Document Title	Specification of UDP Network Management
Document Category	Software specification (SWS)
Document UID	414
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	• Removal of version info API • Removal of RS NM • Simplification of configuration classes and variants • Bug fixes and editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Bug fixes and editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• NM harmonization • Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Fixes for Partial Networking and PNC Shutdown • Removal of obsolete requirements • Bug fixes and editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Added handling of internal requested Pnc • Improved synchronized Pnc shutdown • NM PDU filter algorithm and aggregation of internal and external requested partial networks is now obsolete and replaced • Traceability directly to RS_Nm
2020-11-30	R20-11	AUTOSAR Release Management	• Updates for CONC 641 VSNM • Updates for Light CONC 685 • Minor changes

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Det error handling corrected • Harmonization of API • Minor corrections • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Header file cleanup • Minor corrections
2017-12-08	4.3.1	AUTOSAR Release Management	• Node Detection Configuration per channel • Det error handling corrected • Bug fixes and editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Added Trigger Transmit feature • Car Wakeup support completed • Immediate TX Transmission corrected • Editorial changes
2015-07-31	4.2.2	AUTOSAR Release Management	• Revised Error Classification • Added support for Car Wakeup • Bug fixes and editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Harmonization of API description • Revised Partial Networking Requirements • Extended Production Errors • Editorial Changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Minor bug fixes • Editorial Changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Revised Spontaneous Transmission • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Added support for Partial Networking • Added updated production errors • Editorial changes
2011-12-22	4.0.3	AUTOSAR Administration	• Support coordinated shutdown • New traceability mechanism

▽

△

2010-09-30	3.1.5	AUTOSAR Administration	• ComStack Harmonization • Harmonization of NM Interfaces
2010-02-02	3.1.4	AUTOSAR Administration	• 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 specification . . . 11
4 Constraints and assumptions 12
4.1 Limitations . . . 12
4.2 Applicability to car domains . . . 12
5 Dependencies to other modules 13
5.1 File Structure . . . 13
5.1.1 Code File Structure . . . 13
5.2 Protocol layer dependencies . . . 14
6 Requirements Tracing 15
7 Functional specification 17
7.1 Coordination algorithm . . . 17
7.2 Operational Modes . . . 17
7.2.1 Network Mode . . . 18
7.2.1.1 Repeat Message State . . . 19
7.2.1.2 Normal Operation State . . . 20
7.2.1.3 Ready Sleep State . . . 20
7.2.2 Prepare Bus-Sleep Mode . . . 21
7.2.3 Bus-Sleep Mode . . . 22
7.3 Network states . . . 23
7.4 Initialization . . . 23
7.5 Execution . . . 24
7.5.1 Processor architecture . . . 24
7.5.2 Timing parameters . . . 24
7.6 Communication Scheduling . . . 25
7.6.1 NM Message Transmission . . . 25
7.6.2 NM Message Reception . . . 27
7.7 Additional features . . . 27
7.7.1 Detection of Remote Sleep Indication (optional) . . . 27
7.7.2 User Data (optional) . . . 28
7.7.3 Passive Mode (optional) . . . 29
7.7.4 State change notification (optional) . . . 30
7.7.5 Communication Control (optional) . . . 30
7.7.6 NM Coordinator synchronization support (optional) . . . 31
7.8 Partial Networking . . . 31

7.8.1 Rx Handling of NM PDUs . . . 31
7.8.2 Tx Handling of NM PDUs . . . 34
7.8.3 Handling of Internal Requested Partial Network Clusters . . . 35
7.8.4 Spontaneous Transmission of NM-PDUs via UdpNm_NetworkRequest . . . 35
7.9 Payload (PDU) Structure . . . 36
7.10 Functional requirements on UdpNm API . . . 37
7.11 Car Wakeup . . . 37
7.12 Error Classification . . . 38
7.12.1 Development Errors . . . 39
7.12.2 Runtime Errors . . . 39
7.12.3 Production Errors . . . 39
7.13 Scheduling of the main function . . . 40
7.14 Application notes . . . 40
7.14.1 Wakeup notification . . . 40
7.14.2 Coordination of coupled networks . . . 40
7.15 Parameter check . . . 40
7.16 Security Events . . . 41
8 API specification 42
8.1 Imported types . . . 43
8.2 Type definitions . . . 43
8.2.1 UdpNm_ConfigType . . . 43
8.2.2 UdpNm_PduPositionType . . . 44
8.3 Function definitions . . . 44
8.3.1 UdpNm_Init . . . 45
8.3.2 UdpNm_PassiveStartUp . . . 45
8.3.3 UdpNm_NetworkRequest . . . 46
8.3.4 UdpNm_NetworkRelease . . . 46
8.3.5 UdpNm_DisableCommunication . . . 47
8.3.6 UdpNm_EnableCommunication . . . 48
8.3.7 UdpNm_SetUserData . . . 48
8.3.8 UdpNm_GetUserData . . . 49
8.3.9 UdpNm_GetNodeIdentifier . . . 50
8.3.10 UdpNm_GetLocalNodeIdentifier . . . 50
8.3.11 UdpNm_RepeatMessageRequest . . . 51
8.3.12 UdpNm_GetPduData . . . 52
8.3.13 UdpNm_GetState . . . 52
8.3.14 UdpNm_RequestBusSynchronization . . . 53
8.3.15 UdpNm_CheckRemoteSleepIndication . . . 54
8.3.16 UdpNm_SetSleepReadyBit . . . 54
8.3.17 UdpNm_Transmit . . . 55
8.3.18 UdpNm_PnLearningRequest . . . 55
8.3.19 UdpNm_ActivateTxPnShutdownMsg . . . 56

8.3.20 UdpNm_DeactivateTxPnShutdownMsg . . . 57
8.3.21 UdpNm_SoAdIfTxConfirmation . . . 58
8.3.22 UdpNm_SoAdIfRxIndication . . . 58
8.3.23 UdpNm_SoAdIfTriggerTransmit . . . 59
8.4 Scheduled functions . . . 60
8.4.1 UdpNm_MainFunction_<Instance Id> . . . 60
8.5 Expected interfaces . . . 61
8.5.1 Mandatory interfaces . . . 61
8.5.2 Optional interfaces . . . 61
8.5.3 Configurable interfaces . . . 63
8.6 Service Interfaces . . . 63
8.7 UML State chart diagram . . . 63
9 Sequence diagrams and Transition Tables 65
9.1 UdpNmTransmission . . . 65
9.2 UdpNm Reception . . . 65
10 Configuration specification 66
10.1 How to read this chapter . . . 66
10.2 Containers and configuration parameters . . . 66
10.2.1 UdpNm . . . 66
10.2.2 UdpNmGlobalConfig . . . 67
10.2.3 UdpNmChannelConfig . . . 73
10.2.4 UdpNmRxPdu . . . 89
10.2.5 UdpNmTxPdu . . . 90
10.2.6 UdpNmUserDataTxPdu . . . 91
10.3 Published Information . . . 92
A Not applicable requirements 93
B Change history of AUTOSAR traceable items 94
B.1 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 94
B.1.1 Added Advisories in R22-11 . . . 94
B.1.2 Changed Advisories in R22-11 . . . 94
B.1.3 Deleted Advisories in R22-11 . . . 94
B.1.4 Added Constraints in R22-11 . . . 94
B.1.5 Changed Constraints in R22-11 . . . 94
B.1.6 Deleted Constraints in R22-11 . . . 94
B.1.7 Added Specification Items in R22-11 . . . 95
B.1.8 Changed Specification Items in R22-11 . . . 105
B.1.9 Deleted Specification Items in R22-11 . . . 105
B.2 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 105
B.2.1 Added Specification Items in R23-11 . . . 105
B.2.2 Changed Specification Items in R23-11 . . . 105

B.2.3 Deleted Specification Items in R23-11 . . . 106
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 106
B.3.1 Added Specification Items in R24-11 . . . 106
B.3.2 Changed Specification Items in R24-11 . . . 106
B.3.3 Deleted Specification Items in R24-11 . . . 106
B.4 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 106
B.4.1 Added Specification Items in R25-11 . . . 106
B.4.2 Changed Specification Items in R25-11 . . . 107
B.4.3 Deleted Specification Items in R25-11 . . . 107
B.5 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 107
B.5.1 Added Specification Items in R26-11 . . . 107
B.5.2 Changed Specification Items in R26-11 . . . 108
B.5.3 Deleted Specification Items in R26-11 . . . 110

1 Introduction and functional overview

This document describes the concept, core functionality, optional features, interfaces and configuration issues of the AUTOSAR UDP Network Management (UdpNm). UdpNm is intended to be an optional feature. It is intended to work together with a TCP/IP Stack, independent of the physical layer of the communication system used. The AUTOSAR UDP Network Management is a hardware independent protocol that can be used on TCP/IP based systems (for limitations refer to chapter 4.1 “Limitations”). Its main purpose is to coordinate the transition between normal operation and bus-sleep mode of the network.

In addition to the core functionality optional features are provided e.g. to implement a service to detect all present nodes or to detect if all other nodes are ready to sleep. The UDP Network Management (UdpNm) function provides an adaptation between Network Management Interface (Nm) and a TCP/IP Stack (TCP/IP). For a general understanding of the AUTOSAR Network Management functionality please refer to [1, Specification of Network Management Interface].

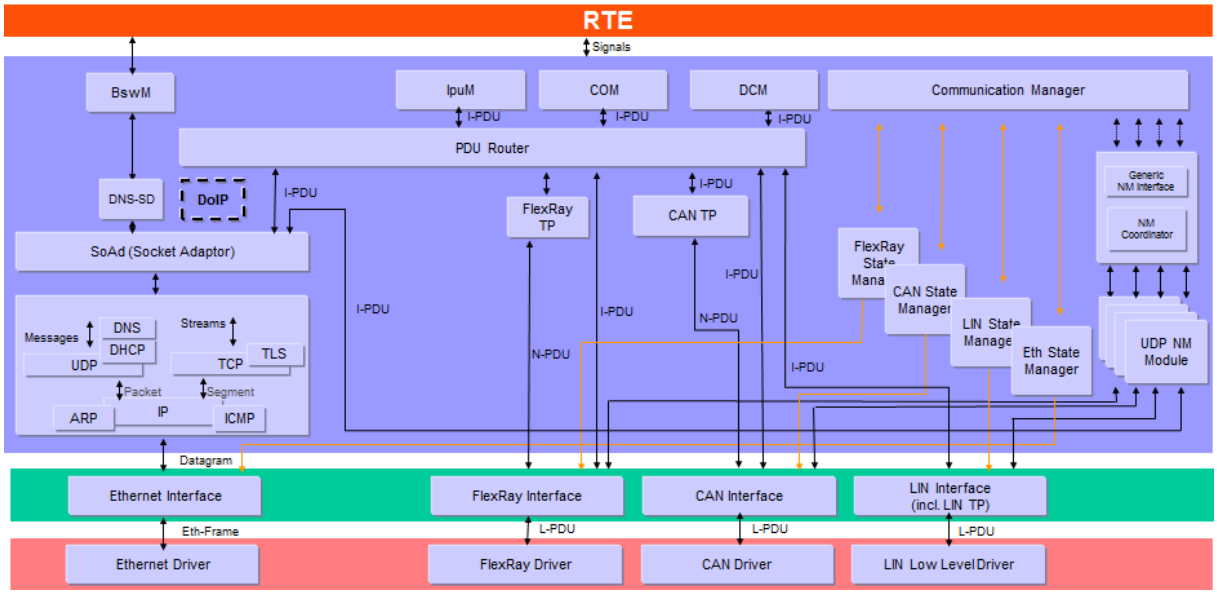

Figure 1.1: Extended AUTOSAR Communication Stack.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the UdpNm module that are not included in the [2, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
CWU	Car Wakeup
EthIf	Ethernet Interface
PNI	Partial Network Information
PNL	Partial Network Learning
TCP/IP	A family of communication protocols used in computer networks

Table 2.1: Acronyms and abbreviations used in the scope of this Document

Term:	Description:
PDU transmission ability is disabled	This means that the NM message transmission has been disabled by the optional service `UdpNm_DisableCommunication`.
Repeat Message Request Bit Indication	UdpNm_SoAdIfRxIndication finds the Repeat Message Bit set in the Control Bit Vector of a received NM message.
NM PDU	Refers to the payload transmitted in a packet. It contains the NM User Data as well as the Control Bit Vector and the Source Node Identifier.
NM Packet	Refers to an Ethernet Frame containing an IP as well as a UDP header in addition to the data (PDU) transmitted by the NM in the payload section.
NM Message	Most abstract term referring to any single information item transferred within the methodology of the NM algorithm.
Bus-Off state	Refers to a situation where no cable is connected to the Ethernet HW.
Top-level PNC coordinator	An ECU acts as top-level PNC coordinator for those PNCs which are actively coordinated on all assigned channels. This ECU has the PNC gateway functionality enabled. The top-level PNC coordinator triggers for those PNCs a synchronized PNC shutdown, if no other ECU in the network requests them and if the synchronized PNC shutdown is enabled. Note: For different PNCs it is possible to have different top-level PNC coordinators.
Intermediate PNC coordinator	An ECU acts as intermediate PNC coordinator for those PNCs which are passively coordinated on at least one channel. This ECU has the PNC gateway functionality enabled. The intermediate PNC coordinator forwards a synchronized PNC shutdown to active coordinated channels for PNCs which are passively coordinated, if the synchronized PNC shutdown is enabled
PNC leaf node	A PNC leaf node is an ECU that acts not as a PNC coordinator at all in the network. It processes PN shutdown message as usual NM messages.
PN shutdown message	A top-level PNC coordinator transmit PN shutdown messages to indicate a synchronized PNC shutdown across the PN topology. A PN shutdown message is as NM message which has PNSR bit in the control bit vector and all PNCs which are indicated for a synchronized shutdown set to '1'.

