg and the Mode set to something else than SOAD_SOCON_ONLINE the state of the socket connection shall be considered as offline and the DoIP module shall behave as described in SWS_DoIP_00115.

⌋(SRS_Eth_00081, SRS_Eth_00028)

[SWS_DoIP_00003]⌈

On successful connection establishment after step SWS_DoIP_00204 (i.e. if the API DoIP_LocalIpAddrAssignmentChg is called with TcpIp_IpAddrStateType equal to TCPIP_IPADDR_STATE_ASSIGNED) the DoIP module shall open all configured UDP Socket connections by according calls to SoAd_OpenSoCon.

⌋(SRS_Eth_00026)

[SWS_DoIP_00205]⌈ If the function DoIP_SoConModeChg is called, after SWS_DoIP_00003 has been performed, for a UDP connection, the DoIP module shallsend the Vehicle announcement message via the reported socket connection as specified in chapter 7.3.2.2.

⌋(SRS_Eth_00026)

[SWS_DoIP_00058]⌈

If a connection needs to be closed based on DoIP specific behavior the DoIP module shall call the function SoAd_CloseSoCon with the parameter abort set to TRUE and the SoConId determined by a call to the function SoAd_GetSoConId with the according DoIPSoAdTxPdu. Additionally also the according inactivity timer will be stopped.

⌋(SRS_Eth_00081, SRS_Eth_00028, SRS_Eth_00083)

[SWS_DoIP_00076]⌈

If the parameter DoIPVinGIDMaster is set to true and the Container DoIPTriggerGIDSynchronization is configured, the DoIP module shall call the <User>_DoIPTriggerGIDSynchronization function (after a successful IP Address assignment as described in SWS_DoIP_000003) and repeate this call within the DoIP_MainFunction until its return value equals to E_OK or until the complete connection is closed for any other reason.

⌋(SRS_Eth_00026)

[SWS_DoIP_00085]⌈

If a change in the IP address assignment indicated by DoIP_LocalIpAddrAssignmentChg with another TCP_IpAddrStateType then TCPIP_IPADDR_STATE_ASSIGNED, the function to start GID synchronisation as described in SWS_DoIP_00076 shall not be called any longer independent from the before return value.

⌋(SRS_Eth_00028, SRS_Eth_00081)

[SWS_DoIP_00115]⌈

If a TCP socket connection gets closed (after the DoIP_SoConModeChg was called with different mode value than SOAD_SOCON_ONLINE or any other reason described by SWS_DoIP_00058) the DoIP module shall
- unregister and release the socket connection to the related Tester,
- discard the ongoing diagnostic message processing and
- reset the inactivity timer of the given socket connection.

⌋(SRS_Eth_00028, SRS_Eth_00081, SRS_Eth_00083)

Note: This includes cleaning up all the buffers/internal variables and scheduled asynchronous or pending function calls as well as reducing the amount of tester connected by 1.

[SWS_DoIP_00142]⌈
The DoIP module shall maintain an inactivity timer for each registered TCP connection.
⌋(SRS_Eth_00083)

[SWS_DoIP_00143]⌈
After a successful TCP socket connection (i.e. DoIP_SoConModeChg) the DoIP module shall start the inactivity timer.
⌋(SRS_Eth_00083)

[SWS_DoIP_00144]⌈
If no Routing Activation request was received on a new opened socket within the configured DoIPInitialInactivityTime, the DoIP module shall close the socket connection.
⌋(SRS_Eth_00083)

[SWS_DoIP_00159]⌈
If a Routing Activation request was received on a new opened socket before the inactivity timer elapsed (i.e. the configured DoIPInitialInactivityTime did not pass) the DoIP module shall reset the inactivity timer to 0.
⌋(SRS_Eth_00083)

[SWS_DoIP_00145]⌈
After a routing activation has been performed (see SWS_DoIP_00159), the DoIP module shall reset the inactivtity timer to 0 always when data communication is performed on the socket (send or receive).
⌋(SRS_Eth_00083)

[SWS_DoIP_00146]⌈
If the inactivity timer reaches the time configured in DoIPGeneralInactivityTime, the according socket connection shall be closed as described in SWS_DoIP_00058.
⌋(SRS_Eth_00083)

[SWS_DoIP_00154]⌈
If the API DoIP_LocalIpAddrAssignmentChg is called with the State set to TCPIP_IPADDR_STATE_ASSIGNED, the DoIP module shall call the function SoAd_ReadDhcpHostNameOption with the received SoConId to get the currently set host name option. The returned Byte buffer shall be considered as ASCII buffer and shall start with “DoIP-“.
⌋(SRS_Eth_00047)

[SWS_DoIP_00155]⌈

If the ASCII buffer returned in SWS_DoIP_00154 does not start with “DoIP-“ and the configuration parameter DoIPDhcpOptionVinUse is set to FALSE the DoIP module shall call the SoAd_WriteDhcpHostNameOption with a pointer to the string “DoIP-“ in order to set the hostname.

⌋(SRS_Eth_00047)

[SWS_DoIP_00156]⌈

If the ASCII buffer returned in SWS_DoIP_00154 does not start with “DoIP-“ and the configuration parameter DoIPDhcpOptionVinUse is set to TRUE the DoIP module shall call the SoAd_WriteDhcpHostNameOption with a pointer to to the ASCII buffer “DoIP-VIN<vinnumberinascii>“ with <vinnumberinascii> representing the ASCII representation of the VIN that is retrieved via Dcm_GetVin. If no valid VIN could be retrieved the DoIP shall use the configured DoIPVinInvalidityPattern in ASCII representation.

⌋(SRS_Eth_00047)

7.3 DoIP Message layout according ISO 13400-2

A DoIP message can be identified by its generic DoIP header structure, which is described in the chapter 7.3.1.

7.3.1 Generic DoIP header

All Pdus received or sent via the SoAd shall support the the DoIP header structure as defined in the ISO 13400-2 [15] table 11. The DoIP header is described in this chapter.

[SWS_DoIP_00004]⌈

The first 8 Bytes of a DoIP message shall contain the DoIP Header followed by the actual payload data.

Item	Position (Byte)	Length (Byte)
Generic DoIP header synchronization pattern		
Protocol version	0	1
Inverse protocol version	1	1
Generic DoIP payload type and payload length		
Payload type	2	2
Payload length	4	4
Payload type specific message content	8	…

Table 1: DoIP message Generic header Layout

⌋(SRS_Eth_00025)

[SWS_DoIP_00005]⌈

Byte 0 of the DoIP header has to contain the protocol version e.g. 0x02.

⌋(SRS_Eth_00025)

[SWS_DoIP_00006]⌈
Byte 1 of the DoIP header shall contain the inverse protocol version (XOR value) e.g. 0xFD for protocol version 0x02.
⌋(SRS_Eth_00025)

[SWS_DoIP_00007]⌈
Byte 2 and Byte 3 shall contain the PayloadType.
⌋(SRS_Eth_00025)

[SWS_DoIP_00008][⌈
The following PayloadTypes shall be supported for reception of DoIP messages:

Payload Type value	Payload type name	Chapter in DoIP SWS	Connection Kind
0x0001	Vehicle Identification request message	7.3.2.2.1	UDP
0x0002	Vehicle identification request message with EID	7.3.2.2.2	UDP
0x0003	Vehicle identification request message with VIN	7.3.2.2.3	UDP
0x0005	Routing activation request	7.3.2.3.1	TCP
0x0008	Alive Check response	7.3.2.4.2	TCP
0x4001	DoIP entity status request	7.3.2.5.3	UDP
0x4003	Diagnostic power mode information request	7.3.2.5.1	UDP
0x8001	Diagnostic message	7.3.2.6.1	TCP

Table 2: DoIP payload types received by a DoIP entity, chapter reference and the connection type they are received on.

⌋(SRS_Eth_00025)

[SWS_DoIP_00009]⌈
The following PayloadTypes shall be supported for sending of DoIP messages:

Payload Type value	Payload type name	Chapter in DoIP SWS	Connection Kind
0x0000	Generic DoIP header negative acknowledge	7.3.2.1	UDP/TCP
0x0004	Vehicle announcement message/vehicle identification response	7.3.2.2.4	UDP
0x0006	Routing activation response	7.3.2.3.2	TCP
0x0007	Alive Check request	7.3.2.4.1	TCP

0x4002	DoIP entity status response	7.3.2.5.4	UDP
0x4004	Diagnostic power mode information response	7.3.2.5.2	UDP
0x8002	Diagnostic message positive acknowledgement	7.3.2.6.2	TCP
0x8003	Diagnostic message negative acknowledgement	7.3.2.6.3	TCP

Table 3: DoIP payload types transmitted by a DoIP entity, chapter reference and the connection type they are transmitted on.

⌋(SRS_Eth_00025)

[SWS_DoIP_00010]⌈
Bytes 4 to 7 shall contain the payload length in Bytes not including the length of the DoIP header information (i.e. if a DoIP message is received with Payload length set to 2 it means that 10 Bytes in total were received).
⌋(SRS_Eth_00025)

7.3.2 Payload types

This chapter describes the different Payload types in detail.

7.3.2.1 Generic acknowledge

This chapter contains the check of the DoIP header with the according negative acknowledge messages with payload type 0x0000 for an invalid DoIP header.

[SWS_DoIP_00012]⌈
If an invalid DoIP header was received, a DoIP message with payload type 0x0000 shall be transmitted with the payload described in SWS_DoIP_00013 on the TxPdu which is related to the RxPdu the message was received on, if the according SocketConnection status has not changed since the reception of the DoIP message
⌋(SRS_Eth_00025)

[SWS_DoIP_00013]⌈
The payload of the generic DoIP header shall contain the corresponding NACK code (1 Byte) as specified from SWS_DoIP_00014 to SWS_DoIP_00019.
⌋(SRS_Eth_00025)

[SWS_DoIP_00014]⌈
If the Protocol information is incorrect, (see SWS_DoIP_00005, SWS_DoIP_00006 and SWS_DoIP_00015 for valid information) the NACK code 0x00 shall be sent and the according socket shall be closed (see SWS_DoIP_00058).
⌋(SRS_Eth_00025)

[SWS_DoIP_00016]⌈
If a payload type is not supported (see SWS_DoIP_00008 for valid payload types) the DoIP module shall send the NACK code 0x01 to indicate that a unkown payload type was requested. The message shall be discarded for further processing.
⌋(SRS_Eth_00025)

[SWS_DoIP_00017]⌈
If the payload length exceeds the value configured by DoIPMaxRequestBytes, the DoIP module shall send the NACK code 0x02 to indicate that the message is too large. The message shall be discarded for further processing.
⌋(SRS_Eth_00025)

[SWS_DoIP_00018]⌈
If the DoIP module is called with DoIP_SoAdTpStartOfReception() with a buffer size which exceeds the currently available DoIP buffer, the NACK code 0x03 shall be sent. The message shall be discarded for further processing.
⌋(SRS_Eth_00025)

[SWS_DoIP_00019]⌈
If the DoIP module is called with a payload length that is not valid for the specifc payload type, the NACK code 0x04 shall be sent and the according socket shall be closed (see SWS_DoIP_00058).
⌋(SRS_Eth_00025)

Note: The single valid payload length ranges for the single payload types are described in the single subchapters of the payloads (see SWS_DoIP_00008 for the list of all receive payload types and the according chapter references).

7.3.2.2 Vehicle Identification

[SWS_DoIP_00015]⌈
On a vehicle identification request the Protocol Type 0xFF and the inverse Protocol Type 0x00 shall be supported as default values, additionally to the ProtocolType described in SWS_DoIP_00005 and SWS_DoIP_00006.
⌋(SRS_Eth_00026)

7.3.2.2.1 Vehicle Identification request (payload type 0x0001)

[SWS_DoIP_00061]⌈
If a DoIP message with payload Type 0x0001 is not received on a configured DoIPUDPConnection, the message shall be.
⌋(SRS_Eth_00026)
Note: This also means that it is not allowed to receive this payload type on a TCP connection.

[SWS_DoIP_00059]⌈
The expected payload length (see SWS_DoIP_00019) for vehicle identification request message with payload type 0x0001 shall be exactly 0.
⌋(SRS_Eth_00026)

[SWS_DoIP_00060]⌈
If a DoIP message with payload Type 0x0001 is received on the configured DoIPUDPConnection, the DoIP module shall respond with a vehicle identification response/vehicle announcement message after the configured DoIPInitialVehicleAnnouncementTime with payload type 0x0004 as described inTable 6.
⌋(SRS_Eth_00026)

7.3.2.2.2 Vehicle Identification request with EID (payload type 0x0002)

The payload data structure of a vehicle identification request message with EID shall be supported as described in Table 4:

Item	Position (Byte)	Length (Byte)
Payload type vehicle identification request message with EID		
EID	0	6

Table 4: Vehicle identification request with EID payload data

[SWS_DoIP_00062]⌈
If a DoIP message with payload Type 0x0002 is not received on a configured DoIPUDPConnection, the message shall be discarded.
⌋(SRS_Eth_00026)

Note: This also means that it is not allowed to receive this payload type on a TCP connection.

[SWS_DoIP_00063]⌈
The expected payload length (see SWS_DoIP_00019) for vehicle identification request message with payload type 0x0002 shall be exactly 6.
⌋(SRS_Eth_00026)

[SWS_DoIP_00064]⌈
If a DoIP message with payload Type 0x0002 is received on the configured DoIPUDPConnection, the DoIP module shall further process the message.
⌋(SRS_Eth_00026)

[SWS_DoIP_00065]⌈
If the Parameter DoIPUseMacAdressForIdentification is set to true the received “EID” 6 payload data bytes shall be compared to the MacAddress received via

SoAd_GetPhysAddr . If they match the DoIP module shall respond with a vehicle identification response/vehicle announcement message with payload type 0x0004 as described inTable 6.

⌋(SRS_Eth_00026)

[SWS_DoIP_00066]⌈

If the Parameter DoIPUseMacAdressForIdentification is set to false the received “EID” 6 payload data bytes shall be compared to the configured DoIPEID. If they match the DoIP module shall respond with a vehicle identification response/vehicle announcement message with payload type 0x0004 as described inTable 6.

⌋(SRS_Eth_00026)

7.3.2.2.3 Vehicle Identification request with VIN (payload type 0x003)

The payload data structure of a vehicle identification request message with VIN shall be supported as described in Table 5:

Item	Position (Byte)	Length (Byte)
Payload type vehicle identification request message with VIN		
VIN	0	17

Table 5: Vehicle identification request with VIN payload data

[SWS_DoIP_00067]⌈

If a DoIP message with payload Type 0x0003 is not received on a configured DoIPUDPConnection the message shall be discarded.

⌋(SRS_Eth_00026)

Note: This also means that it is not allowed to receive this payload type on a TCP connection.

[SWS_DoIP_00068]⌈

The expected payload length (see SWS_DoIP_00019) for vehicle identification request message with payload type 0x0003, shall be exactly 17.

⌋(SRS_Eth_00026)

[SWS_DoIP_00069]⌈

If a DoIP message with payload Type 0x0003 is received on the configured DoIPUDPConnection the DoIP module shall further process the message.

⌋(SRS_Eth_00026)

[SWS_DoIP_00070]⌈

The DoIP 17 payload data bytes shall be compared to the data retrieved by the function Dcm_GetVin. If the function returns E_OK the VIN pointer is considered to contain valid information. If the function returns E_NOT_OK the invalidity value consisting of 17 Bytes with the configured DoIPVinInvalidityPattern shall be used for

the comparison. If the requested VIN matches the derived VIN the DoIP module shall respond with a vehicle identification response/vehicle announcement message with payload type 0x0004 as described inTable 6.

⌋(SRS_Eth_00026)

7.3.2.2.4 Vehicle Identification response/vehicle announcement (payload type 0x0004)

[SWS_DoIP_00071]⌈

If the DoIP module needs to send a vehicle announcement message because of SWS_DoIP_00003, it shall send the first vehicle announcement message via the configured DoIUDPConnection after DoIPInitialVehicleAnnouncementTime as described in Table 6 and repeat this message DoIPVehicleAnnouncementRepetition times with a delay of DoIPVehicleAnnouncementInterval.

⌋(SRS_Eth_00026)

The payload data structure of a vehicle identification response/vehicle announcement message shall be supported as described in Table 6.

Item	Position (Byte)	Length (Byte)
Vehicle identification number		
VIN	0	17
DoIP entity logical address information		
Logical Address	17	2
Entity identification		
EID	19	6
Group identification		
GID	25	6
Further action required	31	1
VIN/GID Status	32	1

Table 6: Vehicle identification response/vehicle announcement message payload data

[SWS_DoIP_00072]⌈

The “VIN” of a vehicle identification response/vehicle announcement message shall be derived by calling Dcm_GetVin. If Dcm_GetVin returns E_OK, the 17 Bytes in the pointer shall be used, if the callback returns E_NOT_OK the 17 Bytes shall be filled with the configured DoIPVinInvalidityPattern.

⌋(SRS_Eth_00026)

[SWS_DoIP_00073]⌈

The “LA” of a vehicle identification response/vehicle announcement message shall contain the configured DoIPLogicalAddress.

⌋(SRS_Eth_00026)

[SWS_DoIP_00074]⌈

The “EID” of a vehicle identification response/vehicle announcement message shall contain the MAC address derived by Soad_GetPhysAddr if the configuration parameter DoIPUseMacAdressForIdentification is set to true.

⌋(SRS_Eth_00026)

[SWS_DoIP_00075]⌈

The “EID” of a vehicle identification response/vehicle announcement message shall contain the configured DoIPEID if the configuration parameter DoIPUseMacAdressForIdentification is set to false.

⌋(SRS_Eth_00026)

[SWS_DoIP_00077]⌈

The “GID” of a vehicle identification response/vehicle announcement message shall contain the same value as for the EID, if both configuration parameter and DoIPUseEIDasGID are set to true (see SWS_DoIP_00074 and SWS_DoIP_00075).

⌋(SRS_Eth_00026)

[SWS_DoIP_00078]⌈

The “GID” of a vehicle identification response/vehicle announcement message shall contain the configured DoIPGID value, if the configuration parameter DoIPVinGIDMaster is set to true, the configuration parameter DoIPUseEIDasGID is set to false and the parameter DoIPGID is configured.

⌋(SRS_Eth_00026)

[SWS_DoIP_00079]⌈

The “GID” of a vehicle identification response/vehicle announcement message shall contain the value retrieved by the configured DoIPGetGID function(for the signature see <User>_DoIPGetGID, SWS_DoIP_00051), if the configuration parameter DoIPVinGIDMaster is set to true, the configuration parameter DoIPUseEIDasGID is set to false and the parameter DoIPGID is not configured. If the function does not return E_OK the GID shall consist of 6 Bytes according to the configured DoIPGIDInvalidityPattern.

⌋(SRS_Eth_00026)

[SWS_DoIP_00080]⌈

The “GID” of a vehicle identification response/vehicle announcement message shall contain the configured DoIPGID value, if the configuration parameter DoIPVinGIDMaster is set to false and the parameter DoIPGID is configured.

