Document Title	Specification of Diagnostic over IP
Document Category	Software specification (SWS)
Document UID	418
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Alignment of connection handling and message processing with ISO 13400-2 • Editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Introduction of Security Event reporting • Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• New API: DoIP_AllowNonTlsDoIPConnection • Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• incorporated DoIP Multiplexed Testers
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• Most APIs reporting development errors no longer return with E_NOT_OK • Removed obsolete elements • Editorial changes
2020-11-30	R20-11	AUTOSAR Release Management	• CONC 649 DoIP Extension fully incorporated • Added internal tester support • Harmonized IP interface concept with ISO 13400 • Duplicate service IDs fixed • Harmonized error classification

△

2019-11-28	R19-11	AUTOSAR Release Management	• Introduced CONC 649 DoIP Extension in draft state • Updated the functionality of routing activation for security use-cases • Increased multiplicity of DoIP target address so more than 255 DoIP addresses could be used • Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Updated the functionality to receive vehicle announcements • Support to add an increased number of DoIP target addresses • DoIP header file clean-up • Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Support for DoIP UDP Connections with limited broadcast IP addresses • Support for Further Action Code values for vehicle identification and vehicle announcement • Alignment of routing activation confirmation with ISO 13400 • Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Support for DoIP Activation line switch • Support for UDP multicast vehicle announcement • Introduction of reliable TxConfirmation • Harmonization of identical APIs functions within BSW
2015-07-31	4.2.2	AUTOSAR Release Management	• DET Renaming and Extension Incorporation • Support for parallel diagnostic sessions

▽

△

2014-10-31	4.2.1	AUTOSAR Release Management	• Harmonization of identical APIs within BSW • Handling UUDT messages within DoIP • Harmonization of callback functions and configuration parameter names • Editorial changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Harmonization of identical APIs • Multiplicity of some configuration parameters were updated • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Formalization of Service Interfaces • Revised return values of Service Interfaces • Editorial changes
2013-03-15	4.1.1	AUTOSAR Release Management	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and abbreviations 10
3 Related documentation 11
3.1 Input documents & related standards and norms . . . 11
3.2 Related standards and norms . . . 11
3.3 Related specification . . . 11
4 Constraints and assumptions 12
4.1 Applicability to car domains . . . 12
5 Dependencies to other modules 13
5.1 Socket Adaptor (SoAd) . . . 13
5.2 Pdu Router (PduR) . . . 13
5.3 Diagnostic Communication Manager (Dcm) . . . 13
5.4 Default Error Tracer (Det) . . . 14
5.5 File structure . . . 14
5.5.1 Code file structure . . . 14
6 Requirements Tracing 15
7 Functional specification 18
7.1 DoIP usage scenarios . . . 18
7.1.1 DoIP Internal Tester Functionality Extension . . . 19
7.2 DoIP Interface Connection Lifecycle Management . . . 20
7.2.1 Activation Line Handling and Connection Opening . . . 20
7.2.2 Socket and Connection Management . . . 22
7.2.3 GID Synchronization . . . 24
7.2.4 DHCP Hostname Handling . . . 24
7.2.5 TCP Connection Lifecycle and Inactivity Handling . . . 25
7.2.6 Routing Activation and Security . . . 26
7.3 DoIP Message layout according ISO 13400-2 . . . 26
7.3.1 Generic DoIP header . . . 26
7.3.2 Payload types . . . 29
7.3.2.1 Generic acknowledge . . . 29
7.3.2.2 Vehicle Identification . . . 32
7.3.2.3 Routing activation . . . 38
7.3.2.4 Alive check . . . 45
7.3.2.5 Node information . . . 46
7.3.2.6 Diagnostic Message . . . 48
7.4 UDP communication . . . 54
7.5 TCP communication . . . 55
7.5.1 Reception of a TCP DoIP message . . . 55

7.5.2 Transmission of a TCP DoIP message . . . 58
7.6 Security Events . . . 61
7.7 Error classification . . . 62
7.7.1 Development Errors . . . 62
7.7.2 Runtime Errors . . . 62
7.7.3 Production Errors . . . 62
8 API specification 63
8.1 Imported types . . . 63
8.2 Type definitions . . . 65
8.2.1 DoIP_ReturnType . . . 65
8.2.2 DoIP_ConfigType . . . 65
8.3 Function definitions . . . 66
8.3.1 DoIP_TpTransmit . . . 66
8.3.2 DoIP_TpCancelTransmit . . . 67
8.3.3 DoIP_TpCancelReceive . . . 67
8.3.4 DoIP_IfTransmit . . . 68
8.3.5 DoIP_IfCancelTransmit . . . 69
8.3.6 DoIP_Init . . . 69
8.3.7 DoIP_ActivationLineSwitch . . . 70
8.3.8 DoIP_TriggerVehicleAnnouncement . . . 70
8.3.9 DoIP_AllowNonTlsDoIPConnection . . . 71
8.3.10 DoIP_SoAdTpCopyTxData . . . 72
8.3.11 DoIP_SoAdTpTxConfirmation . . . 73
8.3.12 DoIP_SoAdTpCopyRxData . . . 74
8.3.13 DoIP_SoAdTpStartOfReception . . . 75
8.3.14 DoIP_SoAdTpRxIndication . . . 76
8.3.15 DoIP_SoAdIfRxIndication . . . 77
8.3.16 DoIP_SoAdIfTxConfirmation . . . 78
8.3.17 DoIP_SoConModeChg . . . 79
8.3.18 DoIP_LocalIpAddrAssignmentChg . . . 79
8.4 Scheduled functions . . . 80
8.4.1 DoIP_MainFunction . . . 80
8.5 Expected Interfaces . . . 81
8.5.1 Mandatory Interfaces . . . 81
8.5.2 Optional Interfaces . . . 82
8.5.3 Configurable interfaces . . . 82
8.5.3.1 <User>_DoIPGetPowerModeCallback . . . 83
8.5.3.2 <User>_DoIPRoutingActivationConfirmation . . . 83
8.5.3.3 <User>_DoIPRoutingActivationAuthentication . . . 84
8.5.3.4 <User>_DoIPTriggerGidSyncCallback . . . 85
8.5.3.5 <User>_DoIPGetGidCallback . . . 85
8.5.3.6 <User>_DoIPGetFurtherActionByteCallback . . . 86
8.5.4 DoIP Service Component . . . 86

9 Sequence diagrams 93
9.1 UDP DoIP communication . . . 93
9.2 Rx TCP message . . . 94
9.3 Tx TCP message . . . 95
9.4 Activation Line Handling - Active . . . 96
9.5 Activation Line Handling - Inactive . . . 97
10 Configuration specification 98
10.1 How to read this chapter . . . 98
10.2 Configuration and configuration parameters . . . 98
10.2.1 Variants . . . 98
10.2.2 DoIP . . . 98
10.2.3 DoIPGeneral . . . 99
10.2.4 DoIPGetGidCallback . . . 105
10.2.5 DoIPPowerModeCallback . . . 106
10.2.6 DoIPTriggerGidSyncCallback . . . 107
10.2.7 DoIPSecurityEvents . . . 108
10.2.8 DoIPConfigSet . . . 110
10.2.9 DoIPInterface . . . 112
10.2.10 DoIPChannel . . . 120
10.2.11 DoIPPduRRxPdu . . . 121
10.2.12 DoIPPduRTxPdu . . . 122
10.2.13 DoIPConnections . . . 124
10.2.14 DoIPTargetAddress . . . 126
10.2.15 DoIPTcpConnection . . . 126
10.2.16 DoIPSoAdTcpRxPdu . . . 128
10.2.17 DoIPSoAdTcpTxPdu . . . 129
10.2.18 DoIPUdpConnection . . . 130
10.2.19 DoIPSoAdUdpRxPdu . . . 131
10.2.20 DoIPSoAdUdpTxPdu . . . 132
10.2.21 DoIPUdpVehicleAnnouncementConnection . . . 133
10.2.22 DoIPSoAdUdpVehicleAnnouncementTxPdu . . . 134
10.2.23 DoIPFurtherActionByteCallback . . . 135
10.2.24 DoIPProtocolVersion . . . 136
10.2.25 DoIPRoutingActivation . . . 138
10.2.26 DoIPRoutingActivationAuthenticationCallback . . . 140
10.2.27 DoIPRoutingActivationConfirmationCallback . . . 142
10.2.28 DoIPTester . . . 144
10.3 Published Information . . . 145
A Change History 146
A.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 146
A.1.1 Added Specification Items in R26-11 . . . 146

A.1.2 Changed Specification Items in R26-11 . . . 146
A.1.3 Deleted Specification Items in R26-11 . . . 150
A.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 150
A.2.1 Added Specification Items in R25-11 . . . 150
A.2.2 Changed Specification Items in R25-11 . . . 151
A.2.3 Deleted Specification Items in R25-11 . . . 153
A.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 153
A.3.1 Added Specification Items in R24-11 . . . 153
A.3.2 Changed Specification Items in R24-11 . . . 153
A.3.3 Deleted Specification Items in R24-11 . . . 153
A.4 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 154
A.4.1 Added Specification Items in R23-11 . . . 154
A.4.2 Changed Specification Items in R23-11 . . . 154
A.4.3 Deleted Specification Items in R23-11 . . . 154
A.5 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 155
A.5.1 Added Specification Items in R22-11 . . . 155
A.5.2 Changed Specification Items in R22-11 . . . 155
A.5.3 Deleted Specification Items in R22-11 . . . 156
A.5.4 Added Constraints in R22-11 . . . 156
A.5.5 Changed Constraints in R22-11 . . . 156
A.5.6 Deleted Constraints in R22-11 . . . 156

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 1.1.

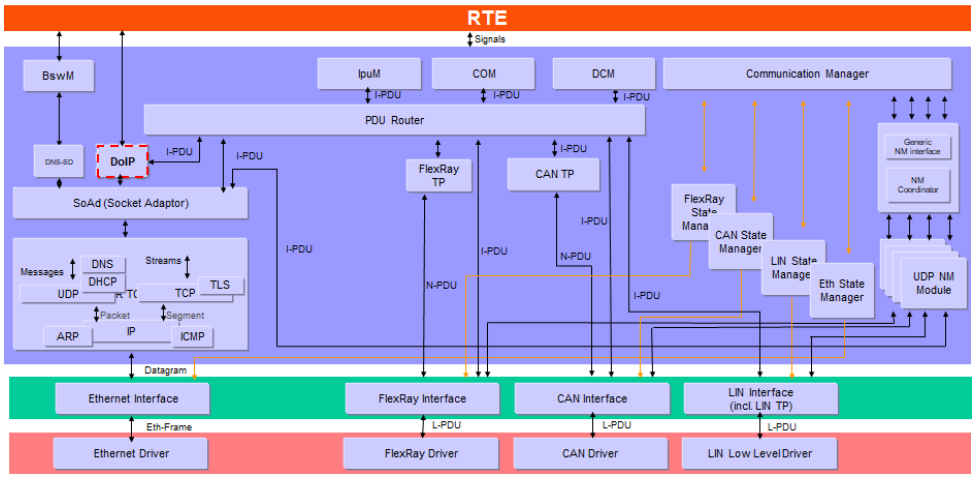

Figure 1.1: DoIP in the AUTOSAR ComStack Stack Architecture

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 & related standards and norms

[1] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[2] Specification of Socket Adaptor
AUTOSAR_CP_SWS_SocketAdaptor

[3] Specification of TCP/IP Stack
AUTOSAR_CP_SWS_TcpIp

[4] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter

[5] Specification of Diagnostic Communication Manager
AUTOSAR_CP_SWS_DiagnosticCommunicationManager

[6] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[7] ISO 13400-2:2019 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services (Edition 2, Release 2019-12)
https://www.iso.org/standard/74785.html

[8] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[9] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related standards and norms

No content.

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [1] (SWS BSW General), which is also valid for the DoIP module.

Thus, the specification SWS BSW General [1] 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 [2] 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 [3] that supports IP, TCP, UDP,IPv4, IPv6 and address assignment mechanisms like AutoIP and DHCP.

5.2 Pdu Router (PduR)

The Pdu Router [4] 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 [5] is the module providing the VIN to the DoIP module. Additionally the Dcm will execute the ECU local diagnostic routed via PduR.

5.4 Default Error Tracer (Det)

If the configuration parameter DoIPDevelopmentErrorDetect is set to true and a DoIP API is called with incorrect parameters, the Default Error Tracer [6] is called with an error ID.

5.5 File structure

5.5.1 Code file structure

For details refer to [1] Chapter 5.1.6 *"Code file structure"*.

6 Requirements Tracing

Requirement	Description	Satisfied by
[RS_Diag_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_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] [SWS_DoIP_00277] [SWS_DoIP_00278] [SWS_DoIP_00279] [SWS_DoIP_00284] [SWS_DoIP_00311]
[RS_Diag_00025]	Valid DoIP messages shall be recognized	[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_00019] [SWS_DoIP_00292] [SWS_DoIP_00293] [SWS_DoIP_00321] [SWS_DoIP_00322]
[RS_Diag_00026]	DoIP Vehicle Identification shall be provided	[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] [SWS_DoIP_00287] [SWS_DoIP_00288] [SWS_DoIP_00289] [SWS_DoIP_00290] [SWS_DoIP_00291]



△

Requirement	Description	Satisfied by
[RS_Diag_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]
[RS_Diag_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_00204] [SWS_DoIP_00234] [SWS_DoIP_00235] [SWS_DoIP_00241] [SWS_DoIP_00243] [SWS_DoIP_00296] [SWS_DoIP_00297] [SWS_DoIP_00298] [SWS_DoIP_00306] [SWS_DoIP_00358]
[RS_Diag_00047]	DoIP shall be able to access the DHCP host name option.	[SWS_DoIP_00154] [SWS_DoIP_00155] [SWS_DoIP_00156]
[RS_Diag_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]
[RS_Diag_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_00204] [SWS_DoIP_00234] [SWS_DoIP_00235] [SWS_DoIP_00241] [SWS_DoIP_00243] [SWS_DoIP_00296] [SWS_DoIP_00297] [SWS_DoIP_00298] [SWS_DoIP_00306] [SWS_DoIP_00358]
[RS_Diag_00082]	DoIP shall implement a mechanism to retrieve Entity Status	[SWS_DoIP_00094] [SWS_DoIP_00095] [SWS_DoIP_00096] [SWS_DoIP_00097] [SWS_DoIP_00098] [SWS_DoIP_00099] [SWS_DoIP_00100]
[RS_Diag_00083]	DoIP shall implement a mechanism to check if diagnostic testers are alive	[SWS_DoIP_00058] [SWS_DoIP_00105] [SWS_DoIP_00107] [SWS_DoIP_00115] [SWS_DoIP_00139] [SWS_DoIP_00140] [SWS_DoIP_00142] [SWS_DoIP_00157] [SWS_DoIP_00358]
[RS_Diag_00084]	DoIP shall implement routing activation mechanism	[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_00114] [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] [SWS_DoIP_00294] [SWS_DoIP_00295]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00488]	Classification of security events	[SWS_DoIP_00500] [SWS_DoIP_00501] [SWS_DoIP_00502] [SWS_DoIP_00503] [SWS_DoIP_00504] [SWS_DoIP_00505] [SWS_DoIP_00506] [SWS_DoIP_00507]
[SRS_BSW_00489]	Reporting of security events	[SWS_DoIP_00500] [SWS_DoIP_00501] [SWS_DoIP_00502] [SWS_DoIP_00503] [SWS_DoIP_00504] [SWS_DoIP_00505] [SWS_DoIP_00506] [SWS_DoIP_00507]

Table 6.1: Requirements Tracing

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 7.1) to broader interconnectivity of the car or single ECUs with the environment (see Figure 7.2).

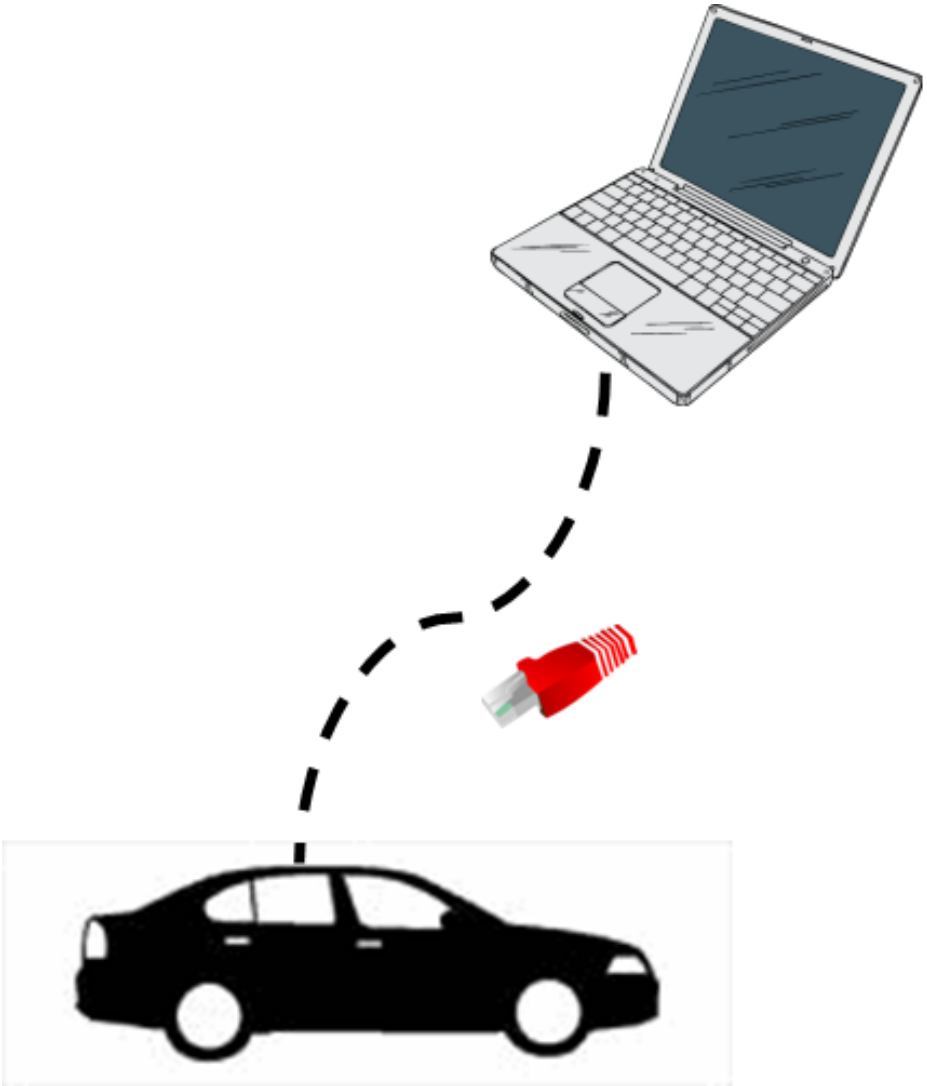

Figure 7.1: Connection of an external test equipment directly to the car (see ISO 13400-1 [7])

DoIP defines several services within the DoIP entities to fulfil the requirements of ISO 13400 [7]:

The key DoIP services include:

- Vehicle identification and announcement: Detects participants in the DoIP communication.
- Routing Activation: Enables selective activation of diagnostic message paths to differentiate protocols (e.g. UDS and OBD) and individual testers.
- Node information: Provides protocol-relevant status information of the DoIP entity to connected testers.
- Alive mechanism: Maintains and monitors active tester connections.

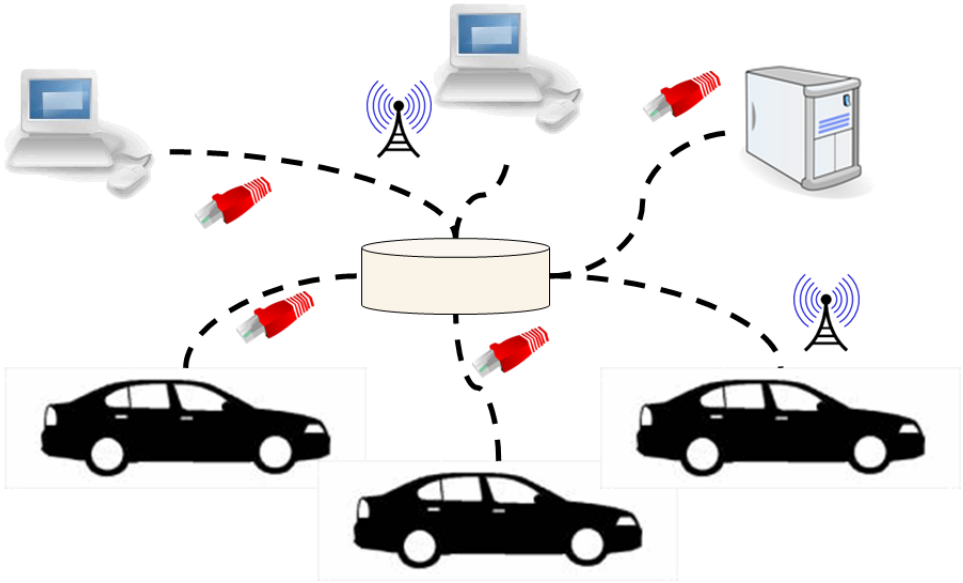

Figure 7.2: Highly interconnected system of several Cars via the DoIP protocol (see ISO 13400-1 [7])

7.1.1 DoIP Internal Tester Functionality Extension

This use case covers in-vehicle DoIP communication, where the tester(s) can also reside within the vehicle network.

An ECU/DoIP node may be multi-homed, i.e. it can have multiple logical IP interfaces (potentially sharing the same physical Ethernet interface/MAC address).

In this case, the ECU shall be able to communicate on each of its IP interfaces via DoIP independently, i.e. DoIP functionalities on each IP interface have to be isolated from each other. This means, for example:

- An Activation-Line-Low trigger is restricted to a certain IP interface. Only the connections aggregated to the IP interface for which an Activation-Line-Low occurred

are closed down — not all DoIP connections on all interfaces. Each interface has its own logical activation line.

- During routing activation, checks for a source address that has already been registered/activated shall also be restricted to that interface. A tester with a given SA can have a valid routing activation on two different interfaces, but not on two different connections of the same interface.

DoIP communication on the vehicle-internal IP interface typically differs from the external interface:

- The internal IP interface is typically always active/enabled through the lifecycle of the ECU.
- The internal IP interface typically has a static IP address assigned.
- The internal IP interface therefore typically has no assigned Activation-Line semantics, i.e. on an abstract level the Activation Line is always “active”.

To support multi-homed ECUs/DoIP nodes, there shall be the possibility for a DoIP module (CP or AP) to:

- Configure on which interfaces it shall operate.
- Per interface configure:
  - Whether Activation Line functionality is needed.
  - Whether dynamic IP address assignment shall take place.
  - Whether vehicle announcement shall be performed (and when).

7.2 DoIP Interface Connection Lifecycle Management

This chapter describes the complete lifecycle of DoIP interface connections, from activation and establishment, through management and maintenance, to closure and security.

7.2.1 Activation Line Handling and Connection Opening

This section covers how the DoIP module monitors and reacts to the activation line status, and how it establishes or tears down connections based on this status.

[SWS_DoIP_00201] Activation Line Status Determination

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈The DoIP module shall determine the DoIP Activation Line status by the call to `DoIP_ActivationLineSwitch()` based on the value of `Active` per `DoIPInterface` with a given `InterfaceId`. The Activation Line status is considered "active", if the

boolean value in the call is set to TRUE. The Activation Line status is considered "inactive", if the boolean value in the call is set to FALSE.⌋

[SWS_DoIP_00202] Reject Requests on Inactive Activation Line

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈If data is received from SoAd or PduR (i.e. communication related interfaces are called) via PduIds related to a certain DoIPInterface configured with DoIPInterfaceActLineCtrl equals to TRUE, where the status of the Activation Line of this DoIPInterface is currently inactive, the DoIP module shall ignore all these requests and return a negative return value.⌋

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_E_NOT_OK.

[SWS_DoIP_00204] Connection Establishment for Activation Line Controlled Interfaces

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈For activation line controlled DoIP Interfaces with DoIPInterfaceActLineCtrl equals to TRUE, DoIP shall establish the corresponding connections for these interfaces according to [SWS_DoIP_00306] if corresponding Activation Line Status switches to "active".⌋

[SWS_DoIP_00296] Connection Establishment for Non-Activation Line Controlled Interfaces

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈For non activation line controlled DoIP Interfaces with DoIPInterfaceActLineCtrl equals to FALSE, DoIP shall establish the corresponding connections for these interfaces according to [SWS_DoIP_00306] in context of first call to DoIP_MainFunction().⌋

[SWS_DoIP_00306] DoIP Socket Connections Establishment

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈To open the socket connections (triggered by [SWS_DoIP_00204] or [SWS_DoIP_00296]) of a DoIP interface the DoIP module shall loop over all its associated DoIPTcpConnection, DoIPUdpConnection and DoIPUdpVehicleAnnouncementConnection. For each DoIP connection belonging to respective DoIP Interfaces which has a DoIPRequestAddressAssignment set to true, the DoIP module shall retrieve the corresponding socket connection Id through SoConIdPtr via call to the SoAd_GetSoConId() and trigger the IP Address assignment via subsequent calls to SoAd_RequestIpAddrAssignment() with the retrieved socket connection Id, LocalIpAddrPtr and DefaultRouterPtr set to NULL_PTR, Netmask set to 0, and Type set to TCPIP_IPADDR_ASSIGNMENT_ALL. For each of these DoIP connections (irrespective of the value of DoIPRequestAddressAssignment)

the DoIP module shall open the respective connection by an according call to `SoAd_OpenSoCon()`.⌋

[SWS_DoIP_00234] Connection Closure on Inactive Activation Line

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈If the Activation Line status of a `DoIPInterface` switches to "inactive", the DoIP module shall loop over all `DoIPTcpConnection`, `DoIPUdpConnection`, and `DoIPUdpVehicleAnnouncementConnection`. For each of these DoIPConnections the DoIP module shall retrieve the corresponding socket connection Id through `SoConIdPtr` via call to the `SoAd_GetSoConId()` and close all the connections by a call to `SoAd_CloseSoCon()` with the retrieved socket connection Id.⌋

[SWS_DoIP_00235] IP Address Release on Inactive Activation Line

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈In addition to [SWS_DoIP_00234], the DoIP module shall release the corresponding IP Address assignment via the call to `SoAd_ReleaseIpAddrAssignment()` for those connections, which belong to the `DoIPInterface` for which the Activation Line status switched to "inactive", that have `DoIPRequestAddressAssignment` set to true.⌋

7.2.2 Socket and Connection Management

This section details how the DoIP module maintains, monitors, and manages the state of socket and connection resources, including handling mode changes and connection closure.

[SWS_DoIP_00001] UDP Connection State Maintenance

Upstream requirements: RS_Diag_00081

⌈The DoIP module shall maintain the State of the SocketConnection of each configured `DoIPUdpConnection`⌋

[SWS_DoIP_00002] TCP Connection State Maintenance

Upstream requirements: RS_Diag_00081, RS_Diag_00028

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

(a) `DoIPSoAdTcpRxPduId`, 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⌋

[SWS_DoIP_00241] Socket Connection Online Handling

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈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.⌋

[SWS_DoIP_00243] Socket Connection Offline Handling

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈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].⌋

[SWS_DoIP_00205] Vehicle Announcement on Socket Online

Upstream requirements: RS_Diag_00026

⌈If the function `DoIP_SoConModeChg()` is called with Mode set to `SOAD_SOCON_ONLINE` for a UDP vehicle announcement connection, the DoIP module shall send the vehicle announcement message via the corresponding Tx PDU configured in the `DoIPUdpVehicleAnnouncementConnection` and belonging to the reported socket connection.⌋

[SWS_DoIP_00058] Graceful Connection Closure

Upstream requirements: RS_Diag_00081, RS_Diag_00028, RS_Diag_00083

⌈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 `FALSE` and the SoConId determined by a call to the function `SoAd_GetSoConId()` for the corresponding `DoIPSoAdTcpTxPdu`.⌋

Note: Graceful closure applies when the connection peer is still reachable, allowing a response to be sent before closing. For the complete list of triggering scenarios, refer to ISO 13400-2 [7].