Table 2.2: Terms used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Specification of Network Management Interface
AUTOSAR_CP_SWS_NetworkManagementInterface

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[4] Specification of the AUTOSAR Network Management Protocol
AUTOSAR_FO_PRS_NetworkManagementProtocol

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] Specification of Communication Manager
AUTOSAR_CP_SWS_COMManager

[7] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[8] Guide to Mode Management
AUTOSAR_CP_EXP_ModeManagementGuide

[9] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for UDP Network Management.

Thus, the specification SWS BSW General shall be considered as additional and required specification for UDP Network Management.

4 Constraints and assumptions

4.1 Limitations

1. One instance of UdpNm is associated with only one NM-Cluster in one network. One NM-Cluster can have only one instance of UdpNm in one node.
2. One instance of UdpNm is associated with only one network within the same ECU.
3. UdpNm is only applicable for TCP/IP based systems.

Figure 4.1 presents an AUTOSAR NM stack within an example ECU belonging to two UDP NM-clusters.

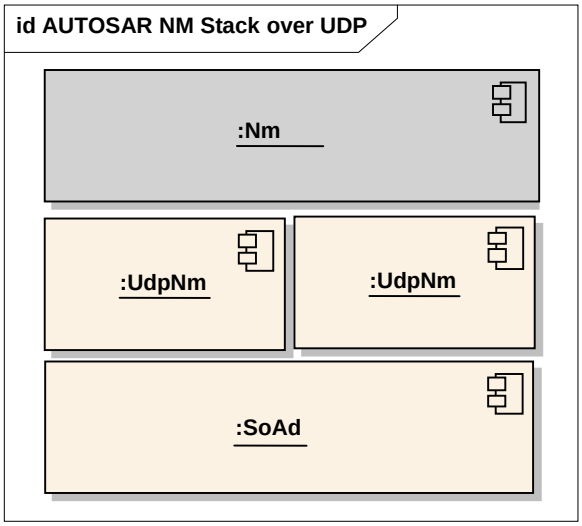

Figure 4.1: AUTOSAR NM stack within an example ECU belonging to two UDP NM-clusters

4.2 Applicability to car domains

N/A

5 Dependencies to other modules

UDP Network Management (UdpNm) uses services of the TCP/IP Stack and provides services to the Generic Network Management Interface (Nm).

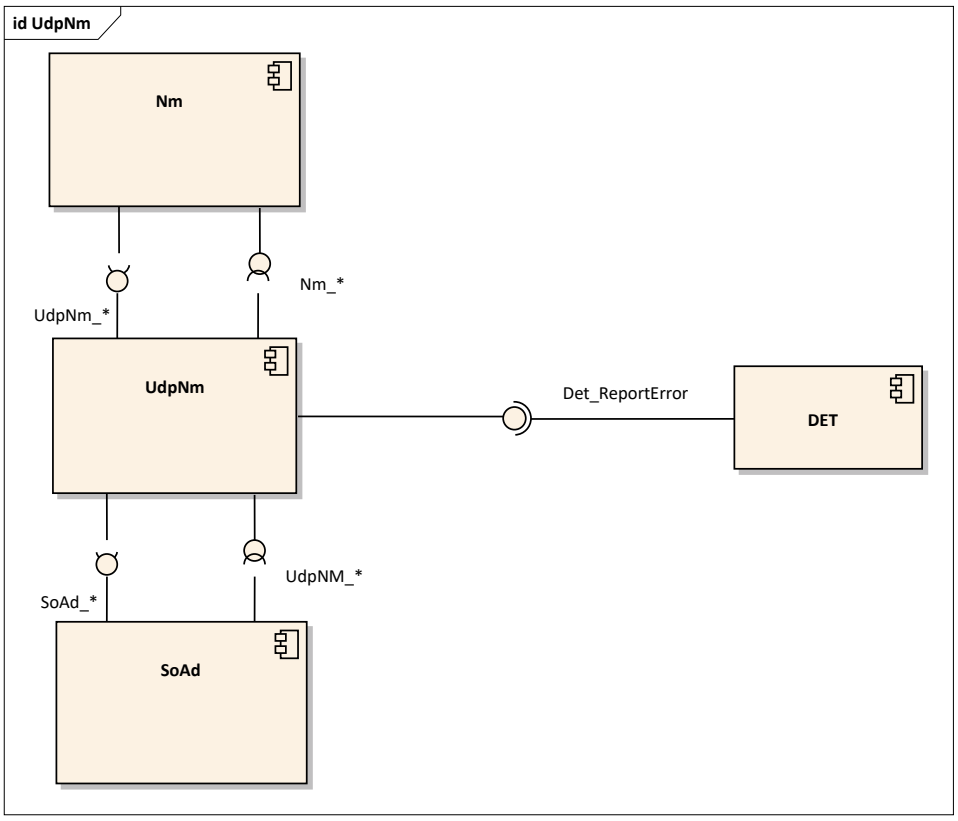

Figure 5.1: Dependencies on other modules.

5.1 File Structure

5.1.1 Code File Structure

[SWS_UdpNm_00081]

Upstream requirements: SRS_BSW_00419, SRS_BSW_00346, SRS_BSW_00308

⌈The code file structure shall not be fully defined within this specification. However, the code file structure shall include the following files:

- `UdpNm_Lcfg.c` (for link time configurable parameters)
- `UdpNm_PBcfg.c` (for post build time configurable parameters)

These files shall contain all link time post build time configurable parameters.⌋

5.2 Protocol layer dependencies

The Udp Network Management is based on the protocol mentioned in PRS NetworkManagementProtocol [4, Specification of the AUTOSAR Network Management Protocol].

6 Requirements Tracing

The following tables reference the requirements specified in [5] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_UdpNm_00208]
[SRS_BSW_00308]	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	[SWS_UdpNm_00081]
[SRS_BSW_00310]	API naming convention	[SWS_UdpNm_00309]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_UdpNm_00192] [SWS_UdpNm_00197] [SWS_UdpNm_00198] [SWS_UdpNm_00199] [SWS_UdpNm_00244] [SWS_UdpNm_00314] [SWS_UdpNm_00463] [SWS_UdpNm_00492]
[SRS_BSW_00337]	Classification of development errors	[SWS_UdpNm_00189] [SWS_UdpNm_00190] [SWS_UdpNm_00192] [SWS_UdpNm_00196] [SWS_UdpNm_00197] [SWS_UdpNm_00198] [SWS_UdpNm_00199] [SWS_UdpNm_00314] [SWS_UdpNm_00463] [SWS_UdpNm_00492]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_UdpNm_00244]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_UdpNm_00208]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_UdpNm_00081]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_UdpNm_00208]
[SRS_BSW_00385]	List possible error notifications	[SWS_UdpNm_00465]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_UdpNm_00026] [SWS_UdpNm_00208]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_UdpNm_00208] [SWS_UdpNm_00308]
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as `const` it should be placed into a separate c-file	[SWS_UdpNm_00081]
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[SWS_UdpNm_00234]
[SRS_BSW_00425]	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	[SWS_UdpNm_00234]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_UdpNm_00465]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_UdpNm_00309]
[SRS_BSW_00460]	Reentrancy Levels	[SWS_UdpNm_00309]



△

Requirement	Description	Satisfied by
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_UdpNm_00309]
[SRS_BSW_00484]	Input parameters of scalar and enum types shall be passed as a value.	[SWS_UdpNm_00309]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Coordination algorithm

The AUTOSAR UdpNm is based on decentralized direct network management strategy, which means that every network node performs activities self-sufficient depending only on the UDP packets received and/or transmitted within the communication system.

The AUTOSAR UdpNm coordination algorithm is based on periodic NM packets, which are received by all nodes in the cluster via multicast transmission. Reception of NM packets indicates that sending nodes want to keep the NM-cluster awake. If any node is ready to go to the Bus-Sleep Mode, it stops sending NM packets, but as long as NM packets from other nodes are received, it postpones transition to the Bus-Sleep Mode. Finally, if a dedicated timer elapses because no NM packets are received anymore, every node initiates transition to the Bus-Sleep Mode. If any node in the NM-cluster requires bus-communication, it can keep the NM-cluster awake by transmitting NM packets. For more details concerning the wakeup procedure itself, please refer to [6] Chapter 7.4 *"Extended functionality"* ff. or [7] Chapter 8.3.4 *"Wakeup Handling"*.

The main concept of the AUTOSAR UdpNm coordination algorithm can be defined by the following two key-requirements:

[SWS_UdpNm_00088] ⌈The parameter `UdpNmStayInPbsEnabled` shall match parameter `NmStayInPbsEnabled` from the [PRS_Nm_00506] specification.⌋

Note: [PRS_Nm_00506] implicitly contains that if `UdpNmStayInPbsEnabled` is enabled UdpNm will never be left due to a timeout, i.e. UdpNm will stay in Prepare Bus-Sleep Mode until either ECU goes to Power Off or any restart reason applies.

The overall state machine of the AUTOSAR UdpNm coordination algorithm can be defined as follows:

[SWS_UdpNm_00089] ⌈The AUTOSAR UdpNm state machine shall contain states, transitions and triggers required for the AUTOSAR UdpNm coordination algorithm as seen from the point of view of one single node in the NM cluster.⌋

Note: A UML state chart of the AUTOSAR UdpNm state machine from the point of view of one single node in the NM cluster can be found in the API specifications chapter 8 " API specification"

7.2 Operational Modes

This chapter describes the operational modes of the AUTOSAR UdpNm coordination algorithm.

[SWS_UdpNm_00092] ⌈The AUTOSAR UdpNm shall contain three operational modes visible at the modules interface:

- Network Mode
- Prepare Bus-Sleep Mode
- Bus-Sleep Mode

⌋

[SWS_UdpNm_00093] ⌈Changes of the AUTOSAR UdpNm operational modes shall be signalled to the upper layer by means of call-back functions.⌋

7.2.1 Network Mode

[SWS_UdpNm_00094] ⌈The Network Mode shall consist of three internal states:

- Repeat Message State
- Normal Operation State
- Ready Sleep State

⌋

[SWS_UdpNm_00095] ⌈When the Network Mode is entered from Bus-Sleep Mode or Prepare Bus-Sleep Mode, by default, the Repeat Message State shall be entered.⌋

[SWS_UdpNm_00096] ⌈When the Network Mode is entered, the NM-Timeout Timer shall be started.⌋

[SWS_UdpNm_00097] ⌈When the Network Mode is entered, the UdpNm shall notify the upper layer by calling `Nm_NetworkMode`.⌋

[SWS_UdpNm_00098] ⌈Upon successful reception of an NM PDU (call of `UdpNm_SoAdIfRxIndication`) in Network Mode, the NM-Timeout Timer shall be restarted.⌋

[SWS_UdpNm_00099] ⌈Upon transmission of an NM PDU (call of `UdpNm_SoAdIfTxConfirmation` with `E_OK`) in the Network Mode, the NM-Timeout Timer shall be restarted.⌋

Note: As no transmission confirmation is available from the SoAd or the TCP/IP stack it is assumed that each Network Management PDU transmission request results in a successful Network Management PDU transmission.

[SWS_UdpNm_00206] ⌈The NM-Timeout Timer shall be reset every time it is started or restarted.⌋

[SWS_UdpNm_00468] ⌈If function `UdpNm_PnLearningRequest` is called on a channel where `UdpNmDynamicPncToChannelMappingEnabled` is set to `TRUE` and

UdpNm is in the Network Mode the UdpNm module shall set the Repeat Message Bit and the Partial Network Learning Bit in the CBV to 1 on this channel and change to or restart the Repeat Message State.⌋

[SWS_UdpNm_00469] ⌈If the bits Partial Network Learning and Repeat Message Request both are received with value 1 on a channel where UdpNmDynamicPncToChannelMappingEnabled is set to TRUE and UdpNm is in the Network Mode the UdpNm module shall set the Partial Network Learning Bit in the CBV to 1 on this channel and change to or restart the Repeat Message State.⌋

Note: Restart in [SWS_UdpNm_00468] or [SWS_UdpNm_00469] means that UdpNm is already in Repeat Message State and then a complete re-entry of the Repeat Message State has to be performed once.