⌋(SRS_Eth_00026)

[SWS_DoIP_00081]⌈

The “GID” of a vehicle identification response/vehicle announcement message shall contain the value retrieved by the configured DoIPGetGID function, if the configuration parameter DoIPVinGIDMaster is set to false and the parameter

DoIPGID is not configured. If the function does not return E_OK, the GID shall consist of 6 Bytes according to the configured DoIPGIDInvalidityPattern.

⌋(SRS_Eth_00026)

[SWS_DoIP_00082]⌈

The "Further action" byte of a vehicle identification response/vehicle announcement message shall contain the value 0x10 if any DoIPRoutingActivation with DoIPRoutingActivationNumber equal to 0xE0 is configured and the according RoutingActivation was not yet successfully performed.

⌋(SRS_Eth_00026)

[SWS_DoIP_00083]⌈

The "Further action" byte of a vehicle identification response/vehicle announcement message shall contain the value 0x00, if no DoIPRoutingActivation with DoIPRoutingActivationNumber equal to 0xE0 is configured.

⌋(SRS_Eth_00026)

[SWS_DoIP_00084]⌈

The "Further action" byte of a vehicle identification response/vehicle announcement message shall contain the value 0x00, if any DoIPRoutingActivation with DoIPRoutingActivationNumber equal to 0xE0 is configured and the according RoutingActivation was successfully performed.

⌋(SRS_Eth_00026)

[SWS_DoIP_00086]⌈

If the configuration parameter DoIPUseVehicleIdentificationSyncStatus is set to true, the "VIN/GID status" byte shall be additionally added to the vehicle identification response/vehicle announcement message.

⌋(SRS_Eth_00026)

[SWS_DoIP_00087]⌈

If a valid VIN could be requested in SWS_DoIP_00072, the value of the "VIN/GID status" byte shall be 0x00.

⌋(SRS_Eth_00026)

[SWS_DoIP_00088]⌈

If no valid VIN could be requested in SWS_DoIP_00072 and the vehicle GID synchronization was not yet successful as described in SWS_DoIP_00076, the value of the "VIN/GID status" byte shall be 0x10.

⌋(SRS_Eth_00026)

[SWS_DoIP_00089]⌈

If no valid VIN could be requested in SWS_DoIP_00072 and the vehicle GID synchronization was already successful as described in SWS_DoIP_00076, the value of the “VIN/GID status” byte shall be 0x00.

⌋(SRS_Eth_00026)

7.3.2.3 Routing activation

7.3.2.3.1 Routing activation request (payload type 0x0005)

The payload data structure of a routing activation request message shall be supported as described in Table 7:

Item	Position (Byte)	Length (Byte)
External test equipment address information		
Source address	0	2
Activation Type	2	1
Reserved and OEM specific data		
Reserved by the ISO (0x00000000)	3	4
OEM specific	7	4

Table 7: Routing activation request message payload data

[SWS_DoIP_00101]⌈
If a DoIP message with payload Type 0x0005 is not received on a configured DoIPTCPConnection the message shall be discarded. ⌋(SRS_Eth_00084)

Note: That means that it is also not allowed to receive this payload type on a UDP connection,

[SWS_DoIP_00117]⌈
The expected payload length (see SWS_DoIP_00019) for Routing Activation Request Message with payload type 0x0005 shall be either exactly 7 or 11.
⌋(SRS_Eth_00084)

[SWS_DoIP_00102]⌈
If a routing activation request message is received with a valid DoIP header, the DoIP module shall process further to SWS_DoIP_00103, if the field “Source address” matches a configured DoIPTesterSA.
⌋(SRS_Eth_00084)

[SWS_DoIP_00106]⌈
If a routing activation request message is received with a valid “Source address” but the connection this Routing activation was received on is already registered to another source address, the DoIP module shall send a routing activation response message (see chapter 7.3.2.3.2) on the same connection the request was received

on, with the routing activation response code set to 0x02. Additionally the socket connection shall be closed as defined in SWS_DoIP_00058.

⌋(SRS_Eth_00084)

[SWS_DoIP_00104]⌈

If a routing activation request message is received with a “Source address” that does not match a configured DoIPTesterSA, the routing activation response message (see chapter7.3.2.3.2) shall be sent on the same connection as the received request with the routing activation response code 0x00. Additionally the socket connection shall be closed as defined in SWS_DoIP_00058.

⌋(SRS_Eth_00084)

[SWS_DoIP_00103]⌈

The DoIP module shall always continue with processing as defined in SWS_DOIP_00105, either if the received “Source Address” is already registered to a connection as described in SWS_DoIP_00002 and it is the same socket connection this routing activation request was received on, or if the received “Source Address” is not registered to a connection yet.

⌋(SRS_Eth_00084)

[SWS_DoIP_00105]⌈

If the received “Source Address” is already registered to another connection, an alive check request to this connection shall be triggered as described in chapter 7.3.2.4.1 and it shall be waiting for the alive check response message or until the time configured in parameter DoIPAliveCheckResponseTimeout expired. If the alive check response was received within the configured time, the DoIP module shall send a routing activation response message with the activation response code set to 0x01 as described in chapter 7.3.2.3.2. Additionally the socket connection shall be closed as defined in SWS_DoIP_00058. If the “Source Address” is not already registered or the DoIPAliveCheckResponseTimeout expired without receiving an alive check response message the DoIP module shall continue with SWS_DoIP_00107.

⌋(SRS_Eth_00084, SRS_Eth_00083)

[SWS_DoIP_00107]⌈

If the amount of registered connections is smaller than the configured DoIPMaxTesterConnections, the DoIP module shall proceed with the message as described in SWS_DoIP_00108 otherwise an alive check request shall be sent to all registered connections as described in chapter7.3.2.4.1. If none of the alive checks times out (i.e. all tester respond with a valid alive check response within the configured DoIPAliveCheckResponseTimeout) the DoIP module shall send a routing activation response message with the activation response code set to 0x01 as described in chapter 7.3.2.3.2. Additionally the socket connection shall be closed as defined in SWS_DoIP_00058. If at least one of them times out the DoIp module shall close the socket connection and continue as described in SWS_DoIP_00108.

⌋(SRS_Eth_00084, SRS_Eth_00083)

[SWS_DoIP_00108]⌜
If the “Activation type” bytes matches the DoIPRoutingActivationNumber of one of the DoIPRoutingActivationRef of the “Source Address” (i.e. DoIPTester has a DoIPRoutingActivationRef configured which has the DoIPRoutingActivationNumber equal to “Activation type”) the DoIP module shall proceed with SWS_DoIP_109.
⌟(SRS_Eth_00084)

[SWS_DoIP_00160]⌜
If the “Activation type” bytes do not fulfill the SWS_DOIP_00108 requirement, the DoIP module shall send a routing activation response message with the activation response code set to 0x06 as described in chapter 7.3.2.3.2. In this case the socket connection shall be closed as defined in SWS_DoIP_00058.
⌟(SRS_Eth_00084)

[SWS_DoIP_00109]⌜
If an DoIPRoutingActivationAuthenticationCallback is configured for the referenced DoIPRoutingActivation, the DoIP module shall call this callback (for the signature see <User>_DoIPRoutingActivationAuthentication, SWS_DoIP_00049). If the DoIPRoutingActivationAuthenticationReqLength is not configured to 0, the DoIP module shall handle additionally the first DoIPRoutingActivationAuthenticationReqLength bytes of the optional field “OEM specific”.
⌟(SRS_Eth_00084)

[SWS_DoIP_00161]⌜
If the DoIPRoutingActivationAuthenticationCallback returns with E_OK the routing activation authentication shall be considered as successful. If the DoIPRoutingActivationAuthenticationResLength is not set to 0 the first DoIPRoutingActivationAuthenticationResLength byte shall be attached in routing activation response message in the field “OEM specific” as described in chapter 7.3.2.3.2.
⌟(SRS_Eth_00084)

[SWS_DoIP_00110]⌜
If the DoIPRoutingActivationAuthenticationCallback returns DOIP_E_PENDING the DoIP module shall trigger the callback at next DoIP_MainFunction call again until something else than DOIP_E_PENDING is returned. Additionally the socket connection shall be considered as registered to this DoIPTesterSA without activating the routing.
⌟(SRS_Eth_00084)

[SWS_DoIP_00111]⌜
If the DoIPRoutingActivationAuthenticationCallback returns something else (e.g. E_NOT_OK) the DoIP module shall send a routing activation response message with the activation response code set to 0x04 as described in chapter 7.3.2.3.2 and the

socket connection shall be considered as registered to this DoIPTesterSA without activating the routing.

⌋(SRS_Eth_00084)

[SWS_DoIP_00112]⌈

If a DoIPRoutingActivationConfirmationCallback is configured for the referenced DoIPRoutingActivation, the DoIp module shall call this callback (for the signature see <User>_DoIPRoutingActivationConfirmation, SWS_DoIP_00048). If the DoIPRoutingActivationConfirmationReqLength is not configured to 0, the DoIP module shall handle additionally the last DoIPRoutingActivationConfirmationReqLength bytes of the optional field “OEM specific”. If the Callback returns with E_OK the routing activation confirmation shall be considered as successful and if the DoIPRoutingActivationConfirmationResLengthis not set to 0, the last DoIPRoutingActivationConfirmationResLength bytes shall be attached in routing activation response message in the field “OEM specific” as described in chapter 7.3.2.3.2.

⌋(SRS_Eth_00084)

[SWS_DoIP_00110]⌈

If the DoIPRoutingActivationConfirmationCallback returns DOIP_E_PENDING the DoIP module shall trigger the callback at next DoIP_MainFunction call again until something else than DOIP_E_PENDING is returned. Additionally the DoIP module shall send a routing activation response message with the activation response code set to 0x11 as described in chapter 7.3.2.3.2. The Routing activation shall be considered as confirmed from the moment the DoIPRoutingActivationConfirmationCallback returns E_OK.

⌋(SRS_Eth_00084)

[SWS_DoIP_00274]⌈

If the DoIPRoutingActivationConfirmationCallback returns something else (le.g. E_NOT_OK) the DoIP module shall send a routing activation response message with the activation response code set to 0x05 as described in chapter 7.3.2.3.2 and the socket connection shall be considered as registered to this DoIPTesterSA without activating the routing.

⌋(SRS_Eth_00084)

[SWS_DoIP_00113]⌈

If no response was sent because of the before mentioned checks this DoIPRoutingActivation is confirmed, authorized and valid so the DoIP module shall send a routing activation response message with the activation response code set to 0x10 as described in chapter 7.3.2.3.2 and the socket connection shall be considered as registered to this DoIPTesterSA and enable the routing for this routing activation. From now on the routing to the configured DoIPTargetAdressRef are active and valid so the diagnostic request messages related to the specified DoIPTargetAdress received via this socket connection are active.

⌋(SRS_Eth_00084)

7.3.2.3.2 Routing activation response (payload type 0x0006)

The payload data structure of a routing activation response message shall be supported as described in Table 8:

Item	Position (Byte)	Length (Byte)
External test equipment address information		
Logical Address Tester	0	2
Routing activation status information		
Logical address of DoIP entity	2	2
Routing activation response code	4	1
Reserved by ISO (0x00000000)	5	4
OEM specific	9	4

Table 8: Routing activation response message payload data

[SWS_DoIP_00116]⌈
The “Logical Address Tester” field shall be set to the Tester SA the according routing activation request message was received from.
⌋(SRS_Eth_00084)

[SWS_DoIP_00118]⌈
The “Logical Address DoIP entity” shall be set to the configured parameter DoIPLogicalAddress.
⌋(SRS_Eth_00084)

[SWS_DoIP_00119]⌈
The “Routing activation response code shall be set according to the response conditions specified in chapter7.3.2.3.1.
⌋(SRS_Eth_00084)

[SWS_DoIP_00120]⌈
The “OEM specific” field shall be filled with the optional values as defined in chapter 7.3.2.2.1. if the according DoIPRoutingActivationAuthenticationResLength and/or DoIPRoutingActivationConfirmationResLength is used.
⌋(SRS_Eth_00084)

7.3.2.4 Alive check

7.3.2.4.1 Alive check request (payload type 0x0007)

[SWS_DoIP_00139]⌈

If the DoIP module needs to send a alive check request, it shall have no payload data but only the generic DoIP header and the payload type set 0x0007.

⌋(SRS_Eth_00083)

[SWS_DoIP_00140]⌈
After sending an alive check request the DoIP module shall wait the configured time DoIPAliveCheckResponseTimeout to receive a valid alive check response as described in chapter 7.3.2.4.2. If it does not receive an alive check response, the socket connection on which the alive check request was sent shall be closed as described in SWS_DoIP_00058.

⌋(SRS_Eth_00083)

7.3.2.4.2 Alive check response (payload type 0x0008)

The payload data structure of a alive check response message shall be supported as described in Table 9:

Item	Position (Byte)	Length (Byte)
External test equipment address information		
Source address	0	2

Table 9: Alive check response message payload data

[SWS_DoIP_00141]⌈
If the received Alive check response field “SourceAddress” matches the registered Source Address of the socket connection the response was received on, the DoIP module shall do nothing. Otherwise it shall close the socket connection as described in SWS_DoIP_00058.

⌋(SRS_Eth_00083)

Note: The alive check response can always be sent (not only after an according request): With this method the test equipment can reset the inactivity time.

7.3.2.5 Node information

7.3.2.5.1 Diagnostic power mode information request (payload type 0x4003)

[SWS_DoIP_00090]⌈
If a DoIP message with payload Type 0x4003 is not received on a configured DoIPUDPConnection the message shall be discarded.

⌋(SRS_Eth_00080)

Note: This means also that it is not allowed to receive this payload type on a TCP connection.

[SWS_DoIP_00091]⌈
The expected payload length (see SWS_DoIP_00019) for diagnostic power mode information request message with payload type 0x4003 shall be exactly 0.

⌋(SRS_Eth_00080)

[SWS_DoIP_00092]⌈
After a valid Diagnostic power mode request message, the DoIP module shall send a Diagnostic Power mode information response message (see chapter 7.3.2.5.2) on the configured DoIPUDPConnection.
⌋(SRS_Eth_00080)

7.3.2.5.2 Diagnostic power mode information response (payload type 0x4004)

The payload data structure of a diagnostic power mode information response shall be supported as described in Table 10:

Item	Position (Byte)	Length (Byte)
Diagnostic Power Mode		
Diagnostic power mode	0	1

Table 10: Diagnostic power mode information response message payload data

[SWS_DoIP_00093]⌈
The “Diagnostic Power Mode” byte of diagnostic power mode information response message contains the 1 Byte value retrieved by a call to the configured DoIPPowerModeCallback (for the signature see <User>DoIPGetPowerModeStatus, SWS_DoIP_00047). If the function returns E_OK, the “Diagnostic Power Mode” shall be set to the retrieved value of PowerStateReady, otherwise it shall be set to 0x00 to indicate that the power mode is not ready.
⌋(SRS_Eth_00080)

7.3.2.5.3 Diagnostic entity status request (payload type 0x4001)

[SWS_DoIP_00094]⌈
If a DoIP message with payload Type 0x4001 is not received on a configured DoIPUDPConnection the message shall be discarded.
⌋(SRS_Eth_00082)

Note: This means also that it is not allowed to receive this payload type on a TCP connection.

[SWS_DoIP_00095]⌈
The expected payload length (see SWS_DoIP_00019) for diagnostic entity status request message with payload type 0x4001 shall be exactly 0.
⌋(SRS_Eth_00082)

[SWS_DoIP_00096]⌈

After a valid Diagnostic entity status request message, the DoIP module shall send a Diagnostic entity status response message (see chapter 7.3.2.5.4) on the configured DoIPUDPConnection.

⌋(SRS_Eth_00082)

7.3.2.5.4 Diagnostic entity status response (payload type 0x4002)

The payload data structure of a diagnostic entity status response message shall be supported as described in Table 11:

Item	Position (Byte)	Length (Byte)
DoIP Entity Status Response		
Node Type	0	1
Max open sockets	1	1
Currently open socket	2	1
Max. data size	3	4

Table 11: Diagnostic entity status response message payload data

[SWS_DoIP_00097]⌈
The “Node Type” byte of a diagnostic entity status response message shall contain the configured DoIPNodeType, whereas DOIP_GATEWAY shall be represented by 0x00 and DOIP_NODE shall be represented by 0x01.

⌋(SRS_Eth_00082)

[SWS_DoIP_00098]⌈
The “Max open sockets” byte of a diagnostic entity status response message shall contain the configured DoIPMaxTesterConnections.

⌋(SRS_Eth_00082)

[SWS_DoIP_00099]⌈
The “Currently open sockets” byte of a diagnostic entity status response message shall contain the currently active connections, based on the information described in SWS_DoIP_00002.

⌋(SRS_Eth_00082)

[SWS_DoIP_00100]⌈
The “Max data size” bytes are only supported if the configuration parameter DoIPEntityStatusMaxByteFieldUse is set to TRUE. In this case, the diagnostic entity status response message shall contain the configured DoIPMaxRequestBytes in the “Max data size” field.

⌋(SRS_Eth_00082)

7.3.2.6 Diagnostic Message

For enhanced diagnostic as well as for emissions related diagnostic communication, the DoIP module uses the same diagnostic message structure and payload types.

Additionally it provides an acknowledge mechanism to provide early feedback to the tester wether the diagnostic message was received and successfully received for the internal ECU or sent out to the target network.

7.3.2.6.1 Diagnostic message (for request and response) (payload type 0x80001)

The payload data structure of a diagnostic message shall be supported as described in Table 12:

Item	Position (Byte)	Length (Byte)
Logical address information		
Source address	0	2
Target address	2	4
Diagnostic message data		
User data	4	…

Table 12: Diagnostic message payload data

[SWS_DoIP_00121]⌈
If a DoIP message with payload Type 0x8001 is not received on a configured DoIPTcpConnection the message shall be discarded.
⌋(SRS_Eth_00027)

Note: This means also that it is not allowed to receive this payload type on a UDP connection.

[SWS_DoIP_00122]⌈
The expected payload length (see SWS_DoIP_00019) for diagnostic messages with payload type 0x8001 shall be at least 5 byte.
⌋(SRS_Eth_00027)

[SWS_DoIP_00123]⌈
If the DoIP module receives a diagnostic message with a “Source Address” (equals DoIPTesterSA) which is not registered on an established socket connection, the DoIP modules shall send a diagnostic message negative acknowledge message with the diagnostic message negative acknowledge code set to 0x02 as described in chapter 7.3.2.6.3. Additionally the socket connection shall be closed as described in SWS_DoIP_00058.
⌋(SRS_Eth_00027)

[SWS_DoIP_00124]⌈
If the DoIP module receives a diagnostic message with a “Target Addess” (equals DoIPTargetAdressValue) which is not connected via DoIPRoutingActivationRef and DoIPTargetAdressRef to the received valid DoIPTesterSA, than the DoIP module shall send a diagnostic message negative acknowledge message with the diagnostic message negative acknowledge code set to 0x03 as described in chapter 7.3.2.6.3. Additionally the message shall be discarded.