[SWS_DoIP_00358] Immediate Socket Connection Termination

Upstream requirements: RS_Diag_00081, RS_Diag_00028, RS_Diag_00083

⌈If a connection needs to be reset 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()` for the corresponding `DoIPSoAdTcpTxPdu`.⌋

Note: Immediate termination applies when the connection peer is considered unreachable (e.g. timeouts), making an orderly shutdown unnecessary. For the complete list of triggering scenarios, refer to ISO 13400-2 [7].

[SWS_DoIP_00115] TCP Socket Connection Cleanup

Upstream requirements: RS_Diag_00028, RS_Diag_00081, RS_Diag_00083

⌈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] or [SWS_DoIP_00358]) the DoIP module shall:

- unregister and release the socket connection to the related Tester and

- discard the ongoing diagnostic message processing.⌋

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.

7.2.3 GID Synchronization

This section describes the process for GID synchronization, which is triggered after successful IP address assignment and managed during the connection lifecycle.

[SWS_DoIP_00076] GID Synchronization Trigger

Upstream requirements: RS_Diag_00026

⌈If the parameter `DoIPVinGidMaster` is set to true and the Container `DoIPTriggerGidSyncCallback` is configured, the DoIP module shall call the <User>_DoIPTriggerGIDSynchronization function (after successful IP Address assignment, see [SWS_DoIP_00306]) and repeat 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.⌋

[SWS_DoIP_00085] GID Synchronization Stop on IP Change

Upstream requirements: RS_Diag_00028, RS_Diag_00081

⌈If a change in the IP address assignment indicated by `DoIP_LocalIpAddrAssignmentChg()` with another `TcpIp_IpAddrStateType` then `TCPIP_IPADDR_STATE_ASSIGNED`, the function to start GID synchronization as described in [SWS_DoIP_00076] shall not be called any longer independent from the before return value.⌋

7.2.4 DHCP Hostname Handling

This section explains how the DoIP module manages DHCP hostnames, ensuring correct network identification.

[SWS_DoIP_00154] DHCP Hostname Retrieval

Upstream requirements: RS_Diag_00047

⌈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-".⌋

[SWS_DoIP_00155] DHCP Hostname Default Setting

Upstream requirements: RS_Diag_00047

⌈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.⌋

[SWS_DoIP_00156] DHCP Hostname VIN Setting

Upstream requirements: RS_Diag_00047

⌈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.⌋

7.2.5 TCP Connection Lifecycle and Inactivity Handling

This section covers the management of TCP connection inactivity, including timers, routing activation, and connection resets due to inactivity.

[SWS_DoIP_00142] General Inactivity Timer

Upstream requirements: RS_Diag_00083

⌈The DoIP module shall maintain a general inactivity timer for each registered TCP connection. The general inactivity timer shall be initialized, started, and reset according to the ISO 13400-2 [7] rules, using `DoIPGeneralInactivityTime` as the corresponding configuration parameter.⌋

[SWS_DoIP_00157] Initial Inactivity Timer

Upstream requirements: RS_Diag_00083

⌈The DoIP module shall maintain an initial inactivity timer for each registered TCP connection. The initial inactivity timer shall be initialized, started, and reset according to the ISO 13400-2 [7] rules, using `DoIPInitialInactivityTime` as the corresponding configuration parameter.⌋

Note: AUTOSAR allows configuration of both `T_TCP_Initial_Inactivity` and `T_TCP_General_Inactivity`. The initial inactivity timer is initialized upon successful socket opening, as notified via `DoIP_SoConModeChg()`.

7.2.6 Routing Activation and Security

This section details the handling of routing activation requests and the associated security requirements for TCP connections.

[SWS_DoIP_00294] Unsecured TCP Connection Routing Activation

Upstream requirements: RS_Diag_00084

⌈When receiving a routing activation request on a TCP connection where `DoIPTcpConnectionSecurityRequired` is not set or set to `FALSE`, the DoIP module shall search for a `DoIPTester` with an assigned container that matches `DoIPTesterSA`. If such a `DoIPTester` container was found and the matching `DoIPRoutingActivation` container (refer to [SWS_DoIP_00108]) has the attribute `DoIPRoutingActivationSecurityRequired` not set or set to `FALSE`, the connection will be established.

If such a `DoIPTester` container was found and the matching `DoIPRoutingActivation` container (refer to [SWS_DoIP_00108]) has the attribute `DoIPRoutingActivationSecurityRequired` is set to `TRUE`, the connection shall be rejected with the response code `0x07`.⌋

[SWS_DoIP_00295] Secured TCP Connection Routing Activation

Upstream requirements: RS_Diag_00084

⌈When receiving a routing activation request on a TCP connection where `DoIPTcpConnectionSecurityRequired` set to `TRUE`, the DoIP module shall search for a `DoIPTester` with an assigned container that matches `DoIPTesterSA`. If such a `DoIPTester` container was found, the connection will be established.⌋

Rationale: A secure TCP connection can be established with a DoIPTester that requests a secure or unsecured connection.

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 [7] table 11. The DoIP header is described in this chapter.

[SWS_DoIP_00004]

Upstream requirements: RS_Diag_00025

⌈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 7.1: DoIP message Generic header Layout

[SWS_DoIP_00005]

Upstream requirements: RS_Diag_00025

⌈Byte 0 of the DoIP header shall contain the protocol version e.g. 0x02.⌋

[SWS_DoIP_00333] Used DoIP protocol version ⌈The DoIP module shall use the configured DoIPProtocolVersion [ECUC_DoIP_00102] as DoIP protocol version in all generic headers of send DoIP messages.⌋

[SWS_DoIP_00006]

Upstream requirements: RS_Diag_00025

⌈The Byte 1 of the DoIP header shall contain the inverse protocol version e.g. 0xFD value shall be added if the protocol version is 0x02.⌋

[SWS_DoIP_00321]

Upstream requirements: RS_Diag_00025

⌈If config parameter DoIP_ISO13400_2_2012 is enabled, the DoIP entity shall accept received frames with protocol version 0x02.⌋

[SWS_DoIP_00322]

Upstream requirements: RS_Diag_00025

⌈If config parameter DoIP_ISO13400_2_2019 is enabled, the DoIP entity shall accept received frames with protocol version 0x03.⌋

[SWS_DoIP_00323] ⌈If a routing activation request with protocol version 0x02 was received on an unsecured connection to activate a secured route, then the DoIP entity shall send back the routing activation response 0x06 and close this socket connection according to [SWS_DoIP_00058].⌋

[SWS_DoIP_00324] ⌈The DoIP entity shall use the protocol version configured in DoIPAnnouncedProtocolVersion as advertised protocol version in vehicle announcements.⌋

[SWS_DoIP_00325] ⌈If a vehicle identification request was received with protocol version 0xFF, the DoIP entity shall use the protocol version configured in DoIPAnnouncedProtocolVersion as protocol version in the vehicle identification response.⌋

[SWS_DoIP_00326] ⌈For transmission of DoIP responses over Tcp, the DoIP entity shall use the same protocol version as received and extracted from the first request of this tester.⌋

[SWS_DoIP_00327] ⌈For transmission of DoIP responses over UDP, the DoIP entity shall use the same protocol version as received and extracted from the request.⌋

[SWS_DoIP_00007]

Upstream requirements: RS_Diag_00025

⌈Byte 2 and Byte 3 shall contain the PayloadType.⌋

[SWS_DoIP_00008]

Upstream requirements: RS_Diag_00025

⌈The following PayloadTypes shall be supported for reception 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
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
0x0004	Vehicle announcement message/vehicle identification response message	7.3.2.2.1	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 7.2: DoIP payload types received by a DoIP entity, chapter reference and the connection type they are received on

[SWS_DoIP_00009]

Upstream requirements: RS_Diag_00025

⌈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
0x8001	Diagnostic Message	7.3.2.6.1	TCP
0x8002	Diagnostic message positive acknowledgement	7.3.2.6.2	TCP
0x8003	Diagnostic message negative acknowledgement	7.3.2.6.3	TCP

Table 7.3: DoIP payload types transmitted by a DoIP entity, chapter reference and the connection type they are transmitted on

[SWS_DoIP_00010]

Upstream requirements: RS_Diag_00025

⌈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).⌋

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]

Upstream requirements: RS_Diag_00025

⌈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 Socket-Connection status has not changed since the reception of the DoIP message⌋

[SWS_DoIP_00013]

Upstream requirements: RS_Diag_00025

⌈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].⌋

[SWS_DoIP_00014]

Upstream requirements: RS_Diag_00025

⌈If the Protocol information is incorrect, (see [SWS_DoIP_00005], [SWS_DoIP_00006], [SWS_DoIP_00321], [SWS_DoIP_00322], 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]).⌋

[SWS_DoIP_00016]

Upstream requirements: RS_Diag_00025

⌈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 an unknown payload type was requested. The message shall be discarded for further processing. For TCP_DATA socket connections, this NACK shall only be sent as response in the ISO 13400 connection state "Registered [Routing Active]".⌋

[SWS_DoIP_00017]

Upstream requirements: RS_Diag_00025

⌈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. For TCP_DATA socket connections, this NACK shall only be sent as response in the ISO 13400 connection state "Registered [Routing Active]".⌋

[SWS_DoIP_00019]

Upstream requirements: RS_Diag_00025

⌈If the DoIP module is called with a payload length that is not valid for the specific payload type, the NACK code 0x04 shall be sent and the corresponding connection shall be closed (see [SWS_DoIP_00058]). For TCP_DATA socket connections, this NACK shall be sent as response for payload type "Routing Activation Request" in any ISO 13400 connection state and for payload type "Alive Check Response" in any ISO 13400 connection state "Registered".

All other DoIP messages received on the TCP_DATA socket connections shall only be responded in the ISO 13400 connection state "Registered [Routing Active]".⌋

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).

[SWS_DoIP_00292]

Upstream requirements: RS_Diag_00025

⌈If a DoIP message with payload Type 0x0000 is received on a configured DoIPUdpConnection or DoIPTcpConnection, the message shall be discarded.⌋

[SWS_DoIP_00504] Header Validation Failure Security Event

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈If `DoIPEnableSecurityEventReporting` is enabled and `SEV_DOIP_HEADER_CHECK_FAILED` exists, the DoIP module shall report the security event SEV_DOIP_HEADER_CHECK_FAILED to IdsM with the context data specified in [SWS_DoIP_00511] when a message is rejected due to a failure in generic acknowledgement checks. These checks include:

- Incorrect protocol information
- Unsupported payload type
- Payload length exceeding `DoIPMaxRequestBytes`
- Inability to provide a buffer of the required length
- Payload length that is not valid for the specific payload type

⌋

Note: Refer to the DoIP generic header check as described in ISO 13400-2 [7] in the figure "DoIP generic header handler".

[SWS_DoIP_00511] Security event context data definition: SEV_DOIP_HEADER_CHECK_FAILED ⌈

SEV Name	***SEV_DOIP_HEADER_CHECK_FAILED***	
ID	127	
Description	The DoIP Header Handler rejected a request (routing or diagnostic message).	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
ProtocolVersion	uint8 [1]	
PayloadType	uint8 [2]	
NACKCode	uint8 [1]	

⌋

7.3.2.2 Vehicle Identification

[SWS_DoIP_00015]

Upstream requirements: RS_Diag_00026

⌈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].⌋

7.3.2.2.1 Vehicle Identification request (payload type 0x0001)

[SWS_DoIP_00061]

Upstream requirements: RS_Diag_00026

⌈When the module receives a DoIP message with payload type 0x0001 on a connection different than a configured DoIPUdpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00059]

Upstream requirements: RS_Diag_00026

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

[SWS_DoIP_00060]

Upstream requirements: RS_Diag_00026

⌈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 DoIPInitialVehicleAnnounceTime with payload type 0x0004.⌋

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 7.4:

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

Table 7.4: Vehicle identification request with EID payload data

[SWS_DoIP_00062]

Upstream requirements: RS_Diag_00026

⌈When the module receives a DoIP message with payload Type 0x0002 on a connection different than a configured DoIPUdpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00063]

Upstream requirements: RS_Diag_00026

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

[SWS_DoIP_00064]

Upstream requirements: RS_Diag_00026

⌈If a DoIP message with payload Type 0x0002 is received on the configured DoIPUdpConnection, the DoIP module shall further process the message.⌋

[SWS_DoIP_00065]

Upstream requirements: RS_Diag_00026

⌈If the Parameter DoIPUseMacAddressForIdentification 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.⌋

[SWS_DoIP_00066]

Upstream requirements: RS_Diag_00026

⌈If the Parameter DoIPUseMacAddressForIdentification 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.⌋

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 7.5:

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

Table 7.5: Vehicle identification request with VIN payload data

[SWS_DoIP_00067]

Upstream requirements: RS_Diag_00026

⌈When the module receives a DoIP message with payload Type 0x0003 on a connection different than a configured DoIPUdpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00068]

Upstream requirements: RS_Diag_00026

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

[SWS_DoIP_00069]

Upstream requirements: RS_Diag_00026

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

[SWS_DoIP_00070]

Upstream requirements: RS_Diag_00026

⌈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 or the returned VIN do not match the requested VIN, the DoIP message with payload Type 0x0003 shall be ignored. 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.⌋

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

[SWS_DoIP_00297]

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈For a DoIP Interface with DoIPInterfaceAnnouncementStart = DOIP_AUTOMATIC_ANNOUNCE, the DoIP module shall start Vehicle announcement according to [SWS_DoIP_00205].⌋

[SWS_DoIP_00298]

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈For a DoIP Interface with DoIPInterfaceAnnouncementStart = DOIP_ONTRIGGER_ANNOUNCE, the sending of vehicle announcement only starts if DoIP_TriggerVehicleAnnouncement () has been called for that Interface.⌋

[SWS_DoIP_00299] ⌈If DoIP_TriggerVehicleAnnouncement() is called, but the corresponding socket is not yet ONLINE then the request shall be remembered and vehicle announcement shall be sent as soon as the socket goes ONLINE.⌋

[SWS_DoIP_00071]

Upstream requirements: RS_Diag_00026

⌈If the DoIP module needs to send a vehicle announcement message (see [SWS_DoIP_00205] and [SWS_DoIP_00298]), it shall send the vehicle announcement message via the configured DoIPUdpVehicleAnnouncementConnection after DoIPInitialVehicleAnnouncementTime. This message shall be sent DoIPVehicleAnnouncementCount times with a delay of DoIPVehicleAnnouncementInterval between each message. The last "VIN/GID Status" byte of the Vehicle identification response message is optional as defined in the ISO 13400-2 standard. It shall exist only if the "DoIPUseVehicleIdentificationSyncStatus" configuration parameter is set to True. (See [SWS_DoIP_00086]).⌋

The payload data structure of a vehicle identification response/vehicle announcement message shall be supported as described in Table 7.6.

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 byte	31	1
VIN/GID Status	32	1

Table 7.6: Vehicle identification response/vehicle announcement message payload data

[SWS_DoIP_00072]

Upstream requirements: RS_Diag_00026

⌈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 with "Further Action Required" field set to 0x00 and VIN/GID sync. Status field set to 0x10 if (DoIPUseVehicleIdentificationSyncStatus) is set to true.⌋

[SWS_DoIP_00073]

Upstream requirements: RS_Diag_00026

⌈The "LA" of a vehicle identification response/vehicle announcement message shall contain the configured DoIPLogicalAddress.⌋

[SWS_DoIP_00074]

Upstream requirements: RS_Diag_00026

⌈The "EID" of a vehicle identification response/vehicle announcement message of a DoIPInterface shall contain the MAC address derived by SoAd_GetPhysAddr() if the corresponding configuration parameter DoIPUseMacAddressForIdentification is set to true.⌋

[SWS_DoIP_00075]

Upstream requirements: RS_Diag_00026

⌈The "EID" of a vehicle identification response/vehicle announcement message of a DoIPInterface shall contain the configured `DoIPEid` if the corresponding configuration parameter DoIPUseMacAddressForIdentification is set to false.⌋

[SWS_DoIP_00077]

Upstream requirements: RS_Diag_00026

⌈The "GID" of a vehicle identification response/vehicle announcement message shall contain the same value as the EID used in the interface referenced by DoIPUseEidAsGidRef.⌋

[SWS_DoIP_00078]

Upstream requirements: RS_Diag_00026

⌈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 reference DoIPUseEidAsGidRef is disabled.⌋

[SWS_DoIP_00079]

Upstream requirements: RS_Diag_00026

⌈The "GID" of a vehicle identification response/vehicle announcement message shall contain the value retrieved by the configured DoIPGetGidCallback function(for the signature see <User>_DoIPGetGidcallback, [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.⌋

[SWS_DoIP_00080]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00081]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00082]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00083]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00084]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00086]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00087]

Upstream requirements: RS_Diag_00026

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

[SWS_DoIP_00088]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00089]

Upstream requirements: RS_Diag_00026

⌈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.⌋

[SWS_DoIP_00291]

Upstream requirements: RS_Diag_00026

⌈The "Further action" byte of a vehicle identification response/vehicle announcement message shall contain the 1 Byte value retrieved by a call to the configured DoIPFurtherActionByteCallback (if configured, for the signature see <User>_DoIPGetFurtherActionByteCallback, [SWS_DoIP_00288]). If the function returns E_OK, the "Further action" byte shall be set to the retrieved value of FurtherActionByte. If the function returns E_NOT_OK, the "Further action" byte shall be set according to [SWS_DoIP_00082], [SWS_DoIP_00083] or [SWS_DoIP_00084].⌋

[SWS_DoIP_00293]

Upstream requirements: RS_Diag_00025

⌈If a DoIP message with payload Type 0x0004 is received on a configured DoIPUdpConnection, the message shall be discarded.⌋

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 7.7:

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 7.7: Routing activation request message payload data

[SWS_DoIP_00101]

Upstream requirements: RS_Diag_00084

⌈When the module receives a DoIP message with payload Type 0x0005 on a connection different than a configured DoIPTcpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00117]

Upstream requirements: RS_Diag_00084

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

[SWS_DoIP_00102]

Upstream requirements: RS_Diag_00084

⌈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.⌋

[SWS_DoIP_00106]

Upstream requirements: RS_Diag_00084

⌈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 [Routing Active] to another source address, the DoIP module shall send a routing activation response message 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].⌋

[SWS_DoIP_00104]

Upstream requirements: RS_Diag_00084

⌈If a routing activation request message is received with a "Source address" that does not match a configured DoIPTesterSA, the routing activation response message 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].⌋

[SWS_DoIP_00103]

Upstream requirements: RS_Diag_00084

⌈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.⌋

[SWS_DoIP_00105]

Upstream requirements: RS_Diag_00084, RS_Diag_00083

⌈If the received "Source Address" is already registered to another connection, belonging to the same DoIPInterface, an alive check request to this connection shall be triggered as described in [SWS_DoIP_00139] and [SWS_DoIP_00140] 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 0x03. 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].⌋

[SWS_DoIP_00107]

Upstream requirements: RS_Diag_00084, RS_Diag_00083

⌈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 [SWS_DoIP_00139] and [SWS_DoIP_00140]. 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. 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].⌋

[SWS_DoIP_00108]

Upstream requirements: RS_Diag_00084

⌈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_00109].⌋

[SWS_DoIP_00160]

Upstream requirements: RS_Diag_00084

⌈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. In this case the socket connection shall be closed as defined in [SWS_DoIP_00058].⌋

[SWS_DoIP_00109]

Upstream requirements: RS_Diag_00084

⌈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".⌋

[SWS_DoIP_00161]

Upstream requirements: RS_Diag_00084

⌈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 [SWS_DoIP_00120].⌋

[SWS_DoIP_00110]

Upstream requirements: RS_Diag_00084

⌈If the DoIPRoutingActivationAuthenticationCallback returns DOIP_E_PENDING, the DoIP module shall trigger the callback at next DoIP_MainFunction calls 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. In this case, the connection is in ISO 13400 connection state "Registered[Pending for Authentication]".⌋

[SWS_DoIP_00111]

Upstream requirements: RS_Diag_00084

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

[SWS_DoIP_00112]

Upstream requirements: RS_Diag_00084

⌈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 [SWS_DoIP_00120].⌋

[SWS_DoIP_00114]

Upstream requirements: RS_Diag_00084

⌈If the DoIPRoutingActivationConfirmationCallback returns DOIP_E_PENDING, the DoIP module shall send a routing activation response message once with the routing activation response code set to 0x11. In this case, the connection is in ISO 13400 connection state "Registered [Pending for Confirmation]".⌋

[SWS_DoIP_00274]

Upstream requirements: RS_Diag_00084

⌈If the DoIPRoutingActivationConfirmationCallback returns `E_NOT_OK`, the DoIP module shall send a routing activation response message with the activation response code set to `0x05` and the socket connection shall be closed as defined in [SWS_DoIP_00058].⌋

[SWS_DoIP_00113]

Upstream requirements: RS_Diag_00084

⌈If no response was sent because of the before mentioned checks, this DoIPRoutingActivation is confirmed, authorized and valid. A routing activation response message shall be sent with the activation response code set to 0x10 and the socket connection shall be considered as Registered [Routing Active] to this DoIPTesterSA and enable the routing for this routing activation.

Note: From now on the routing to the configured DoIPTargetAdressRef are activated and valid so the diagnostic request messages related to the specified DoIPTargetAdress received via this socket connection are routed to the respective target address.⌋

[SWS_DoIP_00330] ⌈The DoIP module shall silently discard any message other than routing activation request message or messages required for authentication or confirmation received on a socket connection in any ISO 13400 connection state other than "Registered [Routing Active]".

Exceptions are:

- The protocol version or inverse protocol version check. If this check fails it shall always be responded and handled according to [SWS_DoIP_00014].
- Alive check response is valid in any ISO 13400 connection state "Registered". Generic header checks for this payload type shall be responded.

⌋

[SWS_DoIP_00505] Routing Activation Failure Security Event

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈If `DoIPEnableSecurityEventReporting` is enabled and `SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED` exists, the DoIP module shall report the security event SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED to IdsM with the context data specified in [SWS_DoIP_00512] when a routing activation request is rejected due to a failure in "Routing activation checks" These checks include:

- Unknown source address
- All concurrently supported sockets are registered and active
- Source address (SA) received differs from the one already activated on the socket
- Source address (SA) received is already registered and active on a different socket
- Missing authentication
- Rejected confirmation
- Activation type requires a secure socket

⌋

Note: Refer to the DoIP routing activation check as described in ISO 13400-2 [7] in the figure "DoIP routing activation handler".

[SWS_DoIP_00512] Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED ⌈

SEV Name	***SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED***	
ID	128	
Description	A routing request was rejected by the routing handler.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
LogicalSourceAddress	uint8 [2]	
ActivationType	uint8 [1]	
ResponseCode	uint8 [1]	

⌋

[SWS_DoIP_00506] Successful Routing Activation Security Event

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈If `DoIPEnableSecurityEventReporting` is enabled and `SEV_DOIP_ROUTING_ACTIVATION_SUCCESS` exists, the DoIP module shall report the security event SEV_DOIP_ROUTING_ACTIVATION_SUCCESS to IdsM with the context data specified in [SWS_DoIP_00513] when a successful routing activation occurs.⌋

[SWS_DoIP_00513] Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_SUCCESS ⌈

SEV Name	*SEV_DOIP_ROUTING_ACTIVATION_SUCCESS*	
ID	129	
Description	A routing request was successful.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
LogicalSourceAddress	uint8 [2]	
ActivationType	uint8 [1]	

⌋

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 7.8:

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 7.8: Routing activation response message payload data

[SWS_DoIP_00116]

Upstream requirements: RS_Diag_00084

⌈The "Logical Address Tester" field shall be set to the Tester SA the according routing activation request message was received from.⌋

[SWS_DoIP_00118]

Upstream requirements: RS_Diag_00084

⌈The "Logical Address DoIP entity" shall be set to the configured parameter DoIPLogicalAddress.⌋

[SWS_DoIP_00119]

Upstream requirements: RS_Diag_00084

⌈The "Routing activation response code shall be set according to the response conditions specified in [SWS_DoIP_00106], [SWS_DoIP_00104], [SWS_DoIP_00105],

[SWS_DoIP_00107], [SWS_DoIP_00160], [SWS_DoIP_00111], [SWS_DoIP_00114], [SWS_DoIP_00274] and [SWS_DoIP_00113].⌋

[SWS_DoIP_00120]

Upstream requirements: RS_Diag_00084

⌈If the DoIPRoutingActivationAuthenticationResLength and/or DoIPRoutingActivation-ConfirmationResLength are used (different from zero), the "OEM specific" field shall be filled with the specified length data.⌋

Note: see details of the RoutingActivation request and response structures in 7.3.2.3.

7.3.2.4 Alive check

7.3.2.4.1 Alive check request (payload type 0x0007)

[SWS_DoIP_00139]

Upstream requirements: RS_Diag_00083

⌈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.⌋

[SWS_DoIP_00140]

Upstream requirements: RS_Diag_00083

⌈After sending an alive check request the DoIP module shall wait the configured time DoIPAliveCheckResponseTimeout to receive a valid alive check response. If it does not receive an alive check response, the socket connection on which the alive check request was sent shall be reset as described in [SWS_DoIP_00358].⌋

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 7.9:

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

Table 7.9: Alive check response message payload data

[SWS_DoIP_00332] ⌈If an Alive check response is received on a socket connection which is in any ISO 13400 connection state "Registered" and the field "SourceAddress" does not match the registered Source Address of this socket connection, then the socket connection shall be closed as described in [SWS_DoIP_00058].⌋

7.3.2.5 Node information

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

[SWS_DoIP_00090]

Upstream requirements: RS_Diag_00080

⌈When the module receives a DoIP message with payload Type 0x4003 on a connection different than a configured DoIPUdpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00091]

Upstream requirements: RS_Diag_00080

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

[SWS_DoIP_00092]

Upstream requirements: RS_Diag_00080

⌈After a valid Diagnostic power mode request message, the DoIP module shall send a Diagnostic Power mode information response message ([SWS_DoIP_00093]) on the configured DoIPUdpConnection.⌋

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 7.10:

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

Table 7.10: Diagnostic power mode information response message payload data

[SWS_DoIP_00093]

Upstream requirements: RS_Diag_00080

⌈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.⌋

7.3.2.5.3 Diagnostic entity status request (payload type 0x4001)

[SWS_DoIP_00094]

Upstream requirements: RS_Diag_00082

⌈When the module receives a DoIP message with payload Type 0x4001 on a connection different than a configured DoIPUdpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00095]

Upstream requirements: RS_Diag_00082

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

[SWS_DoIP_00096]

Upstream requirements: RS_Diag_00082

⌈After a valid Diagnostic entity status request message, the DoIP module shall send a Diagnostic entity status response message on the configured DoIPUdpConnection.⌋

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 7.11:

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 7.11: Diagnostic entity status response message payload data

[SWS_DoIP_00097]

Upstream requirements: RS_Diag_00082

⌈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.⌋

[SWS_DoIP_00098]

Upstream requirements: RS_Diag_00082

⌈The "Max open sockets" byte of a diagnostic entity status response message shall contain the configured DoIPMaxTesterConnections. This parameter represents the maximum number of concurrent TCP_DATA sockets allowed with this DoIP entity, excluding the reserve socket required for socket handling as defined in the ISO 13400-2 standard.⌋

[SWS_DoIP_00099]

Upstream requirements: RS_Diag_00082

⌈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].⌋

[SWS_DoIP_00100]

Upstream requirements: RS_Diag_00082

⌈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.⌋

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 0x8001)

The payload data structure of a diagnostic message shall be supported as described in Table 7.12:

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

Table 7.12: Diagnostic message payload data

[SWS_DoIP_00121]

Upstream requirements: RS_Diag_00027

⌈When the module receives a DoIP message with payload Type 0x8001 on a connection different than a configured DoIPTcpConnection, the module shall discard the DoIP message.⌋

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

[SWS_DoIP_00122]

Upstream requirements: RS_Diag_00027

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

[SWS_DoIP_00123]

Upstream requirements: RS_Diag_00027

⌈If a diagnostic message on a socket connection arrived, which is in the state Registered [Routing Active], and the "Source Address" does not match, a negative acknowledge set to 0x02 shall be sent. Additionally, the socket connection shall be closed as described in [SWS_DoIP_00058].⌋

Note: see details of the negative acknowledge message in 7.3.2.6.3.