7.2.1.1 Repeat Message State

For nodes that are not in passive mode (refer to chapter 7.7.3 "Passive Mode (optional)") the Repeat Message State ensures, that any transition from Bus-Sleep or Prepare Bus-Sleep to the Network Mode becomes visible for the other nodes on the network. Additionally it ensures that any node stays active for a minimum amount of time (UdpNmRepeatMessageTime). Optionally it can be used for detection of present nodes.

[SWS_UdpNm_00100] ⌈When the Repeat Message State is entered from Bus-Sleep Mode, Prepare-Bus-Sleep Mode, Normal Operation State or Ready Sleep State transmission of NM packets shall be (re-) started unless passive mode is enabled.⌋

[SWS_UdpNm_00101] ⌈When the NM-Timeout Timer expires in the Repeat Message State, the NM-Timeout Timer shall be restarted.⌋

[SWS_UdpNm_00102] ⌈The NM shall stay in the Repeat Message State for a configurable amount of time determined by the UdpNmRepeatMessageTime (configuration parameter); after that time the Repeat Message State shall be left.⌋

[SWS_UdpNm_00103] ⌈When Repeat Message State is left, the Normal Operation State shall be entered, if the network has been requested (see [SWS_UdpNm_00104]).⌋

[SWS_UdpNm_00106] ⌈When Repeat Message State is left, the Ready Sleep State shall be entered, if the network has been released (see [SWS_UdpNm_00105]).⌋

[SWS_UdpNm_00107] ⌈If UdpNmNodeDetectionEnabled is set to TRUE UdpNm shall clear the Repeat Message Bit when leaving Repeat Message State.⌋

[SWS_UdpNm_00470] ⌈If UdpNmDynamicPncToChannelMappingEnabled is set to TRUE UdpNm shall clear the Partial Network Learning Bit when leaving the Repeat Message State.⌋

7.2.1.2 Normal Operation State

The Normal Operation State ensures that any node can keep the NM-cluster awake as long as the network functionality is required.

[SWS_UdpNm_00116] ⌈When the Normal Operation State is entered from Ready Sleep State, transmission of NM PDUs shall be started unless passive mode is enabled or the NM message transmission ability has been disabled.⌋

[SWS_UdpNm_00117] ⌈When the NM-Timeout Timer expires in the Normal Operation State, the NM-Timeout Timer shall be restarted.⌋

[SWS_UdpNm_00118] ⌈When the network is released and the current state is Normal Operation State, the Normal Operation State shall be left and the Ready Sleep state shall be entered (refer to [SWS_UdpNm_00105]).⌋

[SWS_UdpNm_00119] ⌈If `UdpNmNodeDetectionEnabled` is set to `TRUE` and Repeat Message Request bit is received in the Normal Operation State, UdpNm shall enter Repeat Message State.⌋

[SWS_UdpNm_00120] ⌈If `UdpNmNodeDetectionEnabled` is set to `TRUE` and function `UdpNm_RepeatMessageRequest` is called in the Normal Operation State, UdpNm shall enter Repeat Message State.⌋

[SWS_UdpNm_00121] ⌈If `UdpNmNodeDetectionEnabled` is set to `TRUE` and function `UdpNm_RepeatMessageRequest` is called in the Normal Operation State, UdpNm shall set the Repeat Message Bit.⌋

7.2.1.3 Ready Sleep State

The Ready Sleep State ensures that any node in the NM-cluster waits with transition to the Prepare Bus-Sleep Mode as long as any other node keeps the NM-cluster awake.

[SWS_UdpNm_00108] ⌈When the Ready Sleep State is entered from Repeat Message State or Normal Operation State, transmission of NM PDUs shall be stopped.⌋

Note: If passive mode is enabled no NM PDUs are transmited, no action is required. If passive mode is disabled, in some cases NM PDUs have to be transmitted in Ready Sleep State to grant a synchronized shutdown in the network, e.g. re-transmission of PN shutdown messages.

[SWS_UdpNm_00109] ⌈When the NM-Timeout Timer expires in the Ready Sleep State, the Ready Sleep State shall be left and the Prepare Bus-Sleep Mode shall be entered.⌋

[SWS_UdpNm_00110] ⌈When the network is requested and the current state is the Ready Sleep State, the Ready Sleep State shall be left and the Normal Operation State shall be entered (refer to SWS_UdpNm_00104).⌋

[SWS_UdpNm_00111] ⌈If `UdpNmNodeDetectionEnabled` is set to `TRUE` and Repeat Message Request bit is received in the Ready Sleep State, UdpNm shall enter Repeat Message State.⌋

[SWS_UdpNm_00112] ⌈If `UdpNmNodeDetectionEnabled` is set to `TRUE` and function `UdpNm_RepeatMessageRequest` is called in the Ready Sleep State, UdpNm shall enter Repeat Message State.⌋

[SWS_UdpNm_00113] ⌈If `UdpNmNodeDetectionEnabled` is set to `TRUE` and function `UdpNm_RepeatMessageRequest` is called in the Ready Sleep State, UdpNm shall set the Repeat Message Bit.⌋

7.2.2 Prepare Bus-Sleep Mode

The purpose of the Prepare Bus Sleep state is to ensure that all nodes have time to stop their network activity before the Bus Sleep state is entered. Bus activity is calmed down (i.e. queued messages are transmitted in order to empty all Tx-buffers) and finally there is no activity on the bus in the Prepare Bus-Sleep Mode.

[SWS_UdpNm_00114] ⌈When Prepare Bus-Sleep Mode is entered, the UdpNm shall notify the upper layer by calling `Nm_PrepareBusSleepMode`.⌋

[SWS_UdpNm_00124] ⌈Upon successful reception of an NM PDU in the Prepare Bus-Sleep Mode, the Prepare Bus-Sleep Mode shall be left and the Network Mode shall be entered; by default the Repeat Message State is entered (refer to [SWS_UdpNm_00095]).⌋

[SWS_UdpNm_00123] ⌈When the network is requested in the Prepare Bus-Sleep Mode, the Prepare Bus-Sleep Mode shall be left and the Network Mode shall be entered; by default the Repeat Message State is entered (refer to [SWS_UdpNm_00095])⌋

[SWS_UdpNm_00122] ⌈When the network has been requested (see [SWS_UdpNm_00104]) in the Prepare Bus-Sleep Mode and the UdpNm module has entered Network Mode and if `UdpNmImmediateRestartEnabled` (configuration parameter) is `TRUE`, the UdpNm module shall transmit a Network Management PDU.⌋

Rationale: Other nodes in the cluster are still in Prepare Bus-Sleep Mode; in the exceptional situation described above transition into the Bus-Sleep Mode shall be avoided and bus-communication shall be restored as fast as possible.

Caused by the transmission offset for Network Management PDUs in UdpNm, the transmission of the first Network Management PDU in Repeat Message State can be

delayed significantly. In order to avoid a delayed re-start of the network the transmission of a Network Management PDU can be requested immediately.

Note: If `UdpNmImmediateRestartEnabled` is `TRUE` and a wake-up line is used, a burst of Network Management PDUs occurs if all network nodes get a network request in Prepare Bus-Sleep Mode.

7.2.3 Bus-Sleep Mode

The purpose of the Bus-Sleep state is to reduce power consumption in the node, when no messages are to be exchanged.

The communication controller is switched to sleep mode, respective wakeup mechanisms are activated and finally power consumption is reduced to the adequate level in the Bus-Sleep Mode.

If `UdpNmStayInPbsEnabled` is disabled and configurable amount of time determined by the `UdpNmTimeoutTime` + `UdpNmWaitBusSleepTime` (both configuration parameters) is identically configured for all nodes in the network management cluster, all nodes in the network management cluster that are coordinated with use of the AUTOSAR NM algorithm perform the transition into the Bus-Sleep Mode at approximately the same time.

Note: The parameters `UdpNmTimeoutTime` and `UdpNmWaitBusSleepTime` should have the same values within all network nodes of the NM-cluster. Depending on the specific implementation, transition into the Bus-Sleep Mode takes place approximately at the same time. The time jitter experienced for this transition depends on the following factors:

- internal clock precision (oscillator's drift),
- NM-task cycle time (if tasks are not synchronized with a global time),
- NM PDUs waiting time in the Tx-queue (if transmission confirmation is made immediately after transmit request).

For a best case estimation only oscillator drift should be taken into account for a configurable amount of time determined by the value `UdpNmTimeoutTime` + `UdpNmWaitBusSleepTime` (both configuration parameters).

[SWS_UdpNm_00126] ⌈When Bus-Sleep Mode is entered, the UdpNm shall notify the upper layer by calling `Nm_BusSleepMode`; this shall not be the case if Bus-Sleep Mode is entered by default at initialization.⌋

[SWS_UdpNm_00127] ⌈When the UdpNm module receives successfully Network Management PDU in the Bus-Sleep Mode (call of `UdpNm_SoAdIfRxIndication`), the UdpNm module shall notify the upper layer by calling the callback function `Nm_NetworkStartIndication`.⌋

Rationale: To avoid race conditions and state inconsistencys between Network and Mode Management, UdpNm will not automatically perform the transition from Bus-Sleep Mode to Network Mode. UdpNm will only inform the upper layers which have to make the wake-up decision. NM packet reception in Bus-Sleep Mode must be handled depending on the current state of the ECU shutdown or startup process.

[SWS_UdpNm_00128] ⌈If `UdpNm_PassiveStartUp` is called in the Bus-Sleep Mode or Prepare Bus Sleep Mode, the UdpNm module shall enter the Network Mode; by default the Repeat Message State is entered (refer to [SWS_UdpNm_00095] and [SWS_UdpNm_00104]).⌋

Note: In the Prepare Bus-Sleep Mode and Bus-Sleep Mode is assumed that the network is released, unless bus communication is explicitly requested.

[SWS_UdpNm_00129] ⌈When the network is requested in Bus-Sleep Mode, the Udp Nm module shall enter the Network Mode; by default the UdpNm module shall enter the Repeat Message State (refer to SWS_UdpNm_00095 and SWS_UdpNm_00104).⌋

7.3 Network states

Network states (i.e. 'requested' and 'released') are two additional states of the AUTOSAR UdpNm state machine that exist in parallel to the state machine. Network states denote, whether the software components need to communicate on the bus (the network state is then 'requested'); or whether the software components don't have to communicate on the bus (the bus network state is then 'released'); note that if the network is released an ECU may still communicate because some other ECU still request the network.

[SWS_UdpNm_00104] ⌈The function call `UdpNm_NetworkRequest` shall request the network. I.e. the UdpNm module shall change network state to 'requested'.⌋

[SWS_UdpNm_00105] ⌈The function call `UdpNm_NetworkRelease` shall release the network. I.e. the UdpNm module shall change network state to 'released'.⌋

7.4 Initialization

[SWS_UdpNm_00141] ⌈After successful initialization the Network Management state shall be set to BusSleep Mode.⌋

Note: The UdpNm module should be initialized after SoAd is initialized and before any other network management service is called.

[SWS_UdpNm_00143] ⌈When initialized, by default, the UdpNm module shall set the network state to 'released'.⌋

[SWS_UdpNm_00144] ⌈When initialized, by default, the UdpNm module shall enter the Bus-Sleep Mode.⌋

[SWS_UdpNm_00060] ⌈The function `UdpNm_Init` shall select the active configuration set by means of a configuration pointer parameter being passed (see [SWS_UdpNm_00208]).⌋

[SWS_UdpNm_00033] ⌈After initialization the transmission of NM messages shall be stopped.⌋

[SWS_UdpNm_00025] ⌈After initialization each byte of the user data bytes shall be set to `0xFF`.⌋

[SWS_UdpNm_00085] ⌈After initialization the `Control Bit Vector` shall be set to `0x00`.⌋

[SWS_UdpNm_00485] ⌈During initialization and if `UdpNmPnEnabled` is `TRUE`, the UdpNm module shall set each byte of the PNC Request Bit Vector to `0x00`.⌋

[SWS_UdpNm_00496] ⌈UdpNmSynchronizedPncShutdownEnabled is set to `TRUE`, the UdpNm module shall consider transmission of PN shutdown message as inactive after initialization.⌋

[SWS_UdpNm_00148] ⌈All instances of UDP NM on different ECUs in one NM cluster shall use the same UDP receive port.⌋

7.5 Execution

7.5.1 Processor architecture

[SWS_UdpNm_00146] ⌈The AUTOSAR UdpNm coordination algorithm shall be processor independent, meaning it shall not rely on any processor specific hardware support and thus shall be realizable on any processor architecture that is within the scope of AUTOSAR.⌋

7.5.2 Timing parameters

[SWS_UdpNm_00246] ⌈The configuration parameter `UdpNmTimeoutTime` shall determine the AUTOSAR UdpNm timing parameter NM-Timeout Time.⌋

[SWS_UdpNm_00247] ⌈The configuration parameter `UdpNmRepeatMessageTime` shall determine the AUTOSAR UdpNm timing parameter Repeat Message Time.⌋

[SWS_UdpNm_00248] ⌈The configuration parameter `UdpNmWaitBusSleepTime` shall determine the AUTOSAR UdpNm timing parameter Wait Bus-Sleep Time.⌋

[SWS_UdpNm_00249] ⌈The optional configuration parameter `UdpNmRemoteSleepIndTime` shall determine the AUTOSAR UdpNm timing parameter Remote Sleep Indication Time.⌋

7.6 Communication Scheduling

7.6.1 NM Message Transmission

Note: The transmission mechanisms described in this chapter are only relevant if the NM message transmission ability is enabled.