⌋(SRS_Eth_00027)

[SWS_DoIP_00125]⌈
If the DoIP module receives a diagnostic message with the payload data length in the DoIP header is set to a value bigger than DoIPMaxRequestBytes-4, than the DoIP module shall send a diagnostic message negative acknowledge message with the diagnostic message negative acknowledge code set to 0x04 as described in chapter7.3.2.6.3. Additionally the message shall be discarded.
⌋(SRS_Eth_00027)

[SWS_DoIP_00126]⌈
If the DoIP module receives a diagnostic message and SWS_DoIP_00125 does not apply but the current buffer size is not sufficient to receive the message, than the DoIP module shall send a diagnostic message negative acknowledge message with the diagnostic message negative acknowledge code set to 0x05 as described in chapter 7.3.2.6.3. Additionally the message shall be discarded.
⌋(SRS_Eth_00027)

Note: This means that the PduR_DoIPStartOfReception is not accepting the buffer.

[SWS_DoIP_00127]⌈
If the DoIP module receives a diagnostic message and the according “TargetAddress” was not activated by routing activation as described in SWS_DoIP_00113, the DoIP module shall send a diagnostic message negative acknowledge message with the diagnostic message negative acknowledge code set to 0x06 as described in chapter 7.3.2.6.3. Additionally the message shall be discarded.
⌋(SRS_Eth_00027)

[SWS_DoIP_00128]⌈
If no negative acknowledge was sent the DoIP module shall evaluate the message and forward the content (i.e. all UDS Data, not the TargetAddress and SourceAddress) to the DoIPPduRRxPdu connected to the received TargetAddress/SourceAddress combination as configured in DoIPChannel
⌋(SRS_Eth_00027)

Note: For how to proceed with the communication please refer to the TCP communication described in chapter 7.5.1

[SWS_DoIP_00174]⌈
If the PduR is not accepting the data totally (for details refer to chapter7.5.1), the DoIP module shall send a diagnostic message negative acknowledge message with the diagnostic message negative acknowledge code set to 0x08 as described in chapter 7.3.2.6.3. Additionally the message shall be discarded.
⌋(SRS_Eth_00027)

[SWS_DoIP_00129]⌈
If the PduR accepted all Data, the DoIP module shall send a diagnostic acknowledge message as described in chapter 7.3.2.6.2.
⌋(SRS_Eth_00027)

[SWS_DoIP_00130]⌈
The DoIP module will get a diagnostic response message (i.e DoIP_Transmit called with DoIPPduRTxPdu which matches to the DoIPPduRRxPdu that handled the data to the PduR) via the upper layer connection to the PduR, so it has to monitor whether the socket connection the request was received on is still established. If the socket connection has been closed, the response shall be discarded and the DoIP shall return with E_NOT_OK in the return value.
⌋(SRS_Eth_00027)

[SWS_DoIP_00131]⌈
If the DoIP module is called with DoIPPduRTxPdu in the DoIP_Transmit as described in SWS_DoIP_00130 and the according socket connection has not been closed since the reception of the according diagnostic message, the DoIP module shall prepare a diagnostic message via the according socket connection with the message layout as described in Table 12 but with the “SourceAddress” set to the DoIPTargetAdressValue of the request and the “TargetAddress” set to the DoIPTesterSA.
⌋(SRS_Eth_00027)

[SWS_DoIP_00173]⌈
The field “User data” of the SWS_DoIP_00131 message contains the actual diagnostic payload data which shall not be modified by DoIP.
⌋(SRS_Eth_00027)

Note: The reveption and transmission of diagnostic payload data is described more in detail in chapter 7.5, the diagnostic communication related part of this specification

Note: Because of enhanced diagnostic and emissions related diagnostic communication behavior, several responses to the tester could be sent out before the final response is sent. The DoIP module is not evaluating the content or the amount of responses or requests to the target address. It is just routing the diagnostic data from SoAd to PduR and back.

7.3.2.6.2 Diagnostic acknowledge message (payload type 0x8002)

The payload data structure of a diagnostic acknowledge message shall be supported as described in Table 13:

Item	Position (Byte)	Length (Byte)

Logical address information		
Source address	0	2
Target address	2	4
Diagnostic message acknowledge information		
ACK code (0x00)	4	1
Previous diagnostic message	5	…

Table 13: Diagnostic acknowledge message payload data

[SWS_DoIP_00132]⌜
If the DoIP module needs to send a diagnostic acknowledge message the “Source Address” shall be set to the according “TargetAddress” of the received message (see chapter 7.3.2.6.1).
⌟(SRS_Eth_00027)

[SWS_DoIP_00133]⌜
If the DoIP module needs to send a diagnostic acknowledge message the “Target Address” shall be set to the according “SourceAddress” of the received message (see chapter7.3.2.6.1).
⌟(SRS_Eth_00027)

[SWS_DoIP_00134]⌜
If the DoIP module needs to send a diagnostic acknowledge message the field “previous diag message” shall be filed with the number of bytes of the original request message as configured in the parameter DoIPNumByteDiagAckNack for the DoIPTester the request was received on.
⌟(SRS_Eth_00027)

7.3.2.6.3 Diagnostic negative acknowledge message (payload type 0x8003)

The payload data structure of a diagnostic negative acknowledge message shall be supported as described in Table 14:

Item	Position (Byte)	Length (Byte)
Logical address information		
Source address	0	2
Target address	2	2
Diagnostic message acknowledge information		
Diagnostic message negative acknowledge code	4	1
Previous diagnostic message	5	…

Table 14 Diagnostic negative acknowledge payload data

[SWS_DoIP_00135]⌜
If the DoIP module needs to send a diagnostic negative acknowledge message the “Source Address” shall be set to the according “TargetAddress” of the received message (see chapter7.3.2.6.1).

⌋(SRS_Eth_00027)

[SWS_DoIP_00136]⌈
If the DoIP module needs to send a diagnostic negative acknowledge message the “Target Address” shall be set to the according “SourceAddress” of the received message (see chapter7.3.2.6.1).
⌋(SRS_Eth_00027)

[SWS_DoIP_00137]⌈
If the DoIP module needs to send a diagnostic negative acknowledge message, the “Diagnostic message negative acknowledge code” shall be set to the value specified by the specification item that is triggering the diagnostic negative acknowledge message.
⌋(SRS_Eth_00027)

[SWS_DoIP_00138]⌈
If the DoIP module needs to send a diagnostic negative acknowledge message the field “previous diag message” shall be filed with the configured number of the original request message as configured in the parameter DoIPNumByteDiagAckNack for the DoIPTester the request was received on.
⌋(SRS_Eth_00027)

7.4 UDP communication

DoIP messages that are communicated via UDP connection are communicated on the SoAd Interface APIs. So all messages which are received via UDP as described in Table 2 and sent via UDP as described in Table 3 shall be treated as described in this chapter.

[SWS_DoIP_00197]⌈ If the SoAd calls the DoIP module via the Interface DoIP_SoAdIfRxIndication, the DoIP module shall copy the message into the internal UDP buffer for further processing.
⌋(SRS_Eth_00024)

Note: Further processing depends on the header information and on the payload type. For details refer to chapter 7.3.2. Which messages are expected to be received on UDP connection is described in Table 2.

[SWS_DoIP_00198]⌈ If the DoIP module shall send a DoIP message via UDP it shall call the SoAd_IfTransmit with the SoAdSrcPduId set to the SoAd internal TxPduId that is retrieved via the according configured DoIPSoAdTxPduRef, the SoAdSrcPduInfoPtr shall contain the length of the message and the pointer to the to be transmitted message buffer and additionally the buffer shall be locked.
⌋(SRS_Eth_00024)

Note: The events that lead to the sending of UDP DoIP messages are described in the rest of the specification. Which DoIP message shall use UDP connection is described in Table 3.

[SWS_DoIP_00199]⌈ If the SoAd calls the DoIP module via the Interface DoIP_SoAdIfTxConfirmation, the DoIP module shall release the buffer which is related to the received TxPduId.

⌋()

7.5 TCP communication

DoIP messages that are communicated via TCP connection are communicated on the SoAd Tp APIs. So all messages which are received via TCP as described in Table 2 and sent via TCP as described in Table 3 shall be treated as described in this chapter.

7.5.1 Reception of a TCP DoIP message

[SWS_DoIP_00207]⌈
If the function DoIP_SoAdTpStartOfReception is called with TpSduLength set to 0, the DoIP module shall fill in the bufferSizePtr the available buffer size in the DoIP for the reception of the TCP message, lock the according buffer for other TCP connections and return BUFREQ_OK.

⌋(SRS_Eth_00024)

Note: The API will be called from SoAd only once per TCP connection, directly when the socket is connected. All the data will be transferred to DoIP via the API DoIP_SoAdTpCopyRxData.

[SWS_DoIP_00208]⌈
If the function DoIP_SoAdTpCopyRxData is called at the start of a new DoIP message (e.g. directly after DoIPSoAdTpStartOfReception succeeded or previous DoIP message processed completely) with PduInfoPtr.SduLength set to 0 the DoIP module shall return in the parameter bufferSizePtr the length to the maximum necessary bytes to evaluate the DoIP relevant data for routing of diagnostic data.

⌋(SRS_Eth_00024)

Note: The DoIP module knows internal when a new DoIP message is started because of the DoIP protocol payload length information (see chapter Generic DoIP header 7.3.1).

[SWS_DoIP_00209]⌈

If the function DoIP_SoAdTpCopyRxData is called at the start of a new DoIP message (e.g. directly after DoIPSoAdTpStartOfReception succeeded or previous DoIP message processed completely) with PduInfoPtr.SduLength is not set to 0 and the DoIP TCP buffer is big enough to copy all the data, the DoIP module shall copy the received data to the internal TCP buffer, return the parameter bufferSizePtr set to the available buffer after copying and return BUFREQ_OK.

⌋(SRS_Eth_00024)

[SWS_DoIP_00210]⌈

If the function DoIP_SoAdTpCopyRxData is called at the start of a new DoIP message (e.g. directly after DoIPSoAdTpStartOfReception succeeded or previous DoIP message processed completely) with PduInfoPtr.SduLength is not set to 0 and the DoIP TCP buffer is not big enough to copy all the data, the DoIP module shall return BUFREQ_E_NOT_OK.

⌋(SRS_Eth_00024)

[SWS_DoIP_00214]⌈

If the DoIP module has received sufficient data to evaluate the DoIP header and the payload type is not diagnostic message the DoIP shall copy all data of this DoIP message to the internal DoIP TCP buffer, lock the according buffer for other TCP connections and process the DoIP message as described in SWS_DoIP_00219.

⌋(SRS_Eth_00024)

Note: The length of the DoIP message is encoded in the DoIP header. It has to be considered that after the first DoIP message, there can be more in one single TCP stream.

[SWS_DoIP_00212]⌈

If the DoIP module has received sufficient data to evaluate the DoIP header, the payload type is diagnostic message and the Routing was already activated for the SourceAddress/TargetAddress combination, the DoIP module shall call the PduR_DoIPStartOfReception with the according id set to the DoIPPduRRxPduId matching the SourceAddress/TargetAddress combination of the diagnostic message, set the info.SduLength to the already received diagnostic data, set the info->SduDataPtr to the buffer containing the received diagnostic data and set the TpSduLength to the total size of the diagnostic message extracted from DoIP Header.

⌋(SRS_Eth_00024)

Note: For the SourceAddress/TargetAddress combinations refer to configuration container DoIPChannel.

[SWS_DoIP_00260]⌈

If PduR_DoIPStartOfReception returns BUFREQ_OK the reception was accepted and the DoIP module shall forward already received data of the diagnostic message to the upper layer by subsequent calls to PduR_DoIPCopyRxData.

⌋(SRS_Eth_00024)

[SWS_DoIP_00218]⌈
If PduR_DoIPStartOfReception returns BUFREQ_OK the reception was accepted and the DoIP shall forward all subsequent calls to DoIP_SoAdTpCopyRxData directly to PduR_DoIPCopyRxData until all diagnostic data was handed to the PduR.⌋(SRS_Eth_00024)

[SWS_DoIP_00259]⌈
At the end of the copy procedure via PduR_DoIPCopyRxData to PduR, the DoIP module has to modify the available buffer size pointer returned to SoAd in order to stop before the next DoIP header.
⌋(SRS_Eth_00024)

[SWS_DoIP_00253]⌈
If the buffer size reported by PduR_DoIPStartOfReception does not suffice for already received data, DoIP shall abort the reception and call PduR_DoIPRxIndication with E_NOT_OK.
⌋(SRS_Eth_00024)

[SWS_DoIP_00216]⌈
If PduR_DoIPStartOfReception returns BUFREQ_E_NOT_OK or BUFREQ_E_OVFL, the DoIP module shall react as described in SWS_DoIP_00174 and discard all the TCP data until the next DoIP message.
⌋(SRS_Eth_00024)

[SWS_DoIP_00217]⌈
If PduR_DoIPCopyRxData returns BUFREQ_E_NOT_OK, the DoIP module shall react as described in SWS_DoIP_00174, discard all the TCP data until the next DoIP message and call the PduR_DoIPRxIndication with the according PduId and the result set to E_NOT_OK.
⌋(SRS_Eth_00024)

[SWS_DoIP_00221]⌈
If all diagnostic data was successfully forwarded to the PduR (see SWS_DoIP_00216) the DoIP module shall call the PduR_DoIPRxIndication with the according PduId and the result set to E_OK.
⌋(SRS_Eth_00024)

[SWS_DoIP_00219]⌈
If the DoIP module has received with the DoIP_SoAdTpCopyRxData operations enough data to evaluate the DoIP header and the payload type is not diagnostic

message (see SWS_DoIP_00214), the DoIP module shall receive via subsequent calls to DoIP_SoAdTpCopyRxData all data for the DoIP message and process it.

⌋(SRS_Eth_00024)

Note: The possible DoIP messages on TCP are described in Table 2 and in the according chapters in this specification.

[SWS_DoIP_00200]⌈

If the function DoIP_SoAdTpRxIndication is called the DoIp module shall release all data connected to the reception and forward the result to PduR_DoIPRxIndication if a reception for diagnostic message is currently ongoing.

⌋(SRS_Eth_00024)

Note: The function DoIP_SoAdTpRxIndication is only called once when the socket is closed.

[SWS_DoIP_00258]⌈

If the DoIP module is called with DoIP_CancelReceive, the DoIP module shall call the SoAd_TpCancelReceive function with the according SoAdRxPduId.

⌋()

7.5.2 Transmission of a TCP DoIP message

[SWS_DoIP_00220]⌈

If the DoIP module needs to send a DoIP message that is not a diagnostic message on the TCP connection, the DoIP shall call the SoAd_TpTransmit with the SoAdSrcPduId containing the Id of the according socket, the SoAdSrcPduInfoPtr.SduLength set to the size of the data to be transmitted and lock the buffer to send.

⌋(SRS_Eth_00024)

Note: If the call to SoAd_TpTransmit returns E_OK the DoIP module shall consider that the data will be transmitted by subsequent calls to the DoIP_SoAdTpCopyTxData.

[SWS_DoIP_00223]⌈

If the call to SoAd_TpTransmit returns E_NOT_OK the DoIP module shall discard the DoIP message.

⌋(SRS_Eth_00024)

[SWS_DoIP_00224]⌈

If the function DoIP_SoAdCopyTxData is called after a sucessfull call to SoAd_TpTransmit, with a valid TxPduId and the PduInfoPtr.SduLength is set to 0 the

DoIP shall return BUFREQ_OK and set the parameter availableDataPtr to the total available data size of the current DoIP message to be transmitted.

⌋(SRS_Eth_00024)

[SWS_DoIP_00225]⌈
If the function DoIP_SoAdCopyTxData is called after a sucessfull call to SoAd_TpTransmit, with a valid TxPduId and the PduInfoPtr.SduLength is not set to 0, the DoIP module shall copy the bytes specified in the PduInfoPtr.SduLength to the PduInfoPtr->SduDataPtr, return BUFREQ_OK and set the parameter availableDataPtr to the total available data size of the current DoIP message after the copy process.

⌋(SRS_Eth_00024)

[SWS_DoIP_00229]⌈
If the function DoIP_SoAdTpTxConfirmation is called the DoIP module shall release the buffer related to the TxPduId. ⌋(SRS_Eth_00024)

[SWS_DoIP_00230]⌈
If the function DoIP_Transmit is called and the data package is allowed to be sent according to the current DoIP protocol related information, the DoIP module shall return E_OK, assemble the DoIP header considering the information of the handed DoIPPduRTxInfoPtr.SduLength and call the SoAd_TpTransmit function with the SoAdSrcPduId set to the according PduId of the socket connection, the SoAdSrcPduInfoPtr.SduLength set to the sum of the following lengthes: DoIP header (8 Byte), the DoIP diagnostic message specific data (4 Byte) and received length of the call to DoIP_Transmit (DoIPPduRTxInfoPtr.SduLength).

⌋(SRS_Eth_00024)

[SWS_DoIP_00226]⌈
If the function DoIP_Transmit is called and the data package is not allowed according to the current DoIP protocol related information, the DoIP module shall return E_NOT_OK.

⌋(SRS_Eth_00024)

Note: If the function SoAd_TpTransmit returns for the use case “diagnostic message” E_OK, the DoIP module shall consider that the data will be transmitted by subsequent calls to the DoIP_SoAdTpCopyTxData.

[SWS_DoIP_00228]⌈ If the call to SoAd_TpTransmit returns for the use case “diagnostic message” E_NOT_OK the DoIP module shall discard the DoIP message and call the PduR_DoIPTxConfirmation with result set to E_NOT_OK.

⌋(SRS_Eth_00024)

[SWS_DoIP_00231]⌈

If the function DoIP_SoAdCopyTxData is called after a sucessfull call to SoAd_TpTransmit for the use case “diagnostic message”, with a valid TxPduId and the PduInfoPtr.SduLength is set to 0 the DoIP shall return BUFREQ_OK and set the parameter availableDataPtr to the total available data size of the current buffered DoIP message to be transmitted.

⌋(SRS_Eth_00024)

Note: This means that only the length for the created DoIP header and the diagnostic SourceAddress/TargetAddress is returned and not the total data length.

[SWS_DoIP_00232]⌈

If the function DoIP_SoAdCopyTxData is called after a sucessfull call to SoAd_TpTransmit for the use case “diagnostic message” with a valid TxPduId and the PduInfoPtr.SduLength is not set to 0, the DoIP module shall copy the bytes specified in the PduInfoPtr.SduLength to the PduInfoPtr->SduDataPtr. If the requested bytes are more than in the DoIP internal buffer, the DoIP shall call the PduR_CopyTxData with the PduInfoPtr.SduLength set to the remaining requested data bytes and the PduInfoPtr-> SduDataPtr set to the position where the PduR shall continue to copy the data.

⌋(SRS_Eth_00024)

[SWS_DoIP_00254]⌈

If the call to PduR_CopyTxData returns BUFREQ_OK or all the requested data was part of the DoIP internal buffer, the DoIP module shall return BUFREQ_OK and set the parameter availableDataPtr to the remaining data size of the DoIP header and diagnostic SourceAddress/TargetAddress if they have not been copied completely or to the remaining data size returned from PduR_CopyTxData.