[SWS_DoIP_00124]

Upstream requirements: RS_Diag_00027

⌈If the DoIP module receives a diagnostic message with a "Target Addess" (equals DoIPTargetAddressValue) which is not connected via DoIPRoutingActivationRef and DoIPTargetddressRef 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. Additionally the message shall be discarded.⌋

[SWS_DoIP_00125]

Upstream requirements: RS_Diag_00027

⌈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. Additionally the message shall be discarded.⌋

[SWS_DoIP_00126]

Upstream requirements: RS_Diag_00027

⌈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. Additionally the message shall be discarded.⌋

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

[SWS_DoIP_00127]

Upstream requirements: RS_Diag_00027

⌈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. Additionally the message shall be discarded.⌋

[SWS_DoIP_00331] ⌈Diagnostic messages shall only be evaluated in the ISO 13400 connection state "Registered [Routing Active]". If a DoIP message of payload type "Diagnostic Message" is received on a TCP_DATA socket connection which is not in the ISO 13400 connection state "Registered [Routing Active]", the DoIP message shall be silently discarded as soon as the payload type is evaluated. No further checks shall be performed on followed messages fields. Only the payload length shall be used to identify the start of the next frame.⌋

[SWS_DoIP_00128]

Upstream requirements: RS_Diag_00027

⌈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⌋

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

[SWS_DoIP_00129]

Upstream requirements: RS_Diag_00027

⌈If the PduR accepted all Data, the DoIP module shall send a diagnostic acknowledge message.⌋

[SWS_DoIP_00130]

Upstream requirements: RS_Diag_00027

⌈The DoIP module will get a diagnostic response message (i.e DoIP_TpTransmit or DoIP_IfTransmit is 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.⌋

[SWS_DoIP_00131]

Upstream requirements: RS_Diag_00027

⌈If the DoIP module is called with DoIPPduRTxPdu in the DoIP_TpTransmit or DoIP_IfTransmit 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 "SourceAddress" set to the DoIPTargetAddressValue of the request and the "TargetAddress" set to the DoIPTesterSA.⌋

[SWS_DoIP_00173]

Upstream requirements: RS_Diag_00027

⌈The field "User data" of the [SWS_DoIP_00131] message contains the actual diagnostic payload data which shall not be modified by DoIP.⌋

Note: The reception 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.

[SWS_DoIP_00507] Diagnostic Message Validation Failure Security Event

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈If `DoIPEnableSecurityEventReporting` is enabled and `SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED` exists, the DoIP module shall report the security event SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED to IdsM with the context data specified in [SWS_DoIP_00514] when a diagnostic message is rejected due to a failure in "Diagnostic message checks." These checks include:

- Invalid Source Address
- Unknown Target Address
- Payload length exceeds target network maximum allowed size
- Current available buffer too small
- Target unreachable

⌋

Note: Refer to the DoIP diagnostic message check as described in ISO 13400-2 [7] in the figure "DoIP diagnostic message handler".

[SWS_DoIP_00514] Security event context data definition: SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED ⌈

SEV Name	***SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED***	
ID	130	
Description	A diagnostic message request was rejected by the diagnostic message handler.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
LogicalSourceAddress	uint8 [2]	
LogicalTargetAddress	uint8 [2]	
NACKCode	uint8 [1]	

⌋

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 7.13:

Item	Position (Byte)	Length (Byte)
Logical address information		
Source address	0	2
Target address	2	2
Diagnostic message acknowledge information		
ACK code (0x00)	4	1
Previous diagnostic message	5	...

Table 7.13: Diagnostic acknowledge message payload data

[SWS_DoIP_00132]

Upstream requirements: RS_Diag_00027

⌈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.⌋

[SWS_DoIP_00133]

Upstream requirements: RS_Diag_00027

⌈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.⌋

[SWS_DoIP_00134]

Upstream requirements: RS_Diag_00027

⌈If the DoIP module needs to send a diagnostic acknowledge message the field "previous diag message" shall be filled with the number of bytes of the original request mes-

sage as configured in the parameter DoIPNumByteDiagAckNack for the DoIPTester the request was received on.⌋

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 7.14:

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 7.14: Diagnostic negative acknowledge payload data

[SWS_DoIP_00135]

Upstream requirements: RS_Diag_00027

⌈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.⌋

[SWS_DoIP_00136]

Upstream requirements: RS_Diag_00027

⌈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.⌋

[SWS_DoIP_00137]

Upstream requirements: RS_Diag_00027

⌈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.⌋

[SWS_DoIP_00138]

Upstream requirements: RS_Diag_00027

⌈If the DoIP module needs to send a diagnostic negative acknowledge message the field "previous diag message" shall be filled with the configured number of the original request message as configured in the parameter DoIPNumByteDiagAckNack for the DoIPTester the request was received on.⌋

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 7.2 and sent via UDP as described in Table 7.3 shall be treated as described in this chapter.

[SWS_DoIP_00197] UDP Reception Buffer Copy

Upstream requirements: RS_Diag_00024

⌈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.⌋

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 ISO 13400-2 [7].

[SWS_DoIP_00198] UDP Message Transmission

Upstream requirements: RS_Diag_00024

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

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 ISO 13400-2 [7].

[SWS_DoIP_00199] UDP Transmission Buffer Release ⌈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.⌋

[SWS_DoIP_00286] Vehicle Announcement Success Criteria ⌈DoIP module shall consider the announcement successful and process `DoIPVehicleAnnouncementCount` if the SoAd calls the DoIP module via the interface `DoIP_SoAdIfTxConfirmation()` with Result set to `E_OK` for the announcement related `SoAd_IfTransmit()` call i.e. if `E_NOT_OK` is returned for the last announcement message, it will not be considered an announcement.⌋

[SWS_DoIP_00276] UDP Request Rate Limiting ⌈If DoIP receives more UDP requests on a connection than the configured amount of `DoIPMaxUDPRequestPerConnection`, only `DoIPMaxUDPRequestPerConnection` requests (including the request that has just been accepted) shall be processed and responded. DoIP shall silently discard the request messages that cannot be processed.⌋

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 7.2 and sent via TCP as described in Table 7.3 shall be treated as described in this chapter.

7.5.1 Reception of a TCP DoIP message

[SWS_DoIP_00207] TCP Reception Buffer Initialization

Upstream requirements: RS_Diag_00024

⌈When `DoIP_SoAdTpStartOfReception()` is invoked, 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`.⌋

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] TCP Copy Zero-Length Query

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpCopyRxData()` is called at the start of a new DoIP message (e.g. directly after `DoIP_SoAdTpStartOfReception()` succeeded or previous DoIP message processed completely) with info.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.⌋

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] TCP Copy With Sufficient Buffer

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpCopyRxData()` is called at the start of a new DoIP message (e.g. directly after `DoIP_SoAdTpStartOfReception()` succeeded or previous DoIP message processed completely) with info.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`.⌋

[SWS_DoIP_00210] TCP Copy With Insufficient Buffer

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpCopyRxData()` is called at the start of a new DoIP message (e.g. directly after `DoIP_SoAdTpStartOfReception()` succeeded or previous DoIP message processed completely) with info.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`.⌋

[SWS_DoIP_00214] Non-Diagnostic Message Buffering

Upstream requirements: RS_Diag_00024

⌈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].⌋

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] Diagnostic Message PduR Handoff

Upstream requirements: RS_Diag_00024

⌈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 on this `DoIPInterface`, the DoIP module shall call the `PduR_DoIPTpStartOfReception()` with the according id set to the `DoIPPduRRxPduId` matching the SourceAddress/TargetAddress combination of the diagnostic message on this `DoIPInterface`, 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.⌋

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

[SWS_DoIP_00260] Accepted Diagnostic Data Forwarding

Upstream requirements: RS_Diag_00024

⌈If `PduR_DoIPTpStartOfReception()` 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_DoIPTpCopyRxData()`.⌋

[SWS_DoIP_00218] Diagnostic Data Stream Forwarding

Upstream requirements: RS_Diag_00024

⌈If PduR_DoIPTpStartOfReception() returns BUFREQ_OK the reception was accepted and the DoIP shall forward all subsequent calls to DoIP_SoAdTpCopyRxData() directly to PduR_DoIPTpCopyRxData() until all diagnostic data was handed to the PduR.⌋

[SWS_DoIP_00259] Buffer Size Boundary Adjustment

Upstream requirements: RS_Diag_00024

⌈At the end of the copy procedure via PduR_DoIPTpCopyRxData() 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.⌋

[SWS_DoIP_00253] Insufficient PduR Buffer Abort

Upstream requirements: RS_Diag_00024

⌈If the buffer size reported by PduR_DoIPTpStartOfReception() does not suffice for already received data, DoIP shall abort the reception and call PduR_DoIPTpRxIndication() with E_NOT_OK.⌋

[SWS_DoIP_00216] PduR Rejection NACK 0x08

Upstream requirements: RS_Diag_00024

⌈If PduR_DoIPTpStartOfReception() returns BUFREQ_E_NOT_OK, the DoIP module shall send a diagnostic negative acknowledge message with the diagnostic message negative acknowledge code set to 0x08 and discard all the TCP data until the next DoIP message.⌋

Note: PduR_DoIPTpRxIndication() will not be called when PduR_DoIPTpStartOfReception() does not return BUFREQ_OK.

[SWS_DoIP_00311] PduR Overflow NACK 0x05

Upstream requirements: RS_Diag_00024

⌈If PduR_DoIPTpStartOfReception() returns BUFREQ_E_OVFL, the DoIP module shall send a diagnostic negative acknowledge message with the diagnostic message negative acknowledge code set to 0x05 and discard all the TCP data until the next DoIP message.⌋

[SWS_DoIP_00217] Copy Failure Discard and Indication

Upstream requirements: RS_Diag_00024

⌈If PduR_DoIPTpCopyRxData() returns BUFREQ_E_NOT_OK, the DoIP module shall discard all the TCP data until the next DoIP message and call the PduR_DoIPTpRxIndication() with the according PduId and the result set to E_NOT_OK.⌋

[SWS_DoIP_00221] Successful Reception Indication

Upstream requirements: RS_Diag_00024

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

[SWS_DoIP_00219] Non-Diagnostic TCP Message Processing

Upstream requirements: RS_Diag_00024

⌈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.⌋

Note: The possible DoIP messages on TCP are described in ISO 13400-2 [7] and in the according chapters in this specification.

[SWS_DoIP_00200] TCP Reception Cleanup

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpRxIndication()` is called the DoIP module shall release all data connected to the reception and forward the result to `PduR_DoIPTpRxIndication()` if a reception for diagnostic message is currently ongoing.⌋

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

[SWS_DoIP_00258] Cancel Receive Forwarding ⌈If the DoIP module is called with `DoIP_TpCancelReceive()`, the DoIP module shall call the `SoAd_TpCancelReceive()` function with the RxPduId that is retrieved via the according configured `DoIPSoAdTcpRxPduRef`.⌋

7.5.2 Transmission of a TCP DoIP message

[SWS_DoIP_00220] Non-Diagnostic TCP Transmission

Upstream requirements: RS_Diag_00024

⌈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 TxPduId containing the Id of the according socket, the PduInfoPtr.SduLength set to the size of the data to be transmitted and lock the buffer to send.⌋

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] Transmission Failure Discard

Upstream requirements: RS_Diag_00024

⌈If the call to `SoAd_TpTransmit()` returns `E_NOT_OK` the DoIP module shall discard the DoIP message.⌋

[SWS_DoIP_00224] Copy Zero-Length Query Response

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpCopyTxData()` is called after a successful call to `SoAd_TpTransmit()`, with a valid id and the info.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.⌋

[SWS_DoIP_00225] Copy Data and Update Available Size

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpCopyTxData()` is called after a successful call to `SoAd_TpTransmit()`, with a valid id and the info.SduLength is not set to 0, the DoIP module shall copy the bytes specified in the info.SduLength to the info->SduDataPtr, return `BUFREQ_OK` and set the parameter availableDataPtr to the total available data size of the current DoIP message after the copy process.⌋

[SWS_DoIP_00229] Transmission Confirmation Buffer Release

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_SoAdTpTxConfirmation()` is called the DoIP module shall release the buffer related to the id.⌋

[SWS_DoIP_00230] Diagnostic Transmit Acceptance

Upstream requirements: RS_Diag_00024

⌈If the function `DoIP_TpTransmit()` or `DoIP_IfTransmit()` 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`.

1.) If the connection to the SoAd is idle, the DoIP shall call the `SoAd_TpTransmit()` function according to [SWS_DoIP_00284].

2.) If the connection to the SoAd is not idle, the DoIP shall store the transmission request and call `SoAd_TpTransmit()` according to [SWS_DoIP_00284] as soon as the connection is idle again.⌋

[SWS_DoIP_00284] Diagnostic Message Header Assembly

Upstream requirements: RS_Diag_00024

⌈To transmit a DoIP diagnostic message the DoIP shall assemble the DoIP header considering the information of the handed PduInfoPtr.SduLength and call `SoAd_TpTransmit()` with the TxPduId set to the according PduId of the socket connection and the PduInfoPtr.SduLength set to the sum of the following lengths: DoIP header (8

Byte), the DoIP diagnostic message specific data (4 Byte) and received length of the call to DoIP_TpTransmit() or DoIP_IfTransmit() (PduInfoPtr.SduLength).⌋

[SWS_DoIP_00226] Diagnostic Transmit Rejection

Upstream requirements: RS_Diag_00024

⌈If the function DoIP_TpTransmit() or DoIP_IfTransmit() 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.⌋

[SWS_DoIP_00279] IF PDU Buffer Storage

Upstream requirements: RS_Diag_00024

⌈If the DoIPPduType of a DoIPPduRTxPdu is DOIP_IFPDU, the content of the PDU provided by DoIP_IfTransmit() shall be stored completely in the DoIP internal buffer. If the buffer is too small, E_NOT_OK shall be returned immediately.⌋

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_00231] Diagnostic Copy Zero-Length Query

Upstream requirements: RS_Diag_00024

⌈If the function DoIP_SoAdTpCopyTxData() is called after a successful call to SoAd_TpTransmit() for the use case "diagnostic message", with a valid id and the info.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.⌋

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] Diagnostic Copy Data Forwarding

Upstream requirements: RS_Diag_00024

⌈If the function DoIP_SoAdTpCopyTxData() is called after a successful call to SoAd_TpTransmit() for the use case "diagnostic message" with a valid id and the info.SduLength is not set to 0, the DoIP module shall copy the bytes specified in the info.SduLength to the info->SduDataPtr. If the requested bytes are more than in the DoIP internal buffer, the DoIP shall call the PduR_DoIPTpCopyTxData() with the info.SduLength set to the remaining requested data bytes and the info-> SduDataPtr set to the position where the PduR shall continue to copy the data.⌋

[SWS_DoIP_00254] Copy Success Available Data Update

Upstream requirements: RS_Diag_00024

⌈If the call to PduR_DoIPTpCopyTxData() 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_DoIPTpCopyTxData()`.⌋

[SWS_DoIP_00233] Diagnostic Transmission PduR Passthrough

Upstream requirements: RS_Diag_00024

⌈If the DoIP module has copied via subsequent calls to `DoIP_SoAdTpCopyTxData()` for the use case "diagnostic message" all information stored in the DoIP internal buffer, the DoIP module shall forward all subsequent calls to `DoIP_SoAdTpCopyTxData()`/`DoIP_SoAdTpTxConfirmation()` for this transmission directly to the PduR using `PduR_DoIPTpCopyTxData()`/`PduR_DoIPTpTxConfirmation()` in case the `DoIPPduRTxPdu` is `DOIP_TPPDU` and `PduR_DoIPIfTxConfirmation()` otherwise, and release the internal buffer for this transmission.⌋

[SWS_DoIP_00257] Cancel Transmit Forwarding

Upstream requirements: RS_Diag_00024

⌈If the DoIP module is called with `DoIP_TpCancelTransmit()` or `DoIP_IfCancelTransmit()`, the DoIP module shall call the `SoAd_TpCancelTransmit()` function of the according SoAdTxPduId.⌋

7.6 Security Events

This section describes the handling of security related events in the DoIP module.

[SWS_DoIP_00500] Conditional Security Event Reporting

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈If `DoIPEnableSecurityEventReporting` has been enabled for the DoIP module the respective security events shall be reported to the IdsM via the interfaces defined in AUTOSAR_SWS_BSWGeneral [1].⌋

[SWS_DoIP_00501] Security Event Context Data Population

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈The DoIP module shall populate the context data associated with a reported security event according to the specific condition or failure that triggered the event.⌋

[SWS_DoIP_00502] Retrieve Socket Connection Context Data

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈The DoIP module shall retrieve relevant context data for a security event if needed from the SoAd module by calling `SoAd_GetLocalAddr()` using the SoConId corresponding to the socket connection on which the security event occurred.⌋

[SWS_DoIP_00503] Context Data Retrieval Failure Handling

Status: DRAFT
Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈If the DoIP module fails to obtain any required context data for a security event, it shall invalidate the affected context element by setting all bytes of that element to 0xFF before passing it to IdsM.⌋

The following table lists the security events which are standardized for the DoIP module together with their trigger conditions:

[SWS_DoIP_00510] Security events for DoIP ⌈

Name	*Description*	*ID*
SEV_DOIP_HEADER_CHECK_FAILED	The DoIP Header Handler rejected a request (routing or diagnostic message).	127
SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED	A routing request was rejected by the routing handler.	128
SEV_DOIP_ROUTING_ACTIVATION_SUCCESS	A routing request was successful.	129
SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED	A diagnostic message request was rejected by the diagnostic message handler.	130

⌋

7.7 Error classification

7.7.1 Development Errors

[SWS_DoIP_00148] Definition of development errors in module DoIP ⌈

Type of error	*Related error code*	*Error value*
API service call without module initialization	DOIP_E_UNINIT	0x01
NULL-Pointer on any API call	DOIP_E_PARAM_POINTER	0x02
Wrong Lower Layer (SoAd) or Upper Layer (Pdu Router) Id received	DOIP_E_INVALID_PDU_SDU_ID	0x03
API call with invalid Parameter	DOIP_E_INVALID_PARAMETER	0x04
DoIP Init service call failure	DOIP_E_INIT_FAILED	0x05

⌋

7.7.2 Runtime Errors

There are no runtime errors.

7.7.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

The following types shall be imported by the DoIP module from the modules given:

[SWS_DoIP_00020] Definition of imported datatypes of module DoIP ⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
IdsM	IdsM.h	IdsM_SecurityEventIdType
	Rte_IdsM_Type.h	IdsM_TimestampDataType
SoAd	SoAd.h	SoAd_SoConIdType
	SoAd.h	SoAd_SoConModeType
Std	Std_Types.h	Std_ReturnType
TcpIp	TcpIp.h	TcpIp_DomainType
	TcpIp.h	TcpIp_IpAddrAssignmentType
	TcpIp.h	TcpIp_IpAddrStateType
	TcpIp.h	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] Definition of ImplementationDataType DoIP_PowerStateType ⌈

Name	DoIP_PowerStateType		
Kind	Type		
Derived from	uint8		
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
Description	Used for handling of the PowerMode in DoIP entity status requests		
Variation	–		
Available via	Rte_DoIP_Type.h		

⌋

[SWS_DoIP_00267] Definition of ImplementationDataType AuthenticationReq DataType_{Name} ⌈

Name	AuthenticationReqDataType_{Name}		
Kind	Array	***Element type***	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRoutingActivation AuthenticationCallback.DoIPRoutingActivationAuthenticationReqLength)} Elements		
Description	–		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation.SHORT-NAME)}		
Available via	Rte_DoIP_Type.h		

⌋

[SWS_DoIP_00268] Definition of ImplementationDataType AuthenticationRes DataType_{Name} ⌈

Name	AuthenticationResDataType_{Name}		
Kind	Array	***Element type***	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/DoIPRoutingActivationAuthentication Callback.DoIPRoutingActivationAuthenticationResLength)} Elements		
Description	–		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		
Available via	Rte_DoIP_Type.h		

⌋

[SWS_DoIP_00269] Definition of ImplementationDataType ConfirmationReqData Type_{Name} ⌈

Name	ConfirmationReqDataType_{Name}		
Kind	Array	***Element type***	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/DoIPRoutingActivationConfirmationCallback.Do IPRoutingActivationConfirmationReqLength)} Elements		
Description	–		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		
Available via	Rte_DoIP_Type.h		

⌋

[SWS_DoIP_00270] Definition of ImplementationDataType ConfirmationResData Type_{Name} ⌈

Name	ConfirmationResDataType_{Name}		
Kind	Array	***Element type***	uint8
Size	{ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation/DoIPRoutingActivationConfirmationCallback.Do IPRoutingActivationConfirmationResLength)} Elements		
Description	–		
Variation	Name = {ecuc(DoIP/DoIPConfigSet/DoIPRoutingActivation.SHORT-NAME)}		
Available via	Rte_DoIP_Type.h		

⌋

[SWS_DoIP_00287] Definition of ImplementationDataType DoIP_FurtherAction ByteType

Upstream requirements: RS_Diag_00026

⌈

Name	DoIP_FurtherActionByteType		
Kind	Type		
Derived from	uint8		
Range	0x11..0xFF	–	Available for additional OEM-specific use
Description	Used to get the OEM specific Further Action Byte for the DoIP vehicle identification response/ vehicle announcement.		
Variation	–		
Available via	Rte_DoIP_Type.h		

⌋

8.2 Type definitions

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

8.2.1 DoIP_ReturnType

[SWS_DoIP_91004] Definition of Std_ReturnType-extension for module DoIP ⌈

Range	DOIP_E_PENDING	0x10	Operation still running; call next DoIP_MainFunction cycle again
Description	Overlayed return values of Std_ReturnType for DoIP APIs.		
Available via	DoIP.h		

⌋

8.2.2 DoIP_ConfigType

[SWS_DoIP_00025] Definition of datatype DoIP_ConfigType ⌈

Name	DoIP_ConfigType	
Kind	Structure	
Elements	Implementation specific	
	Type	–
	Comment	The content of the configuration data structure is implementation specific
Description	Configuration data structure of the DoIP module	
Available via	DoIP.h	

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 DoIP_TpTransmit

[SWS_DoIP_00022] Definition of API function DoIP_TpTransmit

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_TpTransmit	
Syntax	`Std_ReturnType DoIP_TpTransmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x53	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `E_NOT_OK`: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00162] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TpTransmit()` 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_00163] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TpTransmit()` shall check if the `TxPduId` matches a configured `DoIPPduRTxPduId`. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.⌋

[SWS_DoIP_00164] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TpTransmit()` shall check if the `PduInfoPtr` is not a `NULL_PTR`. If the check fails, the function shall raise the development error `DOIP_E_PARAM_POINTER`.⌋

8.3.2 DoIP_TpCancelTransmit

[SWS_DoIP_00023] Definition of API function DoIP_TpCancelTransmit

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_TpCancelTransmit	
Syntax	`Std_ReturnType DoIP_TpCancelTransmit ( PduIdType TxPduId )`	
Service ID [hex]	0x54	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	–
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Cancellation was executed successfully by the destination module. `E_NOT_OK`: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module. The call of this API to cancel an ongoing transmission will close the used TCP connection.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00166] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TpCancelTransmit()` 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_00167] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TpCancelTransmit()` shall check if the `TxPduId` matches a configured `DoIPPduRTxPduId`. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.⌋

8.3.3 DoIP_TpCancelReceive

[SWS_DoIP_00024] Definition of API function DoIP_TpCancelReceive

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_TpCancelReceive
Syntax	`Std_ReturnType DoIP_TpCancelReceive ( PduIdType RxPduId )`
Service ID [hex]	0x4c

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	RxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing reception of a PDU in a lower layer transport protocol module. The call of this API to cancel an ongoing reception will close the used TCP connection.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00169] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_TpCancelReceive() 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_00170] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_TpCancelReceive() shall check if the RxPduId matches a configured DoIPPduRRxPduId. If the check fails, the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID.⌋

8.3.4 DoIP_IfTransmit

[SWS_DoIP_00277] Definition of API function DoIP_IfTransmit

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_IfTransmit	
Syntax	`Std_ReturnType DoIP_IfTransmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x49	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	DoIP.h	

⌋

8.3.5 DoIP_IfCancelTransmit

[SWS_DoIP_00278] Definition of API function DoIP_IfCancelTransmit

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_IfCancelTransmit	
Syntax	`Std_ReturnType DoIP_IfCancelTransmit ( PduIdType TxPduId )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Cancellation was executed successfully by the destination module. `E_NOT_OK`: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	
Available via	DoIP.h	

⌋

8.3.6 DoIP_Init

[SWS_DoIP_00026] Definition of API function DoIP_Init

Upstream requirements: RS_Diag_00024

⌈

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 Do IP module is operational.	
Available via	DoIP.h	

⌋

8.3.7 DoIP_ActivationLineSwitch

[SWS_DoIP_91000] Definition of API function DoIP_ActivationLineSwitch ⌈

Service Name	DoIP_ActivationLineSwitch	
Syntax	`void DoIP_ActivationLineSwitch ( uint8 InterfaceId, boolean* Active )`	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	InterfaceId	Indentifier of the DoIP interface for which DoIP_ActivationLine Switch function is called.
Parameters (inout)	Active	Boolean value acting as input parameter to request active/ inactive status of the given DoIP Interface and acts as an output parameter indicating the activation line status.
Parameters (out)	None	
Return value	None	
Description	This function is to be used by integrators to inform the DoIP implementation about the status of the activation line of a DoIP interface with given InterfaceId.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00285] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_ActivationLineSwitch()` 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_00302] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_ActivationLineSwitch()` shall check if interface identified by `InterfaceId` actually exists and `DoIPInterfaceActLineCtrl` is set to TRUE. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

[SWS_DoIP_00303] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_ActivationLineSwitch()` shall check if the interface identified by `InterfaceId` actually exists. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

8.3.8 DoIP_TriggerVehicleAnnouncement

[SWS_DoIP_91002] Definition of API function DoIP_TriggerVehicleAnnouncement ⌈

Service Name	DoIP_TriggerVehicleAnnouncement
Syntax	`void DoIP_TriggerVehicleAnnouncement ( uint8 InterfaceId )`

▽

△

Service ID [hex]	0x0d	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	InterfaceId	Indentifier of the DoIP interface for which DoIP_TriggerVehicle Announcement is called.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is used to notify the DoIP module to start vehicle announcement for DoIP interfaces with given InterfaceId. This function is just a trigger to start the DoIPInitialVehicle AnnouncementTime timeout.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00304] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TriggerVehicleAnnouncement()` shall check if the interface identified by `InterfaceId` is configured with `DoIPInterfaceActLineCtrl` set to FALSE. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

[SWS_DoIP_00305] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_TriggerVehicleAnnouncement()` shall check if the interface identified by `InterfaceId` actually exists. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

8.3.9 DoIP_AllowNonTlsDoIPConnection

[SWS_DoIP_91003] Definition of API function DoIP_AllowNonTlsDoIPConnection
⌈

Service Name	DoIP_AllowNonTlsDoIPConnection	
Syntax	`void DoIP_AllowNonTlsDoIPConnection ( boolean NonTlsAllowed )`	
Service ID [hex]	0x55	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	NonTlsAllowed	NonTlsAllowed: Requested behavior of DoIP
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Call with argument NonTlsAllowed==TRUE to allow unsecured DoIP connection establishments. If called with NonTlsAllowed==FALSE, DoIP will close the unsecured connection with NRC 0x07 and require the tester to use TLS.	
Available via	DoIP.h	

⌋

8.3.10 DoIP_SoAdTpCopyTxData

[SWS_DoIP_00031] Definition of callback function DoIP_SoAdTpCopyTxData

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_SoAdTpCopyTxData	

Syntax	`BufReq_ReturnType DoIP_SoAdTpCopyTxData (`
	`  PduIdType id,`
	`  const PduInfoType* info,`
	`  const RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	`)`	

Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	id	Identification of the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the Sdu DataPtr may be a NULL_PTR.
	retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout)	None	
Parameters (out)	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
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, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. `BUFREQ_E_NOT_OK`: Data has not been copied. Request failed.
Description	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.	