[SWS_UdpNm_00072] ⌈The transmission of NM messages shall be configurable by means of `UdpNmPassiveModeEnabled` (see [ECUC_UdpNm_00010]).⌋

Note: Passive nodes do not transmit NM messages, i.e. they can not actively influence the shut down decision, but they do receive NM message in order to be able to shut down synchronously.

Note: The transmission mechanisms described in this chapter are only relevant if `UdpNmPassiveModeEnabled` is `FALSE`.

Note: The UdpNm module provides the periodic transmission mode, see PRS_Nm_00237. In this transmission mode the UdpNm module sends Network Management PDUs periodically.

Note: The periodic transmission mode is used in the "Repeat Message State" and "Normal Operation State".

Transmission of NM PDU in Repeat Message State can be started either directly in case of `UdpNmImmediateNmTransmissions` is greater than 0 (see PRS_Nm_00005) or otherwise delayed by `UdpNmMsgCycleOffset` (see PRS_Nm_00334).

Note: `UdpNmImmediateNmTransmissions` has to be greater than 0 in case `UdpNmPnHandleMultipleNetworkRequests` is set to TRUE due to [ECUC_UdpNm_00075]. This ensures that immediate transmissions are applied in this case.

[SWS_UdpNm_00006] ⌈If Normal Operation State is entered from Ready Sleep State the transmission of NM PDUs shall be started immediately.⌋

[SWS_UdpNm_00330] ⌈If NM PDUs shall be transmitted with `UdpNmImmediateNmCycleTime`, UdpNm shall ensure that `UdpNmImmediateNmTransmissions` (including first immediate transmission) with this timing are requested successfully. If a transmission request to SoAd fails (`E_NOT_OK` is returned), UdpNm shall retry the transmission request in the next main function. Afterwards UdpNm shall continue transmitting NM PDUs using the `UdpNmMsgCycleTime`.⌋

Note: While transmitting NM PDUs using the `UdpNmImmediateNmCycleTime` no other Nm PDUs shall be transmitted (i.e. the `UdpNmMsgCycleTime` transmission cycle is stopped).

[SWS_UdpNm_00497] ⌈If transmission of Network Management PDUs has been started, the UdpNm Message Cycle Timer expires and when `UdpNmSynchronizedPncShutdownEnabled` is set either to `FALSE` or if set to `TRUE` and additionally the transmission of PN shutdown messages is inactive, then the UdpNm module shall transmit a Network Management PDU by calling `SoAd_IfTransmit`.⌋

[SWS_UdpNm_00498] ⌈If transmission of Network Management PDUs has been started, the UdpNm Message Cycle Timer expires and when `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE` and the transmission of PN shutdown messages is active, the transmission of this NM PDU shall be postponed to the next UdpNm_Mainfunction call.⌋

Note:

- A NM-PDU transmitted as PN shutdown messsage has to be sent immediately and therefore processing of cylic NM-PDUs transmitted with `UdpNmMsgCycleTime` has to be delayed. In rare cases this could lead to a delay of more than one main function cycle time.
- The NM timing has to consider that an NM message transmitted with `UdpNmMsgCycleTime` may be delayed for more than one main function cycle time. Therefore the following condition has to be fulfilled to tolerate multiple delays of those NM Messages: (`NmPnResetTime` - `UdpNmMsgCycleTime`) > `n` * `UdpNmMainFunctionPeriod`, where `n` denotes the number of tolerated delays before the PnResetTime expires, if no NM message is received.

[SWS_UdpNm_00499] ⌈If the UdpNm module has requested a transmission of a NM-PDU, `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE`, the transmission of PN shutdown messages is active, `UdpNm_TxConfirmation` is called with result `E_NOT_OK` or the transmission request for this NM-PDU was not accepted (`SoAd_IfTransmit` returned `E_NOT_OK`), then the UdpNm module shall perform a retransmission of a NM-PDU for this NM-Channel in the next main function call.⌋

Note:

- UdpNm has to perform a retry transmission handling for a NM-PDU in the context of the main function calls, if the transmission of PN shutdown messages is active and if the transmission of this NM-PDU was not accepted or was not confirmed by the lower layer. The retry transmission requests should cover error cases, where the lower layer cannot transmit the Nm messages.
- The dependency to a pending transmission confirmation indicated by the lower layer, should support reliable communication, e.g. ensure PN shutdown message

was transmitted on the bus or avoid transmissions of outdated PN shutdown messages, if for example queueing in the lower layer is configured.

[SWS_UdpNm_00040] ⌈If the UdpNm Message Cycle Timer expires it shall be restarted with `UdpNmMsgCycleTime`.⌋

[SWS_UdpNm_00051] ⌈If transmission of NM PDUs has been stopped the UdpNm Message Cycle Timer shall be canceled.⌋

[SWS_UdpNm_00007] ⌈If parameter `UdpNmRetryFirstMessageRequest` (see [ECUC_UdpNm_00085]) is `TRUE` and if the first transmit request after transition from Bus Sleep to Repeat Message State is not accepted by SoAd, the message request shall be repeated in the next main function until one transmit request is accepted by SoAd.⌋

Note: This feature can be used in case of partial network wakeup filter to avoid a blocking of all messages in case of passive start-up and first message request is not accepted by SoAd due to EthSM could not enable transmission path fast enough (e.g. in case of asynchronous transceiver handling).

[SWS_UdpNm_00379] ⌈If `UdpNm_SoAdIfTxConfirmation` is called with result `E_NOT_OK`, UdpNm shall call the function `Nm_TxTimeoutException`.⌋

7.6.2 NM Message Reception

If an NM message has been successfully received, the SoAd will call `UdpNm_SoAdIfRxIndication`.

[SWS_UdpNm_00035] ⌈Upon a call of `UdpNm_SoAdIfRxIndication`, the UdpNm module shall copy the data of the Network Management PDU referenced in the function parameter to an internal buffer.⌋

[SWS_UdpNm_00037] ⌈When an NM PDU has been received, the Nm function `Nm_PduRxIndication` shall be called, if `UdpNmPduRxIndicationEnabled` (configuration parameter) is `TRUE`.⌋

7.7 Additional features

7.7.1 Detection of Remote Sleep Indication (optional)

The "Remote Sleep Indication" denotes a situation, where a node in Normal Operation State finds all other nodes in the cluster are ready to sleep. The node still in Normal Operation State will still keep the bus awake.

[SWS_UdpNm_00149] ⌈Detection of remote sleep indication shall be statically configurable with use of the `UdpNmRemoteSleepIndEnabled` switch (configuration parameter).⌋

[SWS_UdpNm_00150] ⌈If no NM PDUs are received in the Normal Operation State for a configurable amount of time determined by the `UdpNmRemoteSleepIndTime` (configuration parameter), the NM shall notify the Generic Network Management Interface that all other nodes in the cluster are ready to sleep (the so-called 'Remote Sleep Indication') by calling `Nm_RemoteSleepIndication`.⌋

[SWS_UdpNm_00151] ⌈If Remote Sleep Indication has been previously detected and if an NM PDU is received in the Normal Operation State or Ready Sleep State again, the NM shall notify the Generic Network Management Interface that some nodes in the cluster are not ready to sleep anymore (the so-called 'Remote Sleep Cancellation') by calling `Nm_RemoteSleepCancellation`.⌋

[SWS_UdpNm_00152] ⌈If Remote Sleep Indication has been previously detected and if Repeat Message State is entered from Normal Operation State or Ready Sleep State, the UdpNm shall notify the Generic Network Management Interface that some nodes in the cluster are not ready to sleep anymore (the so-called 'Remote Sleep Cancellation') by calling `Nm_RemoteSleepCancellation`.⌋

[SWS_UdpNm_00154] ⌈The NM shall reject a check of Remote Sleep Indication in Bus-Sleep Mode, Prepare Bus-Sleep Mode and Repeat Message State; the service shall not be executed and `E_NOT_OK` shall be returned.⌋

7.7.2 User Data (optional)

[SWS_UdpNm_00158] ⌈Support of NM user data shall be statically configurable using the `UdpNmUserDataEnabled` switch (configuration parameter).⌋

[SWS_UdpNm_00159] ⌈When `UdpNm_SetUserData` is called, the NM user data for NM packets transmitted next on the bus shall be set; operation of setting the NM user data shall guarantee data consistency.⌋

[SWS_UdpNm_00160] ⌈When `UdpNm_GetUserData` is called, the NM user data contained in the payload of the most recently received NM PDU shall be provided; operation of providing the NM user data shall guarantee data consistency.⌋

Note: If NM user data is configured it will be sent for sure in the Repeat Message State. In Ready Sleep State the user data will not be sent.

[SWS_UdpNm_00312] ⌈If `UdpNmComUserDataSupport` is enabled the API `UdpNm_SetUserData` shall not be available.⌋

[SWS_UdpNm_00317] ⌈If `UdpNmComUserDataSupport` is enabled and NM-PDU is not configured for triggered transmission in SoAd (`SoAdBswModules/SoAdIfTriggerTransmit = FALSE`), the UdpNm shall collect the NM User Data from the referenced NM I-PDU by calling `PduR_UdpNmTriggerTransmit` and combine the user data with the further NM bytes each time before it requests the transmission of the corresponding NM message.⌋

Note: In case of triggered transmission no data is needed at the transmission request, just the length is needed. The data will be collected within `UdpNm_SoAdIfTriggerTransmit` (see chapter 8.3.23 "UdpNm_SoAdIfTrigger Transmit").

[SWS_UdpNm_00464] ⌈If `UdpNmComUserDataSupport` is enabled and if UdpNm is in RepeatMessage state or NormalOperation state and if `UdpNm_Transmit` is called, UdpNm shall request an additional transmission of the NM PDU with the current data.⌋

Note: The call of `UdpNm_Transmit` request to transmit a NM PDU between the periodic transmissions with the current data (e.g. system bytes, user data and PNC Request Bit Vector)

7.7.3 Passive Mode (optional)

In Passive Mode the node is only receiving NM messages but not transmitting any NM messages.

[SWS_UdpNm_00161] ⌈Passive Mode shall be statically configurable with use of the `UdpNmPassiveModeEnabled` switch (configuration parameter).⌋

[SWS_UdpNm_00162] ⌈Passive Mode shall be statically configured consistent for all instances within one ECU.⌋

[SWS_UdpNm_00163] ⌈If Passive Mode is used (configuration parameter `UdpNmPassiveModeEnabled`) the following options must not be used:

- Bus Synchronization

  (configuration parameter `UdpNmBusSynchronizationEnabled`)
- Remote Sleep Indication

  (configuration parameter `UdpNmRemoteSleepIndEnabled`)
- Node Detection

  (configuration parameter `UdpNmNodeDetectionEnabled`)

⌋

7.7.4 State change notification (optional)

[SWS_UdpNm_00166] ⌈All changes of the AUTOSAR UdpNm states shall be notified to the upper layer by calling `Nm_StateChangeNotification` if the callback `Nm_StateChangeNotification` is enabled (configuration parameter `UdpNmStateChangeIndEnabled` is `TRUE`).⌋

7.7.5 Communication Control (optional)

[SWS_UdpNm_00168] ⌈Communication Control shall be statically configurable with use of the `UdpNmComControlEnabled` switch (configuration parameter).⌋

[SWS_UdpNm_00170] ⌈The optional service `UdpNm_DisableCommunication` shall disable the NM PDU transmission ability.⌋

Note: The NM coordination algorithm cannot work correctly if NM PDU transmission ability is disabled. Therefore it has to be ensured that the ECU is not shutdown as long as the NM PDU transmission ability is disabled.