⌋(SRS_Eth_00024)

[SWS_DoIP_00233]⌈

If the DoIP module has copied via subsequent calls to DoIP_SoAdTpCopyTxData for the use case “diagnostic message” all information of the DoIP header and the diagnostic SourceAddress/TargetAddress, the DoIP module shall forward all subsequent calls to DoIP_SoAdTpCopyTxData/DoIP_SoAdTpTxConfirmation for this transmission directly to the PduR_DoIPCopyTxData/PduR_DoIPTxConfirmation and release the internal buffer for this transmission.

⌋(SRS_Eth_00024)

[SWS_DoIP_00257]⌈

If the DoIP module is called with DoIP_CancelTransmit, the DoIP module shall call the SoAd_TpCancelTransmit function of the according SoAdTxPduId.

⌋(SRS_Eth_00024))

7.6 Error classification

[SWS_DoIP_00148]⌈

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service call without module initialization	Development	DOIP_E_UNINIT	0x01
NULL-Pointer on any API call	Development	DOIP_E_PARAM_POINTER	0x02
Wrong Lower Layer (SoaAd) or Upper Layer (PduRouter) Id received	Development	DOIP_E_INVALID_PDU_SDU_ID	0x03
API call with invalid Parameter	Development	DOIP_E_INVALID_PARAMETER	0x04

⌋()

8 API specification

8.1 Imported types

The follwing types shall be imported by the DoIP module from the modules given:
[SWS_DoIP_00020]:⌜

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
	PduIdType
	PduLengthType
	RetryInfoType
	PduInfoType
SoAd	SoAd_SoConIdType
	SoAd_SoConModeType
Std_Types	Std_ReturnType
	Std_VersionInfoType
TcpIp	TcpIp_IpAddrAssignmentType
	TcpIp_IpAddrStateType
	TcpIp_SockAddrType

⌟()

The following types are contained in the Rte_DoIP_Type.h header file, which is generated by the RTE generator:

[SWS_DoIP_00266]⌜

Name	DoIP_PowerStateType		
Kind	Type		
Derived from	uint8		
Description	Used for handling of the PowerMode in DoIP entity status requests		
Range	DOIP_NOT_READY	0x00	DoIP Power Mode "not ready"
	DOIP_READY	0x01	DoIP Power Mode "ready"
	DOIP_NOT_SUPPORTED	0x02	DoIP Power Mode "not supported"
	0x03-0xFF	0x03-0xFF	Reserved
Variation	--		

⌟()

[SWS_DoIP_00267]⌜

Name	AuthenticationReqDataType_{Name}		
Kind	Array	Element type	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationAuthenticationCallback. DoIPRoutingActivationAuthenticationReqLength)} Elements		
Description	--		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		

⌟()

[SWS_DoIP_00268]⌜

Name	AuthenticationResDataType_{Name}		
Kind	Array	Element type	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationAuthenticationCallback. DoIPRoutingActivationAuthenticationResLength)} Elements		
Description	--		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		

⌟()

[SWS_DoIP_00269]⌜

Name	ConfirmationReqDataType_{Name}		
Kind	Array	Element type	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationConfirmationCallback. DoIPRoutingActivationConfirmationReqLength)} Elements		
Description	--		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		

⌟()

[SWS_DoIP_00270]⌜

Name	ConfirmationResDataType_{Name}		
Kind	Array	Element type	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationConfirmationCallback. DoIPRoutingActivationConfirmationResLength)} Elements		
Description	--		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		

⌋()

[SWS_DoIP_00271]⌈

Name	DoIP_ActivationLineType
Kind	ModeDeclarationGroup
Initial mode	DOIP_ACTIVATION_LINE_INACTIVE
Modes	DOIP_ACTIVATION_LINE_INACTIVE
	DOIP_ACTIVATION_LINE_ACTIVE
Description	--

⌋()

8.2 Type definitions

[SWS_DoIP_00272]⌈ The value of DOIP_E_PENDING shall be 0x10.
⌋()

[SWS_DoIP_00273]⌈ DOIP_E_PENDING shall be defined within DoIP_Types.h to ensure compatibility.
⌋()

The following Data Types shall be used for the functions defined in this specification.

8.2.1 DoIP_ConfigType

[SWS_DoIP_00025]⌈

Name:	`DoIP_ConfigType`	
Type:	`Structure`	
Range:	`Implementation specific`	The content of the configuration data structure is implementation specific
Description:	Configuration data structure of the DoIP module	

⌋()

8.3 Function definitions

This chapter contains a list of functions provided to upper layer modules.

8.3.1 DoIP_Transmit

[SWS_DoIP_00022]⌈

Service name:	DoIP_Transmit	
Syntax:	`Std_ReturnType DoIP_Transmit( PduIdType DoIPPduRTxId, const PduInfoType* DoIPPduRTxInfoPtr )`	
Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	DoIPPduRTxId	DoIP unique identifier of the PDU to be transmitted by the PduR
	DoIPPduRTxInfoPtr	Tx Pdu information structure which contains the length of the DoIPTxMessage.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted, e.g. parameter check has failed or no resources are available for transmission
Description:	This service is called to request the transfer data from the PduRouter to the SoAd. It is used to indicate the transmission which will be performed in the DoIP_Mainfunction. Within the provided DoIPPduRTxInfoPtr only SduLength is valid (no data)! If this function returns E_OK then the DoIP module will raise a subsequent call to PduR_DoIPCopyTxData in order to get the data to send.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00162]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT` and return `E_NOT_OK`.

⌋()

[SWS_DoIP_00163]⌈
If development error detection is enabled: The function shall check if the `DoIPPduRTxId` matches a configured `DoIPPduRTxPduId`. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID` and return `E_NOT_OK`.
⌋()

[SWS_DoIP_00164]⌈
If development error detection is enabled: The function shall check if the `DoIPPduRTxInfoPtr` is not a `NULL_PTR`. If the check fails the function shall raise the development error `DOIP_E_PARAM_POINTER` and return `E_NOT_OK`.
⌋()

8.3.2 DoIP_CancelTransmit

[SWS_DoIP_00023]⌈

Service name:	DoIP_CancelTransmit	
Syntax:	`Std_ReturnType DoIP_CancelTransmit( PduIdType DoIPPduRTxId )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	DoIPPduRTxId	DoIP unique identifier of the PDU to be canceled by the PduR
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Transmit cancellation request of the specified DoIPPduRTxId is accepted. E_NOT_OK: The transmit cancellation request of the DoIPPduRTxId has been rejected.
Description:	This service primitive is used to cancel the transfer of pending DoIPPduRTxIds. The connection is identified by DoIPPduRTxId. When the function returns, no transmission is in progress anymore with the given DoIPPduRTxId identifier.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00166]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT` and return `E_NOT_OK`.
⌋()

[SWS_DoIP_00167]⌈
If development error detection is enabled: The function shall check if the DoIPPduRTxId matches a configured DoIPPduRTxPduId. If the check fails the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.
⌋()

8.3.3 DoIP_CancelReceive

[SWS_DoIP_00024]⌈

Service name:	DoIP_CancelReceive	
Syntax:	`Std_ReturnType DoIP_CancelReceive( PduIdType DoIPPduRRxId )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	DoIPPduRRxId	DoIP unique identifier of the PDU for which reception shall be canceled by the PduR
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Reception was canceled successfully E_NOT_OK: Reception was not canceled
Description:	By calling this API with the corresponding DoIPPduRRxId the currently ongoing data reception is terminated immediately. When the function returns, no reception is in progress anymore with the given DoIPPduRRxId identifier.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00169]⌈
If development error detection is enabled: The function shall check that the service DoIP_Init was previously called. If the check fails, the function shall raise the development error DOIP_E_UNINIT and return E_NOT_OK.
⌋()

[SWS_DoIP_00170]⌈
If development error detection is enabled: The function shall check if the DoIPPduRRxId matches a configured DoIPPduRRxPduId. If the check fails the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.
⌋()

8.3.4 DoIP_Init

[SWS_DoIP_00026]⌈

Service name:	DoIP_Init	

Syntax:	`void DoIP_Init(`
	`  const DoIP_ConfigType* DoIPConfigPtr`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	DoIPConfigPtr	Pointer to the configuration data of the DoIP module
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service initializes all global variables of the DoIP module. After return of this service the DoIP module is operational.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00171]]⌈

If development error detection is enabled: The function shall check if the `DoIPConfigPtr` is not a `NULL_PTR`. If the check fails the function shall raise the development error `DOIP_E_PARAM_POINTER`.

⌋()

8.3.5 DoIP_GetVersionInfo

[SWS_DoIP_00027]⌈

Service name:	DoIP_GetVersionInfo	

Syntax:	`void DoIP_GetVersionInfo(`
	`  Std_VersionInfoType* versioninfo`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

⌋(SRS_BSW_00407, SRS_BSW_00411)

[SWS_DoIP_00172]⌈

If development error detection is enabled: The function shall check if the `versioninfo` is not a `NULL_PTR`. If the check fails the function shall raise the development error `DOIP_E_PARAM_POINTER`.

⌋((SRS_BSW_00323, SRS_BSW_00386)

[SWS_DoIP_00030]⌈
If source code for caller and callee of `DoIP_GetVersionInfo` is available, the DoIP module should realize `DoIP_GetVersionInfo` as a macro, defined in the module's header file.
⌋()

8.4 Call-back notifications

In AUTOSAR, the functions a module provides to layers which are placed below the module in the AUTOSAR software layer model, are called 'call-back functions'. Generally, a software entity A (DoIP), which, in order to be informed about some event C in software entity B (SoAd), is registered as interested in event C at software entity B by calling a register mechanism B provides, and is called by entity B if event C occurs.

This chapter contains a list of Call-Back functions which are called by the lower layer SoAd module.

8.4.1 DoIP_SoAdTpCopyTxData

[SWS_DoIP_00031]⌈

Service name:	DoIP_SoAdTpCopyTxData	

Syntax:	`BufReq_ReturnType DoIP_SoAdTpCopyTxData(`
	`  PduIdType TxPduId,`
	`  const PduInfoType* PduInfoPtr,`
	`  RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TxPduId	Identification of the transmitted I-PDU.
	PduInfoPtr	Provides the destination buffer and the number of bytes to be copied. If not enough transmit data is available, no data is copied. The transport protocol module may retry. A copy size of 0 can be used to indicate state changes in the retry parameter or to query currently available data.
	retry	This parameter is expected to be a NULL_PTR as retry is not supported by the DoIP module.
Parameters (inout):	None	
Parameters (out):	availableDataPtr	Indicates the remaining number of bytes that are available in the DoIP Tx buffer.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled as the required amount of Tx data is not available. TP layer might retry later on. No data has been copied.

		BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	This function is called by the SoAd module to query the transmit data of an I-PDU segment. Each call to this function copies the next part of the transmit data until TpDataState indicates TP_DATARETRY. In this case the API restarts to copy the data beginning at the location indicated by TxTpDataCnt.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00175]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT` and return `BUFREQ_E_NOT_OK`.
⌋()

[SWS_DoIP_00176]⌈
If development error detection is enabled: The function shall check if the `TxPduId` matches a configured DoIPSoAdTxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID` and return `BUFREQ_E_NOT_OK`.
⌋()

[SWS_DoIP_00177]⌈
If development error detection is enabled: The function shall check that neither the `PduInfoPtr` nor the `availableDataPtr` are a `NULL_PTR`. If the check fails the function shall raise the development error `DOIP_E_PARAM_POINTER` and return `BUFREQ_E_NOT_OK`.
⌋()

[SWS_DoIP_00178]⌈
If development error detection is enabled: The function shall check if the retry is a `NULL_PTR`. If the check fails the function shall raise the development error `DOIP_E_INVALID_PARAMETER` and return `BUFREQ_E_NOT_OK`.
⌋()

8.4.2 DoIP_SoAdTpTxConfirmation

[SWS_DoIP_00032]⌈

Service name:	DoIP_SoAdTpTxConfirmation

Syntax:	`void DoIP_SoAdTpTxConfirmation(`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`

Service ID[hex]:	0x07
Sync/Async:	Synchronous

Reentrancy:	Reentrant	
Parameters (in):	TxPduId	Identification of the transmitted I-PDU.
	result	Result of the transmission of the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the SoAd module after the I-PDU has been transmitted on its network, the result will reveal if the transmission was successful or not.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00180]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.
⌋()

[SWS_DoIP_00181]⌈
If development error detection is enabled: The function shall check if the `TxPduId` matches a configured DoIPSoAdTxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.
⌋()

[SWS_DoIP_00182]⌈
If development error detection is enabled: The function shall check if the result is valid. If the check fails the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.
⌋()

8.4.3 DoIP_SoAdTpCopyRxData

[SWS_DoIP_00033]⌈

Service name:	DoIP_SoAdTpCopyRxData	
Syntax:	`BufReq_ReturnType DoIP_SoAdTpCopyRxData( PduIdType RxPduId, const PduInfoType* PduInfoPtr, PduLengthType* bufferSizePtr )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	Identification of the received I-PDU
	PduInfoPtr	Requests Pointer to receive buffer and maximum receiveable length in the DoIP receive buffer.
Parameters (inout):	None	

Parameters (out):	bufferSizePtr	Available receive buffer after data has been copied.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful. BUFREQ_E_NOT_OK: Buffer request not successful. Buffer cannot be accessed.
Description:	This function is called when a transport protocol module has data to copy to the receiving module. Several calls may be made during one transportation of an I-PDU.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00183]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT` and return BUFREQ_E_NOT_OK.
⌋()

[SWS_DoIP_00036]⌈
If development error detection is enabled: The function shall check if the `RxPduId` matches a configured DoIPSoAdRxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID` and return `BUFREQ_E_NOT_OK`.
⌋()

[SWS_DoIP_00184]⌈
If development error detection is enabled: The function shall check that neither the `PduInfoPtr` nor the `bufferSizePtr` are a `NULL_PTR`. If the check fails, the function shall raise the development error `DOIP_E_PARAM_POINTER` and return `BUFREQ_E_NOT_OK`.
⌋()

8.4.4 DoIP_SoAdTpStartOfReception

[SWS_DoIP_00037]⌈

Service name:	DoIP_SoAdTpStartOfReception	

Syntax:	`BufReq_ReturnType DoIP_SoAdTpStartOfReception(`
	`  PduIdType RxPduId,`
	`  const PduInfoType* info,`
	`  PduLengthType TpSduLength,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU

		reception. Depending on the global parameter MetaDataLength, additional bytes containing MetaData are appended after the payload data.
	TpSduLength	Total length of the PDU to be received.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer in the DoIP module.
Return value:	BufReq_ReturnType	BUFREQ_OK: Reception has been accepted. RxBufferSizePtr indicates the available receive buffer. BUFREQ_E_NOT_OK: Reception has been rejected. RxBufferSizePtr remains unchanged. BUFREQ_E_OVFL: No Buffer of the required length can be provided.
Description:	This function will be called by the SoAd module at the start of receiving an I-PDU. The service shall provide the currently available maximum buffer size when invoked with TpSdulength equal to 0.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00186]⌈ If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT` and return `BUFREQ_E_NOT_OK`. ⌋()

[SWS_DoIP_00187]⌈ If development error detection is enabled: The function shall check if the `RxPduId` matches a configured DoIPSoAdRxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID` and return `BUFREQ_E_NOT_OK`. ⌋()

[SWS_DoIP_00188]⌈ If development error detection is enabled: The function shall check if the `bufferSizePtr` is not a `NULL_PTR`. If the check fails the function shall raise the development error `DOIP_E_PARAM_POINTER` and return `BUFREQ_E_NOT_OK`. ⌋()

[SWS_DoIP_00189]⌈ If development error detection is enabled: The function shall check if the `TpSduLength` is not 0. If `TpSduLength` is not 0 the function shall raise the development error `DOIP_E_INVALID_PARAMETER` and return `BUFREQ_E_NOT_OK`. ⌋()

Note: This is because SoAd will call the DoIP module only once with the TpSduLength set to 0 after the TCP connection has been established.

8.4.5 DoIP_SoAdTpRxIndication

[SWS_DoIP_00038]⌈

Service name:	DoIP_SoAdTpRxIndication

Syntax:	`void DoIP_SoAdTpRxIndication(`
	`  PduIdType RxPduId,`
	`  Std_ReturnType result`
	`)`	

Service ID[hex]:	0x0A	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	Identification of the received I-PDU.
	result	Result of the reception.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Called by the SoAd after an I-PDU has been received successfully or when an error occured. It is also used to confirm cancellation of an I-PDU.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00190]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.
⌋()

[SWS_DoIP_00191]⌈
If development error detection is enabled: The function shall check if the `RxPduId` matches a configured DoIPSoAdRxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.
⌋()

[SWS_DoIP_00192]⌈
If development error detection is enabled: The function shall check if the result is valid. If the check fails the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.
⌋()

8.4.6 DoIP_SoAdIfRxIndication

[SWS_DoIP_00244]⌈

Service name:	DoIP_SoAdIfRxIndication

Syntax:	`void DoIP_SoAdIfRxIndication(`
	`  PduIdType RxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`

Service ID[hex]:	0x42
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.

Parameters (in):	RxPduId	ID of the received I-PDU.
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication interface module.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00246]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.
⌋()

[SWS_DoIP_00247]⌈
If development error detection is enabled: The function shall check if the `RxPduId` matches a configured DoIPSoAdRxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.
⌋()

[SWS_DoIP_00248]⌈
If development error detection is enabled: The function shall check the validity of the PduInfoPtr and call the DET with DOIP_E_PARAM_POINTER error id if it is a NULL_PTR.
⌋()

8.4.7 DoIP_SoAdIfTxConfirmation

[SWS_DoIP_00245]⌈

Service name:	DoIP_SoAdIfTxConfirmation	
Syntax:	`void DoIP_SoAdIfTxConfirmation( PduIdType TxPduId )`	
Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the I-PDU that has been transmitted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of an I-PDU.	

⌋(SRS_Eth_00024)

[SWS_DoIP_00249]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.
⌋()

[SWS_DoIP_00250]⌈
If development error detection is enabled: The function shall check if the `TxPduId` matches a configured DoIPSoAdTxPduId. If the check fails the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.
⌋()

8.4.8 DoIP_SoConModeChg

[SWS_DoIP_00039]⌈

Service name:	DoIP_SoConModeChg	

Syntax:	`void DoIP_SoConModeChg(`
	`  SoAd_SoConIdType SoConId,`
	`  SoAd_SoConModeType Mode`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
Parameters (in):	SoConId	socket connection index specifying the socket connection with the mode change.
	Mode	new mode
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification about a SoAd socket connection state change, e.g. socket connection gets online	

⌋(SRS_Eth_00081, SRS_Eth_00028)

[SWS_DoIP_00193]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.
⌋()

[SWS_DoIP_00194]⌈

If development error detection is enabled: The function shall check if the `SoConId` and Mode are valid. If the check fails the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.