▽

△

Available via	DoIP.h

⌋

[SWS_DoIP_00175] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpCopyTxData()` 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_00176] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpCopyTxData()` shall check if the `id` matches a configured `DoIPSoAdTcpTxPduId`. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.⌋

[SWS_DoIP_00177] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpCopyTxData()` shall check that neither the `info` nor the `availableDataPtr` are a `NULL_PTR`. If the check fails, the function shall raise the development error `DOIP_E_PARAM_POINTER`.⌋

[SWS_DoIP_00178] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpCopyTxData()` shall check if the `retry` is a `NULL_PTR`. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

8.3.11 DoIP_SoAdTpTxConfirmation

[SWS_DoIP_00032] Definition of callback function DoIP_SoAdTpTxConfirmation

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_SoAdTpTxConfirmation	
Syntax	`void DoIP_SoAdTpTxConfirmation ( PduIdType id, Std_ReturnType result )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	id	Identification of the transmitted I-PDU.
	result	`E_OK`: The PDU was transmitted successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00180] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdTpTxConfirmation() 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 DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdTpTxConfirmation() shall check if the id matches a configured DoIPSoAdTcpTxPduId. If the check fails, the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID.⌋

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

8.3.12 DoIP_SoAdTpCopyRxData

[SWS_DoIP_00033] Definition of callback function DoIP_SoAdTpCopyRxData

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_SoAdTpCopyRxData	
Syntax	`BufReq_ReturnType DoIP_SoAdTpCopyRxData ( PduIdType id, const PduInfoType* info, PduLengthType* bufferSizePtr )`	
Service ID [hex]	0x44	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	id	Identification of the received I-PDU.
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied.
Return value	BufReq_ReturnType	BUFREQ_OK: Data copied successfully BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Description	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 buffer is written to the position indicated by bufferSizePtr.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00183] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdTpCopyRxData() 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_00036] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdTpCopyRxData() shall check if the id matches a configured DoIPSoAdTcpRxPduId. If the check fails, the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID.⌋

[SWS_DoIP_00184] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdTpCopyRxData() shall check that neither the info nor the bufferSizePtr are a NULL_PTR. If the check fails, the function shall raise the development error DOIP_E_PARAM_POINTER.⌋

8.3.13 DoIP_SoAdTpStartOfReception

[SWS_DoIP_00037] Definition of callback function DoIP_SoAdTpStartOfReception

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_SoAdTpStartOfReception	
Syntax	`BufReq_ReturnType DoIP_SoAdTpStartOfReception ( PduIdType id, const PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr )`	
Service ID [hex]	0x46	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	id	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, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. BUFREQ_E_OVFL: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	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). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.	

▽

△

Available via	DoIP.h

⌋

[SWS_DoIP_00186] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpStartOfReception()` 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_00187] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpStartOfReception()` shall check if the `id` matches a configured `DoIPSoAdTcpRxPduId`. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.⌋

[SWS_DoIP_00188] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpStartOfReception()` 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`.⌋

[SWS_DoIP_00189] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpStartOfReception()` shall check if the `TpSduLength` is not 0. If `TpSduLength` is not 0, the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

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.

8.3.14 DoIP_SoAdTpRxIndication

[SWS_DoIP_00038] Definition of callback function DoIP_SoAdTpRxIndication

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_SoAdTpRxIndication	
Syntax	`void DoIP_SoAdTpRxIndication ( PduIdType id, Std_ReturnType result )`	
Service ID [hex]	0x45	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	id	Identification of the received I-PDU.
	result	`E_OK`: The PDU was received successfully. `E_NOT_OK`: Reception of the PDU failed.
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	Called after an I-PDU has been received via the TP API, the result indicates whether the reception was successful or not.
Available via	DoIP.h

⌋

[SWS_DoIP_00190] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpRxIndication()` 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 `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoAdTpRxIndication()` shall check if the `id` matches a configured `DoIPSoAdTcpRxPduId`. If the check fails, the function shall raise the development error `DOIP_E_INVALID_PDU_SDU_ID`.⌋

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

8.3.15 DoIP_SoAdIfRxIndication

[SWS_DoIP_00244] Definition of callback function DoIP_SoAdIfRxIndication

Upstream requirements: RS_Diag_00024

⌈

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 PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication of a received PDU from a lower layer communication interface module.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00246] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdIfRxIndication() 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 DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdIfRxIndication() shall check if the RxPduId matches a configured DoIPSoAdUdpRxPduId. If the check fails, the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID.⌋

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

8.3.16 DoIP_SoAdIfTxConfirmation

[SWS_DoIP_00245] Definition of callback function DoIP_SoAdIfTxConfirmation

Upstream requirements: RS_Diag_00024

⌈

Service Name	DoIP_SoAdIfTxConfirmation	
Syntax	`void DoIP_SoAdIfTxConfirmation ( PduIdType TxPduId, Std_ReturnType result )`	
Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted successfully. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00249] ⌈If DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdIfTxConfirmation() 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 DoIPDevelopmentErrorDetect is enabled: DoIP_SoAdIfTxConfirmation() shall check if the TxPduId matches a configured DoIPSoAdUdpTxPduId. If the check fails, the function shall raise the development error DOIP_E_INVALID_PDU_SDU_ID.⌋

8.3.17 DoIP_SoConModeChg

[SWS_DoIP_00039] Definition of callback function DoIP_SoConModeChg

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈

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 socket connection mode.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notification about a SoAd socket connection mode change, e.g. socket connection gets online.	
Available via	DoIP.h	

⌋

[SWS_DoIP_00193] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoConModeChg()` 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 `DoIPDevelopmentErrorDetect` is enabled: `DoIP_SoConModeChg()` shall check if the `SoConId` and `Mode` are valid. If the check fails the function shall raise the development error `DOIP_E_INVALID_PARAMETER`.⌋

8.3.18 DoIP_LocalIpAddrAssignmentChg

[SWS_DoIP_00040] Definition of callback function DoIP_LocalIpAddrAssignmentChg

Upstream requirements: RS_Diag_00081, RS_Diag_00028

⌈

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).	
Available via	DoIP.h	

⌋

[SWS_DoIP_00195] ⌈If `DoIPDevelopmentErrorDetect` is enabled: `DoIP_LocalIpAddrAssignmentChg()` 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 `DoIPDevelopmentErrorDetect` is enabled: `DoIP_LocalIpAddrAssignmentChg()` 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 Scheduled functions

The Basic Software Scheduler within the Rte [8] directly calls these functions. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.4.1 DoIP_MainFunction

[SWS_DoIP_00041] Definition of scheduled function DoIP_MainFunction ⌈

Service Name	DoIP_MainFunction
Syntax	`void DoIP_MainFunction ( void )`
Service ID [hex]	0x02
Description	Schedules the Diagnostic over IP module. (Entry point for scheduling)
Available via	SchM_DoIP.h

⌋

[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.⌋

8.5 Expected Interfaces

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

8.5.1 Mandatory Interfaces

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

[SWS_DoIP_00044] Definition of mandatory interfaces required by module DoIP ⌈

API Function	*Header File*	*Description*
Dcm_GetVin	Dcm.h	Function to get the VIN (as defined in SAE J1979-DA)
PduR_DoIPTpCopyRxData	PduR_DoIPTp.h	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 buffer is written to the position indicated by bufferSizePtr.
PduR_DoIPTpCopyTxData	PduR_DoIPTp.h	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.
PduR_DoIPTpRxIndication	PduR_DoIPTp.h	Called after an I-PDU has been received via the TP API, the result indicates whether the reception was successful or not.
PduR_DoIPTpStartOfReception	PduR_DoIPTp.h	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). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.
PduR_DoIPTpTxConfirmation	PduR_DoIPTp.h	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.
SoAd_CloseSoCon	SoAd.h	This service closes the socket connection specified by SoConId.
SoAd_GetPhysAddr	SoAd.h	Retrieves the physical source address of the EthIf controller used by the SoAd socket connection specified by SoConId.
SoAd_GetSoConId	SoAd.h	Returns socket connection index related to the specified TxPduId.

▽

△

API Function	Header File	Description
SoAd_IfTransmit	SoAd.h	Requests transmission of a PDU.
SoAd_OpenSoCon	SoAd.h	This service opens the socket connection specified by SoConId.
SoAd_ReadDhcpHostNameOption	SoAd.h	By this API service an upper layer of the SoAd can read the currently configured hostname, i.e. FQDN option in the DHCP submodule of the TCP/IP stack.
SoAd_ReleaseIpAddrAssignment	SoAd.h	By this API service the local IP address assignment used for the socket connection specified by SoConId is released.
SoAd_RequestIpAddrAssignment	SoAd.h	By this API service the local IP address assignment which shall be used for the socket connection specified by SoConId is initiated.
SoAd_TpTransmit	SoAd.h	Requests transmission of a PDU.
SoAd_WriteDhcpHostNameOption	SoAd.h	By this API service an upper layer of the SoAd can set the hostname, i.e. FQDN option in the DHCP submodule of the TCP/IP stack.

⌋

8.5.2 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] Definition of optional interfaces requested by module DoIP ⌈

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.
IdsM_ReportSecurityEvent	IdsM.h	This API is the interface to report security events to the IdsM.
PduR_DoIPIfTxConfirmation	PduR_DoIPIf.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.

⌋

Note: The PduR_DoIPIfTxConfirmation optional interface is needed only if the DoIPPduType is set to DOIP_IFPDU for at least one Tx PDU, which is the case when UUDT frames are sent via Ethernet

8.5.3 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.

8.5.3.1 <User>_DoIPGetPowerModeCallback

[SWS_DoIP_00047] Definition of configurable interface <User>_DoIPGetPowerModeCallback

Upstream requirements: RS_Diag_00080

⌈

Service Name	<User>_DoIPGetPowerModeCallback	

Syntax	`Std_ReturnType <User>_DoIPGetPowerModeCallback (`
	`  DoIP_PowerStateType* PowerStateReady`
	`)`	

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. The <User> part of the function name is configurable via parameter DoIPPowerModeDirect ([ECUC_DoIP_00027]).	
Available via	DoIP_Externals.h	

⌋

8.5.3.2 <User>_DoIPRoutingActivationConfirmation

[SWS_DoIP_00048] Definition of configurable interface <User>_DoIPRoutingActivationConfirmation

Upstream requirements: RS_Diag_00084

⌈

Service Name	<User>_DoIPRoutingActivationConfirmation	

Syntax	`Std_ReturnType <User>_DoIPRoutingActivationConfirmation (`
	`  boolean* Confirmed,`
	`  const uint8* ConfirmationReqData,`
	`  uint8* ConfirmationResData`
	`)`	

Sync/Async	Synchronous	
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.	
Available via	DoIP_Externals.h	

⌋

8.5.3.3 <User>_DoIPRoutingActivationAuthentication

[SWS_DoIP_00049] Definition of configurable interface <User>_DoIPRoutingActivationAuthentication

Upstream requirements: RS_Diag_00084

⌈

Service Name	<User>_DoIPRoutingActivationAuthentication	

Syntax	`Std_ReturnType <User>_DoIPRoutingActivationAuthentication (`
	`  boolean* Authentified,`
	`  const uint8* AuthenticationReqData,`
	`  uint8* AuthenticationResData`
	`)`	

Sync/Async	Synchronous	
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.	
Available via	DoIP_Externals.h	

⌋

8.5.3.4 <User>_DoIPTriggerGidSyncCallback

[SWS_DoIP_00050] Definition of configurable interface <User>_DoIPTriggerGidSyncCallback

Upstream requirements: RS_Diag_00026

⌈

Service Name	<User>_DoIPTriggerGidSyncCallback	
Syntax	`Std_ReturnType <User>_DoIPTriggerGidSyncCallback ( void )`	
Sync/Async	Synchronous	
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 DoIPTrigger GidSyncCallback is configured to trigger the synchronization process of the GroupIdentifier. The <User> part of the function name is configurable via parameter DoIPTriggerGidSyncDirect ([ECUC_DoIP_00029]).	
Available via	DoIP_Externals.h	

⌋

8.5.3.5 <User>_DoIPGetGidCallback

[SWS_DoIP_00051] Definition of configurable interface <User>_DoIPGetGidCallback

Upstream requirements: RS_Diag_00026

⌈

Service Name	<User>_DoIPGetGidCallback	
Syntax	`Std_ReturnType <User>_DoIPGetGidCallback ( uint8* GroupId )`	
Sync/Async	Synchronous	
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 DoIPGetGidCallback 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. The <User> part of the function name is configurable via parameter DoIPGetGidDirect ([ECUC_DoIP_00028]).
Available via	DoIP_Externals.h

⌋

8.5.3.6 <User>_DoIPGetFurtherActionByteCallback

[SWS_DoIP_00288] Definition of configurable interface <User>_DoIPGetFurther ActionByteCallback

Upstream requirements: RS_Diag_00026

⌈

Service Name	<User>_DoIPGetFurtherActionByteCallback	
Syntax	`Std_ReturnType <User>_DoIPGetFurtherActionByteCallback ( DoIP_FurtherActionByteType* FurtherActionByte )`	
Sync/Async	Synchronous	
Reentrancy	Don't care	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	FurtherActionByte	Pointer containing the information of the FurtherActionByte. Only valid if the return value equals E_OK.
Return value	Std_ReturnType	`E_OK`: FurtherActionByte contains valid information `E_NOT_OK`: FurtherActionByte contains no valid information
Description	Callback function to get the OEM specific Further Action Byte for the DoIP vehicle identification response/vehicle announcement.	
Available via	DoIP_Externals.h	

⌋

8.5.4 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.⌋

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.

[SWS_DoIP_00054] Definition of ClientServerInterface CallbackGetPowerMode

Upstream requirements: RS_Diag_00080

⌈

Name	CallbackGetPowerMode		
Comment	–		
IsService	true		
Variation	{ecuc(DoIP/DoIPGeneral/DoIPPowerModeCallback/DoIPPowerModeDirect)} == NULL		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetPowerMode	
Comment	–	
Relates to	–	
Variation	–	
Parameters	PowerStateReady	
	Type	DoIP_PowerStateType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

The DoIP Service Component shall be equipped with a service port as described below to request the value of the Power mode for DoIP diagnostic power mode handling.

[SWS_DoIP_00261] Definition of Port CBGetPowerMode required by module Do IP

Upstream requirements: RS_Diag_00080

⌈

Name	CBGetPowerMode		
Kind	RequiredPort	***Interface***	CallbackGetPowerMode
Description	–		
Variation	{ecuc(DoIP/DoIPGeneral/DoIPPowerModeCallback/DoIPPowerModeDirect)} == NULL		

⌋

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

[SWS_DoIP_00055] Definition of ClientServerInterface {Name}_RoutingActivation

Upstream requirements: RS_Diag_00084

⌈

Name	{Name}_RoutingActivation		
Comment	–		
IsService	true		
Variation	(({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRoutingActivation AuthenticationCallback)} != null) && ({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation/DoIPRoutingActivationAuthenticationCallback/DoIPRoutingActivationAuthentication Func)} == "")) || (({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/Do IPRoutingActivationConfirmationCallback)} != null) && ({ecuc(DoIP/DoIPConfigSet/Do IPInterface/DoIPRoutingActivation/DoIPRoutingActivationConfirmationCallback/DoIPRouting ActivationConfirmationFunc)} == "")) Name = {ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	16	DOIP_E_PENDING	RoutingActivation still pending.

Operation	RoutingActivationAuthentication	
Comment	–	
Relates to	–	
Variation	(({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRoutingActivation AuthenticationCallback)} != NULL) && ({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation/DoIPRoutingActivationAuthenticationCallback/DoIPRoutingActivationAuthentication Func)} ==NULL))	
Parameters	Authentified	
	Type	boolean
	Direction	OUT
	Comment	–
	Variation	–
	AuthenticationReqData	
	Type	AuthenticationReqDataType_{Name}
	Direction	IN
	Comment	–
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRouting ActivationAuthenticationCallback.DoIPRoutingActivationAuthenticationReq Length)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation.SHORT-NAME)}
	AuthenticationResData	
	Type	AuthenticationResDataType_{Name}
	Direction	OUT
	Comment	–
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRouting ActivationAuthenticationCallback.DoIPRoutingActivationAuthenticationRes Length)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation.SHORT-NAME)}
Possible Errors	E_OK E_NOT_OK DOIP_E_PENDING	

Operation	RoutingActivationConfirmation
Comment	–
Relates to	–

▽

△

Variation	(({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRoutingActivation ConfirmationCallback)} != NULL) && ({ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation/DoIPRoutingActivationConfirmationCallback/DoIPRoutingActivationConfirmation Func)} ==NULL))	
Parameters	Confirmed	
	Type	boolean
	Direction	OUT
	Comment	–
	Variation	–
	ConfirmedReqData	
	Type	ConfirmationReqDataType_{Name}
	Direction	IN
	Comment	–
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRouting ActivationConfirmationCallback.DoIPRoutingActivationConfirmationReq Length)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation.SHORT-NAME)}
	ConfirmedResData	
	Type	ConfirmationResDataType_{Name}
	Direction	OUT
	Comment	–
	Variation	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRoutingActivation/DoIPRouting ActivationConfirmationCallback.DoIPRoutingActivationConfirmationRes Length)} > 0 Name = {ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPRouting Activation.SHORT-NAME)}
Possible Errors	E_OK E_NOT_OK DOIP_E_PENDING	

⌋

The DoIP Service Component shall be equipped with a service port as described below for each DoIPRoutingActivation that has at least DoIPRoutingActivationConfirmationCallback or DoIPRoutingActivationAuthenticationCallback configured without direct Callback functions.

[SWS_DoIP_00262] Definition of Port CB{Name}RoutingActivation required by module DoIP

Upstream requirements: RS_Diag_00084

⌈

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

⌋

The DoIP Service Component shall provide the service port interface CallbackTriggerGIDSyncronization as described below if the container DoIPTriggerGIDSyncCallback is configured without direct Callback function.

[SWS_DoIP_00056] Definition of ClientServerInterface CallbackTriggerGIDSynchronization

Upstream requirements: RS_Diag_00026

⌈

Name	CallbackTriggerGIDSynchronization		
Comment	–		
IsService	true		
Variation	({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSyncCallback)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSyncCallback/DoIPTriggerGidSyncDirect)} == NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPVinGidMaster)} == TRUE)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	TriggerGIDSynchronization
Comment	–
Relates to	–
Variation	–
Possible Errors	E_OK E_NOT_OK

⌋

The DoIP Service Component shall be equipped with a service port as described below if the container DoIPTriggerGIDSyncCallback is configured without direct Callback function.

[SWS_DoIP_00263] Definition of Port CBTriggerGIDSynchronization required by module DoIP

Upstream requirements: RS_Diag_00026

⌈

Name	CBTriggerGIDSynchronization		
Kind	RequiredPort	***Interface***	CallbackTriggerGIDSynchronization
Description	–		
Variation	({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSyncCallback)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPTriggerGidSyncCallback/DoIPTriggerGidSyncDirect)} == NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPVinGidMaster)} == TRUE)		

⌋

The DoIP Service Component shall provide the service port interface CallbackGetGID as described below to request the GID if the container DoIPGetGidCallback is configured without direct Callback function.

[SWS_DoIP_00057] Definition of ClientServerInterface CallbackGetGID

Upstream requirements: RS_Diag_00026

⌈

Name	CallbackGetGID		
Comment	–		
IsService	true		
Variation	({ecuc(DoIP/DoIPGeneral/DoIPGetGidCallback)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPGetGidCallback/DoIPGetGidDirect)} == NULL)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetGID	
Comment	–	
Relates to	–	
Variation	–	
Parameters	Data	
	Type	uint8
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

The DoIP Service Component shall provide the service port as described below to request the GID if the container DoIPGetGidCallback is configured without direct Callback function

[SWS_DoIP_00264] Definition of Port CBGetGID required by module DoIP

Upstream requirements: RS_Diag_00026

⌈

Name	CBGetGID		
Kind	RequiredPort	***Interface***	CallbackGetGID
Description	–		
Variation	({ecuc(DoIP/DoIPGeneral/DoIPGetGidCallback)} != NULL) && ({ecuc(DoIP/DoIPGeneral/DoIPGetGidCallback/DoIPGetGidDirect)} == NULL)		

⌋

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

The DoIP Service Component shall provide the interface CallbackGetFurtherActionByte as described below to request the value of the OEM specific Further Action Byte for the DoIP vehicle identification response/vehicle announcement.

[SWS_DoIP_00290] Definition of ClientServerInterface CallbackGetFurtherActionByte

Upstream requirements: RS_Diag_00026

⌈

Name	CallbackGetFurtherActionByte		
Comment	–		
IsService	true		
Variation	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPFurtherActionByteCallback/DoIPFurtherActionByteDirect)} == NULL		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetFurtherActionByte	
Comment	–	
Relates to	<User>_DoIPGetFurtherActionByteCallback	
Variation	–	
Parameters	FurtherActionByte	
	Type	DoIP_FurtherActionByteType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

The DoIP Service Component shall be equipped with a service port per DoIPInterface as described below to request the value of the Further Action Byte for DoIP diagnostic vehicle identification response/vehicle announcement.

[SWS_DoIP_00289] Definition of Port CBGetfurtherActionByte*_{DoIPInterface_short_name}* required by module DoIP

Upstream requirements: RS_Diag_00026

⌈

Name	CBGetfurtherActionByte*_{DoIPInterface_short_name}*		
Kind	RequiredPort	***Interface***	CallbackGetFurtherActionByte
Description	–		
Variation	{ecuc(DoIP/DoIPConfigSet/DoIPInterface/DoIPFurtherActionByteCallback/DoIPFurtherActionByteDirect)} == NULL		