If `UdpNm_NetworkRelease` is called and NM PDU transmission ability has been disabled, ECU will shut down. This ensures that ECU can shut down also in case of race conditions (e.g. diagnostic session left shortly before enabling communication) or a wrong usage of communication control.

[SWS_UdpNm_00172] ⌈The optional service `UdpNm_DisableCommunication` shall return `E_NOT_OK`, if the current mode is not Network Mode.⌋

[SWS_UdpNm_00173] ⌈When the Network Management PDU transmission ability is disabled, the UdpNm module shall stop the UdpNm Message Cycle Timer in order to stop the transmission of Network Management PDUs.⌋

[SWS_UdpNm_00174] ⌈When the NM PDU transmission ability is disabled, the NM-Timeout Timer shall be stopped.⌋

[SWS_UdpNm_00175] ⌈When the NM PDU transmission ability is disabled, the detection of Remote Sleep Indication Timer shall be suspended.⌋

[SWS_UdpNm_00178] ⌈When the Network Management PDU transmission ability is enabled, the transmission of NM PDUs shall be started latest within the next NM main function.⌋

[SWS_UdpNm_00179] ⌈When the NM PDU transmission ability is enabled, the NM-Timeout Timer shall be restarted.⌋

[SWS_UdpNm_00180] ⌈When the NM PDU transmission ability is enabled, the detection of Remote Sleep Indication Timer shall be resumed.⌋

[SWS_UdpNm_00181] ⌈The optional service `UdpNm_RequestBusSynchronization` shall return `E_NOT_OK` if the NM PDU transmission ability is disabled.⌋

7.7.6 NM Coordinator synchronization support (optional)

When having more than one coordinator connected to the same bus a special bit in the `CBV`, the `NmCoordinatorSleepReady` bit is used to indicate that the main coordinator requests to start shutdown sequence. The main functionality of the algorithm is described in the Nm module.

[SWS_UdpNm_00320] ⌈If the UdpNm called `NM_CoordReadyToSleepIndication` and is still in Network Mode it shall notify the Nm by calling `Nm_CoordReadyToSleepCancellation` on the first reception of a NM message with the `NmCoordinatorSleepReady` bit (see `CBV`) set it to `0`⌋

[SWS_UdpNm_00364] ⌈If UdpNm has entered Network mode or called `Nm_CoordReadyToSleepCancellation` before it shall notify the NM by calling `NM_CoordReadyToSleepIndication` on the first reception of NM message with the `NmCoordinatorSleepReady` bit (see `CBV`) set to `1`⌋

[SWS_UdpNm_00321] ⌈If `UdpNmCoodinatorSyncSupport` is set to `TRUE` and the API `UdpNm_SetSleepReadyBit` is called UdpNm shall set the "NM Coordinator Sleep Ready Bit" bit to passed value and trigger a single Network Management PDU.⌋

[SWS_UdpNm_00322] ⌈The API `UdpNm_SetSleepReadyBit()` and the feature "Coordinated Bus Shutdown" shall only be available if `UdpNmCoordinatorSyncSupport` is set to `TRUE`.⌋

7.8 Partial Networking

An overview regarding the partial network cluster functionality can be found in document [8, Guide to Mode Management].

7.8.1 Rx Handling of NM PDUs

[SWS_UdpNm_00328] ⌈If the `UdpNmPnEnabled` is `FALSE`, the UdpNm shall perform the normal Rx Indication handling and the partial networking extensions shall be disabled.⌋

[SWS_UdpNm_00329] ⌈If `UdpNmPnEnabled` is `TRUE`, the PNI bit in the received NM-PDU is `0` and `UdpNmAllNmMessagesKeepAwake` is `TRUE`, the UdpNm module shall perform the normal Rx Indication handling and omitting the extensions for partial networking.⌋

[SWS_UdpNm_00462] ⌈If `UdpNmPnEnabled` is `TRUE`, the PNI bit in the received NM-PDU is `0` and `UdpNmAllNmMessagesKeepAwake` is `FALSE`, the UdpNm module shall ignore the received NM-PDU.⌋

[SWS_UdpNm_00486] ⌈If `UdpNmPnEnabled` is set to `TRUE`, the PNI bit in the received NM-PDU is set to `1` and one of the following pre-conditions is valid:

- `UdpNmSynchronizedPncShutdownEnabled` is set to `FALSE`
- `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE` and the `PNSR` bit is set to `0`

then the UdpNm module shall extract the PNC Request Bit Vector from the received NM-PDU according to the partial network configuration (`NmPncBitVectorOffset` and `NmPncBitVectorLength` of the corresponding NM-channel) and forward the PNC Request Bit Vector by calling `Nm_PncBitVectorRxIndication`.⌋

Note: The `PNSR` bit shall be evaluated only if `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE`.

[SWS_UdpNm_00487] ⌈If `UdpNmPnEnabled` is set to `TRUE` and `Nm_PncBitVectorRxIndication` was called, then a received NM PDU shall only be considered for further processing under the following conditions:

- `UdpNmAllNmMessagesKeepAwake` is set to `TRUE` OR
- the output value of `RelevantPncRequestDetectedPtr` is set to `TRUE`

⌋

Note:

- `UdpNmAllNmMessagesKeepAwake` is required to enable a gateway to stay awake on any kind of NM-PDU.
- As consequence of [SWS_UdpNm_00487], a NM PDU is not considered for further processing if not all messages shall keep the ECU awake or no relevant PNC Request Bit has been detected.

Example:

- `UdpNmPduCbvPosition = 0`
- `UdpNmPduNidPosition = 1`
- `NmPncBitVectorOffset = 4`
- `NmPncBitVectorLength = 4`
- Calculated length of user data range `= 2`

Byte 2 and Byte 3 of the NM PDU contain user data and

Byte 4 to Byte 7 of the NM PDU contain the PNC Request Bit Vector:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CBV	NID	User Data		PNC Request Bit Vector			
0x40	0x00	0xFF	0xFF	0x12	0x8E	0x80	0x01

Table 7.1: Example NM PDU containing PNC Request Bit Vector

For this example four NmPnFilterMaskBytes shall be defined. The values of the PN filter mask are used according to the partial network design e.g:

- NmPnFilterMaskByteIndex = 0 with NmPnFilterMaskByteValue = 0x01
- NmPnFilterMaskByteIndex = 1 with NmPnFilterMaskByteValue = 0x97
- NmPnFilterMaskByteIndex = 2 with NmPnFilterMaskByteValue = 0x00
- NmPnFilterMaskByteIndex = 3 with NmPnFilterMaskByteValue = 0x00

Note: The offset for the PNC Request Bit Vector is derived from the Nm module (NmPncBitVectorOffset). The PNC Request Bit Vector length is derived form the Nm module per NM-channel (NmPncBitVectorLength). The PN filter mask (NmPnFilterMaskByteIndex and NmPnFilterMaskByteValue) located and used in the Nm module.

[SWS_UdpNm_00473] ⌈If UdpNmSynchronizedPncShutdownEnabled is TRUE, the PNI bit in the received NM-PDU is 1, the PNSR bit in the received NM-PDU is 1 and the corresponding ComMChannel configured via UdpNmComMNetworkHandleRef where this NM-PDU was received is actively coordinated (ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE), then the UdpNm module shall report the run-time error UDPNM_E_INVALID_PN_SYNC_SHUTDOWN_REQUEST to the Default Error Tracer, ignore the PNSR bit and handle the PDU as usual NM PDU.⌋

Note: The handling should support the robustness of the PN regarding a synchronized shutdown handling, if the NM of an ECU is malfunction.

[SWS_UdpNm_00488] ⌈If UdpNmSynchronizedPncShutdownEnabled is TRUE, the PNI bit in the received NM-PDU is set to 1 and the PNSR bit is set to 1, UdpNm module shall extract the PNC Request Bit Vector from the received NM-PDU according to the partial network configuration (NmPncBitVectorOffset and NmPncBitVectorLength of the corresponding NM-channel) and forward the PNC Request Bit Vector by calling Nm_ForwardSynchronizedPncShutdown.⌋

Note: PNSR Bit set to 1 is only possible if a synchronized PNC shutdown is requested. A synchronized PNC shutdown should be handled across the PN topology. Therefore, it is assumed that either all coordinators have the synchronized PNC shutdown enabled or all coordinators have the synchronized PNC shutdown disabled. A mixture of both would lead to an unsynchronized PNC shutdown, which has to be avoided.

7.8.2 Tx Handling of NM PDUs

[SWS_UdpNm_00332] ⌈If `UdpNmPnEnabled` is `TRUE` the UdpNm module shall set the value of the transmitted PNI bit in the `CBV` to `1`.⌋

Note: The usage of the `CBV` is mandatory in case Partial Networking is used.

[SWS_UdpNm_00333] ⌈If `UdpNmPnEnabled` is `FALSE` the UdpNm module shall set the value of the transmitted PNI bit in the `CBV` always to `0`.⌋

[SWS_UdpNm_00500] ⌈If the NM-channel parameter `UdpNmPnEnabled` is set to `TRUE`, the UdpNm module shall store the latest PNC Request Bit Vector for this NM-channel everytime the PNC Request Bit Vector has been fetched from the Nm module via call of `Nm_PncBitVectorTxIndication`.⌋

[SWS_UdpNm_00501] ⌈If the NM-channel parameter `UdpNmPnEnabled` is set to `TRUE`, a NM-PDU has been transmitted on this NM-Channel and `UdpNm_TxConfirmation` is called with result `E_OK` for this NM-PDU, then the UdpNm module shall forward the confirmation to Nm by calling `Nm_PncBitVectorTxConfirmation` with the stored PNC Request Bit Vector (see [SWS_UdpNm_00500]) for this NM-channel with result set to `E_OK`.⌋

Note: The confirmation towards the Nm is always performed, independent of the reason for transmission of a NM-PDU (e.g. cyclic NM-PDU transmitted with `UdpNmMsgCycleTime` or NM-PDU transmitted as PN shutdown message).

[SWS_UdpNm_00502] ⌈If the NM-channel parameter `UdpNmPnEnabled` is set to `TRUE`, a NM-PDU has been transmitted on this NM-Channel and `UdpNm_TxConfirmation` is called with result `E_NOT_OK` or the transmission request for this NM-PDU was not accepted (`SoAd_IfTransmit` returned `E_NOT_OK`), then the UdpNm module shall forward the confirmation to Nm by calling `Nm_PncBitVectorTxConfirmation` with the stored PNC Request Bit Vector (see [SWS_UdpNm_00500]) for this NM-Channel with result set to `E_NOT_OK`.⌋

Note: The call of `Nm_PncBitVectorTxConfirmation` with `E_NOT_OK` is used by the Nm module to perform the synchronized PNC shutdown handling, if PNC shutdown handling is configured.

[SWS_UdpNm_00503] ⌈If `UdpNmPnEnabled` is `TRUE` and a NM-PDU has to be transmitted (either as cylic NM-PDU transmitted with `UdpNmMsgCycleTime` (see [SWS_UdpNm_00497]) or as PN shutdown message), the UdpNm module shall additionally fetch the PNC Request Bit Vector by calling `Nm_PncBitVectorTxIndication` and copy the PNC Request Bit Vector with respect to `NmPncBitVectorOffset` and `NmPncBitVectorLength` of the corresponding NM-channel to the NM-PDU before requesting the transmission of the NM-PDU.⌋

Note:

- The transmission of a NM-PDU has to consider user data if the usage of user data is configured. Please refer to 7.7.2 “User Data (optional)”.
- PNC Request Bit Vector is always fetched up front to a transmission request independent if `SoAdTxPduTriggerTransmit` is set to `TRUE` or `FALSE`. This should ensure to re-start the PN reset timer of the affected PNC in the Nm on a transmission request.

[SWS_UdpNm_00504] ⌈If `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE`, the transmission of PN shutdown messages is active for this NM-Channel and no transmission confirmation of a previous call to transmit a NM-PDU as PN shutdown message on this NM-Channel is pending, then the UdpNm module shall request in the next main function call a transmission of a NM-PDU as PN shutdown message by calling `SoAd_IfTransmit`.⌋

7.8.3 Handling of Internal Requested Partial Network Clusters

All internal PNC requests are maintained by ComM. ComM forwards the aggregated internal PNC requests per channel as PNC Request Bit Vector to NmIf. This PNC Request Bit Vector carries the so-called "Internal Request Array". The UdpNm has to retrieve the latest IRA from NmIf every time an NM_PDU is transmitted. NmIf provides the IRA information to UdpNm and updates the PNC reset timer (each time a relevant PNC is transmitted, the PNC reset timer is re-started).

Note: For all configured NM-channel where `UdpNmPnEnabled` is set to `TRUE`, the Udp Nm will call `Nm_PncBitVectorTxIndication`(<NM-channel>, <buffer to store the unfiltered PNC Request Bit Vector of aggregated internal PNC requests>) (see [SWS_UdpNm_00503], [SWS_UdpNm_00506] and [SWS_UdpNm_00508]) to indicate the transmission and to retrieve the current internal PNC requests as PNC Request Bit Vector with respect to the configured `NmPncBitVectorLength`. The UdpNm will copy received internal PNC requests to the PNC Request Bit Vector bytes of the NM-PDU.