⌋()

8.4.9 DoIP_LocalIpAddrAssignmentChg

[SWS_DoIP_00040]⌈

Service name:	DoIP_LocalIpAddrAssignmentChg	
Syntax:	`void DoIP_LocalIpAddrAssignmentChg( SoAd_SoConIdType SoConId, TcpIp_IpAddrStateType State )`	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
Parameters (in):	SoConId	socket connection index specifying the socket connection where the IP address assigment has changed
	State	state of IP address assignment
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function gets called by the SoAd if an IP address assignment related to a socket connection changes (i.e. new address assigned or assigned address becomes invalid).	

⌋(SRS_Eth_00081, SRS_Eth_00028)

[SWS_DoIP_00195]⌈
If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.

⌋()

[SWS_DoIP_00196]⌈
If development error detection is enabled: The function shall check if the `SoConId` and State are valid. If the check fails the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.

⌋()

8.4.10 DoIP_ActivationLineSwitch

[SWS_DoIP_00251]⌈

Service name:	DoIP_ActivationLineSwitch
Syntax:	`void DoIP_ActivationLineSwitch(`

	void)
Service ID[hex]:	0x0f
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This function is used to notify the DoIP on a switch of the DoIPActivationLine

」()

[SWS_DoIP_00252]「 If development error detection is enabled: The function shall check that the service `DoIP_Init` was previously called. If the check fails, the function shall raise the development error `DOIP_E_UNINIT`.

」()

8.5 Scheduled functions

The Basic Software Scheduler within the Rte [6] directly calls these functions. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.5.1 DoIP_MainFunction

[SWS_DoIP_00041]「

Service name:	DoIP_MainFunction
Syntax:	void DoIP_MainFunction(void)
Service ID[hex]:	0x02
Description:	Schedules the Diagnostic over IP module. (Entry point for scheduling)

」(SRS_BSW_00376)

[SWS_DoIP_00042]「
The main function for scheduling the DoIP module (Entry point for scheduling) shall be called by the Schedule Manager according to the configured call period.

」()

[SWS_DoIP_00043]「
The call period of the `DoIP_MainFunction`() is determined by the configuration parameter DoIPMainFunctionPeriod.

」()

8.6 Expected Interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required to fulfill the core functionality of the module.

[SWS_DoIP_00044]「

API function	***Description***
Dcm_GetVin	Callbackfunction to get the VIN.
PduR_DoIPRxIndication	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.
PduR_DoIPStartOfReception	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF).
PduR_DoIPTxConfirmation	This function is called after the I-PDU has been transmitted via the TP API, the result indicates whether the transmission was successful or not.
PduR_DoIPCopyRxData	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.
PduR_DoIPCopyTxData	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
SoAd_CloseSoCon	This service closes the socket connection specified by SoConId.
SoAd_GetLocalAddr	Retrieves the local address (IP address and port) actually used for the SoAd socket connection specified by SoConId, the netmask and default router
SoAd_GetPhysAddr	Retrieves the physical source address of the EthIf controller used by the SoAd socket connection specified by SoConId.
SoAd_GetRemoteAddr	Retrieves the remote address (IP address and port) actually used for the SoAd socket connection specified by SoConId
SoAd_GetSoConId	Returns socket connection index related to specified transmit PduId. In case a fan-out is configured for TxPduId (i.e. multiple SoAdPduRouteDest specified) E_NOT_OK shall be returned.
SoAd_IfTransmit	This service initiates a request for transmission of the L-PDU specified by the SoAdSrcPduId. The corresponding socket has to be resolved by the SoAdSrcPduId. This call is used to mimic the call to an IF in AUTOSAR.

	Development errors: Invalid values of SoAdSrcPduId or SoAdSrcPduInfoPtr will be reported to the development error tracer (SOAD_E_INVALID_TXPDUID or SOAD_E_PARAM_POINTER).
SoAd_OpenSoCon	This service opens the socket connection specified by SoConId.
SoAd_ReadDhcpHostNameOption	By this API service an upper layer of the SoAd can read the currently configured hostname option in the DHCP submodule of the TCP/IP stack.
SoAd_ReleaseIpAddrAssignment	By this API service the local IP address assignment used for the socket connection specified by SoConId is released.
SoAd_RequestIpAddrAssignment	By this API service the local IP address assignment which shall be used for the socket connection specified by SoConId is initiated.
SoAd_SetRemoteAddr	By this API service the remote address (IP address and port) of the specified socket connection shall be set.
SoAd_SetUniqueRemoteAddr	This API service shall either return the socket connection index of the SoAdSocketConnectionGroup where the specified remote address (IP address and port) is set or assign the remote address to an unused socket connection from the same SoAdSocketConnectionGroup.
SoAd_TpCancelReceive	Requests cancellation of the reception via TP for a specific I-PDU.
SoAd_TpCancelTransmit	Requests cancellation of the transmission via TP for a specific I-PDU.
SoAd_TpTransmit	This service is utilized to request the transfer of data. It sets a flag for indicating that a transmit request is present. This function has to be called with the PDU-ID of the SoAd, i.e. the upper layer has to translate its own PDU-ID into the one of the SoAd for the corresponding message. This call shall fail if no entry for the PDU-ID can be found in the socket connection table. Within the provided SoAdSrcPduInfoPtr only SduLength is valid (no data)! If this function returns E_OK then there will arise a call of <Up>_[SoAd][Tp]CopyTxData in order to get data for sending. This call is used to mimic the call to a TP in AUTOSAR.
SoAd_WriteDhcpHostNameOption	By this API service an upper layer of the SoAd can set the hostname option in the DHCP submodule of the TCP/IP stack.

⌋()

8.6.2 Optional Interfaces

This chapter defines all interfaces which are required by the DoIP module to fulfill an optional functionality of the DoIP module.

[SWS_DoIP_00045]⌈

API function	***Description***
Det_ReportError	Service to report development errors.

⌋()

8.6.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kind of interfaces is not fixed because they are configurable.

8.6.3.1 <User>_DoIPGetPowerModeStatus

[SWS_DoIP_00047]⌜

Service name:	<User>_DoIPGetPowerModeStatus	
Syntax:	`Std_ReturnType <User>_DoIPGetPowerModeStatus( DoIP_PowerStateType* PowerStateReady )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Don't care	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	PowerStateReady	Pointer containing the information of the PowerModeStatus. Only valid if the return value equals E_OK.
Return value:	Std_ReturnType	E_OK: PowerStateReady contains valid information E_NOT_OK: PowerStateReady contains no valid information
Description:	Callback function to check if the PowerMode of the DoIP entity is ready or not.	

⌟(SRS_Eth_00080)

8.6.3.2 <User>_DoIPRoutingActivationConfirmation

[SWS_DoIP_00048]⌜

Service name:	<User>_DoIPRoutingActivationConfirmation	
Syntax:	`Std_ReturnType <User>_DoIPRoutingActivationConfirmation( boolean* Confirmed, uint8* ConfirmationReqData, uint8* ConfirmationResData )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous/Asynchronous	
Reentrancy:	Don't care	
Parameters (in):	ConfirmationReqData	Pointer to OEM specific bytes for Routing activation request. Only needed if DoIPRoutingActivationConfirmationReqLength is not 0.
Parameters (inout):	None	
Parameters (out):	Confirmed	Pointer containing the information if Confirmation was successful (TRUE) or not (FALSE). Only valid if the return value equals E_OK.
	ConfirmationResData	Pointer to OEM specific bytes for Response on Routing activation. Only needed if DoIPRoutingActivationConfirmationResLength if not 0. Contains valid data if function return with E_OK.
Return value:	Std_ReturnType	E_OK: Confirmed and ConfirmationResData contain valid Data. DOIP_E_PENDING: Confirmation still running. Call next DoIP_MainFunction cycle again.

		E_NOT_OK: Confirmed and/or ConfirmationResData do not contain valid information.
Description:	Callback function to get the confirmation for the Routing Activation.	

⌋(SRS_Eth_00084)

8.6.3.3 <User>_DoIPRoutingActivationAuthentication

[SWS_DoIP_00049]⌈

Service name:	<User>_DoIPRoutingActivationAuthentication	
Syntax:	`Std_ReturnType <User>_DoIPRoutingActivationAuthentication( boolean* Authentified, uint8* AuthenticationReqData, uint8* AuthenticationResData )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous/Asynchronous	
Reentrancy:	Don't care	
Parameters (in):	AuthenticationReqData	Pointer to OEM specific bytes for Routing activation request. Only needed if DoIPRoutingActivationAuthenticationReqLength is not 0.
Parameters (inout):	None	
Parameters (out):	Authentified	Pointer containing the information if Confirmation was successful (TRUE) or not (FALSE). Only valid if the return value equals E_OK.
	AuthenticationResData	Pointer to OEM specific bytes for Response on Routing activation. Only needed if DoIPRoutingActivationAuthenticationResLength if not 0. Contains valid data if function return with E_OK.
Return value:	Std_ReturnType	E_OK: Authentified and AuthenticationResData contain valid Data. DOIP_E_PENDING: Authentication still running. Call next DoIP_MainFunction cycle again. E_NOT_OK: Authentified and/or AuthenticationResData do not contain valid information.
Description:	Callback function to get the confirmation for the Routing Activation.	

⌋(SRS_Eth_00084)

8.6.3.4 <User>_DoIPTriggerGIDSynchronization

[SWS_DoIP_00050]⌈

Service name:	<User>_DoIPTriggerGIDSynchronization	
Syntax:	`Std_ReturnType <User>_DoIPTriggerGIDSynchronization( void )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous/Asynchronous	
Reentrancy:	Don't care	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: GroupIdentifier Synchronization was triggered

		E_NOT_OK: GroupIdentifier Synchronization could not be triggered so try again next MainFunction
Description:	Function is used in the case that DoIPVinGIDMaster is set to true and a container DoIPTriggerGIDSynchronization is configured to trigger the synchronization process of the GroupIIdentifier.	

⌋(SRS_Eth_00026)

8.6.3.5 <User>_DoIPGetGID

[SWS_DoIP_00051]⌈

Service name:	<User>_DoIPGetGID	

Syntax:	`Std_ReturnType <User>_DoIPGetGID(`
	`  uint8* GroupId`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous/Asynchronous	
Reentrancy:	Don't care	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	GroupId	Pointer to GroupIdentifier
Return value:	Std_ReturnType	E_OK: GroupId contains a valid value E_NOT_OK: GroupId does not contain a valid value
Description:	Function is used in the case that DoIPVinGIDMaster is set to false and DoIPGetGID is configured to get on a vehicle identification the GID. If the return value is not E_OK the DoIP shall use the default GID.	

⌋(SRS_Eth_00026)

8.6.4 DoIP Service Component

The following section describes the DoIP service representation and the condition for which configuration Services have to be requested and provided by the DoIP module.

[SWS_DoIP_00052]⌈
A *DoIP Service Component* with the ShortName DoIP shall be provided based on the configuration of the DoIP module.
⌋()

[SWS_DoIP_00054]⌈
The *DoIP Service Component* shall provide the interface *CallbackGetPowerMode* as described below to request the value of the Power mode for DoIP diagnostic power mode handling.

Name	CallbackGetPowerMode
Comment	--

IsService	true	
Variation	{ecuc(DoIP/DoIPGeneral/DoIPPowerModeCallback/DoIPPowerMode)} == NULL	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

GetPowerMode		
Comments	--	
Variation	--	
Parameters	PowerStateReady	
	Comment	--
	Type	DoIP_PowerStateType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」(SRS_Eth_00080)

[SWS_DoIP_00261]「

Name	CBGetPowerMode		
Kind	RequiredPort	Interface	CallbackGetPowerMode
Description	--		
Variation	{ecuc(DoIP/DoIPGeneral/DoIPPowerModeCallback/DoIPPowerMode)} == NULL		

」(SRS_Eth_00080)

[SWS_DoIP_00055]「

The *DoIP Service Component* shall provide the interface <RoutingActivation>_RoutingActivation as described below for each DoIPRoutingActivation that has at least DoIPRoutingActivationConfirmationCallback or DoIPRoutingActivationAuthenticationCallback configured without direct Callback functions.

Name	{Name}_RoutingActivation

Comment	--	
IsService	true	
Variation	(({ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationAuthenticationCallback)} != null) && ({ecuc(DoIP/DoIPConfigSet/ DoIPRoutingActivation/DoIPRoutingActivationAuthenticationCallback/ DoIPRoutingActivationAuthenticationFunc)} == "")) || (({ecuc(DoIP/DoIPConfigSet/ DoIPRoutingActivation/DoIPRoutingActivationConfirmationCallback)} != null) && ({ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationConfirmationCallback/DoIPRoutingActivationConfirmationFunc)} == "")) Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	16	DOIP_E_PENDING

Operations

RoutingActivationAuthentication		
Comments	--	
Variation	(({ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationAuthenticationCallback)} != NULL) && ({ecuc(DoIP/ DoIPConfigSet/DoIPRoutingActivation/DoIPRoutingActivationAuthenticationCallback/ DoIPRoutingActivationAuthenticationFunc)} ==NULL))	
Parameters	Authentified	
	Comment	--
	Type	boolean
	Variation	--
	Direction	OUT
	AuthenticationReqData	
	Comment	--
	Type	AuthenticationReqDataType_{Name}
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationAuthenticationCallback. DoIPRoutingActivationAuthenticationReqLength)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/ DoIPRoutingActivation.SHORT-NAME)}
	Direction	IN
	AuthenticationResData	
	Comment	--
	Type	AuthenticationResDataType_{Name}

	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationAuthenticationCallback. DoIPRoutingActivationAuthenticationResLength)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/ DoIPRoutingActivation.SHORT-NAME)}
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	DOIP_E_PENDING	RoutingActivation still pending.
RoutingActivationConfirmation		
Comments	--	
Variation	--	
Parameters	Confirmed	
	Comment	--
	Type	boolean
	Variation	--
	Direction	OUT
	ConfirmedReqData	
	Comment	--
	Type	ConfirmationReqDataType_{Name}
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationConfirmationCallback. DoIPRoutingActivationConfirmationReqLength)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/ DoIPRoutingActivation.SHORT-NAME)}
	Direction	IN
	ConfirmedResData	
	Comment	--
	Type	ConfirmationResDataType_{Name}
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/ DoIPRoutingActivationConfirmationCallback. DoIPRoutingActivationConfirmationResLength)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/ DoIPRoutingActivation.SHORT-NAME)}
	Direction	OUT

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	DOIP_E_PENDING	RoutingActivation still pending.

⌋(SRS_Eth_00084)

[SWS_DoIP_00262]⌈

Name	CB{Name}RoutingActivation		
Kind	RequiredPort	Interface	{Name}_RoutingActivation
Description	--		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		

⌋(SRS_Eth_00084)

[SWS_DoIP_00056]⌈
The *DoIP Service Component* shall provide the interface as *CallbackTriggerGIDSynchronization* as described below if the container DoIPTriggerGIDSynchronization is configured without direct Callback function.

Name	CallbackTriggerGIDSynchronization	
Comment	--	
IsService	true	
Variation	({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSynchronization)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSynchronization/DoIPTriggerGIDSynchronzation)} == NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPVinGidMaster)} == TRUE)	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

TriggerGIDSynchronization	
Comments	--
Variation	--

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」(SRS_Eth_00026)

[SWS_DoIP_00263]「

Name	CBTriggerGIDSynchronization		
Kind	RequiredPort	Interface	CallbackTriggerGIDSynchronization
Description	--		
Variation	({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSynchronization)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSynchronization/DoIPTriggerGIDSynchronzation)} == NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPVinGidMaster)} == TRUE)		

」(SRS_Eth_00026)

[SWS_DoIP_00057]「
The *DoIP Service Component* shall provide the interface *CallbackGetGID* as described below to request the GID if the container DoIPGetGID is configured without direct Callback function.

Name	CallbackGetGID	
Comment	--	
IsService	true	
Variation	{ecuc(DoIP/DoIPGeneral/DoIPGetGid)} != NULL && {ecuc(DoIP/DoIPGeneral/DoIPGetGid/DoIPGetGID)} == NULL	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

GetGID		
Comments	--	
Variation	--	
Parameters	Data	
	Comment	--

	Type	uint8
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

⌋(SRS_Eth_00026)

[SWS_DoIP_00264]⌈

Name	CBGetGID		
Kind	RequiredPort	Interface	CallbackGetGID
Description	--		
Variation	({ecuc(DoIP/DoIPGeneral/DoIPGetGid)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPGetGid/DoIPGetGID)} == NULL)		

⌋(SRS_Eth_00026)

[SWS_DoIP_00242]⌈The DoIP Service Component shall provide the interface DoIPActivationLineStatus as described below to be informed on the transition of the ActivationLine for DoIP.