⌋

9 Sequence diagrams

9.1 UDP DoIP communication

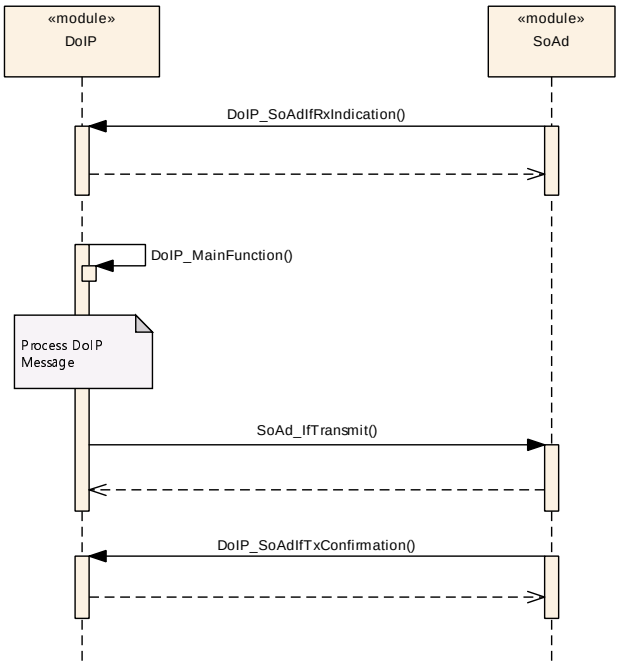

Figure 9.1: DoIP UDP communication

9.2 Rx TCP message

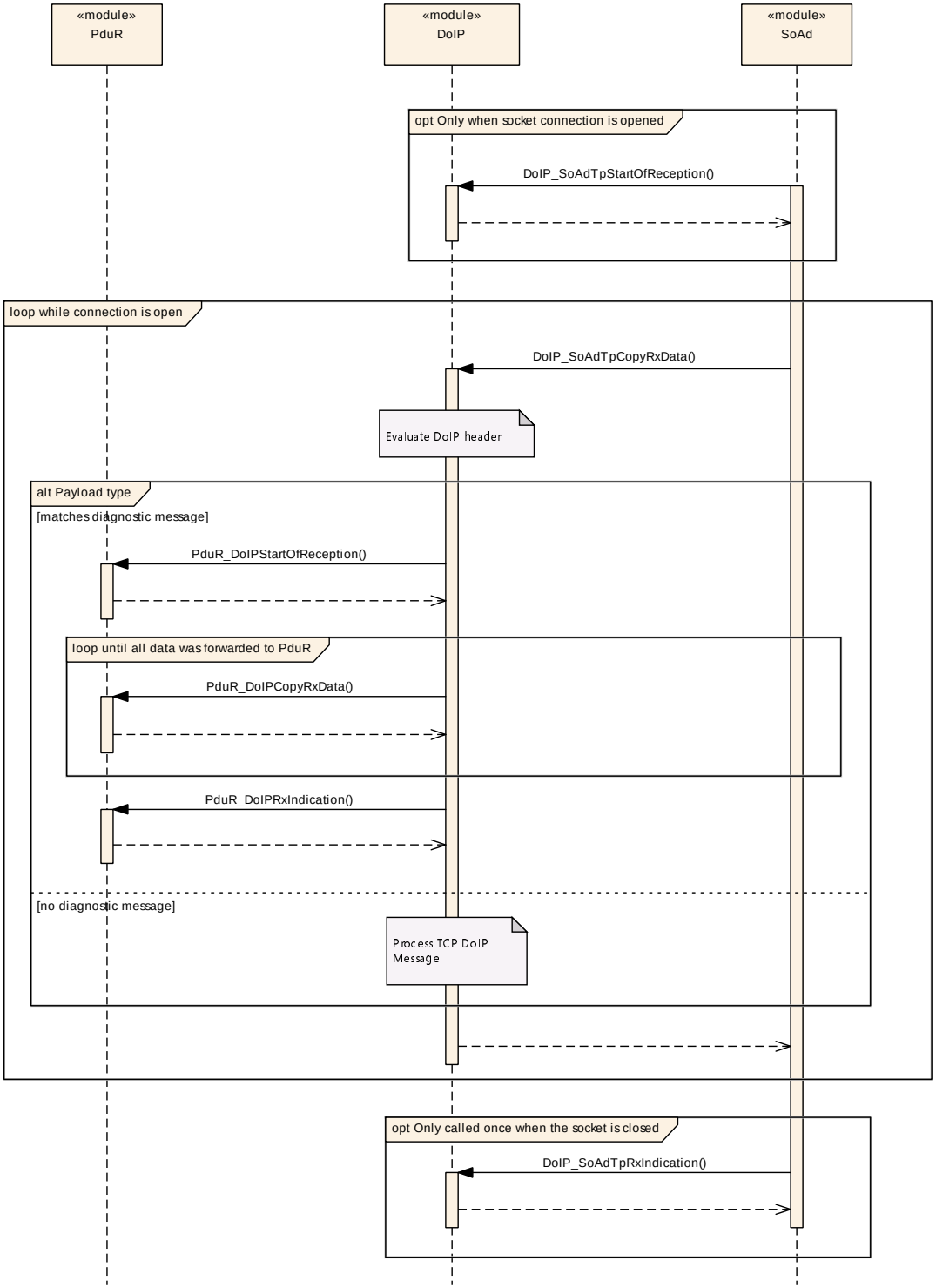

Figure 9.2: DoIP TCP message reception

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

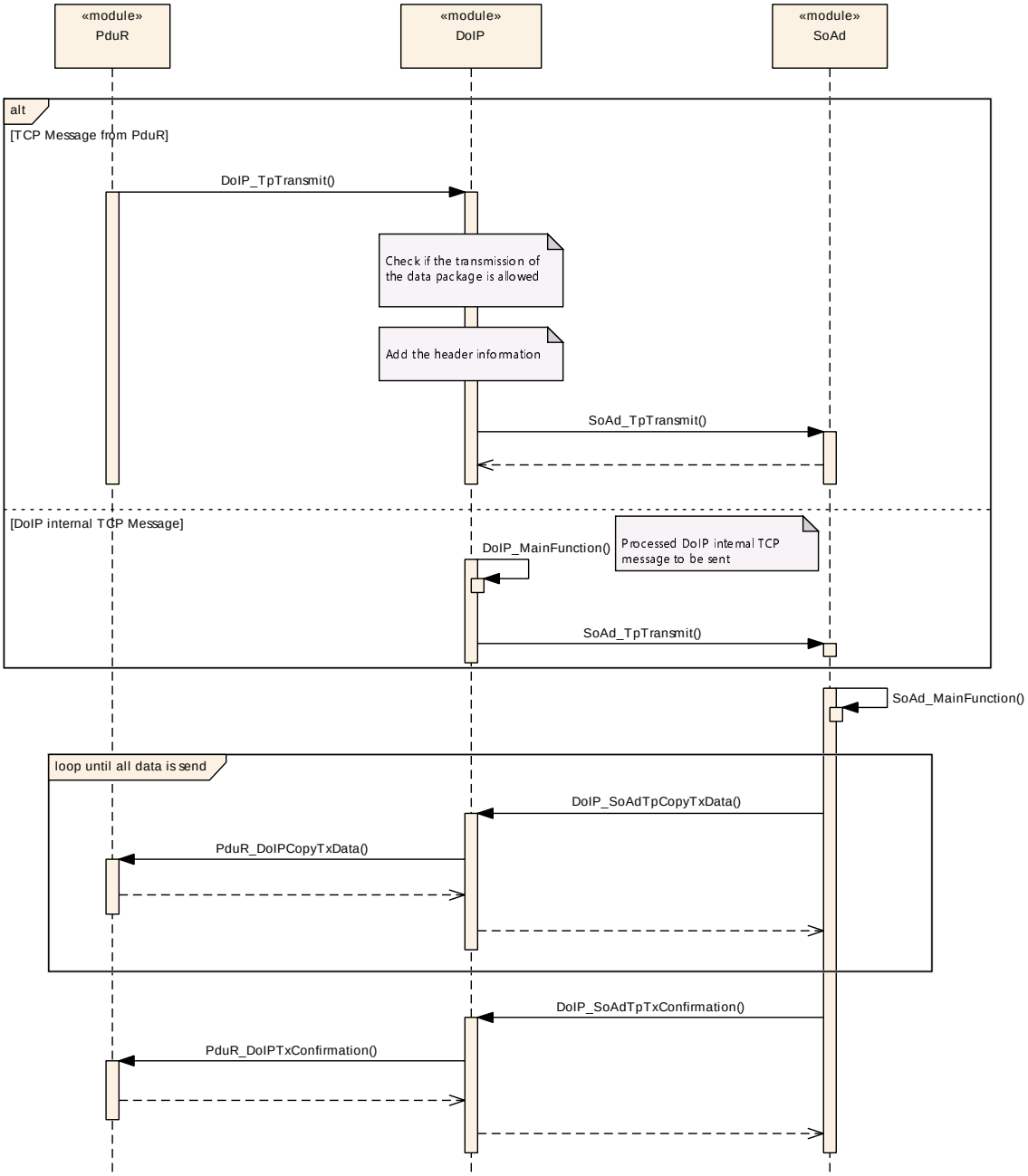

Figure 9.3: DoIP TCP message transmission

9.4 Activation Line Handling - Active

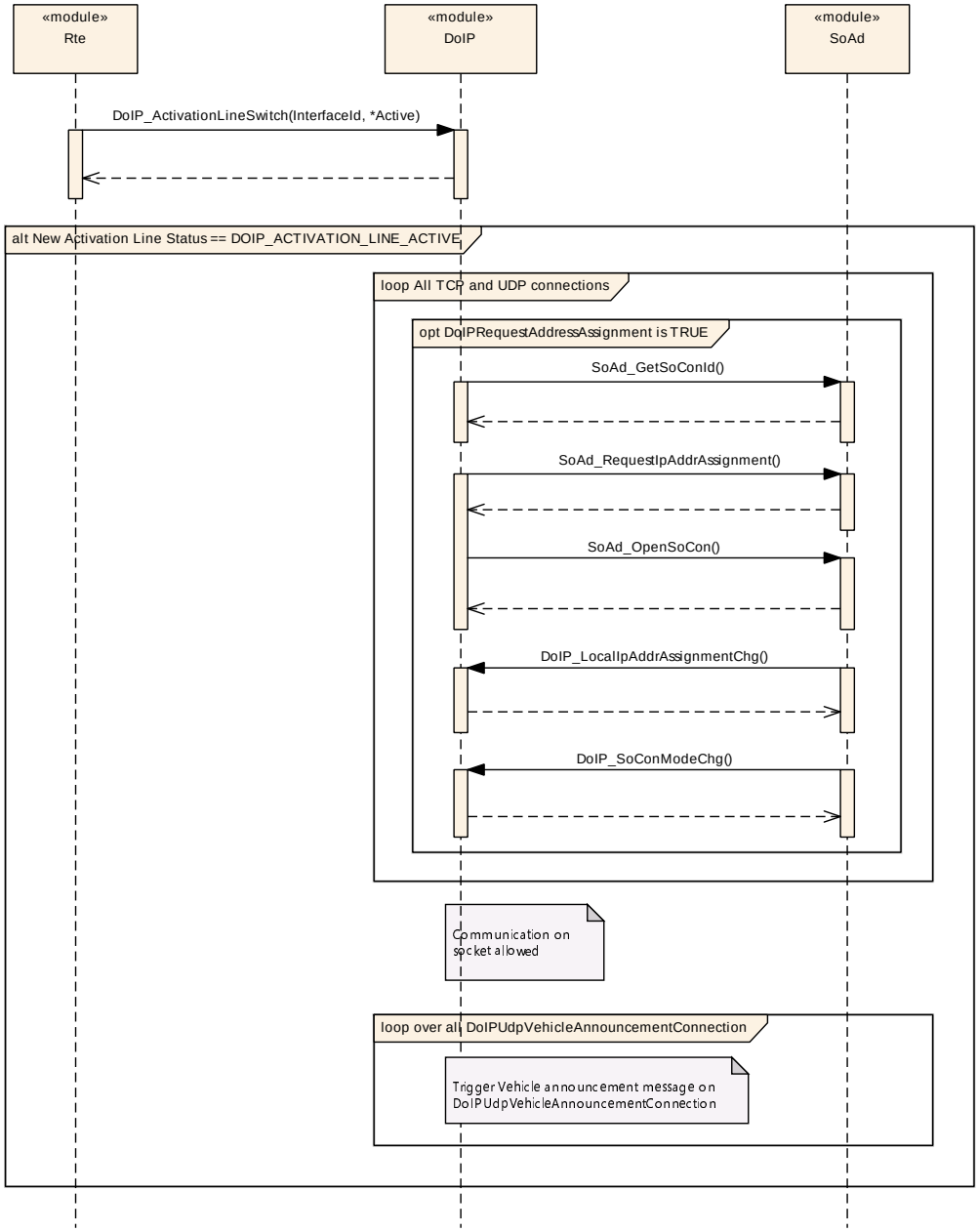

Figure 9.4: Activation Line Handling - Active

9.5 Activation Line Handling - Inactive

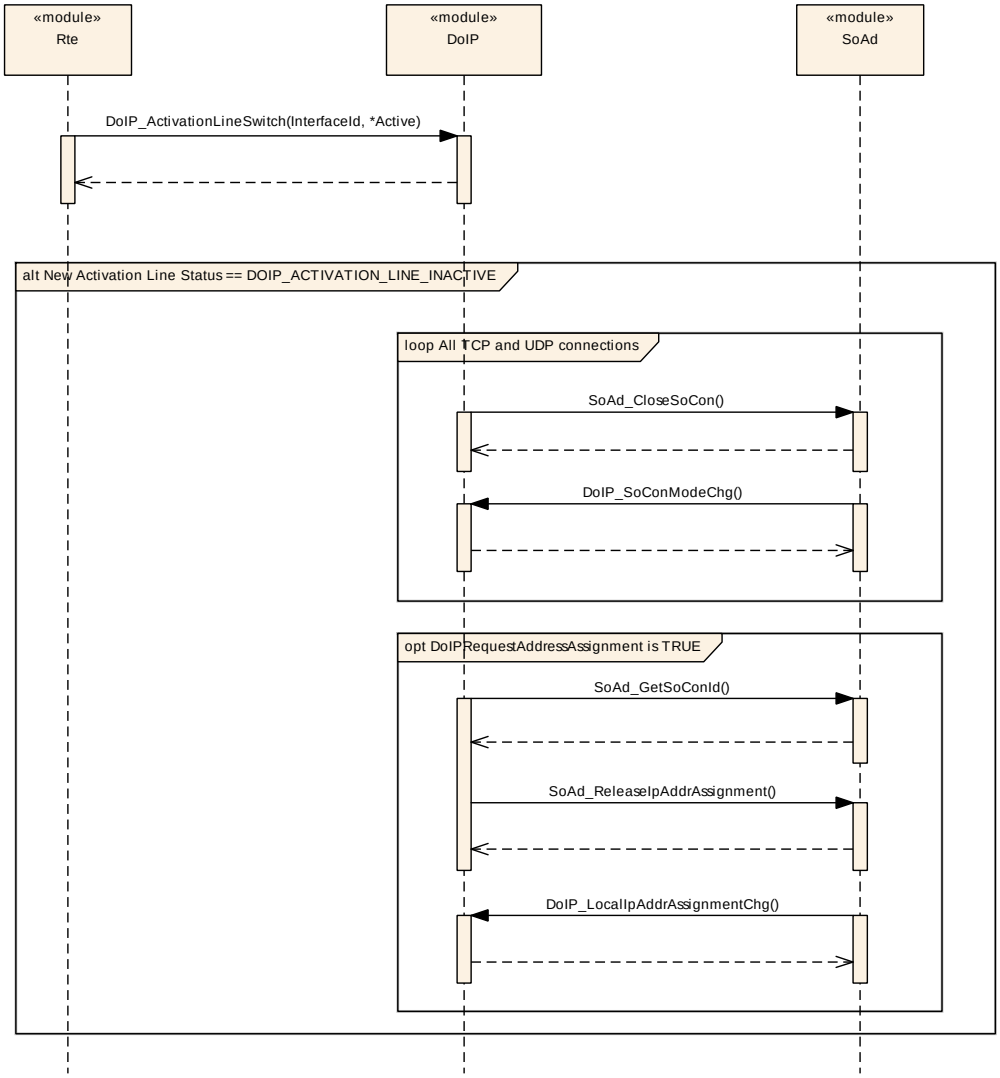

Figure 9.5: Activation Line Handling - Inactive

10 Configuration specification

Chapter 10.2 specifies the structure (containers) and the parameters of the module DoIP. Chapter 10.3 specifies additionally published information of the module DoIP.

10.1 How to read this chapter

For details refer to [1] Chapter 10.1 *"Introduction to configuration specification"*.

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 [1].

10.2.2 DoIP

[ECUC_DoIP_00001] Definition of EcucModuleDef DoIP ⌈

Module Name	DoIP
Description	Configuration of the DoIP (Diagnostic over IP) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR DoIP module.
DoIPGeneral	1	This container specifies the general configuration parameters of the DoIP module.

⌋

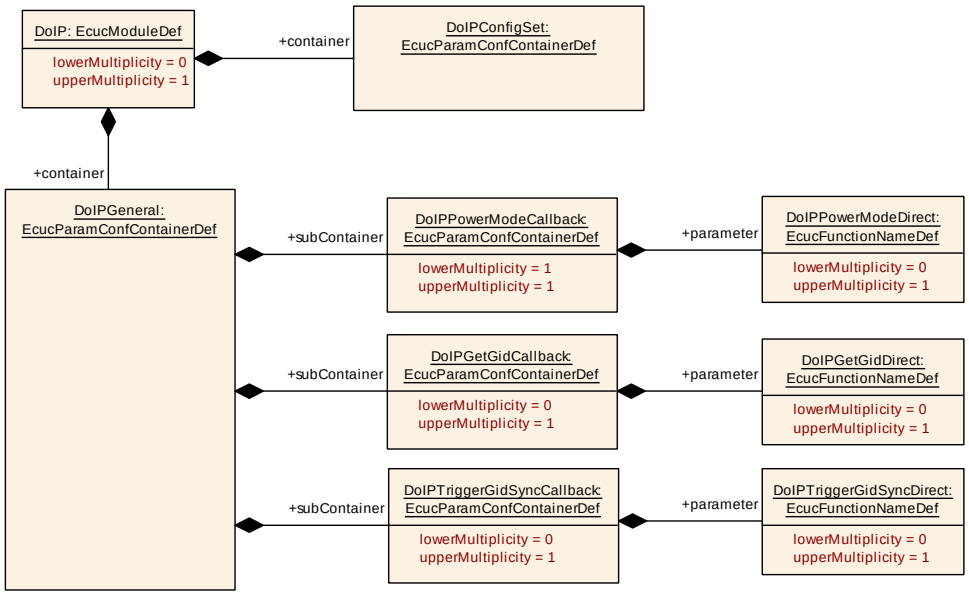

Figure 10.1: DoIP

10.2.3 DoIPGeneral

[ECUC_DoIP_00002] Definition of EcucParamConfContainerDef DoIPGeneral ⌈

Container Name	DoIPGeneral
Parent Container	DoIP
Upstream mapping defined	–
Description	This container specifies the general configuration parameters of the DoIP module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPDevelopmentErrorDetect	1	[ECUC_DoIP_00004]
DoIPDhcpOptionVinUse	1	[ECUC_DoIP_00067]
DoIPEnableSecurityEventReporting	0..1	[ECUC_DoIP_00107]
DoIPEntityStatusMaxByteFieldUse	1	[ECUC_DoIP_00064]
DoIPGIDInvalidityPattern	1	[ECUC_DoIP_00065]
DoIPHostNameSizeMax	1	[ECUC_DoIP_00073]
DoIPMainFunctionPeriod	1	[ECUC_DoIP_00006]
DoIPMaxRequestBytes	1	[ECUC_DoIP_00019]
DoIPMaxUDPRequestPerConnection	1	[ECUC_DoIP_00074]
DoIPNodeType	1	[ECUC_DoIP_00021]
DoIPVinGidMaster	1	[ECUC_DoIP_00017]
DoIPVinInvalidityPattern	1	[ECUC_DoIP_00066]
DoIPUseEidAsGidRef	0..1	[ECUC_DoIP_00106]

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPGetGidCallback	0..1	This container describes the usage of a callback function to get the GID. (If this container is not present no callback function shall be used by DoIP module to retrive the GID.)
DoIPPowerModeCallback	1	This container describes the usage of a callback function to retrieve the current power mode. This container shall always be present.
DoIPTriggerGidSyncCallback	0..1	This container describes the usage of a callback function to trigger the GID synchronization. (If this container does not exist no callback function shall be used by DoIP module to trigger the GID synchronization.)
SecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the DoIP module shall report to the IdsM in case the coresponding security related event occurs (and if DoIPEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events.

⌋

[ECUC_DoIP_00004] Definition of EcucBooleanParamDef DoIPDevelopmentErrorDetect ⌈

Parameter Name	DoIPDevelopmentErrorDetect
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00067] Definition of EcucBooleanParamDef DoIPDhcpOptionVinUse ⌈

Parameter Name	DoIPDhcpOptionVinUse
Parent Container	DoIPGeneral
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00107] Definition of EcucBooleanParamDef DoIPEnableSecurityEventReporting ⌈

Parameter Name	DoIPEnableSecurityEventReporting
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	Switches the reporting of security events to the IdsM: - true: reporting is enabled. - false: reporting is disabled.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00064] Definition of EcucBooleanParamDef DoIPEntityStatusMaxByteFieldUse ⌈

Parameter Name	DoIPEntityStatusMaxByteFieldUse
Parent Container	DoIPGeneral
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00065] Definition of EcucIntegerParamDef DoIPGIDInvalidityPattern ⌈

Parameter Name	DoIPGIDInvalidityPattern	
Parent Container	DoIPGeneral	
Upstream mapping defined	–	
Description	Specifies the Byte pattern that is used for response messages if no valid GID could be retrieved. Only the value '0' or '255' is allowed.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00073] Definition of EcucIntegerParamDef DoIPHostNameSizeMax

⌈

Parameter Name	DoIPHostNameSizeMax	
Parent Container	DoIPGeneral	
Upstream mapping defined	–	
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	EcucIntegerParamDef	
Range	5 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00006] Definition of EcucFloatParamDef DoIPMainFunctionPeriod

⌈

Parameter Name	DoIPMainFunctionPeriod	
Parent Container	DoIPGeneral	
Upstream mapping defined	–	
Description	Determines the frequency at which the DoIP_MainFunction() is called in [s].	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00019] Definition of EcucIntegerParamDef DoIPMaxRequestBytes

⌈

Parameter Name	DoIPMaxRequestBytes	
Parent Container	DoIPGeneral	
Upstream mapping defined	–	
Description	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00074] Definition of EcucIntegerParamDef DoIPMaxUDPRequestPerConnection ⌈

Parameter Name	DoIPMaxUDPRequestPerConnection	
Parent Container	DoIPGeneral	
Upstream mapping defined	–	
Description	This parameter captures the maximum amount of UDP Requests necessary to handle parallel within a single UDP connection.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00021] Definition of EcucEnumerationParamDef DoIPNodeType ⌈

Parameter Name	DoIPNodeType	
Parent Container	DoIPGeneral	
Upstream mapping defined	–	
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.
Pre-Compile only	true	

⌋

[ECUC_DoIP_00017] Definition of EcucBooleanParamDef DoIPVinGidMaster ⌈

Parameter Name	DoIPVinGidMaster
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	Specifies if the DoIP entity is the Vehicle identification Master for the GID (Group ID). Dependency: DoIPUseEIDasGID, DoIPTriggerGIDSynchronization
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00066] Definition of EcucIntegerParamDef DoIPVinInvalidityPattern ⌈

Parameter Name	DoIPVinInvalidityPattern
Parent Container	DoIPGeneral
Upstream mapping defined	–

▽

△

Description	Specifies the Byte pattern that is used for response messages if no valid VIN could be retrieved. Only the value '0' or '255' is allowed.
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 255
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00106] Definition of EcucReferenceDef DoIPUseEidAsGidRef ⌈

Parameter Name	DoIPUseEidAsGidRef
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	Reference to the interface containing the EID which shall be used as GID. Depending on the configuration, the GID is derived either from DoIPEid or from DoIPUseMac AddressForIdentification. Dependency: Parameter shall only be enabled if DoIPVinGidMaster is true. Either Do IPEid or DoIPUseMacAddressForIdentification shall be set for the referenced interface.
Multiplicity	0..1
Type	Reference to DoIPInterface
Pre-Compile only	true

⌋

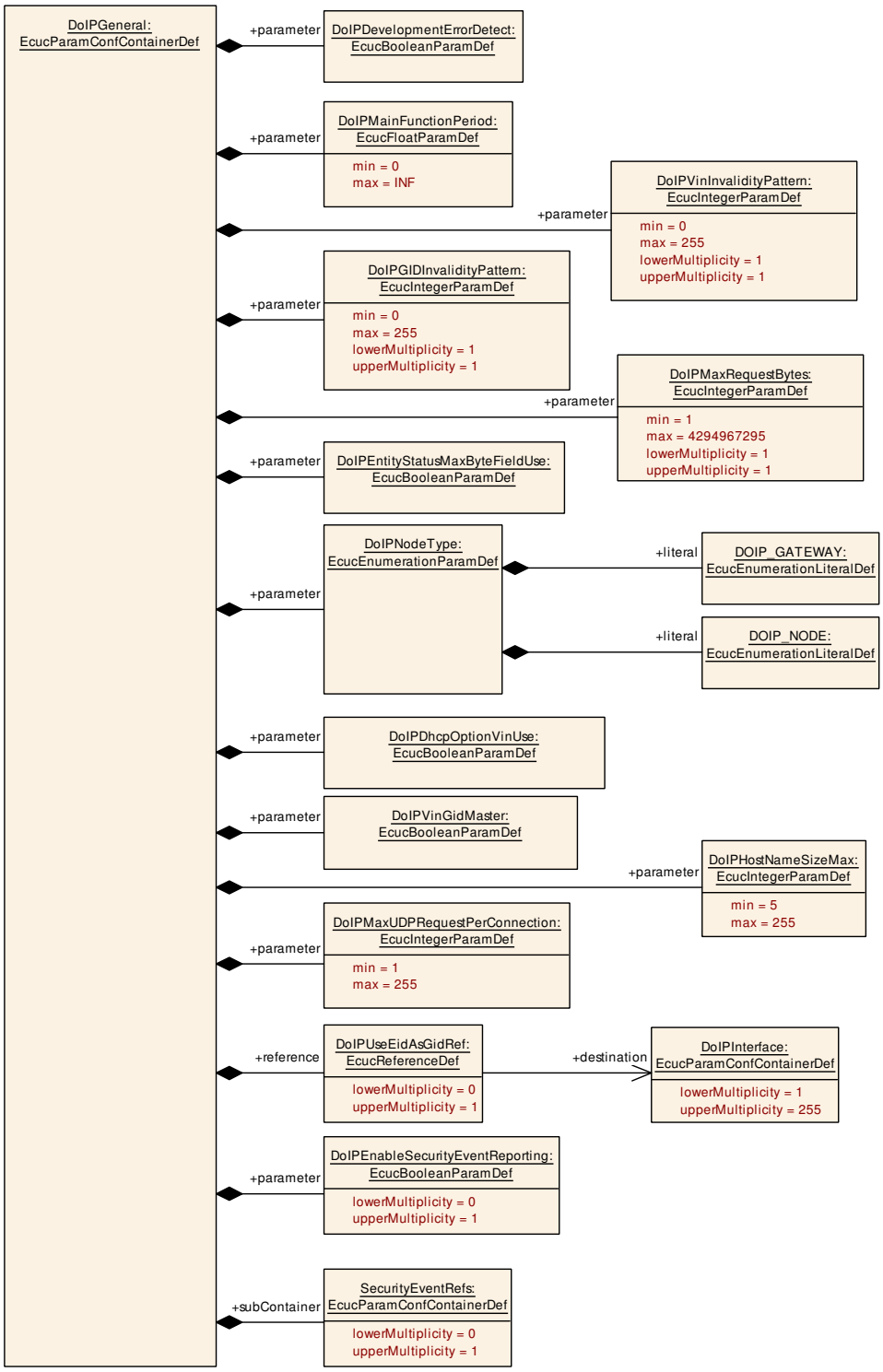

Figure 10.2: DoIPGeneral

10.2.4 DoIPGetGidCallback

[ECUC_DoIP_00024] Definition of EcucParamConfContainerDef DoIPGetGidCallback ⌈

Container Name	DoIPGetGidCallback
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	This container describes the usage of a callback function to get the GID. (If this container is not present no callback function shall be used by DoIP module to retrive the GID.)
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DoIPGetGidDirect	0..1	[ECUC_DoIP_00028]

No Included Containers

⌋

[ECUC_DoIP_00028] Definition of EcucFunctionNameDef DoIPGetGidDirect ⌈

Parameter Name	DoIPGetGidDirect
Parent Container	DoIPGetGidCallback
Upstream mapping defined	–
Description	If the DoIPGetGidDirect parameter exist the DoIP module shall call the configured callback function (<User>_DoIPGetGID) direct. (It is not needed to specify a service port to the DoIP service component.) If the DoIPGetGidDirect parameter does NOT exist the DoIP module shall use a RPort with a CallbackGetGID type of client-server port interface to retrive the GID.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.5 DoIPPowerModeCallback

[ECUC_DoIP_00023] Definition of EcucParamConfContainerDef DoIPPowerMode Callback ⌈

Container Name	DoIPPowerModeCallback
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	This container describes the usage of a callback function to retrieve the current power mode. This container shall always be present.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPPowerModeDirect	0..1	[ECUC_DoIP_00027]

No Included Containers

⌋

[ECUC_DoIP_00027] Definition of EcucFunctionNameDef DoIPPowerModeDirect ⌈

Parameter Name	DoIPPowerModeDirect
Parent Container	DoIPPowerModeCallback
Upstream mapping defined	–
Description	If the DoIPPowerModeDirect parameter exist the DoIP module shall call the configured callback function (<User>_DoIPGetPowerModeCallback) direct. (It is not needed to specify a service port to the DoIP service component.) If the DoIPPowerModeDirect parameter does NOT present the DoIP module shall use a RPort with a CallbackGet PowerMode type of client-server port interface to retrive the current power mode.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.6 DoIPTriggerGidSyncCallback

[ECUC_DoIP_00025] Definition of EcucParamConfContainerDef DoIPTriggerGid SyncCallback ⌈

Container Name	DoIPTriggerGidSyncCallback
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	This container describes the usage of a callback function to trigger the GID synchronization. (If this container does not exist no callback function shall be used by DoIP module to trigger the GID synchronization.)
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPTriggerGidSyncDirect	0..1	[ECUC_DoIP_00029]

No Included Containers

⌋

[ECUC_DoIP_00029] Definition of EcucFunctionNameDef DoIPTriggerGidSyncDirect ⌈

Parameter Name	DoIPTriggerGidSyncDirect
Parent Container	DoIPTriggerGidSyncCallback
Upstream mapping defined	–
Description	If the DoIPTriggerGidSyncDirect parameter exist the DoIP module shall call the configured callback function (<User>_DoIPTriggerGidSyncCallback) direct. (It is not needed to specify a service port to the DoIP service component.) If the DoIPTriggerGid SyncDirect parameter does NOT present the DoIP module shall use a RPort with a CallbackTriggerGIDSynchnonization type of client-server port interface to trigger the GID synchronization.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.7 DoIPSecurityEvents

[ECUC_DoIP_00108] Definition of EcucParamConfContainerDef SecurityEvent Refs ⌈

Container Name	SecurityEventRefs
Parent Container	DoIPGeneral
Upstream mapping defined	–
Description	Container for the references to IdsMEvent elements representing the security events that the DoIP module shall report to the IdsM in case the coresponding security related event occurs (and if DoIPEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED	0..1	[ECUC_DoIP_00112]
SEV_DOIP_HEADER_CHECK_FAILED	0..1	[ECUC_DoIP_00109]
SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED	0..1	[ECUC_DoIP_00110]
SEV_DOIP_ROUTING_ACTIVATION_SUCCESS	0..1	[ECUC_DoIP_00111]

No Included Containers

⌋

[ECUC_DoIP_00112] Definition of EcucReferenceDef SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED ⌈

Parameter Name	SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED
Parent Container	SecurityEventRefs
Upstream mapping defined	–
Description	DoIP diagnostic message check failed
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_DoIP_00109] Definition of EcucReferenceDef SEV_DOIP_HEADER_CHECK_FAILED ⌈

Parameter Name	SEV_DOIP_HEADER_CHECK_FAILED
Parent Container	SecurityEventRefs
Upstream mapping defined	–
Description	DoIP header check failed.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_DoIP_00110] Definition of EcucReferenceDef SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED ⌈

Parameter Name	SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED
Parent Container	SecurityEventRefs
Upstream mapping defined	–
Description	DoIP routing activation check failed.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_DoIP_00111] Definition of EcucReferenceDef SEV_DOIP_ROUTING_ACTIVATION_SUCCESS ⌈

Parameter Name	SEV_DOIP_ROUTING_ACTIVATION_SUCCESS
Parent Container	SecurityEventRefs
Upstream mapping defined	–
Description	Successful DoIP routing activation.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

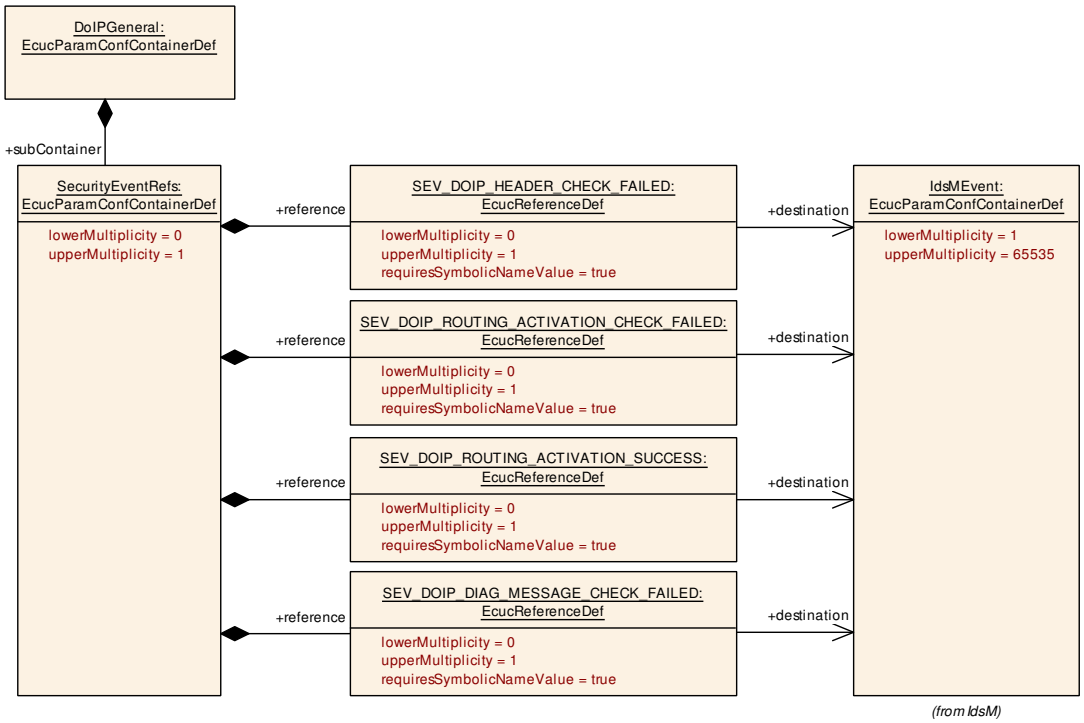

Figure 10.3: `SecurityEventRefs`

10.2.8 DoIPConfigSet

[ECUC_DoIP_00003] Definition of EcucParamConfContainerDef DoIPConfigSet ⌈

Container Name	DoIPConfigSet
Parent Container	DoIP
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR DoIP module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPGid	0..1	[ECUC_DoIP_00015]
DoIPLogicalAddress	1	[ECUC_DoIP_00020]

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPInterface	1..255	This container defines a logical IP interface and collects properties to configure this interface.