7.8.4 Spontaneous Transmission of NM-PDUs via UdpNm_NetworkRequest

In case `UdpNmPnHandleMultipleNetworkRequests` is TRUE, immediate transmissions can be triggered by calling `UdpNm_NetworkRequest`, see PRS_Nm_00507 for further details.

Note: If `UdpNmPnHandleMultipleNetworkRequests` is set to `TRUE` the UdpNm feature ’Immediate Transmission’ is mandatory.

Note: The PNC Control Module (e.g. ComM) is responsible to call UdpNm_Network Request if the PNC Request Bits change.

7.9 Payload (PDU) Structure

The figure below shows an example for n bytes PDU length where the source node identifier is located in the first byte, the control bit vector in the second byte, user data is used and partial network is enabled. User data range is located between the system bytes and the PNC Request Bit Vector:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Source Node Identifier (default)							
Byte 1	Control Bit Vector (default)							
Byte 2	User data 0							
Byte 3	User data 1							
Byte 4	...							
Byte i+2	User data i							
Byte i+3	PNC Request Bit Vector - byte 0							
Byte i+4	PNC Request Bit Vector - byte 1							
...	...							
Byte n	PNC Request Bit Vector - byte j							

Table 7.2: Example of NM packet payload (NM PDU)

Note:

The length of the Network Management PDU (NM PDU) is defined by the PduLength parameter in the "global" ECUC module ([EcuC003_Conf], see Ecu Configuration specification).

[SWS_UdpNm_00074] ⌈The location of the source node identifier shall be configurable by means of `UdpNmPduNidPosition` to Byte `0`, Byte `1`, or `off`.⌋

[SWS_UdpNm_00075] ⌈The location of the control Bit vector shall be configurable by means of `UdpNmPduCbvPosition` to Byte `0`, Byte `1`, or `off`.⌋

Note: The location of the PNC Request Bit Vector is configurable by means of `NmPncBitVectorOffset` and `NmPncBitVectorLength` of the corresponding NM-channel. The location of the PNC Request Bit Vector is placed after the system bytes (control bit vector and source node identifier) and within the PduLenght of the NM-PDU.

[SWS_UdpNm_00491] ⌈The remaining bytes not assigned to Nm System Bytes or PNC Request Bit Vector shall be available for User Data.⌋

Note: According to [9, System Template] (TPS_SYST_03069, TPS_SYST_03070, TPS_SYST_03071, TPS_SYST_03072) the use and location of user data is configurable. If user data are used, the user data are placed within the PduLenght of the NM-PDU and do not overlap with the range of system bytes or PNC Request Bit Vector. If partial network functionaliy is enabled (`UdpNmPnEnabled` is set to `TRUE`) and user data are used, the user data range is exclusively located either between the system bytes and the PNC Request Bit Vector or between the PNC Request Bit Vector and

the end of the NM-PDU. The length of user data range shall be calculated according the following restrictions:

- If the user data range resides between the system bytes and the PNC Request Bit Vector, then the length of the user data range is determined by the difference of the PNC Request Bit offset and the length of the system bytes.
- If the user data range resides between the PNC Request Bit Vector and the end of the NM-PDU, then the length of the user data range is determined by the difference of the NM-PDU length and the position/index of the last byte of the PNC Request Bit Vector (defined by PNC Request Bit offset + PNC Request Bit Vector length)

If partial network functionaliy is disabled (`UdpNmPnEnabled` is set to `FALSE`) and user data are used, the user data range is determined by the difference of NM-PDU length and the length of the system bytes.

[SWS_UdpNm_00013] ⌈The source node identifier shall be set with the configuration parameter `UDPNM_NODE_ID` unless `UdpNmPduNidPosition` is set to `off`.⌋

[SWS_UdpNm_00366] ⌈If the UdpNm performs a state change from BusSleep state or PrepareBusSleep state to NetworkMode due to a call to `UdpNm_NetworkRequest()` (i.e. due to an active wakeup) and `UdpNmActiveWakeupBitEnabled` is `TRUE`, the UdpNm shall set the ActiveWakeupBit in the `CBV`.⌋

[SWS_UdpNm_00367] ⌈If the UdpNm module leaves the NetworkMode and `UdpNmActiveWakeupBitEnabled` is `TRUE`, the UdpNm module shall clear the Active WakeupBit in the `CBV`.⌋

7.10 Functional requirements on UdpNm API

[SWS_UdpNm_00014] ⌈If `UdpNmRepeatMsgIndEnabled` is set to `TRUE` and the Repeat Message Request bit set to `1` is received UdpNm module shall call the callback function Nm_RepeatMessageIndication. In case the Partial Network Learning Bit is also received and `UdpNmDynamicPncToChannelMappingEnabled` is set to `TRUE` the parameter pnLearningBitSet in this function call shall be set to `TRUE`, otherwise to `FALSE`.⌋

7.11 Car Wakeup

[SWS_UdpNm_00373] ⌈The position of the Car Wakeup bit in the NM-PDU is defined by the configuration parameters `UdpNmCarWakeUpBytePosition` and `UdpNmCarWakeUpBitPosition`.⌋

[SWS_UdpNm_00374] ⌈If the Car Wakeup bit within any received NM-PDU is `1`, `UdpNmCarWakeUpRxEnabled` is `TRUE`, and `UdpNmCarWakeUpFilterEnabled` is `FALSE` UdpNm shall call `Nm_CarWakeUpIndication` and perform the standard Rx indication handling.⌋

[SWS_UdpNm_00375] ⌈If `UdpNm_GetPduData` is called in the context of `Nm_CarWakeUpIndication` and if `UdpNmNodeDetectionEnabled` or `UdpNmUserDataEnabled` or `UdpNmNodeIdEnabled` is set to `TRUE`, UdpNm shall return the PDU data of the PDU that causes the call of `Nm_CarWakeUpIndication`.⌋

Note: This is required to enable ECU to identify detail about the sender of the Car Wakeup request

[SWS_UdpNm_00376] ⌈If `UdpNmCarWakeUpFilterEnabled` is `TRUE`, the Car Wakeup bit within any received NM-PDU is `1`, `UdpNmCarWakeUpRxEnabled` is `TRUE` and the Node ID in the received NM-PDU is equal to `UdpNmCarWakeUpFilterNodeId` the UdpNm module shall call `Nm_CarWakeUpIndication` and perform the standard Rx Indication handling.⌋

Note: The Car Wakeup filter is necessary to realize sub gateways that only consider the Car Wakeup of the central Gateway to avoid wrong wakeups

7.12 Error Classification

This section describes how the UdpNm module has to manage the error classes that may occur during the life cycle of this basic software.

The general requirements document of AUTOSAR [5, General Requirements on Basic Software Modules] specifies that all basic software modules must distinguish (according to the product life cycle) two error types:

- **Development errors:** these errors should be detected and fixed during the development phase. In most cases, these errors are software errors. The detection errors that should only occur during development can be switched off for production code (by static configuration, namely preprocessor switches).
- **Production errors:** these errors are hardware errors and software exceptions that cannot be avoided and are expected to occur in the production (i.e. series) code. This kind of error is commonly known as a run-time error.

7.12.1 Development Errors

[SWS_UdpNm_00018] Definition of development errors in module UdpNm ⌈

Type of error	*Related error code*	*Error value*
API service used without module initialization	UDPNM_E_UNINIT	0x01
API service called with wrong channel handle	UDPNM_E_INVALID_CHANNEL	0x02
API service called with wrong PDU ID.	UDPNM_E_INVALID_PDUID	0x03
UdpNm initialization has failed, e.g. selected configuration set doesn't exist	UDPNM_E_INIT_FAILED	0x04
Null pointer has been passed as an argument	UDPNM_E_PARAM_POINTER	0x12

⌋

[SWS_UdpNm_00189]

Upstream requirements: SRS_BSW_00337

⌈Development errors shall not be returned by API functions; in case of a development error, the respective API function will return `E_NOT_OK`, if applicable.⌋

7.12.2 Runtime Errors

[SWS_UdpNm_00465] Definition of runtime errors in module UdpNm

Upstream requirements: SRS_BSW_00385, SRS_BSW_00452

⌈

Type of error	*Related error code*	*Error value*
NM-Timeout timer has expired outside Ready Sleep State (either in Repeat Message state or in Normal Operation state)	UDPNM_E_NETWORK_TIMEOUT	0x11
A NM message with PN Shutdown Request Bit was received on a channel that is actively coordinated by the ComM PNC Gateway.	UDPNM_E_INVALID_PN_SYNC_SHUTDOWN_REQUEST	0x20

⌋

[SWS_UdpNm_00466] ⌈When the NM-Timeout Timer expires in the Repeat Message State, the UdpNm module shall report the runtime error `UDPNM_E_NETWORK_TIMEOUT` to the Default Error Tracer.⌋

[SWS_UdpNm_00467] ⌈When the NM-Timeout Timer expires in the Normal Operation State, the UdpNm module shall report runtime error `UDPNM_E_NETWORK_TIMEOUT` to the Default Error Tracer.⌋

7.12.3 Production Errors

There are no production errors.

7.13 Scheduling of the main function

For details refer to [3] Chapter 8.4 *"Scheduled functions"*.

7.14 Application notes

7.14.1 Wakeup notification

Wakeup notification is defined in detail in the ECU State Manager specification, see [7] Chapter 7 *"Functional Specification"* ff.

7.14.2 Coordination of coupled networks

[SWS_UdpNm_00185] ⌈Support of bus synchronization on demand shall be statically configurable with use of the `UdpNmBusSynchronizationEnabled` switch (configuration parameter).⌋

Note: Since the shutdown of UdpNm can be done at any time, the call of the API `Nm_SynchronizationPoint` is not supported.

7.15 Parameter check

[SWS_UdpNm_00196]

Upstream requirements: SRS_BSW_00337

⌈If detection of development errors is enabled by `UDPNM_DEV_ERROR_DETECT` (configuration parameter), validity checks for all input parameters shall be performed for each UDP NM API service call.⌋

[SWS_UdpNm_00197]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈Parameter type checking shall be performed at compile time; if types do not match, the compilation process shall be stopped and respective compilation warnings or errors shall be returned as far as supported by the compiler.⌋

[SWS_UdpNm_00198]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈Parameter value check (for parameters of the constant value) shall be performed at configuration time; if the value is invalid, the configuration process shall be stopped and the respective configuration error shall be reported.⌋

[SWS_UdpNm_00199]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈Parameter value check (for parameters of the variable value) shall be performed at execution time; if the value is invalid, execution of a service shall be denied and the respective development error shall be reported.⌋

7.16 Security Events

The module does not report security events.

8 API specification

[SWS_UdpNm_00244]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00339

⌈The UdpNm module shall reject the execution of a service called with an invalid parameter and shall inform the DET.⌋

AUTOSAR UdpNm API consists of services, which are UDP specific and can be called whenever they are required; each service apart from `UdpNm_Init` refers to one NM channel only.

[SWS_UdpNm_00190]

Upstream requirements: SRS_BSW_00337

⌈Production errors shall not be returned by API functions; in case of a production error, the respective API function will return `E_NOT_OK`, if applicable.⌋

[SWS_UdpNm_00192]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈When NM API service with an invalid network handle is called, the called function shall not be executed, but instead of that it shall report `UDPNM_E_INVALID_CHANNEL` to the Default Error Tracer (if development error detection is enabled) otherwise it shall return `E_NOT_OK` to the calling function⌋

Note: The network handle is invalid if it is different from allowed configured values.

[SWS_UdpNm_00492]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈When a Null pointer has been passed to a UdpNm service, the called function shall not be executed and it shall return `E_NOT_OK` to the calling function if applicable. If development error detection is enabled (`UdpNmDevErrorDetect` is set to `TRUE`) the corresponding error `UDPNM_E_PARAM_POINTER` shall be reported to DET.⌋

[SWS_UdpNm_00463]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈When UdpNm Callback Notifications with an invalid Pdu ID are called, the called function shall not be executed and `E_NOT_OK` shall be returned if possible. If Development Error Detection is enabled then additionally UdpNm shall report `UDPNM_E_INVALID_PDUID` to the Default Error Tracer.⌋

[SWS_UdpNm_00314]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈If `UdpNmComUserDataSupport` is enabled and the UdpNm User Data length does not match with the length of the referenced I-PDU an error shall be reported at generation time.⌋

Note: NULL Pointer checking is specified within BSW General [3, General Specification of Basic Software Modules].