Name	DoIPActivationLineStatus	
Comment	--	
IsService	true	
Variation	--	
ModeGroup	currentDoIPActivationLineStatus	DoIP_ActivationLineType

⌋()

[SWS_DoIP_00265]⌈

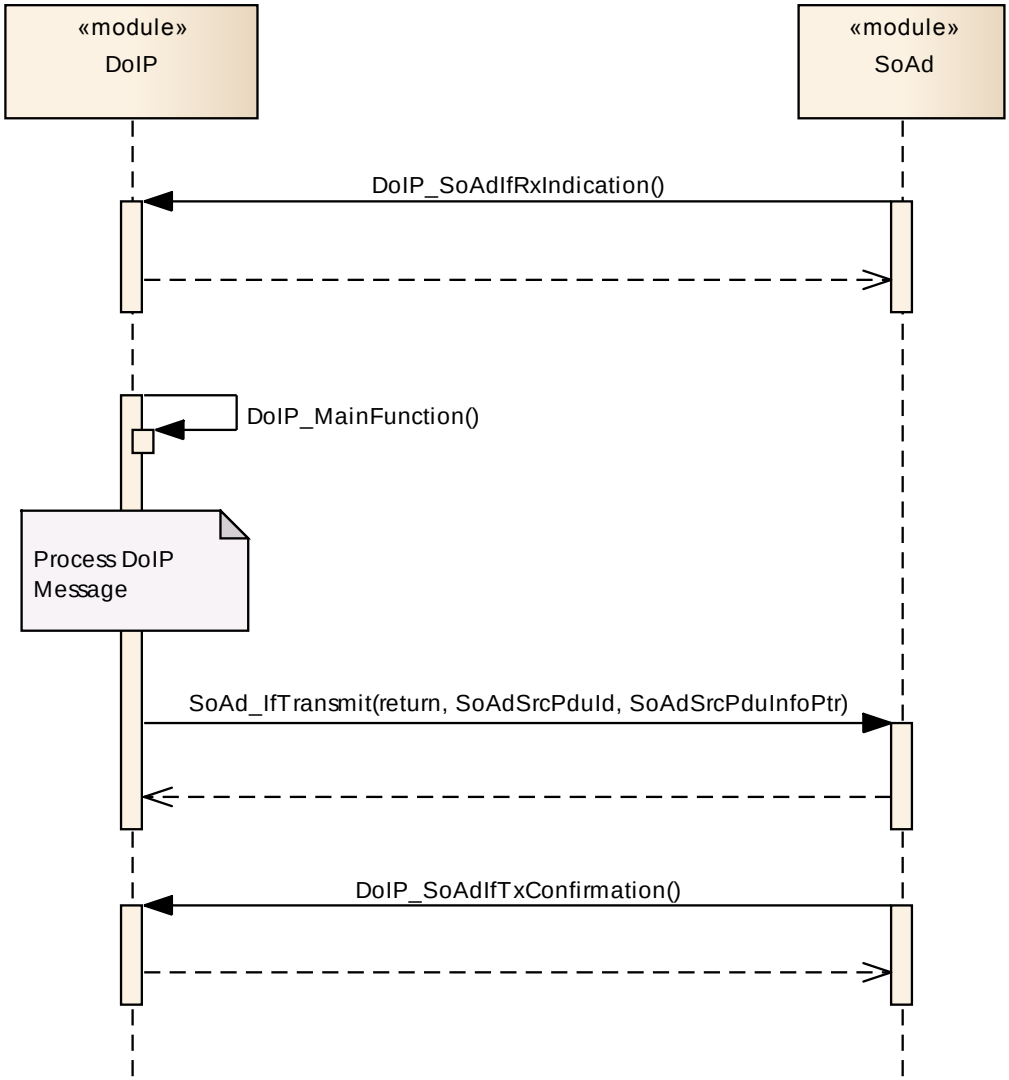
Name	DoIPActivationLineSwitchNotification		
Kind	RequiredPort	Interface	DoIPActivationLineStatus

Description	--
Variation	--

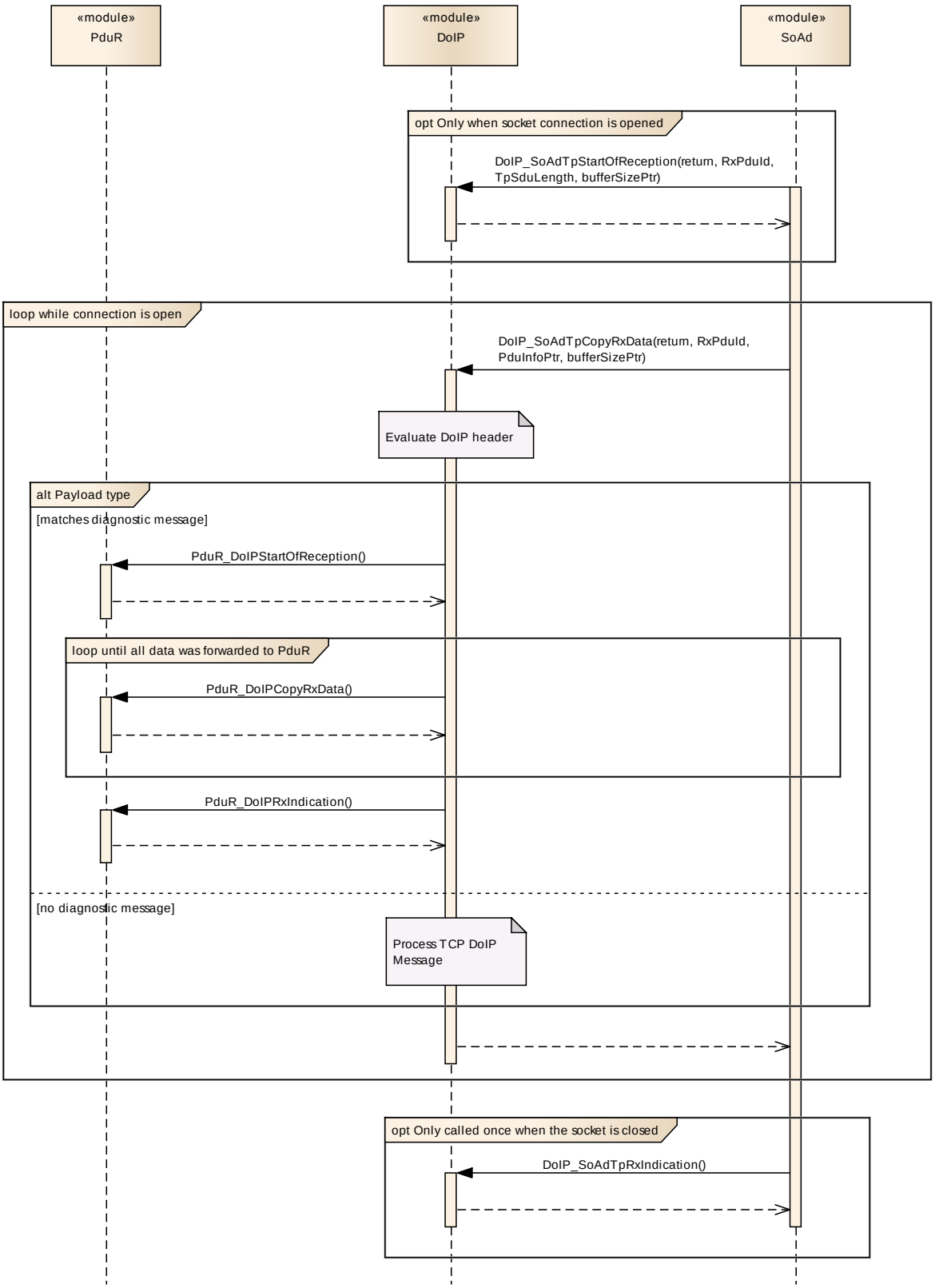
]()

9 Sequence diagrams

9.1 UDP DoIP communication

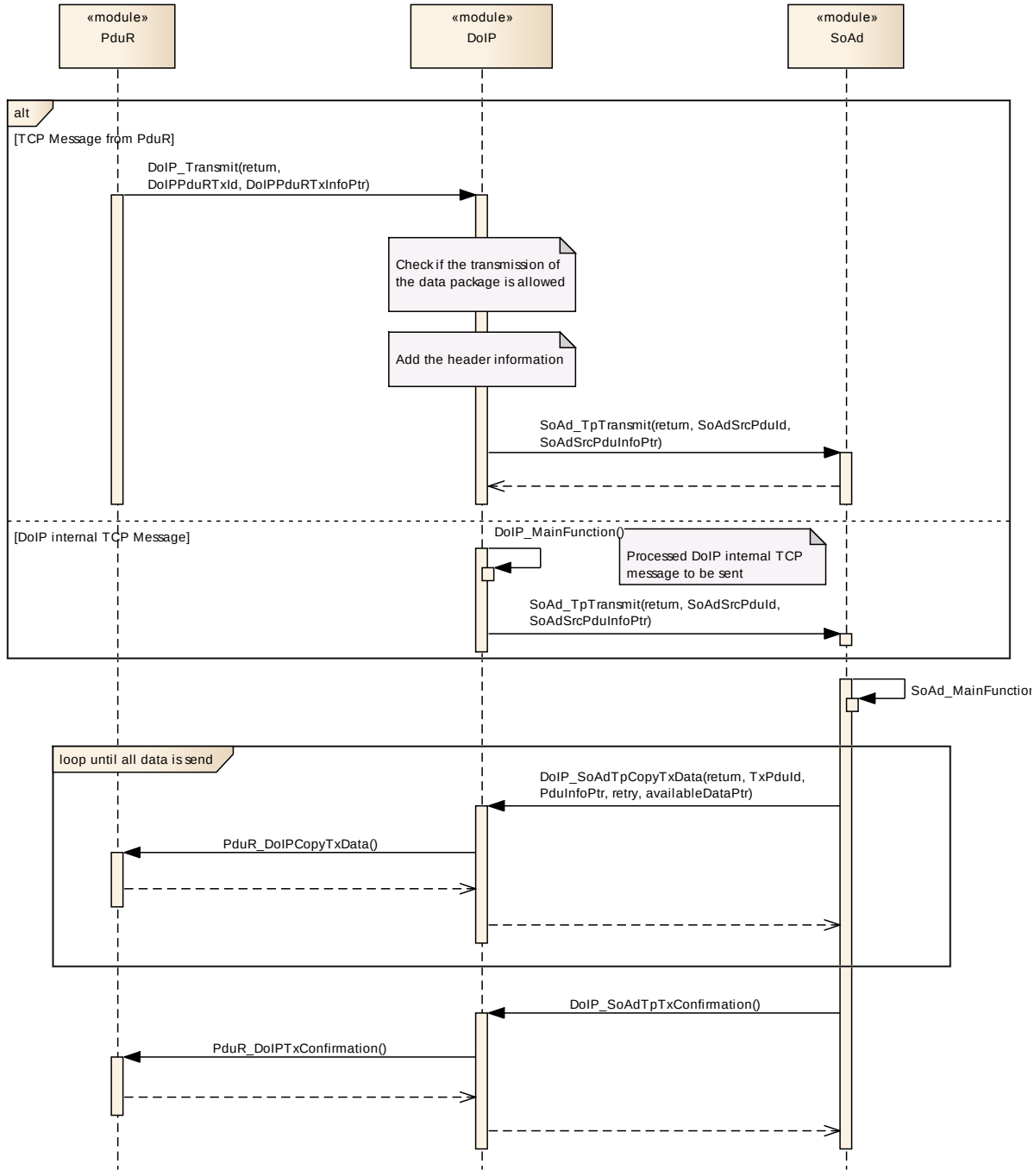


9.2 Rx TCP message

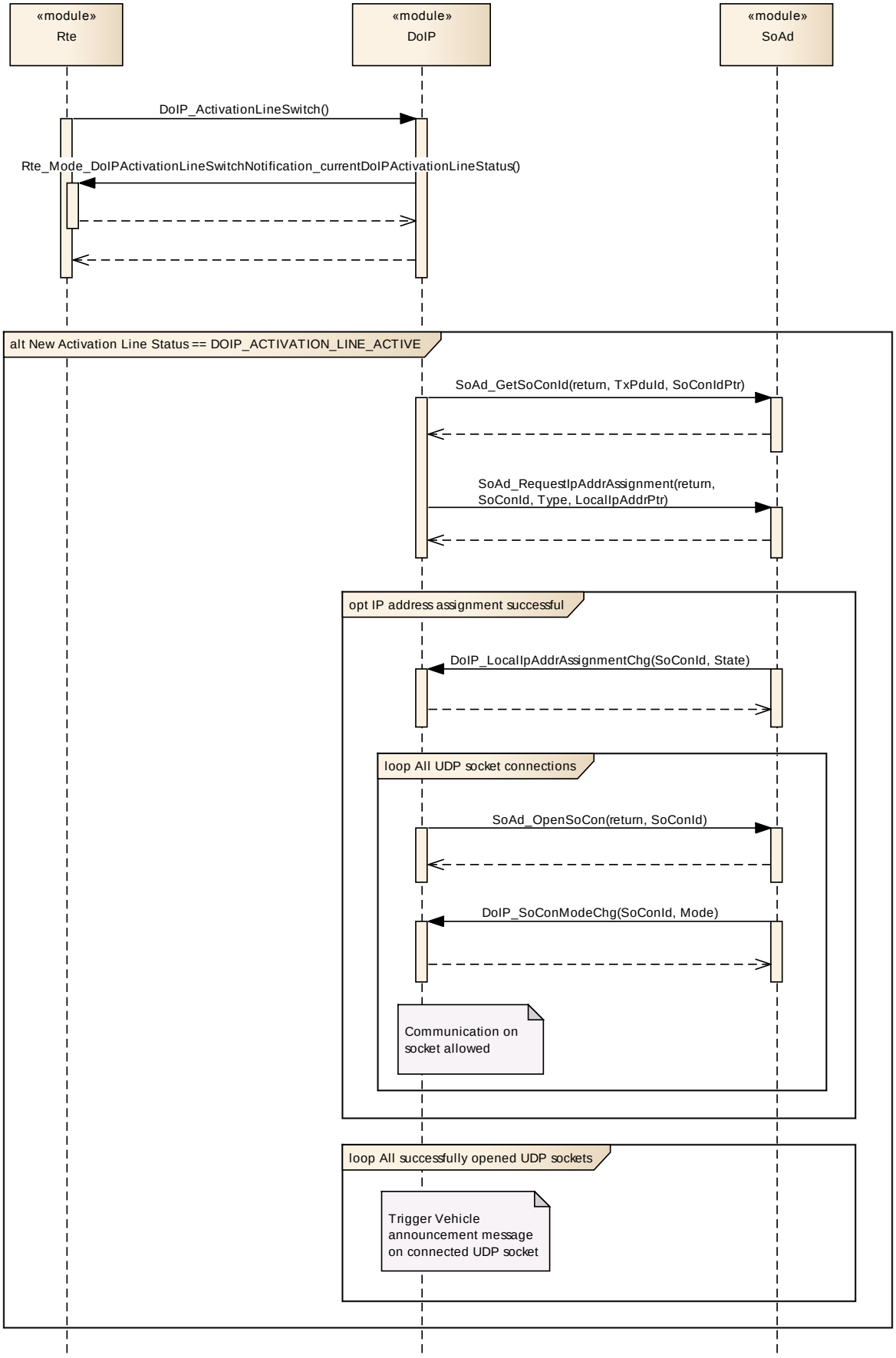


Note that more than one CopyRxData could provide the data of one request, but to reduce complexity this detail was omitted.

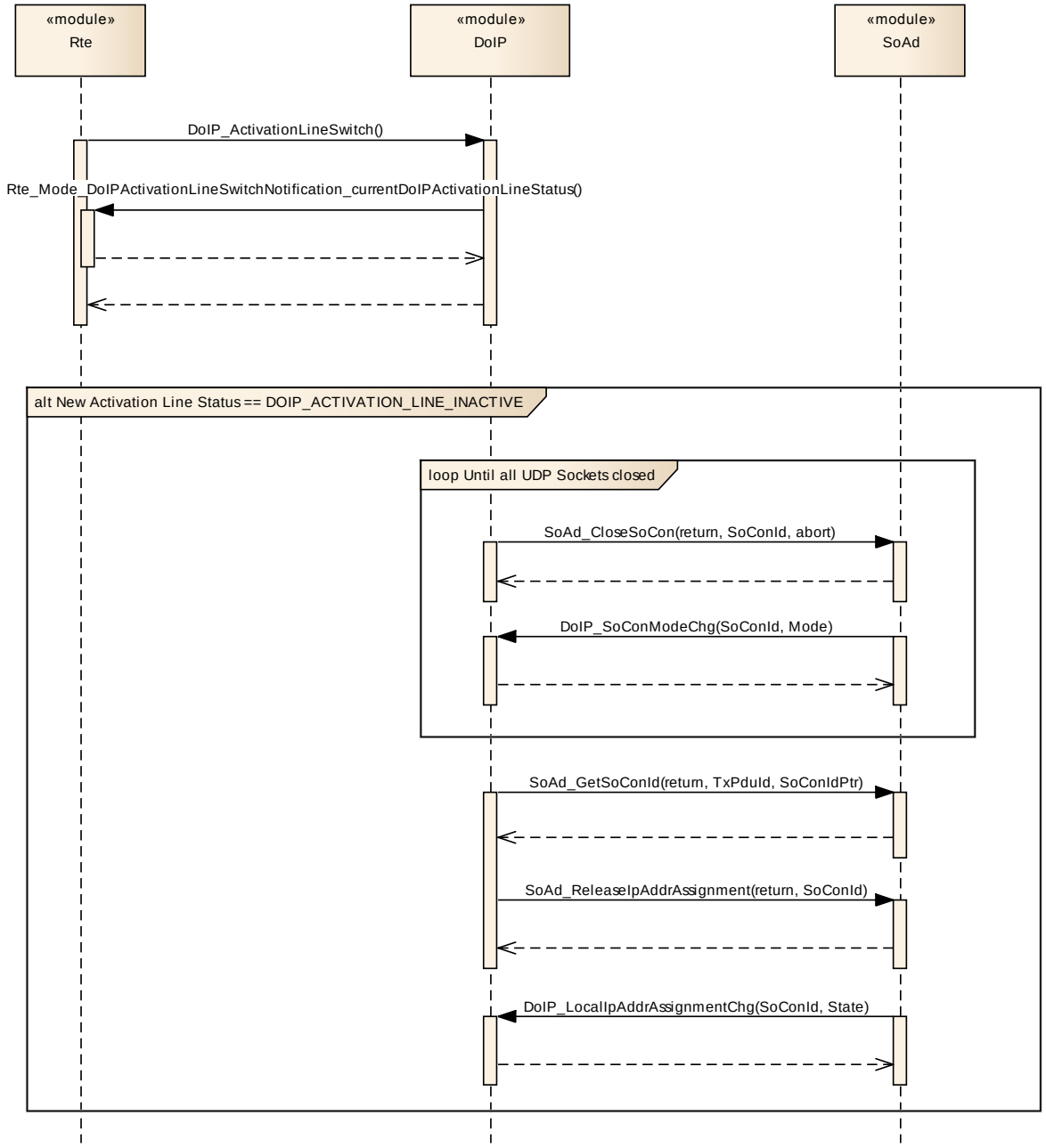
9.3 Tx TCP message



9.4 Activation Line Handling – Active


«module»
Rte
«module»
DoIP
«module»
SoAd
DoIP_ActivationLineSwitch()
Rte_Mode_DoIPActivationLineSwitchNotification_currentDoIPActivationLineStatus()
alt New Activation Line Status == DOIP_ACTIVATION_LINE_ACTIVE
SoAd_GetSoConId(return, TxPduId, SoConIdPtr)
SoAd_RequestIpAddrAssignment(return, SoConId, Type, LocalIpAddrPtr)
opt IP address assignment successful
DoIP_LocalIpAddrAssignmentChg(SoConId, State)
loop All UDP socket connections
SoAd_OpenSoCon(return, SoConId)
DoIP_SoConModeChg(SoConId, Mode)
Communication on socket allowed
loop All successfully opened UDP sockets
Trigger Vehicle announcement message on connected UDP socket

9.5 Activation Line Handling – Inactive



10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification, Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module DoIP.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral [14].

10.2 Configuration and configuration parameters

The following chapters summarize all configuration parameters. For a detailed description of parameters please refer to chapter 7 and chapter 8.

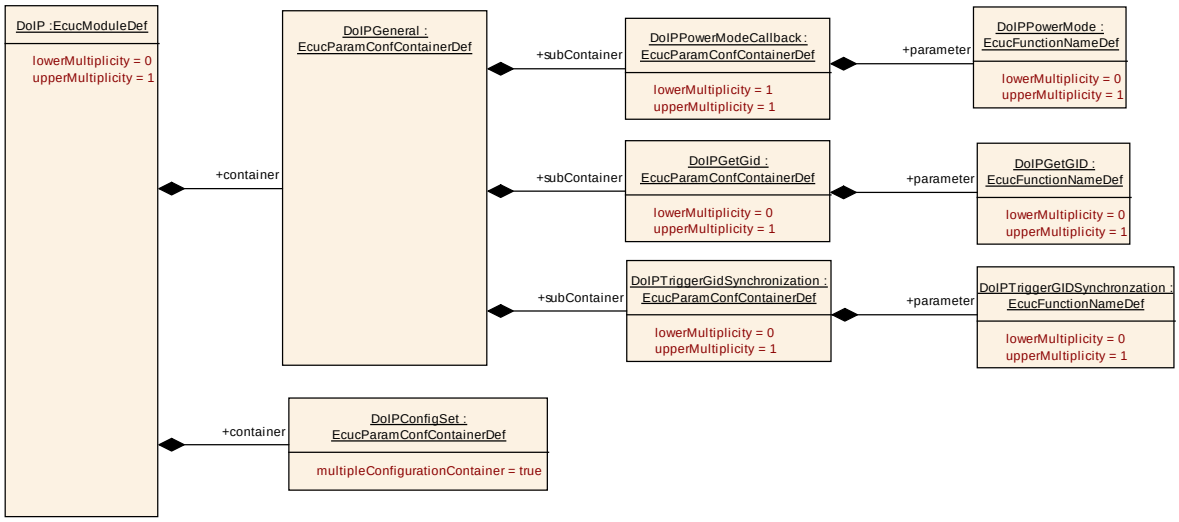
10.2.1 Variants

For details refer to the chapter 10.1.2 “Variants” in SWS_BSWGeneral [14].