⌋

[ECUC_DoIP_00015] Definition of EcucIntegerParamDef DoIPGid ⌈

Parameter Name	DoIPGid	
Parent Container	DoIPConfigSet	
Upstream mapping defined	–	
Description	Configured GID (Group ID of) for vehicle identification/vehicle announcement. Dependency: DoIPVinGIDMaster, DoIPGetGID DoIPGid shall be used if DoIPVinGid Master is true and DoIPUseEidAsGidRef is disabled.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 281474976710655	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_DoIP_00020] Definition of EcucIntegerParamDef DoIPLogicalAddress ⌈

Parameter Name	DoIPLogicalAddress	
Parent Container	DoIPConfigSet	
Upstream mapping defined	–	
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	–	
Post-Build Value	Loadable	Selectable

⌋

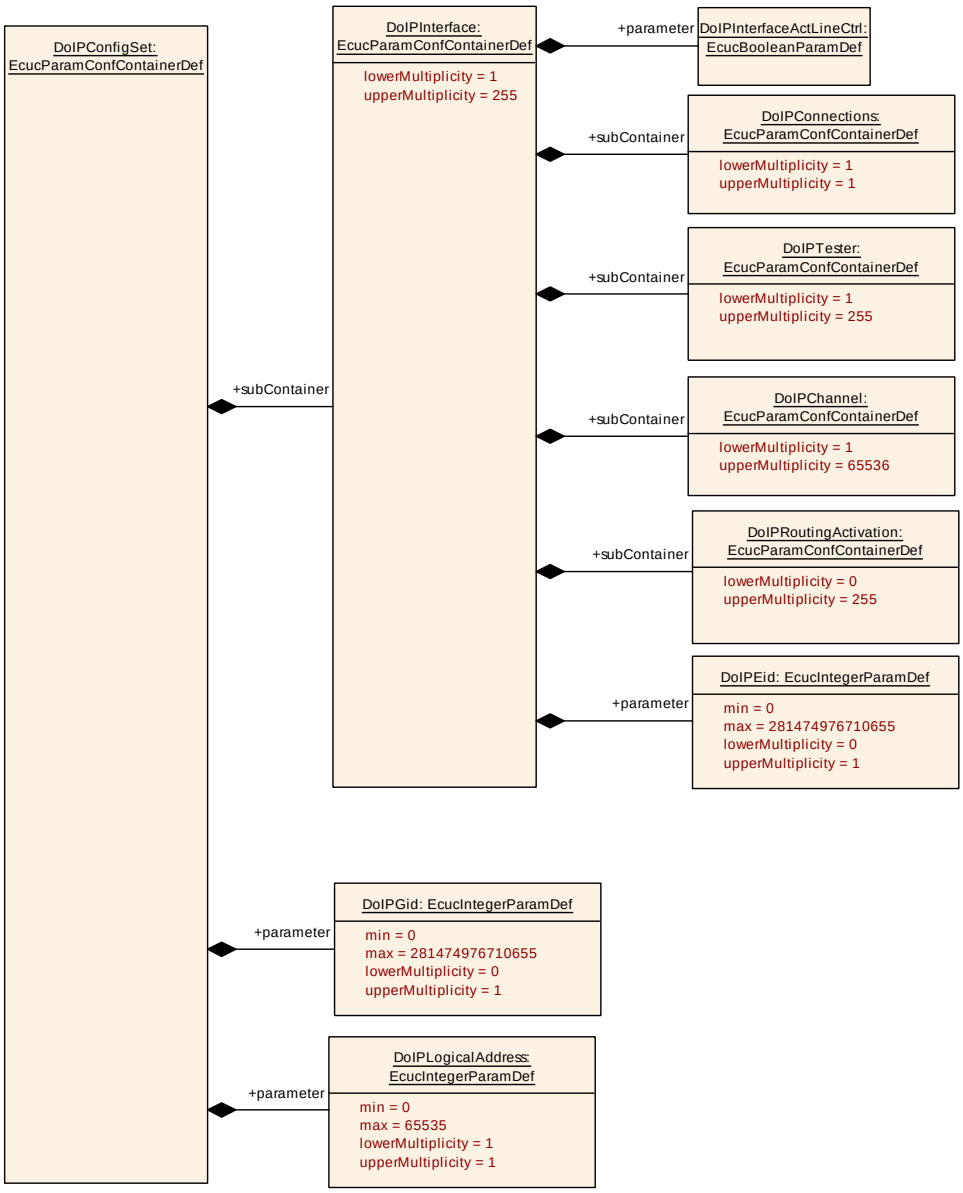

Figure 10.4: DoIPConfigSet

10.2.9 DoIPInterface

[ECUC_DoIP_00100] Definition of EcucParamConfContainerDef DoIPInterface ⌈

Container Name	DoIPInterface
Parent Container	DoIPConfigSet
Upstream mapping defined	–
Description	This container defines a logical IP interface and collects properties to configure this interface.
Multiplicity	1..255

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPAliveCheckResponseTimeout	1	[ECUC_DoIP_00009]
DoIPEid	0..1	[ECUC_DoIP_00014]
DoIPGeneralInactivityTime	1	[ECUC_DoIP_00068]
DoIPInitialInactivityTime	1	[ECUC_DoIP_00010]
DoIPInitialVehicleAnnouncementTime	1	[ECUC_DoIP_00008]
DoIPInterfaceActLineCtrl	1	[ECUC_DoIP_00101]
DoIPInterfaceAnnouncementStart	1	[ECUC_DoIP_00099]
DoIPInterfaceId	1	[ECUC_DoIP_00098]
DoIPMaxTesterConnections	1	[ECUC_DoIP_00012]
DoIPUseMacAddressForIdentification	1	[ECUC_DoIP_00013]
DoIPUseVehicleIdentificationSyncStatus	1	[ECUC_DoIP_00016]
DoIPVehicleAnnouncementCount	0..1	[ECUC_DoIP_00094]
DoIPVehicleAnnouncementInterval	0..1	[ECUC_DoIP_00007]

Included Containers		
Container Name	**Multiplicity**	**Description**
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.
DoIPFurtherActionByteCallback	0..1	This container describes the Callbackfunction to get the Further Action byte. This container shall always be present. If the Do IPFurtherActionByteDirect parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackGetFurther ActionByte with the name "CBGetFurtherAction Byte_<shortname of enclosing DoIPInterface container>".
DoIPProtocolVersion	1	This container defines the protocol version of the DoIP entity.
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	1..255	This container describes the properties of the possible connectable Tester for the DoIP entity.

⌋

[ECUC_DoIP_00009] Definition of EcucFloatParamDef DoIPAliveCheckResponseTimeout ⌈

Parameter Name	DoIPAliveCheckResponseTimeout
Parent Container	DoIPInterface
Upstream mapping defined	–
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_Alive Check of ISO 13400-2.

▽

△

Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00014] Definition of EcucIntegerParamDef DoIPEid ⌈

Parameter Name	DoIPEid	
Parent Container	DoIPInterface	
Upstream mapping defined	–	
Description	Configured EID (Entity ID of) for vehicle identification/vehicle announcement. Only necessary if DoIPUseMacAddressForIdentification is set to FALSE. Dependency: DoIPEid shall be used if DoIPUseMacAddressForIdentification of the same DoIPInterface is false.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 281474976710655	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_DoIP_00068] Definition of EcucFloatParamDef DoIPGeneralInactivity Time ⌈

Parameter Name	DoIPGeneralInactivityTime	
Parent Container	DoIPInterface	
Upstream mapping defined	–	
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.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	300	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00010] Definition of EcucFloatParamDef DoIPInitialInactivityTime ⌈

Parameter Name	DoIPInitialInactivityTime
Parent Container	DoIPInterface
Upstream mapping defined	–

▽

△

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.
Multiplicity	1
Type	EcucFloatParamDef
Range	]0 .. INF[
Default value	2
Pre-Compile only	true

⌋

[ECUC_DoIP_00008] Definition of EcucFloatParamDef DoIPInitialVehicleAnnouncementTime ⌈

Parameter Name	DoIPInitialVehicleAnnouncementTime
Parent Container	DoIPInterface
Upstream mapping defined	–
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.
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. INF]
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00101] Definition of EcucBooleanParamDef DoIPInterfaceActLineCtrl ⌈

Parameter Name	DoIPInterfaceActLineCtrl
Parent Container	DoIPInterface
Upstream mapping defined	–
Description	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed (TRUE) or • is always available (FALSE).
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00099] Definition of EcucEnumerationParamDef DoIPInterfaceAnnouncementStart ⌈

Parameter Name	DoIPInterfaceAnnouncementStart
Parent Container	DoIPInterface
Upstream mapping defined	–

▽

△

Description	This attribute defines, when vehicle announcement is started on a DoIPInterface • Automatic: As soon as the underlying UDP vehicle announcement connection switches to SOAD_SOCON_ONLINE • OnTrigger: As soon as the API DoIP_TriggerVehicleAnnouncement is called for the given DoIPInterface instance	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DOIP_AUTOMATIC_ANNOUNCE	AUTOMATIC announcement
	DOIP_ONTRIGGER_ANNOUNCE	TRIGGERED announcement
Default value	DOIP_AUTOMATIC_ANNOUNCE	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00098] Definition of EcucIntegerParamDef DoIPInterfaceId ⌈

Parameter Name	DoIPInterfaceId	
Parent Container	DoIPInterface	
Upstream mapping defined	–	
Description	This parameter is an identifier of the DoIPInterface. The value of this parameter will be assigned to the symbolic name derived from the container short name.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00012] Definition of EcucIntegerParamDef DoIPMaxTesterConnections ⌈

Parameter Name	DoIPMaxTesterConnections	
Parent Container	DoIPInterface	
Upstream mapping defined	–	
Description	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00013] Definition of EcucBooleanParamDef DoIPUseMacAddressForIdentification ⌈

Parameter Name	DoIPUseMacAddressForIdentification
Parent Container	DoIPInterface
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Provided the information if a configured EID at vehicle identification response/vehicle announcement 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. Dependency: DoIPEid
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00016] Definition of EcucBooleanParamDef DoIPUseVehicleIdentificationSyncStatus ⌈

Parameter Name	DoIPUseVehicleIdentificationSyncStatus
Parent Container	DoIPInterface
Upstream mapping defined	–
Description	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/announcement.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00094] Definition of EcucIntegerParamDef DoIPVehicleAnnouncementCount ⌈

Parameter Name	DoIPVehicleAnnouncementCount
Parent Container	DoIPInterface
Upstream mapping defined	–
Description	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_Announce_Num of ISO 13400-2.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	1 .. 255
Default value	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00007] Definition of EcucFloatParamDef DoIPVehicleAnnouncementInterval ⌈

Parameter Name	DoIPVehicleAnnouncementInterval	
Parent Container	DoIPInterface	
Upstream mapping defined	–	
Description	Time to wait in [s] for sending subsequent vehicle anouncement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

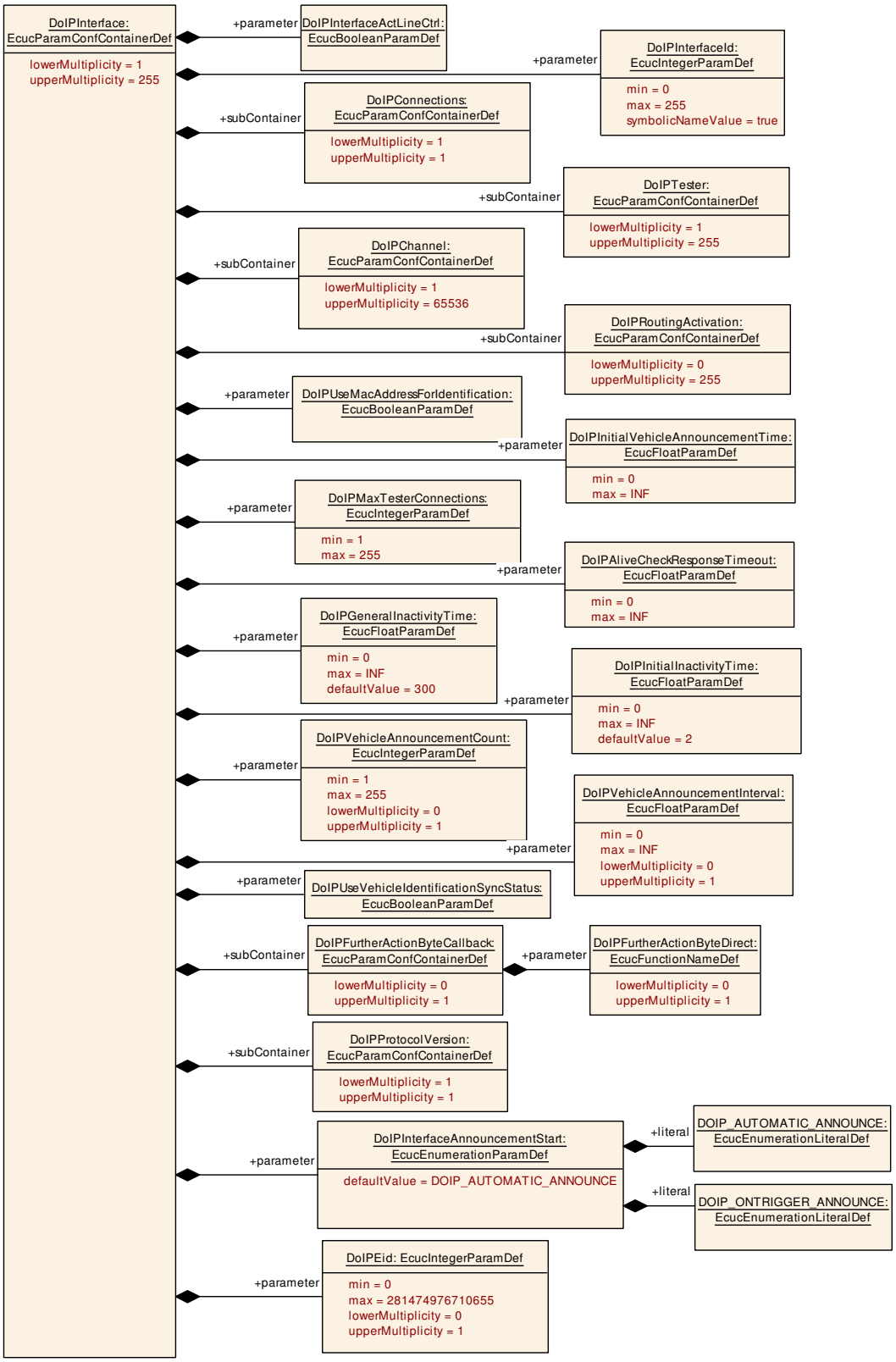

Figure 10.5: DoIPInterface

10.2.10 DoIPChannel

[ECUC_DoIP_00069] Definition of EcucParamConfContainerDef DoIPChannel ⌈

Container Name	DoIPChannel
Parent Container	DoIPInterface
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Configuration of one DoIPChannel.
Multiplicity	1..65536
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPChannelSARef	1	[ECUC_DoIP_00070]
DoIPChannelTARef	1	[ECUC_DoIP_00071]

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPPduRRxPdu	1	This container contains the Rx Pdus to connect with the Rx Pdus of the PduR.
DoIPPduRTxPdu	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.

⌋

[ECUC_DoIP_00070] Definition of EcucReferenceDef DoIPChannelSARef ⌈

Parameter Name	DoIPChannelSARef
Parent Container	DoIPChannel
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Reference to the DoIPTester.
Multiplicity	1
Type	Reference to DoIPTester
Pre-Compile only	true

⌋

[ECUC_DoIP_00071] Definition of EcucReferenceDef DoIPChannelTARef ⌈

Parameter Name	DoIPChannelTARef
Parent Container	DoIPChannel
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Reference to the target address.
Multiplicity	1
Type	Reference to DoIPTargetAddress
Pre-Compile only	true

⌋

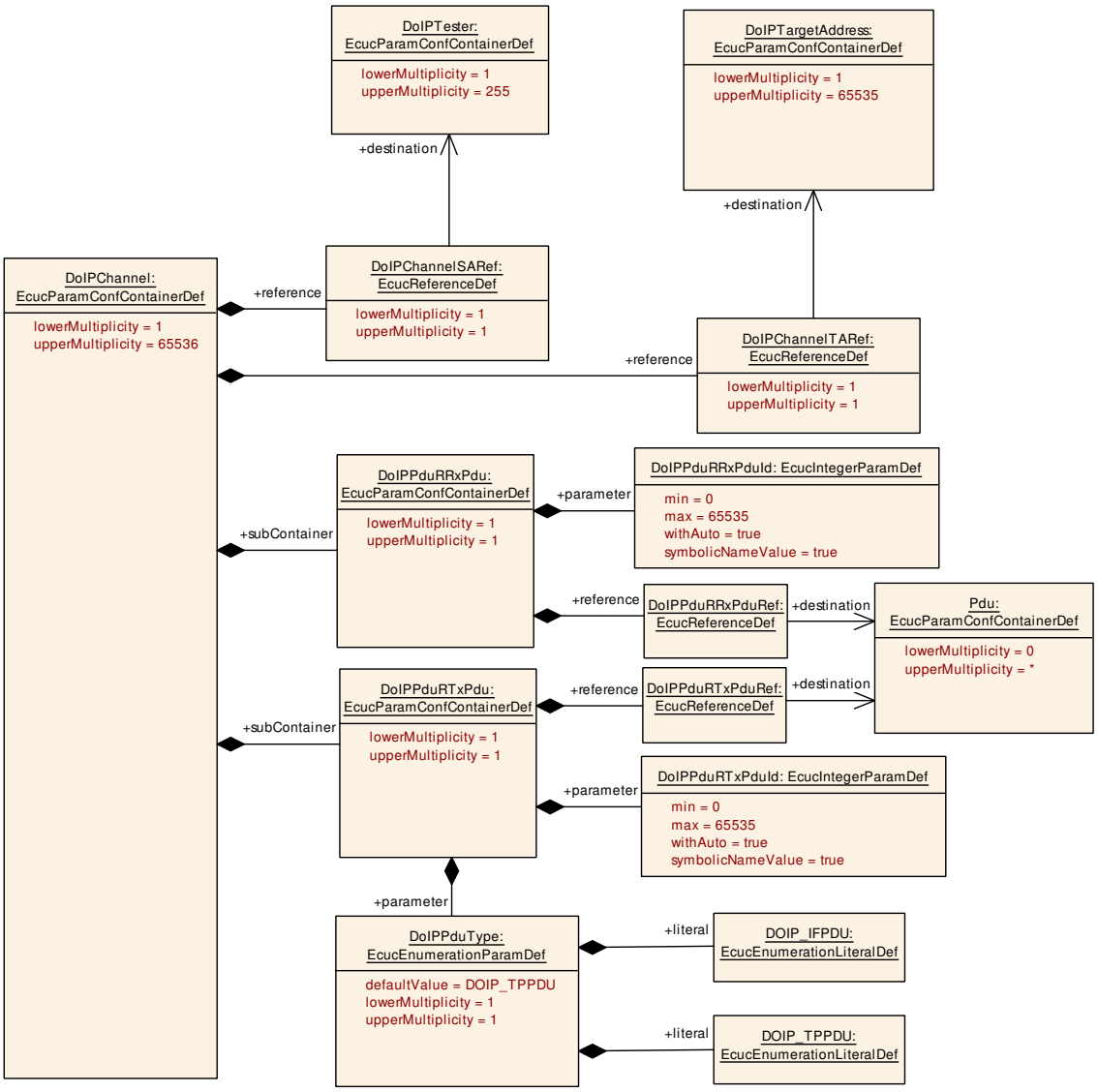

Figure 10.6: DoIPChannel

10.2.11 DoIPPduRRxPdu

[ECUC_DoIP_00055] Definition of EcucParamConfContainerDef DoIPPduRRx Pdu ⌈

Container Name	DoIPPduRRxPdu
Parent Container	DoIPChannel
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container contains the Rx Pdus to connect with the Rx Pdus of the PduR.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPPduRRxPduId	1	[ECUC_DoIP_00057]
DoIPPduRRxPduRef	1	[ECUC_DoIP_00058]

No Included Containers

⌋

[ECUC_DoIP_00057] Definition of EcucIntegerParamDef DoIPPduRRxPduId ⌈

Parameter Name	DoIPPduRRxPduId	
Parent Container	DoIPPduRRxPdu	
Upstream mapping defined	–	
Description	The DoIPPduRRxPduId is required by the API call DoIP_TpCancelReceive.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00058] Definition of EcucReferenceDef DoIPPduRRxPduRef ⌈

Parameter Name	DoIPPduRRxPduRef	
Parent Container	DoIPPduRRxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.12 DoIPPduRTxPdu

[ECUC_DoIP_00056] Definition of EcucParamConfContainerDef DoIPPduRTxPdu ⌈

Container Name	DoIPPduRTxPdu
Parent Container	DoIPChannel
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
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.
Multiplicity	1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPPduRTxPduId	1	[ECUC_DoIP_00060]
DoIPPduType	1	[ECUC_DoIP_00075]
DoIPPduRTxPduRef	1	[ECUC_DoIP_00059]

No Included Containers

⌋

[ECUC_DoIP_00060] Definition of EcucIntegerParamDef DoIPPduRTxPduId ⌈

Parameter Name	DoIPPduRTxPduId	
Parent Container	DoIPPduRTxPdu	
Upstream mapping defined	–	
Description	The DoIPPduRTxPduId is required by DoIP_TpTransmit or DoIP_IfTransmit and DoIP_TpCancelTransmit.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00075] Definition of EcucEnumerationParamDef DoIPPduType ⌈

Parameter Name	DoIPPduType	
Parent Container	DoIPPduRTxPdu	
Upstream mapping defined	–	
Description	API Type to use for communication with PduR. DOIP_IFPDU for UUDT messages, DOIP_TPPDU for all other diagnostic messages.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DOIP_IFPDU	DOIP_IFPDU for UUDT messages,
	DOIP_TPPDU	DOIP_TPPDU for all other diagnostic messages.
Default value	DOIP_TPPDU	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_DoIP_00059] Definition of EcucReferenceDef DoIPPduRTxPduRef ⌈

Parameter Name	DoIPPduRTxPduRef	
Parent Container	DoIPPduRTxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.13 DoIPConnections

[ECUC_DoIP_00032] Definition of EcucParamConfContainerDef DoIPConnections ⌈

Container Name	DoIPConnections
Parent Container	DoIPInterface
Upstream mapping defined	–
Description	Container contains all lower layer connection specific information, i.e. the single Pdu References and Handle IDs to the SoAd.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
DoIPTargetAddress	1..65535	This container describes a possible TargetAddress that is supported by DoIP.
DoIPTcpConnection	2..255	This container describes a TCP connection to the lower layer So Ad module.
DoIPUdpConnection	1..255	This Container describes a Udp connection to the lower layer So Ad module.
DoIPUdpVehicleAnnouncement Connection	0..255	This container describes the UDP multicast connections to the lower layer SoAd module.