8.1 Imported types

The following types of `Std_Types.h` are imported:

`boolean`

`uint8`

`uint16`

`uint32`

[SWS_UdpNm_91011] Definition of imported datatypes of module UdpNm ⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	NetworkHandleType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
Nm	NmStack_types.h	Nm_ModeType
	NmStack_types.h	Nm_StateType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 UdpNm_ConfigType

This type shall contain the parameters of the container UdpNm_GlobalConfig and its sub containers.

[SWS_UdpNm_00308] Definition of datatype UdpNm_ConfigType

Upstream requirements: SRS_BSW_00414

⌈

Name	UdpNm_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	This type shall contain the parameters of the container UdpNm_Global Config and its sub containers.
Description	–	
Available via	UdpNm.h	

⌋

8.2.2 UdpNm_PduPositionType

[SWS_UdpNm_00304] Definition of datatype UdpNm_PduPositionType ⌈

Name	UdpNm_PduPositionType		
Kind	Enumeration		
Range	UDPNM_PDU_BYTE_0	0x00	Byte 0 is used
	UDPNM_PDU_BYTE_1	0x01	Byte 1 is used
	UDPNM_PDU_OFF	0xFF	Node Identification is not used
Description	Used to define the position of the control bit vector within the NM PACKET.		
Available via	UdpNm.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 UdpNm_Init

[SWS_UdpNm_00208] Definition of API function UdpNm_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00344, SRS_BSW_00358, SRS_BSW_00405, SRS_BSW_00414

⌈

Service Name	UdpNm_Init	
Syntax	`void UdpNm_Init ( const UdpNm_ConfigType* UdpNmConfigPtr )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	UdpNmConfigPtr	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initialize the complete UdpNm module, i.e. all channels which are activated at configuration time are initialized. A UDP socket shall be set up with the TCP/IP stack. Caveats: This function has to be called after initialization of the TCP/IP stack. Configuration: Mandatory	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00210] ⌈If an error has to be indicated to the DET the value 0x00 shall be used as the instance id.⌋

Rationale: the value 0 x 00 is not error value but instance ID

8.3.2 UdpNm_PassiveStartUp

[SWS_UdpNm_00211] Definition of API function UdpNm_PassiveStartUp ⌈

Service Name	UdpNm_PassiveStartUp	
Syntax	`Std_ReturnType UdpNm_PassiveStartUp ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0x0e	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Passive startup of network management has failed

▽

△

Description	Passive startup of the AUTOSAR UdpNm. It triggers the transition from Bus-Sleep Mode or Prepare Bus Sleep Mode to the Network Mode in Repeat Message State. Caveats: UdpNm is initialized correctly.
Available via	UdpNm.h

⌋

[SWS_UdpNm_00147] ⌈If `UdpNm_PassiveStartUp` is called in the Network Mode, the UdpNm module shall not execute this service and shall return `E_NOT_OK`.⌋

8.3.3 UdpNm_NetworkRequest

[SWS_UdpNm_00213] Definition of API function UdpNm_NetworkRequest ⌈

Service Name	UdpNm_NetworkRequest	
Syntax	`Std_ReturnType UdpNm_NetworkRequest ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0x02	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Requesting of network has failed
Description	Request the network, since ECU needs to communicate on the bus. Network state shall be changed to 'requested' Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UdpNmPassiveModeEnabled == false)	
Available via	UdpNm.h	

⌋

8.3.4 UdpNm_NetworkRelease

[SWS_UdpNm_00214] Definition of API function UdpNm_NetworkRelease ⌈

Service Name	UdpNm_NetworkRelease
Syntax	`Std_ReturnType UdpNm_NetworkRelease ( NetworkHandleType nmChannelHandle )`
Service ID [hex]	0x03
Sync/Async	Asynchronous
Reentrancy	Reentrant (but not for the same NM-Channel)

▽

△

Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Releasing of network has failed
Description	Release the network, since ECU doesn't have to communicate on the bus. Network state shall be changed to 'released'. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UdpNmPassiveModeEnabled == false)	
Available via	UdpNm.h	

⌋

8.3.5 UdpNm_DisableCommunication

[SWS_UdpNm_00215] Definition of API function UdpNm_DisableCommunication
⌈

Service Name	UdpNm_DisableCommunication	
Syntax	`Std_ReturnType UdpNm_DisableCommunication ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0x0c	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Disabling of NM PDU transmission ability has failed
Description	Disable the NM PDU transmission ability due to a ISO14229 Communication Control (0x28) service Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UdpNmComControlEnabled == true)	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00307] ⌈If the module operates in passive mode (`UdpNmPassiveModeEnabled`) the service `UdpNm_DisableCommunication` shall have no effects and shall directly return `E_NOT_OK`.⌋

8.3.6 UdpNm_EnableCommunication

[SWS_UdpNm_00216] Definition of API function UdpNm_EnableCommunication ⌈

Service Name	UdpNm_EnableCommunication	
Syntax	`Std_ReturnType UdpNm_EnableCommunication ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0x0d	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Enabling of NM PDU transmission ability has failed
Description	Enable the NM PDU transmission ability due to a ISO14229 Communication Control (0x28) service Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UdpNmComControlEnabled == true).	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00176] ⌈The optional service `UdpNm_EnableCommunication` shall enable the NM PDU transmission ability if the NM PDU transmission ability is disabled.⌋

[SWS_UdpNm_00177] ⌈The optional service `UdpNm_EnableCommunication` shall return `E_NOT_OK` if the NM PDU transmission ability is already enabled when the service is called.⌋

[SWS_UdpNm_00305] ⌈The service `UdpNm_EnableCommunication` shall return `E_NOT_OK`, if the current mode is not Network Mode.⌋

[SWS_UdpNm_00306] ⌈If the module operates in passive mode (`UdpNmPassiveModeEnabled` is `TRUE`) the service `UdpNm_EnableCommunication` shall have no effects and shall directly return `E_NOT_OK`.⌋

8.3.7 UdpNm_SetUserData

[SWS_UdpNm_00217] Definition of API function UdpNm_SetUserData ⌈

Service Name	UdpNm_SetUserData
Syntax	`Std_ReturnType UdpNm_SetUserData ( NetworkHandleType nmChannelHandle, const uint8* nmUserDataPtr )`

▽

△

Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
	nmUserDataPtr	Pointer where the user data for the next transmitted NM message shall be copied from.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Setting of user data has failed
Description	Set user data for all NM messages transmitted on the bus after this function has returned without error. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UdpNmUserDataEnabled==true and UdpNmPassive ModeEnabled==false).	
Available via	UdpNm.h	

⌋

8.3.8 UdpNm_GetUserData

[SWS_UdpNm_00218] Definition of API function UdpNm_GetUserData ⌈

Service Name	UdpNm_GetUserData	

Syntax	`Std_ReturnType UdpNm_GetUserData (`
	`  NetworkHandleType nmChannelHandle,`
	`  uint8* nmUserDataPtr`
	`)`	

Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmUserDataPtr	Pointer where user data out of the most recently received NM message shall be copied to.
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Getting of user data has failed
Description	Get user data from the most recently received NM message. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UdpNmUserDataEnabled == true).	
Available via	UdpNm.h	

⌋

8.3.9 UdpNm_GetNodeIdentifier

[SWS_UdpNm_00219] Definition of API function UdpNm_GetNodeIdentifier ⌈

Service Name	UdpNm_GetNodeIdentifier	
Syntax	`Std_ReturnType UdpNm_GetNodeIdentifier ( NetworkHandleType nmChannelHandle, uint8* nmNodeIdPtr )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmNodeIdPtr	Pointer where the source node identifier from the most recently received NM PDU shall be copied to.
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Getting of the node identifier out of the most recently received NM PDU has failed or is not configured for this network handle.
Description	Get node identifier from the most recently received NM PDU. Caveats: UdpNm is initialized correctly.	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00132] ⌈The service call `UdpNm_GetNodeIdentifier` shall provide the node identifier out of the most recently received Network Management PDU if `UdpNmNodeIdEnabled` is set to `TRUE`.⌋

8.3.10 UdpNm_GetLocalNodeIdentifier

[SWS_UdpNm_00220] Definition of API function UdpNm_GetLocalNodeIdentifier ⌈

Service Name	UdpNm_GetLocalNodeIdentifier	
Syntax	`Std_ReturnType UdpNm_GetLocalNodeIdentifier ( NetworkHandleType nmChannelHandle, uint8* nmNodeIdPtr )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmNodeIdPtr	Pointer where node identifier of the local node shall be copied to.
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Getting of the node identifier of the local node has failed or is not configured for this network handle.

▽

△

Description	Get node identifier configured for the local node. Caveats: UdpNm is initialized correctly.
Available via	UdpNm.h

⌋

[SWS_UdpNm_00133] ⌈The service call `UdpNm_GetLocalNodeIdentifier` shall provide the node identifier configured for the local host node if `UdpNmNodeIdEnabled` is set to `TRUE`.⌋

8.3.11 UdpNm_RepeatMessageRequest

[SWS_UdpNm_00221] Definition of API function UdpNm_RepeatMessageRequest ⌈

Service Name	UdpNm_RepeatMessageRequest	
Syntax	`Std_ReturnType UdpNm_RepeatMessageRequest ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0x08	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Setting of Repeat Message Request Bit has failed or is not configured for this network handle.
Description	Set Repeat Message Request Bit for all NM messages transmitted on the bus after this function has returned without error.	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00137] ⌈If the service `UdpNm_RepeatMessageRequest` is called in Repeat Message State, Prepare Bus-Sleep Mode or Bus-Sleep Mode, the UdpNm module shall not execute the service and return `E_NOT_OK`.⌋

8.3.12 UdpNm_GetPduData

[SWS_UdpNm_00309] Definition of API function UdpNm_GetPduData

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00484, SRS_BSW_00459

⌈

Service Name	UdpNm_GetPduData	
Syntax	`Std_ReturnType UdpNm_GetPduData ( NetworkHandleType nmChannelHandle, uint8* nmPduDataPtr )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmPduDataPtr	Pointer where NM PDU shall be copied to.
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Getting of NM PDU Data has failed or is not configured for this network handle.
Description	Get the whole PDU data out of the most recently received NM message. Caveats: UdpNm is initialized correctly.	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00138] ⌈The service call `UdpNm_GetPduData` shall provide whole payload (Source Node ID, `Control Bit Vector` and User Data) of the most recently received Network Management PDU if `UdpNmNodeDetectionEnabled` or `UdpNmUserDataEnabled` or `UdpNmNodeIdEnabled` is set to `TRUE`.⌋

8.3.13 UdpNm_GetState

[SWS_UdpNm_00310] Definition of API function UdpNm_GetState ⌈

Service Name	UdpNm_GetState	
Syntax	`Std_ReturnType UdpNm_GetState ( NetworkHandleType nmChannelHandle, Nm_StateType* nmStatePtr, Nm_ModeType* nmModePtr )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmStatePtr	Pointer where state of the network management shall be copied to.

▽

△

	nmModePtr	Pointer where the mode of the network management shall be copied to.
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: Getting of NM state has failed
Description	Returns the state and the mode of the network management. Caveats: UdpNm is initialized correctly. Configuration: Mandatory	
Available via	UdpNm.h	

⌋

8.3.14 UdpNm_RequestBusSynchronization

[SWS_UdpNm_00226] Definition of API function UdpNm_RequestBusSynchronization ⌈

Service Name	UdpNm_RequestBusSynchronization	
Syntax	`Std_ReturnType UdpNm_RequestBusSynchronization ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0x14	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: Requesting of bus synchronization has failed
Description	Request bus synchronization. Caveats: UdpNm is initialized correctly. Configuration: Optional (only available if UdpNmBusSynchronizationEnabled==true and UdpNmPassiveModeEnabled==false).	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00130] ⌈The service call `UdpNm_RequestBusSynchronization` shall trigger transmission of a single Network Management PDU if `UdpNmPassiveModeEnabled` (configuration parameter) is `FALSE`.⌋

Rationale: This service is typically used for supporting the NM gateway extensions.