10.2.2 DoIP

SWS Item	**ECUC_DoIP_00001 :**
Module Name	*DoIP*
Module Description	Configuration of the DoIP (Diagnostic over IP) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR DoIP module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
DoIPGeneral	1	This container specifies the general configuration parameters of the DoIP module.



10.2.3 DoIPGeneral

SWS Item	**ECUC_DoIP_00002 :**
Container Name	DoIPGeneral
Description	This container specifies the general configuration parameters of the DoIP module.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00009 :**		
Name	DoIPAliveCheckResponseTimeout		
Description	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2012.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00004 :**		
Name	DoIPDevelopmentErrorDetect		
Description	Pre-processor switch for enabling development error detection support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00067 :**
Name	DoIPDhcpOptionVinUse
Description	If DoIPDhcpOptionVinUse is set to true the DoIP module will add the VIN to the Dhcp host name if no valid Dhcp host name is already set.

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00064 :**		
Name	DoIPEntityStatusMaxByteFieldUse		
Description	This parameter is used to distinguish the optional support of the Max data size element of a diagnostic entity status response.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00065 :**		
Name	DoIPGIDInvalidityPattern		
Description	Specifies the Byte pattern that is used for response messages if no valid GID could be retrieved.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00068 :**		
Name	DoIPGeneralInactivityTime		
Description	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_TCP_General_Inactivity of ISO 13400-2:2012		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00072 :**		
Name	DoIPHeaderFileInclusion		
Description	Name of the header file(s) to be included by the DoIP module containing the used C-callback declarations.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		

regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00073 :**		
Name	DoIPHostNameSizeMax		
Description	Maximum Size of the DHCP HostName in ASCII. This parameter is necessary to reserve the correct amount of bytes for working with the DHCP HostName option. Minimum range is 5 because Dhcp Host Name should be at least "DoIP-" on any configuration.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	255		
minLength	5		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00010 :**		
Name	DoIPInitialInactivityTime		
Description	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2012		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00008 :**		
Name	DoIPInitialVehicleAnnouncementTime		
Description	Time to wait in [s] for sending first vehicle anouncement message after IP address assignment. Represents parameter A_DoIP_Announce_Wait of ISO 13400-2:2012		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00006 :**		
Name	DoIPMainFunctionPeriod		
Description	Determines the frequency at which the DoIP_MainFunction() is called in [s].		
Multiplicity	1		

Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00019 :**		
Name	DoIPMaxRequestBytes		
Description	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 18446744073709551615		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00012 :**		
Name	DoIPMaxTesterConnections		
Description	Maximum ammount of tester connections that shall be maintained at one time before alive check is performed.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00021 :**		
Name	DoIPNodeType		
Description	Describes the Type of the DoIP node.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DOIP_GATEWAY		The DoIP Entity is a DoIP Gateway.
	DOIP_NODE		The DoIP Entity is a DoIP Node.
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00018 :**		
Name	DoIPUseEIDasGID		
Description	Specifies if the DoIP entity shall use its EID if it is the Master for vehicle identification gid on the vehicle identification/vehicle announcement.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00013 :**		
Name	DoIPUseMacAddressForIdentification		
Description	Provided the information if a configured EID at vehicle identification response/vehicle announment is used or the MAC address. TRUE: Use MAC Address instead of EID for Vehicle identification/announcement. FALSE: Use configured EID for vehicle identification/announcement. Dependencies: DoIPEID		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00016 :**		
Name	DoIPUseVehicleIdentificationSyncStatus		
Description	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/announcement.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00007 :**		
Name	DoIPVehicleAnnouncementInterval		
Description	Time to wait in [s] for sending subsequent vehicle anouncement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2:2012		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00011 :**		
Name	DoIPVehicleAnnouncementRepetition		
Description	Amount of repetitions of the vehicle announcement message on IP address assignment. Represents parameter A_DoIP_Annunce_Num of ISO 13400-2:2012		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

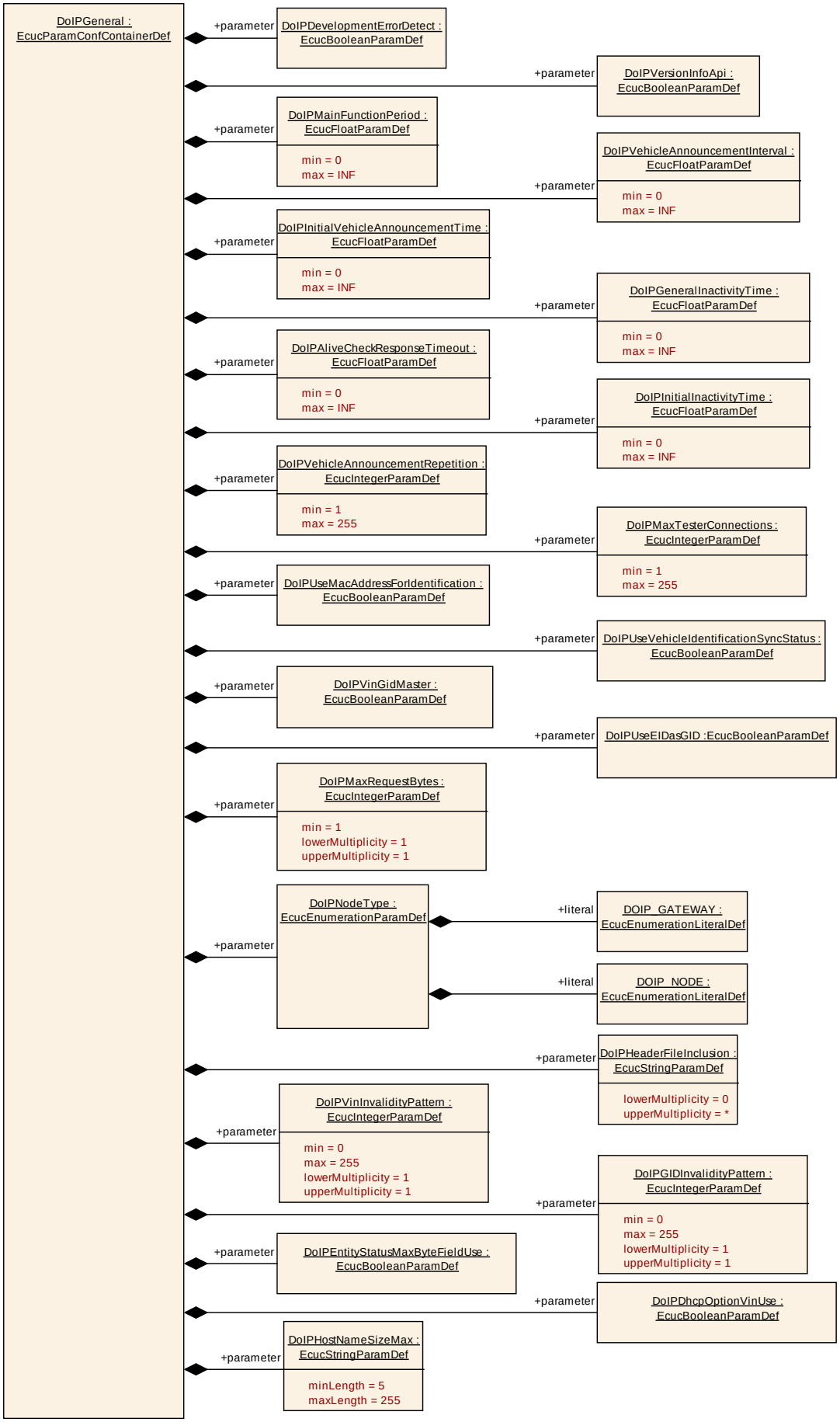
SWS Item	**ECUC_DoIP_00005 :**		
Name	DoIPVersionInfoApi		
Description	Activates the DoIP_GetVersionInfo() API. TRUE: Enables the DoIP_GetVersionInfo() API. FALSE: DoIP_GetVersionInfo() API is not included.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00017 :**		
Name	DoIPVinGidMaster		
Description	Specifies if the DoIP entity is the Vehicle identification Master for the GID (Group ID).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: DoIPUseEIDasGID, DoIPTriggerGIDSynchronization		

SWS Item	**ECUC_DoIP_00066 :**		
Name	DoIPVinInvalidityPattern		
Description	Specifies the Byte pattern that is used for response messages if no valid VIN could be retrieved.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPGetGid	0..1	This container describes the Callbackfunction to get the GID. If this container is not configured no Callbackfunction will be used. If this container is configured but the DoIPGetGID parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackGetGID with the name "CBGetGID".
DoIPPowerModeCallback	1	This container describes the Callbackfunction to get the Power Mode. This container shall always be present. If the DoIPPowerMode parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackGetPowerMode with the name "CBGetPowerMode".
DoIPTriggerGidSynchronization	0..1	This container describes the Callbackfunction to trigger the GID synchronisation.

		If this container is not configured no Callbackfunction will be used. If this container is configured but the DoIPTriggerGIDSynchronzation parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackTriggerGIDSynchronization with the name "CBTriggerGIDSynchronization".


DoIPGeneral : EcucParamConfContainerDef
+parameter
DoIPDevelopmentErrorDetect : EcucBooleanParamDef
DoIPVersionInfoApi : EcucBooleanParamDef
DoIPMainFunctionPeriod : EcucFloatParamDef
min = 0
max = INF
DoIPVehicleAnnouncementInterval : EcucFloatParamDef
min = 0
max = INF
DoIPInitialVehicleAnnouncementTime : EcucFloatParamDef
min = 0
max = INF
DoIPGeneralInactivityTime : EcucFloatParamDef
min = 0
max = INF
DoIPAliveCheckResponseTimeout : EcucFloatParamDef
min = 0
max = INF
DoIPInitialInactivityTime : EcucFloatParamDef
min = 0
max = INF
DoIPVehicleAnnouncementRepetition : EcucIntegerParamDef
min = 1
max = 255
DoIPMaxTesterConnections : EcucIntegerParamDef
min = 1
max = 255
DoIPUseMacAddressForIdentification : EcucBooleanParamDef
DoIPUseVehicleIdentificationSyncStatus : EcucBooleanParamDef
DoIPVinGidMaster : EcucBooleanParamDef
DoIPUseEIDasGID :EcucBooleanParamDef
DoIPMaxRequestBytes : EcucIntegerParamDef
min = 1
lowerMultiplicity = 1
upperMultiplicity = 1
DoIPNodeType : EcucEnumerationParamDef
+literal
DOIP_GATEWAY : EcucEnumerationLiteralDef
DOIP_NODE : EcucEnumerationLiteralDef
DoIPHeaderFileInclusion : EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = *
DoIPVinInvalidityPattern : EcucIntegerParamDef
min = 0
max = 255
lowerMultiplicity = 1
upperMultiplicity = 1
DoIPGIDInvalidityPattern : EcucIntegerParamDef
min = 0
max = 255
lowerMultiplicity = 1
upperMultiplicity = 1
DoIPEntityStatusMaxByteFieldUse : EcucBooleanParamDef
DoIPDhcpOptionVinUse : EcucBooleanParamDef
DoIPHostNameSizeMax : EcucStringParamDef
minLength = 5
maxLength = 255

10.2.4 DoIPGetGid

SWS Item	**ECUC_DoIP_00024 :**
Container Name	DoIPGetGid
Description	This container describes the Callbackfunction to get the GID. If this container is not configured no Callbackfunction will be used. If this container is configured but the DoIPGetGID parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackGetGID with the name "CBGetGID".
Configuration Parameters	

SWS Item	**ECUC_DoIP_00028 :**		
Name	DoIPGetGID		
Description	Direct C Callback function to get the GID of the DoIP entity. If the DoIPGetGID parameter is present, the DoIP module will not use an RPort of ServiceInterface CallbackGetGID but call the configured function.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.5 DoIPPowerModeCallback

SWS Item	**ECUC_DoIP_00023 :**
Container Name	DoIPPowerModeCallback
Description	This container describes the Callbackfunction to get the Power Mode. This container shall always be present. If the DoIPPowerMode parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackGetPowerMode with the name "CBGetPowerMode".
Configuration Parameters	

SWS Item	**ECUC_DoIP_00027 :**
Name	DoIPPowerMode
Description	Direct C Callback function to get the Power Mode of the DoIP entity. If the DoIPPowerMode parameter is present, the DoIP module will not use an RPort of ServiceInterface CallbackGetPowerMode but will call the configured function.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	--
maxLength	--
minLength	--
regularExpression	--

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 DoIPTriggerGidSynchronization

SWS Item	**ECUC_DoIP_00025 :**
Container Name	DoIPTriggerGidSynchronization
Description	This container describes the Callbackfunction to trigger the GID synchronisation. If this container is not configured no Callbackfunction will be used. If this container is configured but the DoIPTriggerGIDSynchronzation parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackTriggerGIDSynchronization with the name "CBTriggerGIDSynchronization".
Configuration Parameters	

SWS Item	**ECUC_DoIP_00029 :**		
Name	DoIPTriggerGIDSynchronzation		
Description	Direct C Callback function to trigger the synchronization of the GID. If the DoIPTriggerGIDSynchronzation parameter is present, the DoIP module will not use an RPort of ServiceInterface CallbackTriggerGIDSynchronization but call the configured function.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.7 DoIPConfigSet

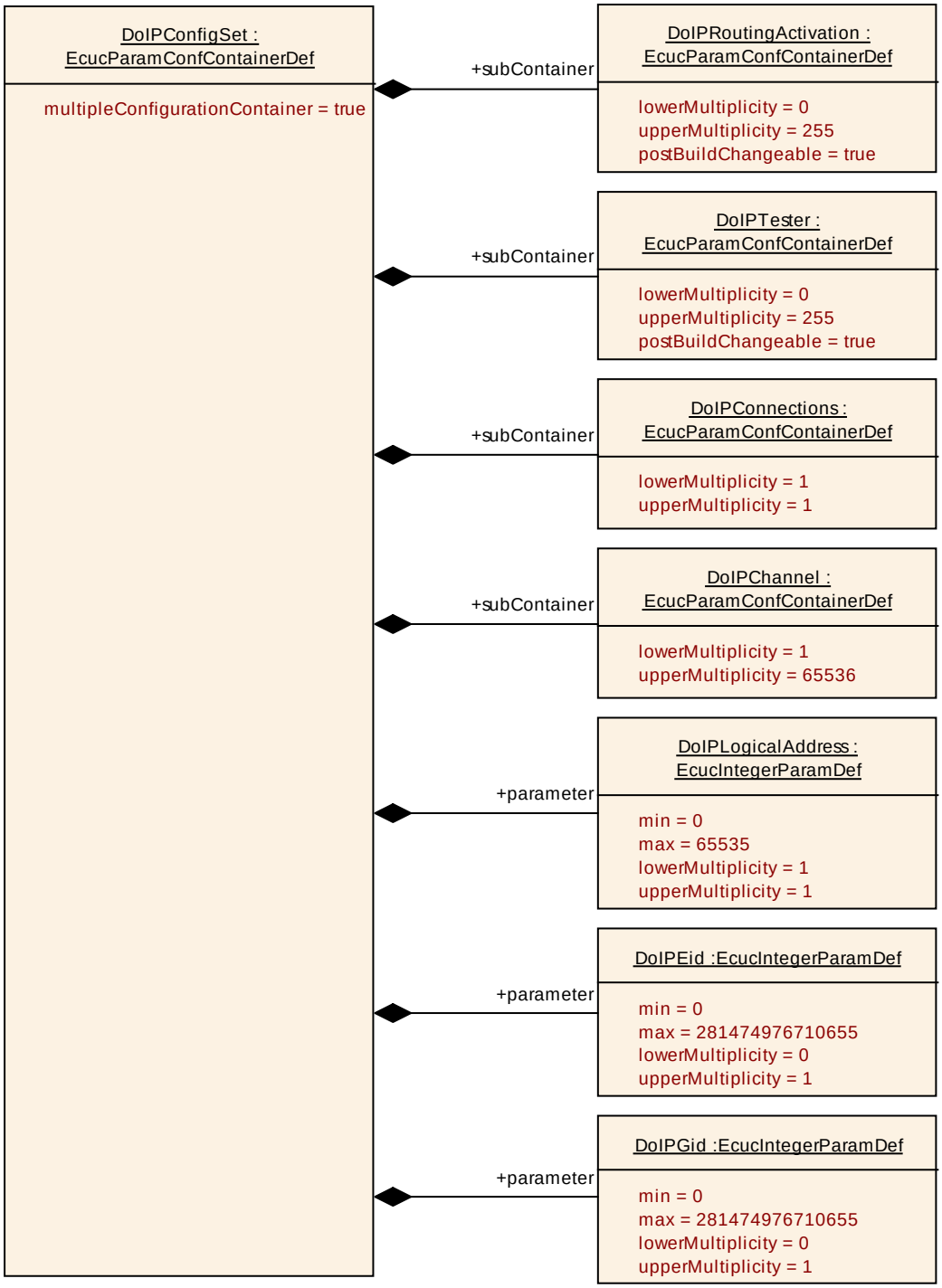
SWS Item	**ECUC_DoIP_00003 :**
Container Name	DoIPConfigSet [Multi Config Container]
Description	This container contains the configuration parameters and sub containers of the AUTOSAR DoIP module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00014 :**		
Name	DoIPEid		
Description	Configured EID (Entity ID of) for vehicle identification/vehicle announcement. Only necessary if DoIPUseMacAddressForIdentification is set to FALSE.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 281474976710655		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: DoIPUseMacAdressForIdentification		

SWS Item	**ECUC_DoIP_00015 :**		
Name	DoIPGid		
Description	Configured GID (Group ID of) for vehicle identification/vehicle announcement.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 281474976710655		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: DoIPUseEIDasGID, DoIPVinGIDMaster, DoIPGetGID		

SWS Item	**ECUC_DoIP_00020 :**		
Name	DoIPLogicalAddress		
Description	Describes the logical address of the DoIP entity, i.e. the LA that will route diagnostic requests to the Dcm of the DoIP entity.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPChannel	1..65536	Configuration of one DoIPChannel.
DoIPConnections	1	Container contains all lower layer connection specific information, i.e. the single Pdu References and Handle IDs to the SoAd.
DoIPRoutingActivation	0..255	This container describes the routing activation possibilities by representing for each container a possible routing activation request message to the DoIP entity and the according references to the activated diagnostic messages.
DoIPTester	0..255	This container describes the properties of the possible connectable Tester for the DoIP entity.