⌋

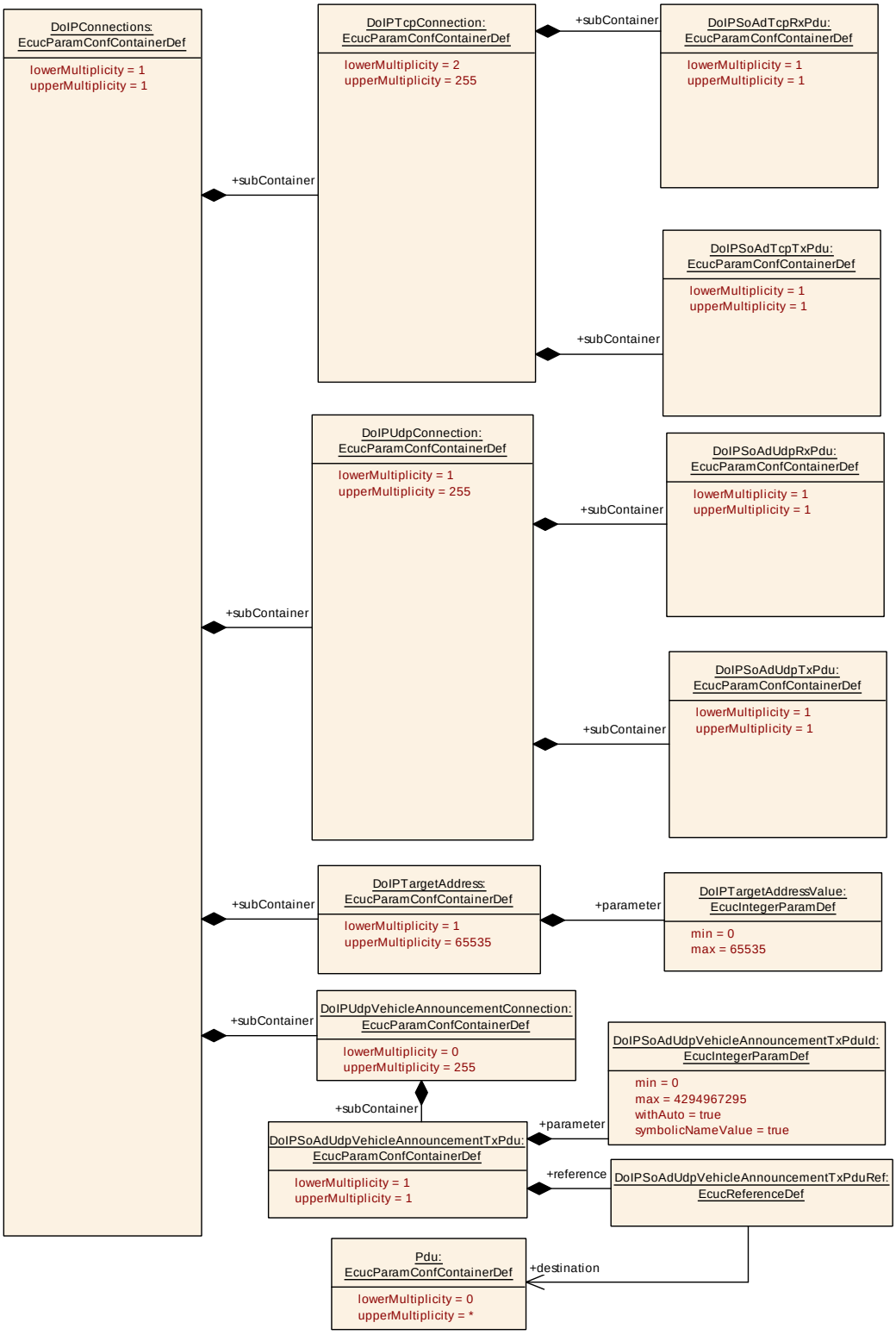

Figure 10.7: DoIPConnections

10.2.14 DoIPTargetAddress

[ECUC_DoIP_00053] Definition of EcucParamConfContainerDef DoIPTargetAddress ⌈

Container Name	DoIPTargetAddress
Parent Container	DoIPConnections
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes a possible TargetAddress that is supported by DoIP.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPTargetAddressValue	1	[ECUC_DoIP_00054]

No Included Containers

⌋

[ECUC_DoIP_00054] Definition of EcucIntegerParamDef DoIPTargetAddressValue ⌈

Parameter Name	DoIPTargetAddressValue	
Parent Container	DoIPTargetAddress	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Valid Target Address of a DoIP target address.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.2.15 DoIPTcpConnection

[ECUC_DoIP_00045] Definition of EcucParamConfContainerDef DoIPTcpConnection ⌈

Container Name	DoIPTcpConnection
Parent Container	DoIPConnections
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes a TCP connection to the lower layer SoAd module.
Multiplicity	2..255

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPRequestAddressAssignment	1	[ECUC_DoIP_00095]
DoIPTcpConnectionSecurityRequired	0..1	[ECUC_DoIP_00097]

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPSoAdTcpRxPdu	1	This container describes a Rx PDU received via SoAd over TCP
DoIPSoAdTcpTxPdu	1	This container describes a Tx PDU sent via SoAd over TCP

⌋

[ECUC_DoIP_00095] Definition of EcucBooleanParamDef DoIPRequestAddress Assignment ⌈

Parameter Name	DoIPRequestAddressAssignment
Parent Container	DoIPTcpConnection, DoIPUdpConnection, DoIPUdpVehicleAnnouncementConnection
Upstream mapping defined	–
Description	The DoIP module shall request IP address assignment by calling SoAd_RequestIpAddr Assignment() for the TcpIpLocalAddr related to this DoIpConnection.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_DoIP_00097] Definition of EcucBooleanParamDef DoIPTcpConnection SecurityRequired ⌈

Parameter Name	DoIPTcpConnectionSecurityRequired
Parent Container	DoIPTcpConnection
Upstream mapping defined	–
Description	Indicates if the associated TCP socket uses a secure connection (e.g. TLS) Dependency: DoIPTcpConnectionSecurityRequired shall only be enabled if DoIP_ ISO13400_2_2019 is enabled.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

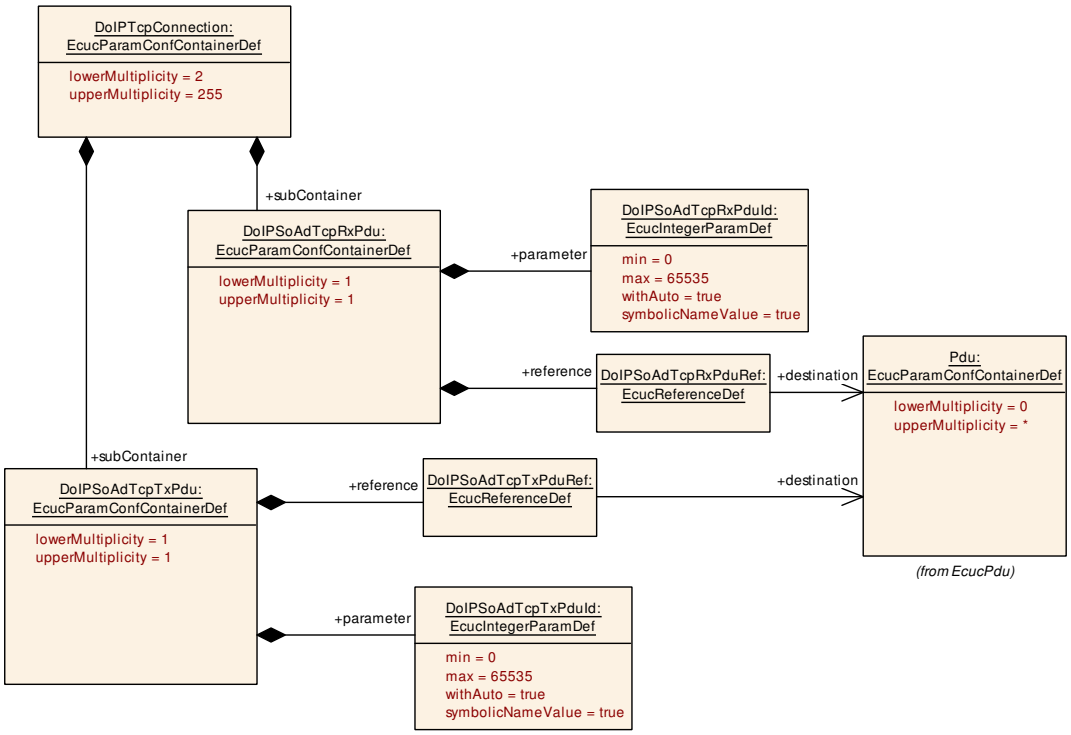

Figure 10.8: DoIPTcpConnection

10.2.16 DoIPSoAdTcpRxPdu

[ECUC_DoIP_00080] Definition of EcucParamConfContainerDef DoIPSoAdTcpRx Pdu ⌈

Container Name	DoIPSoAdTcpRxPdu
Parent Container	DoIPTcpConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes a Rx PDU received via SoAd over TCP
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPSoAdTcpRxPduId	1	[ECUC_DoIP_00082]
DoIPSoAdTcpRxPduRef	1	[ECUC_DoIP_00083]

No Included Containers

⌋

[ECUC_DoIP_00082] Definition of EcucIntegerParamDef DoIPSoAdTcpRxPduId ⌈

Parameter Name	DoIPSoAdTcpRxPduId	
Parent Container	DoIPSoAdTcpRxPdu	
Upstream mapping defined	–	
Description	The DoIPSoAdTcpRxPduId 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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00083] Definition of EcucReferenceDef DoIPSoAdTcpRxPduRef ⌈

Parameter Name	DoIPSoAdTcpRxPduRef	
Parent Container	DoIPSoAdTcpRxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.17 DoIPSoAdTcpTxPdu

[ECUC_DoIP_00081] Definition of EcucParamConfContainerDef DoIPSoAdTcpTxPdu ⌈

Container Name	DoIPSoAdTcpTxPdu
Parent Container	DoIPTcpConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes a Tx PDU sent via SoAd over TCP
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPSoAdTcpTxPduId	1	[ECUC_DoIP_00085]
DoIPSoAdTcpTxPduRef	1	[ECUC_DoIP_00084]

No Included Containers

⌋

[ECUC_DoIP_00085] Definition of EcucIntegerParamDef DoIPSoAdTcpTxPduId ⌈

Parameter Name	DoIPSoAdTcpTxPduId	
Parent Container	DoIPSoAdTcpTxPdu	
Upstream mapping defined	–	
Description	The DoIPSoAdTcpTxPduId 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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00084] Definition of EcucReferenceDef DoIPSoAdTcpTxPduRef ⌈

Parameter Name	DoIPSoAdTcpTxPduRef	
Parent Container	DoIPSoAdTcpTxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.18 DoIPUdpConnection

[ECUC_DoIP_00052] Definition of EcucParamConfContainerDef DoIPUdpConnection ⌈

Container Name	DoIPUdpConnection
Parent Container	DoIPConnections
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This Container describes a Udp connection to the lower layer SoAd module.
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPRequestAddressAssignment	1	[ECUC_DoIP_00095]

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPSoAdUdpRxPdu	1	This container describes a Rx PDU received via SoAd over UDP.
DoIPSoAdUdpTxPdu	1	This container describes a Tx PDU sent via SoAd over UDP.

⌋

For parameter table [ECUC_DoIP_00095] DoIPRequestAddress Assignment, see definition below container DoIPTcpConnection.

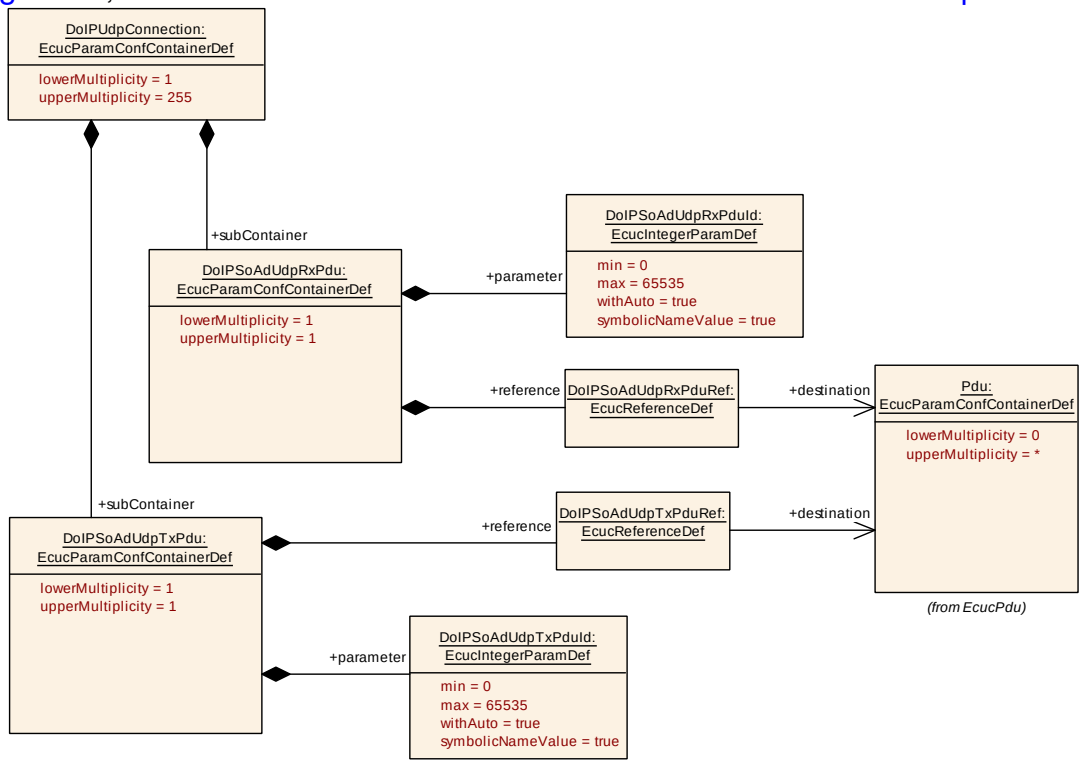

Figure 10.9: DoIPUdpConnection

10.2.19 DoIPSoAdUdpRxPdu

[ECUC_DoIP_00046] Definition of EcucParamConfContainerDef DoIPSoAdUdpRxPdu ⌈

Container Name	DoIPSoAdUdpRxPdu
Parent Container	DoIPUdpConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes a Rx PDU received via SoAd over UDP.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPSoAdUdpRxPduId	1	[ECUC_DoIP_00048]
DoIPSoAdUdpRxPduRef	1	[ECUC_DoIP_00049]

No Included Containers

⌋

[ECUC_DoIP_00048] Definition of EcucIntegerParamDef DoIPSoAdUdpRxPduId ⌈

Parameter Name	DoIPSoAdUdpRxPduId	
Parent Container	DoIPSoAdUdpRxPdu	
Upstream mapping defined	–	
Description	The DoIPSoAdUdpRxPduId is required by the API call DoIP_SoAdIfRxIndication to receive I-PDUs from the SoAd.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00049] Definition of EcucReferenceDef DoIPSoAdUdpRxPduRef ⌈

Parameter Name	DoIPSoAdUdpRxPduRef	
Parent Container	DoIPSoAdUdpRxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.20 DoIPSoAdUdpTxPdu

[ECUC_DoIP_00047] Definition of EcucParamConfContainerDef DoIPSoAdUdpTxPdu ⌈

Container Name	DoIPSoAdUdpTxPdu
Parent Container	DoIPUdpConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	This container describes a Tx PDU sent via SoAd over UDP.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPSoAdUdpTxPduId	1	[ECUC_DoIP_00051]
DoIPSoAdUdpTxPduRef	1	[ECUC_DoIP_00050]

No Included Containers

⌋

[ECUC_DoIP_00051] Definition of EcucIntegerParamDef DoIPSoAdUdpTxPduId ⌈

Parameter Name	DoIPSoAdUdpTxPduId	
Parent Container	DoIPSoAdUdpTxPdu	
Upstream mapping defined	–	
Description	The DoIPSoAdUdpTxPduId is required by the API call DoIP_SoAdIfTxConfirmation 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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00050] Definition of EcucReferenceDef DoIPSoAdUdpTxPduRef ⌈

Parameter Name	DoIPSoAdUdpTxPduRef	
Parent Container	DoIPSoAdUdpTxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.21 DoIPUdpVehicleAnnouncementConnection

[ECUC_DoIP_00076] Definition of EcucParamConfContainerDef DoIPUdpVehicleAnnouncementConnection ⌈

Container Name	DoIPUdpVehicleAnnouncementConnection
Parent Container	DoIPConnections
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes the UDP multicast connections to the lower layer SoAd module.
Multiplicity	0..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DoIPRequestAddressAssignment	1	[ECUC_DoIP_00095]

Included Containers		
Container Name	Multiplicity	Description
DoIPSoAdUdpVehicle AnnouncementTxPdu	1	This container describes the vehicle announcement TxPdu sent via the SoAd.

⌋

For parameter table [ECUC_DoIP_00095] DoIPRequestAddressAssignment, see definition below container DoIPTcpConnection.

10.2.22 DoIPSoAdUdpVehicleAnnouncementTxPdu

[ECUC_DoIP_00077] Definition of EcucParamConfContainerDef DoIPSoAdUdpVehicleAnnouncementTxPdu ⌈

Container Name	DoIPSoAdUdpVehicleAnnouncementTxPdu
Parent Container	DoIPUdpVehicleAnnouncementConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes the vehicle announcement TxPdu sent via the SoAd.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DoIPSoAdUdpVehicleAnnouncementTxPduId	1	[ECUC_DoIP_00078]
DoIPSoAdUdpVehicleAnnouncementTxPduRef	1	[ECUC_DoIP_00079]

No Included Containers

⌋

[ECUC_DoIP_00078] Definition of EcucIntegerParamDef DoIPSoAdUdpVehicleAnnouncementTxPduId ⌈

Parameter Name	DoIPSoAdUdpVehicleAnnouncementTxPduId	
Parent Container	DoIPSoAdUdpVehicleAnnouncementTxPdu	
Upstream mapping defined	–	
Description	The DoIPSoAdUdpVehicleAnnouncementTxPduId is required by the API call DoIP_SoAdIfTxConfirmation() 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 .. 4294967295	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00079] Definition of EcucReferenceDef DoIPSoAdUdpVehicleAnnouncementTxPduRef ⌈

Parameter Name	DoIPSoAdUdpVehicleAnnouncementTxPduRef	
Parent Container	DoIPSoAdUdpVehicleAnnouncementTxPdu	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to the "global" PDU structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.23 DoIPFurtherActionByteCallback

[ECUC_DoIP_00092] Definition of EcucParamConfContainerDef DoIPFurtherActionByteCallback ⌈

Container Name	DoIPFurtherActionByteCallback
Parent Container	DoIPInterface
Upstream mapping defined	–
Description	This container describes the Callbackfunction to get the Further Action byte. This container shall always be present. If the DoIPFurtherActionByteDirect parameter is not present, the DoIP module will use an RPort of ServiceInterface CallbackGetFurtherActionByte with the name "CBGetFurtherActionByte_<shortname of enclosing DoIPInterface container>".
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPFurtherActionByteDirect	0..1	[ECUC_DoIP_00093]

No Included Containers

⌋

[ECUC_DoIP_00093] Definition of EcucFunctionNameDef DoIPFurtherActionByteDirect ⌈

Parameter Name	DoIPFurtherActionByteDirect
Parent Container	DoIPFurtherActionByteCallback
Upstream mapping defined	–
Description	Direct C Callback function to get the OEM specific Further Action Byte for the DoIP vehicle identification response/vehicle announcement. If the DoIPFurtherActionByte Direct parameter is present, the DoIP module will not use an RPort of ServiceInterface “CBGetFurtherActionByte” but will call the configured function.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.24 DoIPProtocolVersion

[ECUC_DoIP_00102] Definition of EcucParamConfContainerDef DoIPProtocolVersion ⌈

Container Name	DoIPProtocolVersion
Parent Container	DoIPInterface
Upstream mapping defined	–
Description	This container defines the protocol version of the DoIP entity.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIP_ISO13400_2_2012	1	[ECUC_DoIP_00103]
DoIP_ISO13400_2_2019	1	[ECUC_DoIP_00105]
DoIPAnnouncedProtocolVersion	1	[ECUC_DoIP_00104]

No Included Containers

⌋

[ECUC_DoIP_00103] Definition of EcucBooleanParamDef DoIP_ISO13400_2_2012 ⌈

Parameter Name	DoIP_ISO13400_2_2012
Parent Container	DoIPProtocolVersion
Upstream mapping defined	–
Description	If this parameter is enabled, the DoIP entity supports communication with ISO13400-2:2012 compliant tester using protocol version 0x02.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_DoIP_00105] Definition of EcucBooleanParamDef DoIP_ISO13400_2_2019 ⌈

Parameter Name	DoIP_ISO13400_2_2019
Parent Container	DoIPProtocolVersion
Upstream mapping defined	–
Description	If this parameter is enabled, the DoIP entity supports communication with ISO13400-2:2019 compliant tester using protocol version 0x03.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_DoIP_00104] Definition of EcucEnumerationParamDef DoIPAnnouncedProtocolVersion ⌈

Parameter Name	DoIPAnnouncedProtocolVersion	
Parent Container	DoIPProtocolVersion	
Upstream mapping defined	–	
Description	This parameter defines the used protocol version for vehicle announcements and vehicle identification responses if the vehicle identification request contains protocol version 0xFF.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DOIP_ISO13400_2_2012	ISO13400-2:2012
	DOIP_ISO13400_2_2019	ISO13400-2:2019
Pre-Compile only	true	

⌋

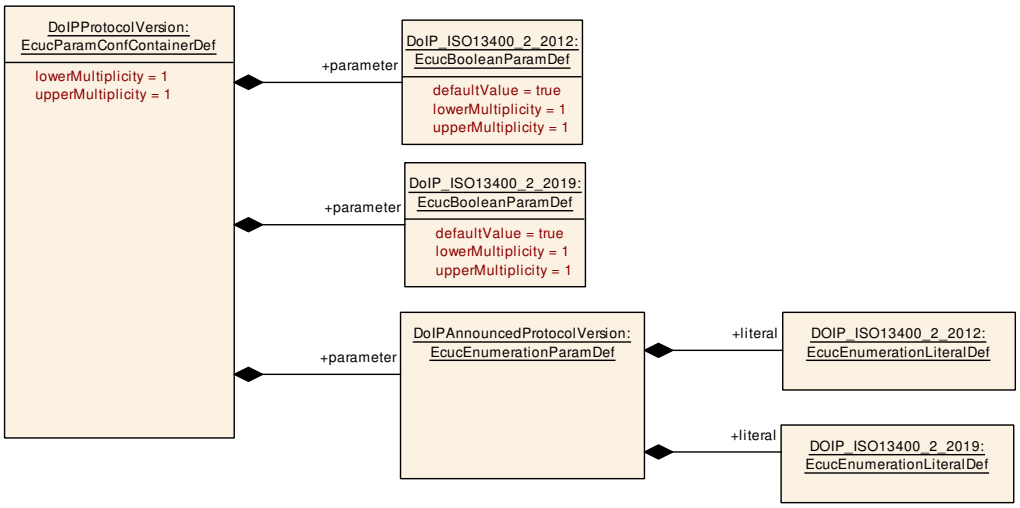

Figure 10.10: DoIPProtocolVersion

10.2.25 DoIPRoutingActivation

[ECUC_DoIP_00030] Definition of EcucParamConfContainerDef DoIPRoutingActivation ⌈

Container Name	DoIPRoutingActivation
Parent Container	DoIPInterface
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
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.
Multiplicity	0..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPRoutingActivationNumber	1	[ECUC_DoIP_00033]
DoIPRoutingActivationSecurityRequired	0..1	[ECUC_DoIP_00096]
DoIPTargetAddressRef	1..65535	[ECUC_DoIP_00034]

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPRoutingActivation AuthenticationCallback	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". <Routing Activation> is the ShortName of the DoIPRoutingActiviation container.

⌋

△

Included Containers		
Container Name	**Multiplicity**	**Description**
DoIPRoutingActivation ConfirmationCallback	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". <Routing Activation> is the ShortName of the DoIPRoutingActiviation container.

⌋

[ECUC_DoIP_00033] Definition of EcucIntegerParamDef DoIPRoutingActivation Number ⌈

Parameter Name	DoIPRoutingActivationNumber	
Parent Container	DoIPRoutingActivation	
Upstream mapping defined	–	
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	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_DoIP_00096] Definition of EcucBooleanParamDef DoIPRoutingActivation SecurityRequired ⌈

Parameter Name	DoIPRoutingActivationSecurityRequired
Parent Container	DoIPRoutingActivation
Upstream mapping defined	–
Description	Indicates if a routing activation requires a secure TCP connection
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_DoIP_00034] Definition of EcucReferenceDef DoIPTargetAddressRef ⌈

Parameter Name	DoIPTargetAddressRef
Parent Container	DoIPRoutingActivation
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Reference to all DoIPTargetAddress which are activated on this Routing activation.
Multiplicity	1..65535

▽

△

Type	Reference to DoIPTargetAddress	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

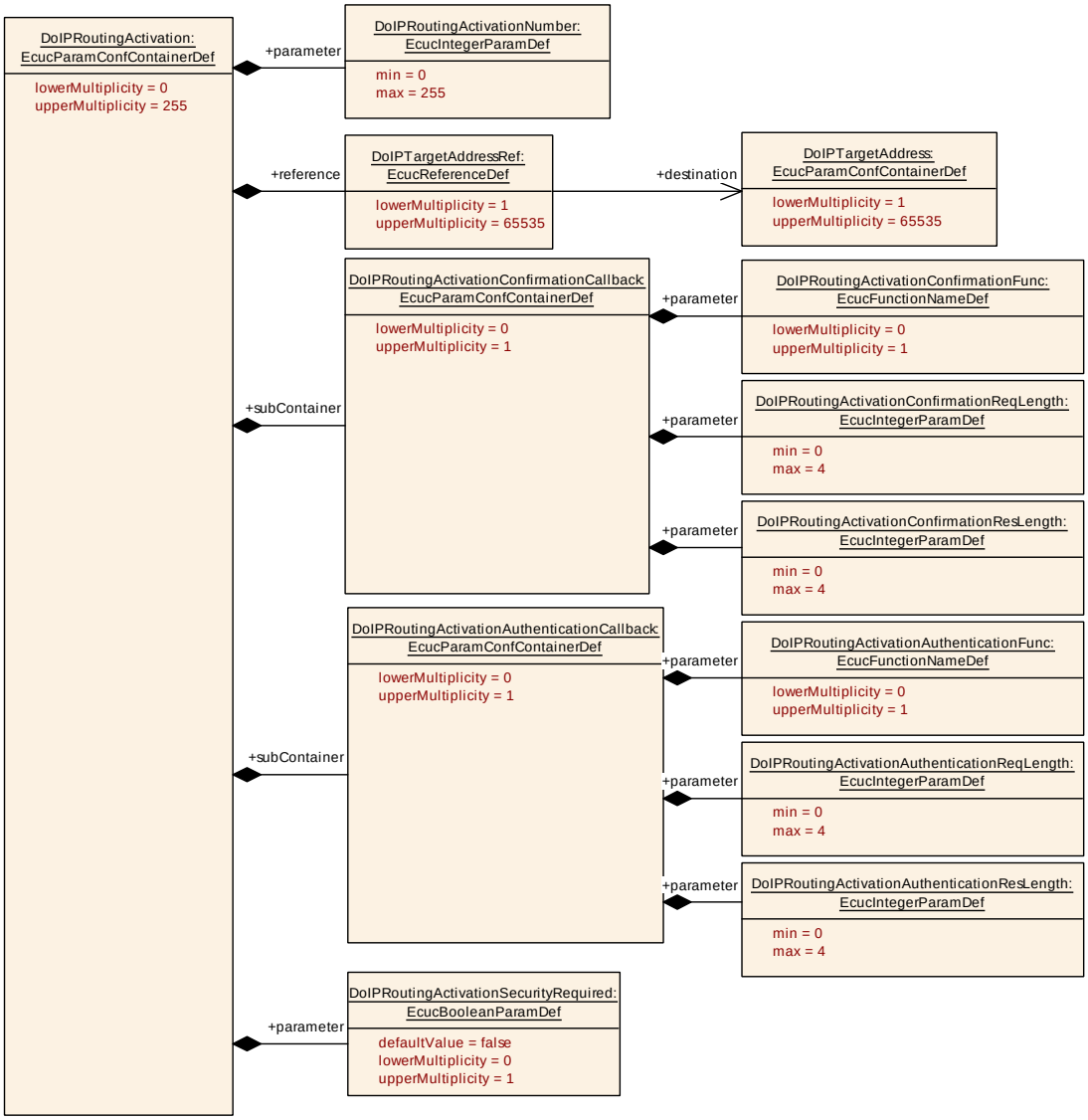

Figure 10.11: `DoIPRoutingActivation`

10.2.26 DoIPRoutingActivationAuthenticationCallback

[ECUC_DoIP_00035] Definition of EcucParamConfContainerDef DoIPRoutingActivationAuthenticationCallback ⌈