10.2.8 DoIPChannel

SWS Item	**ECUC_DoIP_00069 :**
Container Name	DoIPChannel
Description	Configuration of one DoIPChannel.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00070 :**
Name	DoIPChannelSARef

Description	Reference to the DoIPTester.		
Multiplicity	1		
Type	Reference to [DoIPTester]		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ECUC_DoIP_00071 :**		
Name	DoIPChannelTARef		
Description	Reference to the target address.		
Multiplicity	1		
Type	Reference to [DoIPTargetAddress]		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPPduRRxPdu	1	This container contains the Rx Pdus to connect with the Rx Pdus of the PduR.
DoIPPduRTxPdu	0..1	This container contains the Tx Pdus to connect with the Tx Pdus of the PduR. If the parameter is not configured the channel is for functional addressing.

10.2.9 DoIPPduRRxPdu

SWS Item	**ECUC_DoIP_00055 :**
Container Name	DoIPPduRRxPdu
Description	This container contains the Rx Pdus to connect with the Rx Pdus of the PduR.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00057 :**		
Name	DoIPPduRRxPduId		
Description	The DoIPPduRRxPduId is required by the API call DoIP_CancelReceive.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_DoIP_00058 :**		
Name	DoIPPduRRxPduRef		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.10 DoIPPduRTxPdu

SWS Item	**ECUC_DoIP_00056 :**		
Container Name	DoIPPduRTxPdu		
Description	This container contains the Tx Pdus to connect with the Tx Pdus of the PduR. If the parameter is not configured the channel is for functional addressing.		
Configuration Parameters			

SWS Item	**ECUC_DoIP_00060 :**		
Name	DoIPPduRTxPduId		
Description	The DoIPPduRTxPduId is required by DoIP_Transmit and DoIP_CancelTransmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_DoIP_00059 :**		
Name	DoIPPduRTxPduRef		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

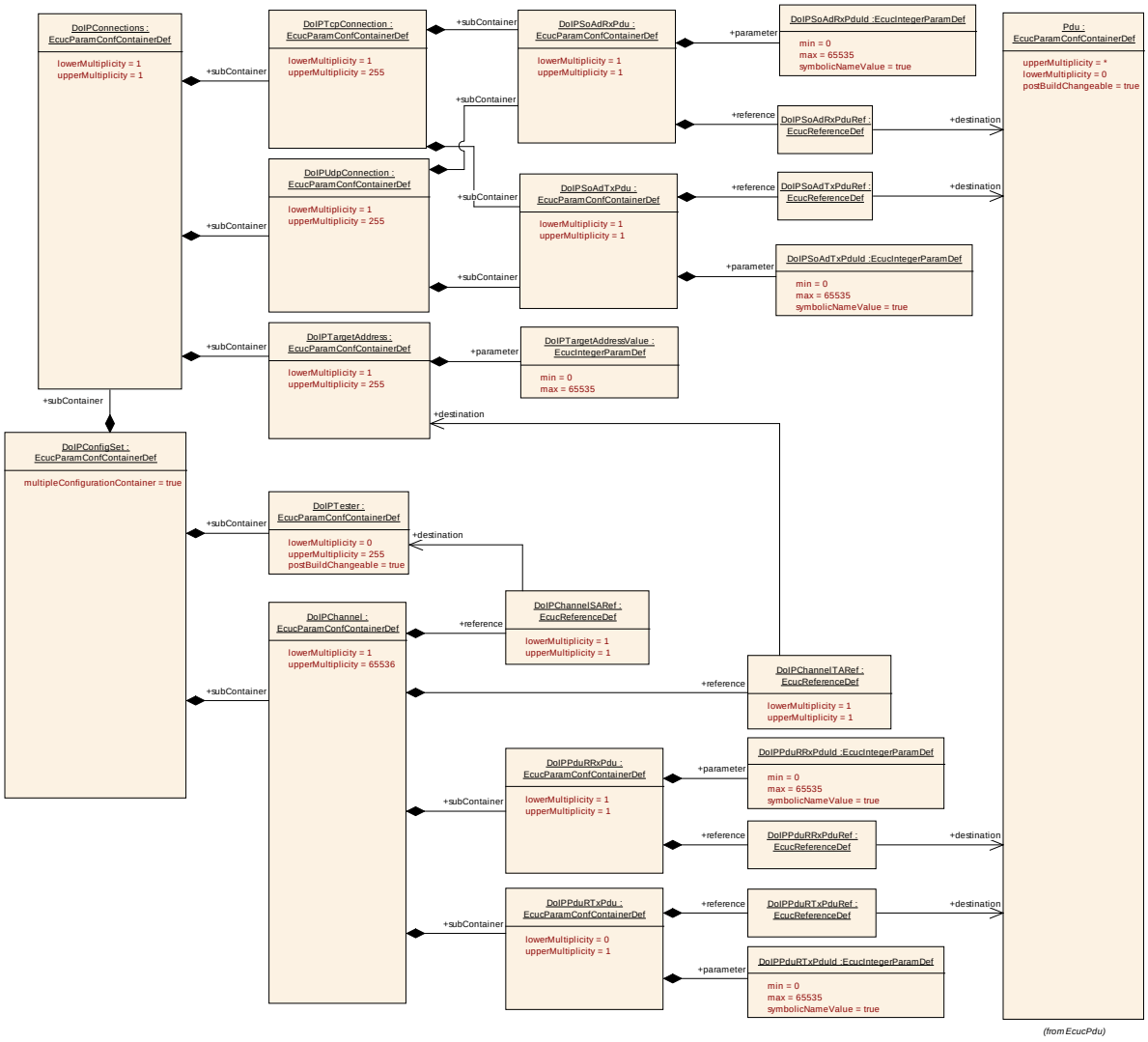
No Included Containers

10.2.11 DoIPConnections

SWS Item	**ECUC_DoIP_00032 :**
Container Name	DoIPConnections
Description	Container contains all lower layer connection specific information, i.e. the single Pdu References and Handle IDs to the SoAd.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPTargetAddress	1..255	This container describes a possible TargetAddress that is

		supported by DoIP.
DoIPTcpConnection	1..255	This container describes a tcp connection to the lower layer SoAd module.
DoIPUdpConnection	1..255	This Container describes the udp connection to the lower layer SoAd module.



10.2.12 DoIPTargetAddress

SWS Item	**ECUC_DoIP_00053 :**
Container Name	DoIPTargetAddress
Description	This container describes a possible TargetAddress that is supported by DoIP.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00054 :**	
Name	DoIPTargetAddressValue	
Description	Valid Target Address of a DoIP target address.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	--	

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.13 DoIPTcpConnection

SWS Item	**ECUC_DoIP_00045 :**
Container Name	DoIPTcpConnection
Description	This container describes a tcp connection to the lower layer SoAd module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPSoAdRxPdu	1	This container contains the Rx Pdus received by DoIP
DoIPSoAdTxPdu	1	This container describes the TxPdu sent via the SoAd

10.2.14 DoIPUdpConnection

SWS Item	**ECUC_DoIP_00052 :**
Container Name	DoIPUdpConnection
Description	This Container describes the udp connection to the lower layer SoAd module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPSoAdRxPdu	1	This container contains the Rx Pdus received by DoIP
DoIPSoAdTxPdu	1	This container describes the TxPdu sent via the SoAd

10.2.15 DoIPSoAdRxPdu

SWS Item	**ECUC_DoIP_00046 :**
Container Name	DoIPSoAdRxPdu
Description	This container contains the Rx Pdus received by DoIP
Configuration Parameters	

SWS Item	**ECUC_DoIP_00048 :**	
Name	DoIPSoAdRxPduId	
Description	The DoIPSoAdRxPduId is required by the API call DoIP_SoAdTpRxIndication to receive I-PDUs from the SoAd.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	

Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_DoIP_00049 :**		
Name	DoIPSoAdRxPduRef		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.16 DoIPSoAdTxPdu

SWS Item	**ECUC_DoIP_00047 :**
Container Name	DoIPSoAdTxPdu
Description	This container describes the TxPdu sent via the SoAd
Configuration Parameters	

SWS Item	**ECUC_DoIP_00051 :**		
Name	DoIPSoAdTxPduId		
Description	The DoIPSoAdTxPduId is required by the API call DoIP_SoAdTpTxConfirmation that is called by the SoAd to confirm that the IPdu has been transmitted successfully.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_DoIP_00050 :**		
Name	DoIPSoAdTxPduRef		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.17 DoIPRoutingActivation

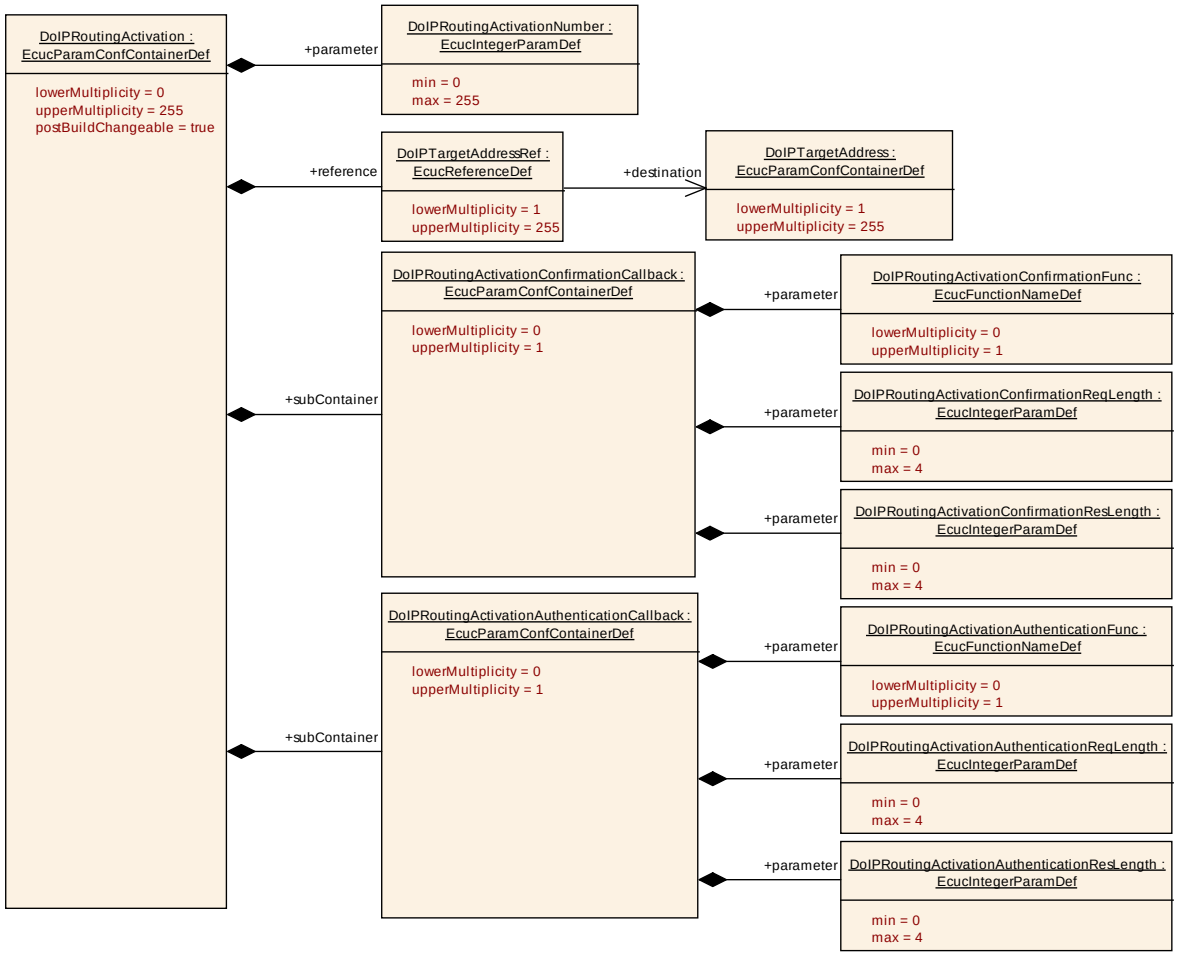
SWS Item	**ECUC_DoIP_00030 :**		
Container Name	DoIPRoutingActivation		
Description	This container describes the routing activation possibilities by representing for each container a possible routing activation request message to the DoIP entity and the according references to the activated diagnostic messages. **Attributes:** postBuildChangeable=true		
Configuration Parameters			

SWS Item	**ECUC_DoIP_00033 :**		
Name	DoIPRoutingActivationNumber		
Description	Identifies the Routing activation Number which is received for a DoIP routing activation request message.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00034 :**		
Name	DoIPTargetAddressRef		
Description	Reference to all DoIPTargetAddress which are activated on this Routing activation.		
Multiplicity	1..255		
Type	Reference to [DoIPTargetAddress]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DoIPRoutingActivationAuthenticationCallback	0..1	Container describes the Callbackfunction to call on a Routing Activation Request for Authentication. If this container is configured but the DoIPRoutingActivationAuthenticationFunc parameter is not present, the DoIP module will use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation with the name "CB<RoutingActivation>RoutingActivation". <RoutingActivation> is the ShortName of the DoIPRoutingActiviation container.
DoIPRoutingActivationConfirmationCallback	0..1	Container describes the Callbackfunction to call on a Routing Activation Request for Confirmation. If this container is configured but

		the DoIPRoutingActivationConfirmationFunc parameter is not present the DoIP module will use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation with the name "CB<RoutingActivation>RoutingActivation". <RoutingActivation> is the ShortName of the DoIPRoutingActiviation container.



10.2.18 DoIPRoutingActivationAuthenticationCallback

SWS Item	**ECUC_DoIP_00035 :**
Container Name	DoIPRoutingActivationAuthenticationCallback
Description	Container describes the Callbackfunction to call on a Routing Activation Request for Authentication. If this container is configured but the DoIPRoutingActivationAuthenticationFunc parameter is not present, the DoIP module will use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation with the name "CB<RoutingActivation>RoutingActivation". <RoutingActivation> is the ShortName of the DoIPRoutingActiviation container.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00039 :**
Name	DoIPRoutingActivationAuthenticationFunc

Description	Direct C Callback function to trigger the authentication function for routing activation. If the DoIPRoutingActivationAuthenticationFunc parameter is present, the DoIP module will not use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation but call the configured function.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00040 :**		
Name	DoIPRoutingActivationAuthenticationReqLength		
Description	Describes the amount of bytes used to handle to the authentication function on routing activation. If 0 is configured as length the parameter AuthenticationReqData will not be handled to the API.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00041 :**		
Name	DoIPRoutingActivationAuthenticationResLength		
Description	Describes the amount of bytes used to read by the authentication function on routing activation. If 0 is configured as length the parameter AuthenticationResData will not be fetched via the API.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.19 DoIPRoutingActivationConfirmationCallback

SWS Item	**ECUC_DoIP_00061 :**
Container Name	DoIPRoutingActivationConfirmationCallback
Description	Container describes the Callbackfunction to call on a Routing Activation

	Request for Confirmation. If this container is configured but the DoIPRoutingActivationConfirmationFunc parameter is not present the DoIP module will use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation with the name "CB<RoutingActivation>RoutingActivation". <RoutingActivation> is the ShortName of the DoIPRoutingActiviation container.
Configuration Parameters	

SWS Item	**ECUC_DoIP_00036 :**		
Name	DoIPRoutingActivationConfirmationFunc		
Description	Direct C Callback function to trigger the confirmation function for routing activation. If the DoIPRoutingActivationConfirmationFunc parameter is present the DoIP module will not use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation but call the configured function.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00037 :**		
Name	DoIPRoutingActivationConfirmationReqLength		
Description	Describes the amount of bytes used to handle to the confirmation function on routing activation. If 0 is configured as length the parameter ConfirmedReqData will not be handled to the API.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00038 :**		
Name	DoIPRoutingActivationConfirmationResLength		
Description	Describes the amount of bytes used to read by the confirmation function on routing activation. If 0 is configured as length the parameter ConfirmedResData will not be fetched via the API.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.20 DoIPTester

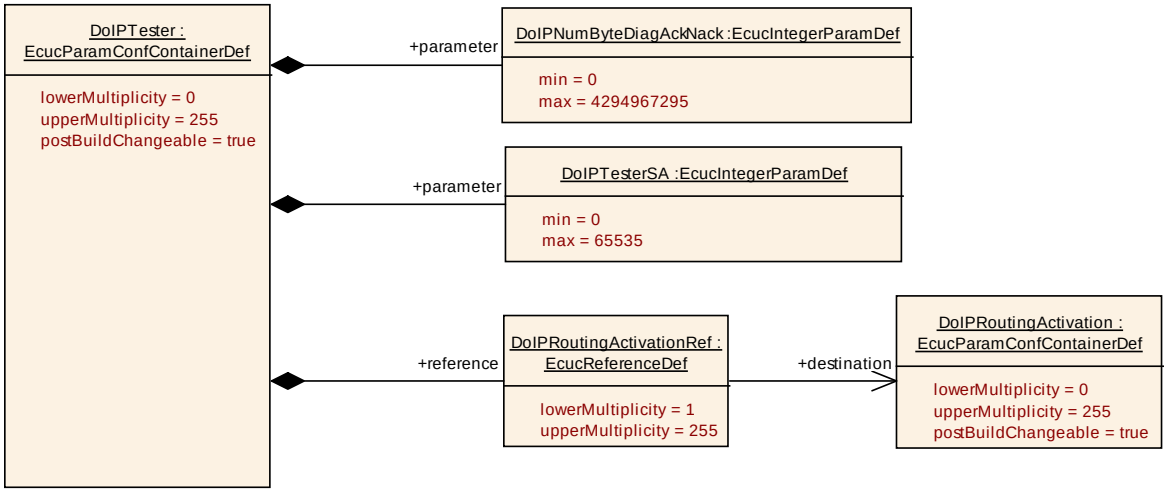
SWS Item	**ECUC_DoIP_00031 :**
Container Name	DoIPTester
Description	This container describes the properties of the possible connectable Tester for the DoIP entity. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_DoIP_00042 :**		
Name	DoIPNumByteDiagAckNack		
Description	Specifies the number of original Diagnostic request bytes the DoIP entity responses on a NACK of a diagnostic response message to the Tester.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00043 :**		
Name	DoIPTesterSA		
Description	Source Address of the Tester sent via routing activation or diagnostic message.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_DoIP_00062 :**		
Name	DoIPRoutingActivationRef		
Description	Reference to a DoIPRoutingActivation describing the possible routing activations of the DoIPTester		
Multiplicity	1..255		
Type	Reference to [DoIPRoutingActivation]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers


DoIPTester :
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 255
postBuildChangeable = true
+parameter
DoIPNumByteDiagAckNack :EcucIntegerParamDef
min = 0
max = 4294967295
+parameter
DoIPTesterSA :EcucIntegerParamDef
min = 0
max = 65535
+reference
DoIPRoutingActivationRef :
EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 255
+destination
DoIPRoutingActivation :
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 255
postBuildChangeable = true

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral* [14].