Container Name	DoIPRoutingActivationAuthenticationCallback
Parent Container	DoIPRoutingActivation
Upstream mapping defined	–
Description	Container describes the Callbackfunction to call on a Routing Activation Request for Authentication. If this container is configured but the DoIPRoutingActivation AuthenticationFunc parameter is not present, the DoIP module will use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation with the name "CB<Routing Activation>RoutingActivation". <RoutingActivation> is the ShortName of the Do IPRoutingActiviation container.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPRoutingActivationAuthenticationFunc	0..1	[ECUC_DoIP_00039]
DoIPRoutingActivationAuthenticationReqLength	1	[ECUC_DoIP_00040]
DoIPRoutingActivationAuthenticationResLength	1	[ECUC_DoIP_00041]

No Included Containers

⌋

[ECUC_DoIP_00039] Definition of EcucFunctionNameDef DoIPRoutingActivation AuthenticationFunc ⌈

Parameter Name	DoIPRoutingActivationAuthenticationFunc
Parent Container	DoIPRoutingActivationAuthenticationCallback
Upstream mapping defined	–
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	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00040] Definition of EcucIntegerParamDef DoIPRoutingActivation AuthenticationReqLength ⌈

Parameter Name	DoIPRoutingActivationAuthenticationReqLength
Parent Container	DoIPRoutingActivationAuthenticationCallback
Upstream mapping defined	–
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	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00041] Definition of EcucIntegerParamDef DoIPRoutingActivation AuthenticationResLength ⌈

Parameter Name	DoIPRoutingActivationAuthenticationResLength	
Parent Container	DoIPRoutingActivationAuthenticationCallback	
Upstream mapping defined	–	
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	–	
Pre-Compile only	true	

⌋

10.2.27 DoIPRoutingActivationConfirmationCallback

[ECUC_DoIP_00061] Definition of EcucParamConfContainerDef DoIPRoutingActivationConfirmationCallback ⌈

Container Name	DoIPRoutingActivationConfirmationCallback
Parent Container	DoIPRoutingActivation
Upstream mapping defined	–
Description	Container describes the Callbackfunction to call on a Routing Activation Request for Confirmation. If this container is configured but the DoIPRoutingActivationConfirmation Func parameter is not present the DoIP module will use an RPort of ServiceInterface <RoutingActivation>_RoutingActivation with the name "CB<RoutingActivation>Routing Activation". <RoutingActivation> is the ShortName of the DoIPRoutingActiviation container.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPRoutingActivationConfirmationFunc	0..1	[ECUC_DoIP_00036]
DoIPRoutingActivationConfirmationReqLength	1	[ECUC_DoIP_00037]
DoIPRoutingActivationConfirmationResLength	1	[ECUC_DoIP_00038]

No Included Containers

⌋

[ECUC_DoIP_00036] Definition of EcucFunctionNameDef DoIPRoutingActivationConfirmationFunc ⌈

Parameter Name	DoIPRoutingActivationConfirmationFunc
Parent Container	DoIPRoutingActivationConfirmationCallback
Upstream mapping defined	–
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	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_DoIP_00037] Definition of EcucIntegerParamDef DoIPRoutingActivationConfirmationReqLength ⌈

Parameter Name	DoIPRoutingActivationConfirmationReqLength	
Parent Container	DoIPRoutingActivationConfirmationCallback	
Upstream mapping defined	–	
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	–	
Pre-Compile only	true	

⌋

[ECUC_DoIP_00038] Definition of EcucIntegerParamDef DoIPRoutingActivationConfirmationResLength ⌈

Parameter Name	DoIPRoutingActivationConfirmationResLength
Parent Container	DoIPRoutingActivationConfirmationCallback
Upstream mapping defined	–
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	–	
Pre-Compile only	true	

⌋

10.2.28 DoIPTester

[ECUC_DoIP_00031] Definition of EcucParamConfContainerDef DoIPTester ⌈

Container Name	DoIPTester
Parent Container	DoIPInterface
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes the properties of the possible connectable Tester for the DoIP entity.
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DoIPNumByteDiagAckNack	1	[ECUC_DoIP_00042]
DoIPTesterSA	1	[ECUC_DoIP_00043]
DoIPRoutingActivationRef	1..255	[ECUC_DoIP_00062]

No Included Containers

⌋

[ECUC_DoIP_00042] Definition of EcucIntegerParamDef DoIPNumByteDiagAckNack ⌈

Parameter Name	DoIPNumByteDiagAckNack	
Parent Container	DoIPTester	
Upstream mapping defined	–	
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	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_DoIP_00043] Definition of EcucIntegerParamDef DoIPTesterSA ⌈

Parameter Name	DoIPTesterSA	
Parent Container	DoIPTester	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Source Address of the Tester sent via routing activation or diagnostic message.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_DoIP_00062] Definition of EcucReferenceDef DoIPRoutingActivationRef ⌈

Parameter Name	DoIPRoutingActivationRef	
Parent Container	DoIPTester	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Reference to a DoIPRoutingActivation describing the possible routing activations of the DoIPTester	
Multiplicity	1..255	
Type	Reference to DoIPRoutingActivation	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

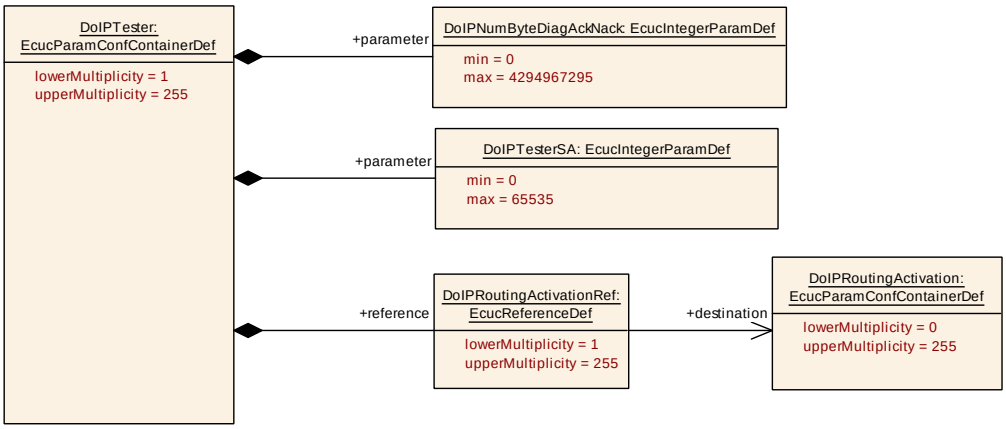

Figure 10.12: DoIPTester

10.3 Published Information

For details refer to [1] Chapter 10.3 *"Published Information"*.

A Change History

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

A.1 Traceable item history of this document according to AUTOSAR Release R26-11

A.1.1 Added Specification Items in R26-11

Number	Heading
[SWS_DoIP_00157]	Initial Inactivity Timer
[SWS_DoIP_91004]	Definition of Std_ReturnType-extension for module DoIP

Table A.1: Added Specification Items in R26-11

A.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_DoIP_00001]	Definition of EcucModuleDef DoIP
[ECUC_DoIP_00002]	Definition of EcucParamConfContainerDef DoIPGeneral
[ECUC_DoIP_00003]	Definition of EcucParamConfContainerDef DoIPConfigSet
[ECUC_DoIP_00004]	Definition of EcucBooleanParamDef DoIPDevelopmentErrorDetect
[ECUC_DoIP_00006]	Definition of EcucFloatParamDef DoIPMainFunctionPeriod
[ECUC_DoIP_00007]	Definition of EcucFloatParamDef DoIPVehicleAnnouncementInterval
[ECUC_DoIP_00008]	Definition of EcucFloatParamDef DoIPInitialVehicleAnnouncementTime
[ECUC_DoIP_00009]	Definition of EcucFloatParamDef DoIPAliveCheckResponseTimeout
[ECUC_DoIP_00010]	Definition of EcucFloatParamDef DoIPInitialInactivityTime
[ECUC_DoIP_00012]	Definition of EcucIntegerParamDef DoIPMaxTesterConnections
[ECUC_DoIP_00013]	Definition of EcucBooleanParamDef DoIPUseMacAddressForIdentification
[ECUC_DoIP_00014]	Definition of EcucIntegerParamDef DoIPEid
[ECUC_DoIP_00015]	Definition of EcucIntegerParamDef DoIPGid
[ECUC_DoIP_00016]	Definition of EcucBooleanParamDef DoIPUseVehicleIdentificationSync Status
[ECUC_DoIP_00017]	Definition of EcucBooleanParamDef DoIPVinGidMaster
[ECUC_DoIP_00019]	Definition of EcucIntegerParamDef DoIPMaxRequestBytes
[ECUC_DoIP_00020]	Definition of EcucIntegerParamDef DoIPLogicalAddress
[ECUC_DoIP_00021]	Definition of EcucEnumerationParamDef DoIPNodeType

▽

△

Number	Heading
[ECUC_DoIP_00023]	Definition of EcucParamConfContainerDef DoIPPowerModeCallback
[ECUC_DoIP_00024]	Definition of EcucParamConfContainerDef DoIPGetGidCallback
[ECUC_DoIP_00025]	Definition of EcucParamConfContainerDef DoIPTriggerGidSyncCallback
[ECUC_DoIP_00027]	Definition of EcucFunctionNameDef DoIPPowerModeDirect
[ECUC_DoIP_00028]	Definition of EcucFunctionNameDef DoIPGetGidDirect
[ECUC_DoIP_00029]	Definition of EcucFunctionNameDef DoIPTriggerGidSyncDirect
[ECUC_DoIP_00030]	Definition of EcucParamConfContainerDef DoIPRoutingActivation
[ECUC_DoIP_00031]	Definition of EcucParamConfContainerDef DoIPTester
[ECUC_DoIP_00032]	Definition of EcucParamConfContainerDef DoIPConnections
[ECUC_DoIP_00033]	Definition of EcucIntegerParamDef DoIPRoutingActivationNumber
[ECUC_DoIP_00034]	Definition of EcucReferenceDef DoIPTargetAddressRef
[ECUC_DoIP_00035]	Definition of EcucParamConfContainerDef DoIPRoutingActivation AuthenticationCallback
[ECUC_DoIP_00036]	Definition of EcucFunctionNameDef DoIPRoutingActivationConfirmation Func
[ECUC_DoIP_00037]	Definition of EcucIntegerParamDef DoIPRoutingActivationConfirmationReq Length
[ECUC_DoIP_00038]	Definition of EcucIntegerParamDef DoIPRoutingActivationConfirmationRes Length
[ECUC_DoIP_00039]	Definition of EcucFunctionNameDef DoIPRoutingActivationAuthentication Func
[ECUC_DoIP_00040]	Definition of EcucIntegerParamDef DoIPRoutingActivationAuthenticationReq Length
[ECUC_DoIP_00041]	Definition of EcucIntegerParamDef DoIPRoutingActivationAuthenticationRes Length
[ECUC_DoIP_00042]	Definition of EcucIntegerParamDef DoIPNumByteDiagAckNack
[ECUC_DoIP_00043]	Definition of EcucIntegerParamDef DoIPTesterSA
[ECUC_DoIP_00045]	Definition of EcucParamConfContainerDef DoIPTcpConnection
[ECUC_DoIP_00046]	Definition of EcucParamConfContainerDef DoIPSoAdUdpRxPdu
[ECUC_DoIP_00047]	Definition of EcucParamConfContainerDef DoIPSoAdUdpTxPdu
[ECUC_DoIP_00048]	Definition of EcucIntegerParamDef DoIPSoAdUdpRxPduId
[ECUC_DoIP_00049]	Definition of EcucReferenceDef DoIPSoAdUdpRxPduRef
[ECUC_DoIP_00050]	Definition of EcucReferenceDef DoIPSoAdUdpTxPduRef
[ECUC_DoIP_00051]	Definition of EcucIntegerParamDef DoIPSoAdUdpTxPduId
[ECUC_DoIP_00052]	Definition of EcucParamConfContainerDef DoIPUdpConnection
[ECUC_DoIP_00053]	Definition of EcucParamConfContainerDef DoIPTargetAddress
[ECUC_DoIP_00054]	Definition of EcucIntegerParamDef DoIPTargetAddressValue
[ECUC_DoIP_00055]	Definition of EcucParamConfContainerDef DoIPPduRRxPdu
[ECUC_DoIP_00056]	Definition of EcucParamConfContainerDef DoIPPduRTxPdu
[ECUC_DoIP_00057]	Definition of EcucIntegerParamDef DoIPPduRRxPduId

▽

△

Number	Heading
[ECUC_DoIP_00058]	Definition of EcucReferenceDef DoIPPduRRxPduRef
[ECUC_DoIP_00059]	Definition of EcucReferenceDef DoIPPduRTxPduRef
[ECUC_DoIP_00060]	Definition of EcucIntegerParamDef DoIPPduRTxPduId
[ECUC_DoIP_00061]	Definition of EcucParamConfContainerDef DoIPRoutingActivation ConfirmationCallback
[ECUC_DoIP_00062]	Definition of EcucReferenceDef DoIPRoutingActivationRef
[ECUC_DoIP_00064]	Definition of EcucBooleanParamDef DoIPEntityStatusMaxByteFieldUse
[ECUC_DoIP_00065]	Definition of EcucIntegerParamDef DoIPGIDInvalidityPattern
[ECUC_DoIP_00066]	Definition of EcucIntegerParamDef DoIPVinInvalidityPattern
[ECUC_DoIP_00067]	Definition of EcucBooleanParamDef DoIPDhcpOptionVinUse
[ECUC_DoIP_00068]	Definition of EcucFloatParamDef DoIPGeneralInactivityTime
[ECUC_DoIP_00069]	Definition of EcucParamConfContainerDef DoIPChannel
[ECUC_DoIP_00070]	Definition of EcucReferenceDef DoIPChannelSARef
[ECUC_DoIP_00071]	Definition of EcucReferenceDef DoIPChannelTARef
[ECUC_DoIP_00073]	Definition of EcucIntegerParamDef DoIPHostNameSizeMax
[ECUC_DoIP_00074]	Definition of EcucIntegerParamDef DoIPMaxUDPRequestPerConnection
[ECUC_DoIP_00075]	Definition of EcucEnumerationParamDef DoIPPduType
[ECUC_DoIP_00076]	Definition of EcucParamConfContainerDef DoIPUdpVehicleAnnouncement Connection
[ECUC_DoIP_00077]	Definition of EcucParamConfContainerDef DoIPSoAdUdpVehicle AnnouncementTxPdu
[ECUC_DoIP_00078]	Definition of EcucIntegerParamDef DoIPSoAdUdpVehicleAnnouncementTx PduId
[ECUC_DoIP_00079]	Definition of EcucReferenceDef DoIPSoAdUdpVehicleAnnouncementTxPdu Ref
[ECUC_DoIP_00080]	Definition of EcucParamConfContainerDef DoIPSoAdTcpRxPdu
[ECUC_DoIP_00081]	Definition of EcucParamConfContainerDef DoIPSoAdTcpTxPdu
[ECUC_DoIP_00082]	Definition of EcucIntegerParamDef DoIPSoAdTcpRxPduId
[ECUC_DoIP_00083]	Definition of EcucReferenceDef DoIPSoAdTcpRxPduRef
[ECUC_DoIP_00084]	Definition of EcucReferenceDef DoIPSoAdTcpTxPduRef
[ECUC_DoIP_00085]	Definition of EcucIntegerParamDef DoIPSoAdTcpTxPduId
[ECUC_DoIP_00092]	Definition of EcucParamConfContainerDef DoIPFurtherActionByteCallback
[ECUC_DoIP_00093]	Definition of EcucFunctionNameDef DoIPFurtherActionByteDirect
[ECUC_DoIP_00094]	Definition of EcucIntegerParamDef DoIPVehicleAnnouncementCount
[ECUC_DoIP_00095]	Definition of EcucBooleanParamDef DoIPRequestAddressAssignment
[ECUC_DoIP_00096]	Definition of EcucBooleanParamDef DoIPRoutingActivationSecurityRequired
[ECUC_DoIP_00097]	Definition of EcucBooleanParamDef DoIPTcpConnectionSecurityRequired
[ECUC_DoIP_00098]	Definition of EcucIntegerParamDef DoIPInterfaceId
[ECUC_DoIP_00099]	Definition of EcucEnumerationParamDef DoIPInterfaceAnnouncementStart
[ECUC_DoIP_00100]	Definition of EcucParamConfContainerDef DoIPInterface

▽

△

Number	Heading
[ECUC_DoIP_00101]	Definition of EcucBooleanParamDef DoIPInterfaceActLineCtrl
[ECUC_DoIP_00102]	Definition of EcucParamConfContainerDef DoIPProtocolVersion
[ECUC_DoIP_00103]	Definition of EcucBooleanParamDef DoIP_ISO13400_2_2012
[ECUC_DoIP_00104]	Definition of EcucEnumerationParamDef DoIPAnnouncedProtocolVersion
[ECUC_DoIP_00105]	Definition of EcucBooleanParamDef DoIP_ISO13400_2_2019
[ECUC_DoIP_00106]	Definition of EcucReferenceDef DoIPUseEidAsGidRef
[ECUC_DoIP_00107]	Definition of EcucBooleanParamDef DoIPEnableSecurityEventReporting
[ECUC_DoIP_00108]	Definition of EcucParamConfContainerDef SecurityEventRefs
[ECUC_DoIP_00109]	Definition of EcucReferenceDef SEV_DOIP_HEADER_CHECK_FAILED
[ECUC_DoIP_00110]	Definition of EcucReferenceDef SEV_DOIP_ROUTING_ACTIVATION_ CHECK_FAILED
[ECUC_DoIP_00111]	Definition of EcucReferenceDef SEV_DOIP_ROUTING_ACTIVATION_ SUCCESS
[ECUC_DoIP_00112]	Definition of EcucReferenceDef SEV_DOIP_DIAG_MESSAGE_CHECK_ FAILED
[SWS_DoIP_00020]	Definition of imported datatypes of module DoIP
[SWS_DoIP_00045]	Definition of optional interfaces requested by module DoIP
[SWS_DoIP_00076]	GID Synchronization Trigger
[SWS_DoIP_00115]	TCP Socket Connection Cleanup
[SWS_DoIP_00140]	
[SWS_DoIP_00142]	General Inactivity Timer
[SWS_DoIP_00241]	Socket Connection Online Handling
[SWS_DoIP_00358]	Immediate Socket Connection Termination
[SWS_DoIP_00500]	Conditional Security Event Reporting
[SWS_DoIP_00501]	Security Event Context Data Population
[SWS_DoIP_00502]	Retrieve Socket Connection Context Data
[SWS_DoIP_00503]	Context Data Retrieval Failure Handling
[SWS_DoIP_00504]	Header Validation Failure Security Event
[SWS_DoIP_00505]	Routing Activation Failure Security Event
[SWS_DoIP_00506]	Successful Routing Activation Security Event
[SWS_DoIP_00507]	Diagnostic Message Validation Failure Security Event
[SWS_DoIP_00510]	Security events for DoIP
[SWS_DoIP_00511]	Security event context data definition: SEV_DOIP_HEADER_CHECK_ FAILED
[SWS_DoIP_00512]	Security event context data definition: SEV_DOIP_ROUTING_ ACTIVATION_CHECK_FAILED
[SWS_DoIP_00513]	Security event context data definition: SEV_DOIP_ROUTING_ ACTIVATION_SUCCESS

▽

△

Number	Heading
[SWS_DoIP_00514]	Security event context data definition: SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED

Table A.2: Changed Specification Items in R26-11

A.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_DoIP_00005]	Definition of EcucBooleanParamDef DoIPVersionInfoApi
[SWS_DoIP_00027]	Definition of API function DoIP_GetVersionInfo
[SWS_DoIP_00030]	
[SWS_DoIP_00141]	
[SWS_DoIP_00143]	
[SWS_DoIP_00144]	
[SWS_DoIP_00145]	
[SWS_DoIP_00146]	
[SWS_DoIP_00159]	
[SWS_DoIP_00172]	
[SWS_DoIP_00272]	
[SWS_DoIP_00310]	

Table A.3: Deleted Specification Items in R26-11

A.2 Traceable item history of this document according to AUTOSAR Release R25-11

A.2.1 Added Specification Items in R25-11

Number	Heading
[ECUC_DoIP_00107]	Definition of EcucBooleanParamDef DoIPEnableSecurityEventReporting
[ECUC_DoIP_00108]	Definition of EcucParamConfContainerDef SecurityEventRefs
[ECUC_DoIP_00109]	Definition of EcucReferenceDef SEV_DOIP_HEADER_CHECK_FAILED
[ECUC_DoIP_00110]	Definition of EcucReferenceDef SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED
[ECUC_DoIP_00111]	Definition of EcucReferenceDef SEV_DOIP_ROUTING_ACTIVATION_SUCCESS
[ECUC_DoIP_00112]	Definition of EcucReferenceDef SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED

▽

△

Number	Heading
[SWS_DoIP_00500]	Conditional Security Event Reporting
[SWS_DoIP_00501]	Security Event Context Data Population
[SWS_DoIP_00502]	Retrieve Socket Connection Context Data
[SWS_DoIP_00503]	Context Data Retrieval Failure Handling
[SWS_DoIP_00504]	Header Validation Failure Security Event
[SWS_DoIP_00505]	Routing Activation Failure Security Event
[SWS_DoIP_00506]	Successful Routing Activation Security Event
[SWS_DoIP_00507]	Diagnostic Message Validation Failure Security Event
[SWS_DoIP_00510]	Security events for Diagnostics over IP
[SWS_DoIP_00511]	Security event context data definition: SEV_DOIP_HEADER_CHECK_FAILED
[SWS_DoIP_00512]	Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED
[SWS_DoIP_00513]	Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_SUCCESS
[SWS_DoIP_00514]	Security event context data definition: SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED

Table A.4: Added Specification Items in R25-11

A.2.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_DoIP_00002]	Definition of EcucParamConfContainerDef DoIPGeneral
[ECUC_DoIP_00056]	Definition of EcucParamConfContainerDef DoIPPduRTxPdu
[ECUC_DoIP_00069]	Definition of EcucParamConfContainerDef DoIPChannel
[ECUC_DoIP_00075]	Definition of EcucEnumerationParamDef DoIPPduType
[SWS_DoIP_00031]	Definition of callback function DoIP_SoAdTpCopyTxData
[SWS_DoIP_00032]	Definition of callback function DoIP_SoAdTpTxConfirmation
[SWS_DoIP_00033]	Definition of callback function DoIP_SoAdTpCopyRxData
[SWS_DoIP_00036]	
[SWS_DoIP_00037]	Definition of callback function DoIP_SoAdTpStartOfReception
[SWS_DoIP_00038]	Definition of callback function DoIP_SoAdTpRxIndication
[SWS_DoIP_00039]	Definition of callback function DoIP_SoConModeChg
[SWS_DoIP_00054]	Definition of ClientServerInterface CallbackGetPowerMode
[SWS_DoIP_00055]	Definition of ClientServerInterface {Name}_RoutingActivation
[SWS_DoIP_00056]	Definition of ClientServerInterface CallbackTriggerGIDSynchronization
[SWS_DoIP_00057]	Definition of ClientServerInterface CallbackGetGID

▽

△

Number	Heading
[SWS_DoIP_00162]	
[SWS_DoIP_00163]	
[SWS_DoIP_00164]	
[SWS_DoIP_00166]	
[SWS_DoIP_00167]	
[SWS_DoIP_00169]	
[SWS_DoIP_00170]	
[SWS_DoIP_00172]	
[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_00207]	
[SWS_DoIP_00246]	
[SWS_DoIP_00247]	
[SWS_DoIP_00248]	
[SWS_DoIP_00249]	
[SWS_DoIP_00250]	
[SWS_DoIP_00285]	
[SWS_DoIP_00290]	Definition of ClientServerInterface CallbackGetFurtherActionByte
[SWS_DoIP_00302]	
[SWS_DoIP_00303]	

▽

△

Number	Heading
[SWS_DoIP_00304]	
[SWS_DoIP_00305]	

Table A.5: Changed Specification Items in R25-11

A.2.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_DoIP_00018]	
[SWS_DoIP_00228]	

Table A.6: Deleted Specification Items in R25-11

A.3 Traceable item history of this document according to AUTOSAR Release R24-11

A.3.1 Added Specification Items in R24-11

Number	Heading
[SWS_DoIP_00333]	Used DoIP protocol version
[SWS_DoIP_91003]	Definition of API function DoIP_AllowNonTlsDoIPConnection

Table A.7: Added Specification Items in R24-11

A.3.2 Changed Specification Items in R24-11

none

A.3.3 Deleted Specification Items in R24-11

none

A.4 Traceable item history of this document according to AUTOSAR Release R23-11

A.4.1 Added Specification Items in R23-11

Number	Heading
[SWS_DoIP_00321]	
[SWS_DoIP_00322]	
[SWS_DoIP_00323]	
[SWS_DoIP_00324]	
[SWS_DoIP_00325]	
[SWS_DoIP_00326]	
[SWS_DoIP_00327]	
[SWS_DoIP_00330]	
[SWS_DoIP_00331]	
[SWS_DoIP_00332]	

Table A.8: Added Specification Items in R23-11

A.4.2 Changed Specification Items in R23-11

Number	Heading
[SWS_DoIP_00016]	
[SWS_DoIP_00017]	
[SWS_DoIP_00019]	
[SWS_DoIP_00077]	
[SWS_DoIP_00078]	
[SWS_DoIP_00110]	
[SWS_DoIP_00113]	
[SWS_DoIP_00114]	
[SWS_DoIP_00141]	

Table A.9: Changed Specification Items in R23-11

A.4.3 Deleted Specification Items in R23-11

none

A.5 Traceable item history of this document according to AUTOSAR Release R22-11

A.5.1 Added Specification Items in R22-11

none

A.5.2 Changed Specification Items in R22-11

Number	Heading
[SWS_DoIP_00020]	
[SWS_DoIP_00022]	
[SWS_DoIP_00023]	
[SWS_DoIP_00024]	
[SWS_DoIP_00025]	
[SWS_DoIP_00026]	
[SWS_DoIP_00027]	
[SWS_DoIP_00031]	
[SWS_DoIP_00032]	
[SWS_DoIP_00033]	
[SWS_DoIP_00037]	
[SWS_DoIP_00038]	
[SWS_DoIP_00039]	
[SWS_DoIP_00040]	
[SWS_DoIP_00041]	
[SWS_DoIP_00044]	
[SWS_DoIP_00045]	
[SWS_DoIP_00047]	
[SWS_DoIP_00048]	
[SWS_DoIP_00049]	
[SWS_DoIP_00050]	
[SWS_DoIP_00051]	
[SWS_DoIP_00054]	
[SWS_DoIP_00055]	
[SWS_DoIP_00056]	
[SWS_DoIP_00057]	
[SWS_DoIP_00148]	
[SWS_DoIP_00244]	
[SWS_DoIP_00245]	
[SWS_DoIP_00261]	

▽

△

Number	Heading
[SWS_DoIP_00262]	
[SWS_DoIP_00263]	
[SWS_DoIP_00264]	
[SWS_DoIP_00266]	
[SWS_DoIP_00267]	
[SWS_DoIP_00268]	
[SWS_DoIP_00269]	
[SWS_DoIP_00270]	
[SWS_DoIP_00277]	
[SWS_DoIP_00278]	
[SWS_DoIP_00287]	
[SWS_DoIP_00288]	
[SWS_DoIP_00289]	
[SWS_DoIP_00290]	
[SWS_DoIP_91000]	
[SWS_DoIP_91002]	

Table A.10: Changed Specification Items in R22-11

A.5.3 Deleted Specification Items in R22-11

none

A.5.4 Added Constraints in R22-11

none

A.5.5 Changed Constraints in R22-11

none

A.5.6 Deleted Constraints in R22-11

none