[SWS_UdpNm_00187] ⌈If `UdpNm_RequestBusSynchronization` is called in Bus-Sleep Mode and Prepare Bus-Sleep Mode the UdpNm module shall not execute the service and shall return `E_NOT_OK`.⌋

8.3.15 UdpNm_CheckRemoteSleepIndication

[SWS_UdpNm_00227] Definition of API function UdpNm_CheckRemoteSleepIndication ⌈

Service Name	UdpNm_CheckRemoteSleepIndication	
Syntax	`Std_ReturnType UdpNm_CheckRemoteSleepIndication ( NetworkHandleType nmChannelHandle, boolean* NmRemoteSleepIndPtr )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	NmRemoteSleepIndPtr	Pointer where check result of remote sleep indication shall be copied to.
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Checking of remote sleep indication bits has failed
Description	Check if remote sleep indication takes place or not. Caveats: UdpNm is initialized correctly. Configuration: Optional (only available if UdpNmRemoteSleepIndEnabled == true)	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00153] ⌈The service call `UdpNm_CheckRemoteSleepIndication` shall provide the information about current status of Remote Sleep Indication (i.e. already detected or not).⌋

8.3.16 UdpNm_SetSleepReadyBit

[SWS_UdpNm_00324] Definition of API function UdpNm_SetSleepReadyBit ⌈

Service Name	UdpNm_SetSleepReadyBit	
Syntax	`Std_ReturnType UdpNm_SetSleepReadyBit ( NetworkHandleType nmChannelHandle, boolean nmSleepReadyBit )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
	nmSleepReadyBit	Value written to ReadySleep Bit in CBV
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Writing of remote sleep indication bit has failed

▽

△

Description	Set the NM Coordinator Sleep Ready bit in the Control Bit Vector
Available via	UdpNm.h

⌋

8.3.17 UdpNm_Transmit

[SWS_UdpNm_00313] Definition of API function UdpNm_Transmit ⌈

Service Name	UdpNm_Transmit	

Syntax	`Std_ReturnType UdpNm_Transmit (`
	`  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	UdpNm.h	

⌋

[SWS_UdpNm_00315] ⌈If `UdpNmComUserDataSupport` or `UdpNmPnEnabled` is enabled the UdpNm implementation shall provide an API `UdpNm_Transmit`.⌋

8.3.18 UdpNm_PnLearningRequest

[SWS_UdpNm_91004] Definition of API function UdpNm_PnLearningRequest

Status: DRAFT

⌈

Service Name	UdpNm_PnLearningRequest (draft)

Syntax	`Std_ReturnType UdpNm_PnLearningRequest (`
	`  NetworkHandleType nmChannelHandle`
	`)`

Service ID [hex]	0x4a
Sync/Async	Asynchronous
Reentrancy	Reentrant (but not for the same NM-channel)

▽

△

Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: PN Learning Requesthas failed or is not configured for this network handle.
Description	Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. This will force all nodes on the bus to enter the PNC Learning Phase. This is needed for the optional Dynamic PNC-to-channel-mapping feature. **Tags:** atp.Status=draft	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00471] ⌈If the function UdpNm_PnLearningRequest is called in "Prepare Bus-Sleep Mode" or "Bus Sleep Mode" no functionality shall be executed and E_NOT_OK shall be returned.⌋

[SWS_UdpNm_00509] ⌈The function UdpNm_PnLearningRequest shall only be available if UdpNmDynamicPncToChannelMappingSupport is set to TRUE.⌋

8.3.19 UdpNm_ActivateTxPnShutdownMsg

[SWS_UdpNm_91009] Definition of API function UdpNm_ActivateTxPnShutdown Msg ⌈

Service Name	UdpNm_ActivateTxPnShutdownMsg	
Syntax	`Std_ReturnType UdpNm_ActivateTxPnShutdownMsg ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0xf4	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different nmChannelHandle. Non reentrant for the same nmChannelHandle.	
Parameters (in)	nmChannelHandle	Identifier of the NM-Channel where the PNC shutdown process is started.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK:Request has been accepted. E_NOT_OK: Request has not been accepted.
Description	NM indicate to activate the transmission of PN shutdown messages on the given NM-Channel. This results in transmission of a NM-PDU with PNSR bit set to 1 (PN shutdown message).	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00505] ⌈If UdpNmSynchronizedPncShutdownEnabled is set to TRUE the UdpNm implementation shall provide the API UdpNm_ActivateTxPnShutdownMsg.⌋

[SWS_UdpNm_00506] ⌈If `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE` and `UdpNm_ActivateTxPnShutdownMsg` is called with a valid NM-Channel (`nmChannelHandle`), then the UdpNm module shall consider the PN shutdown message transmission as active on the given NM-channel, set `PNSR` bit in the `CBV` to `1` and return with `E_OK`.⌋

8.3.20 UdpNm_DeactivateTxPnShutdownMsg

[SWS_UdpNm_91010] Definition of API function UdpNm_DeactivateTxPnShutdownMsg ⌈

Service Name	UdpNm_DeactivateTxPnShutdownMsg	
Syntax	`Std_ReturnType UdpNm_DeactivateTxPnShutdownMsg ( NetworkHandleType nmChannelHandle )`	
Service ID [hex]	0xf5	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different nmChannelHandle. Non reentrant for the same nmChannelHandle.	
Parameters (in)	nmChannelHandle	Identifier of the NM-Channel where the PNC shutdown process is stopped.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`:Request has been accepted. `E_NOT_OK`: Request has not been accepted.
Description	NM indicate to deactive the transmission of PN shutdown messages on the given NM-Channel. This result in transmission of a usual NM-PDUs with PNSR bit set to 0.	
Available via	UdpNm.h	

⌋

[SWS_UdpNm_00507] ⌈If `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE` the UdpNm implementation shall provide the API `UdpNm_DeactivateTxPnShutdownMsg`.⌋

[SWS_UdpNm_00508] ⌈If `UdpNmSynchronizedPncShutdownEnabled` is set to `TRUE` and `UdpNm_DeactivateTxPnShutdownMsg` is called with a valid NM-Channel (`nmChannelHandle`), then the UdpNm module shall consider the PN shutdown message transmission as inactive on the given NM-channel, set `PNSR` bit in the `CBV` to `0` and return with `E_OK`.⌋

8.3.21 UdpNm_SoAdIfTxConfirmation

[SWS_UdpNm_00228] Definition of callback function UdpNm_SoAdIfTxConfirmation ⌈

Service Name	UdpNm_SoAdIfTxConfirmation	
Syntax	`void UdpNm_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	UdpNm.h	

⌋

Note: The callback function `UdpNm_SoAdIfTxConfirmation` is called by the SoAd and is implemented by the UdpNm module.

Note: The callback function `UdpNm_SoAdIfTxConfirmation` is either called on interrupt level (interrupt mode) or on task level (Polling Mode) with respect to the context.

The value passed to UdpNm via the API parameter `TxPduId` shall refer to the NM channel handle, i.e. a mapping from PduId to NM channel handle is not necessary.

[SWS_UdpNm_00316] ⌈If `UdpNmComUserDataSupport` is enabled the UdpNm shall call `PduR_UdpNmTxConfirmation` within the message transmission confirmation function `UdpNm_SoAdIfTxConfirmation` called by the SoAd and with result passed by SoAd⌋

8.3.22 UdpNm_SoAdIfRxIndication

[SWS_UdpNm_00231] Definition of callback function UdpNm_SoAdIfRxIndication ⌈

Service Name	UdpNm_SoAdIfRxIndication
Syntax	`void UdpNm_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	UdpNm.h	

⌋

The callback function `UdpNm_SoAdIfRxIndication` called by the SoAd and implemented by the UdpNm module. It is called in case of a receive indication event of the SoAd.

The value passed to UdpNm via the API parameter udpNmRxPduId shall refer to the UdpNm channel handle, i.e. a mapping from PduId to UdpNm channel handle is not necessary.

8.3.23 UdpNm_SoAdIfTriggerTransmit

[SWS_UdpNm_91001] Definition of callback function UdpNm_SoAdIfTrigger Transmit ⌈

Service Name	UdpNm_SoAdIfTriggerTransmit	
Syntax	`Std_ReturnType UdpNm_SoAdIfTriggerTransmit ( PduIdType TxPduId, PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: SDU has been copied and SduLength indicates the number of copied bytes. `E_NOT_OK`: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.

▽

△

Description	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.
Available via	UdpNm.h

⌋

Note: The PNC Request Bit Vector is not updated within the call of `UdpNm_TriggerTransmit` but upfront of each NM message transmission request (see [SWS_UdpNm_00503]). This ensure a common handling independent of the `SoAdTxPduTriggerTransmit` setting (`TRUE` or `FALSE`).

[SWS_UdpNm_00495] ⌈If `UdpNm_SoAdIfTriggerTransmit` is called and `UdpNmComUserDataSupport` is enabled, UdpNm shall collect the NM User Data from the referenced NM I-PDU by calling `PduR_UdpNmTriggerTransmit` and copy the data to the user data range of the NM-PDU.⌋

[SWS_UdpNm_00378] ⌈The function `UdpNm_SoAdIfTriggerTransmit` shall copy the NM PDU data of the according NM PDU requested by `TxPduId`.⌋

Note: The function `UdpNm_SoAdIfTriggerTransmit` might be called by the SoAd in an interrupt context.

8.4 Scheduled functions

8.4.1 UdpNm_MainFunction_<Instance Id>

[SWS_UdpNm_00234] Definition of scheduled function UdpNm_MainFunction<Instance_Id>

Upstream requirements: SRS_BSW_00424, SRS_BSW_00425

⌈

Service Name	UdpNm_MainFunction<Instance_Id>
Syntax	`void UdpNm_MainFunction<Instance_Id> ( void )`
Service ID [hex]	0x13
Description	Main function of the UdpNm which processes the algorithm describes in that document. E.g.: UdpNm_MainFunction_0() represents the UdpNm instance for the UDP channel 0 UdpNm_MainFunction_1() represents the UdpNm instance for the UDP channel 1
Available via	SchM_UdpNm.h

⌋

8.5 Expected interfaces

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

8.5.1 Mandatory interfaces

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

[SWS_UdpNm_91007] Definition of mandatory interfaces required by module UdpNm ⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
Nm_BusSleepMode	Nm.h	Notification that the network management has entered Bus-Sleep Mode.
Nm_NetworkMode	Nm.h	Notification that the network management has entered Network Mode.
Nm_NetworkStartIndication	Nm.h	Notification that a NM-message has been received in the Bus-Sleep Mode, what indicates that some nodes in the network have already entered the Network Mode.
Nm_PrepareBusSleepMode	Nm.h	Notification that the network management has entered Prepare Bus-Sleep Mode.
SoAd_IfTransmit	SoAd.h	Requests transmission of a PDU.

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

[SWS_UdpNm_91006] Definition of optional interfaces requested by module Udp Nm ⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
Nm_CarWakeUpIndication	Nm.h	This function is called by a <Bus>Nm to indicate reception of a CWU request.
Nm_CoordReadyToSleepCancellation	Nm.h	Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0.
Nm_CoordReadyToSleepIndication	Nm.h	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set

▽

△

API Function	Header File	Description
Nm_ForwardSynchronizedPnc Shutdown	Nm.h	Notification that the network management has received a PN shutdown message on a particular NM-channel. This is used to grant a nearly synchronized PNC shutdown across the entire PN topology.
Nm_PduRxIndication	Nm.h	Notification that a NM message has been received.
Nm_PncBitVectorRxIndication	Nm.h	Indication that a bus specific network management has received a NM message on a particular NM-channel that contain a PNC Request Bit Vector. This is used to aggregate the external PNC requests. The function evaluate if a relevant PNC request (PNC Request Bit set to '1') is available in the given PNC Request Bit Vector. If a relevant PNC request is available (PNC Request Bit passes the PNC Request Bit Vector filter), then the RelevantPnc RequestDetectedPtr refers to a boolean with value set to TRUE. Otherwise refer to boolean with value set to FALSE. RelevantPncRequestDetectedPtr is evaluated by the callee <Bus>Nm module to qualify the further processing of the received NM-PDU.
Nm_PncBitVectorTxConfirmation	Nm.h	Function called by <Bus>Nms to confirm the state of the transmission for the given PNC Request Bit Vector on the given NM-Channel.
Nm_PncBitVectorTxIndication	Nm.h	Function called by <Bus>Nms to request the aggregated internal PNC requests for transmission within the Nm message.
Nm_RemoteSleepCancellation	Nm.h	Notification that the network management has detected that not all other nodes on the network are longer ready to enter Bus-Sleep Mode.
Nm_RemoteSleepIndication	Nm.h	Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode.
Nm_RepeatMessageIndication	Nm.h	Service to indicate that an NM message with set Repeat Message Re- quest Bit has been received. This is needed for node detection and the Dynamic PNC-to-channel-mapping feature.
Nm_StateChangeNotification	Nm.h	Notification that the state of the lower layer <Bus>Nm has changed.
Nm_TxTimeoutException	Nm.h	Service to indicate that an attempt to send an NM message failed.
PduR_UdpNmRxIndication	PduR_UdpNm.h	Indication of a received PDU from a lower layer communication interface module.
PduR_UdpNmTriggerTransmit	PduR_UdpNm.h	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->Sdu Length. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->Sdu Length. If not, it returns E_NOT_OK without changing PduInfoPtr.
PduR_UdpNmTxConfirmation	PduR_UdpNm.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.

⌋

8.5.3 Configurable interfaces

Not applicable

8.6 Service Interfaces

Not applicable

8.7 UML State chart diagram

The following figure shows an UML state diagram with respect to the API specification. Mode change related transitions are denoted in green, error handling related transitions in red and optional node detection related transitions in blue.

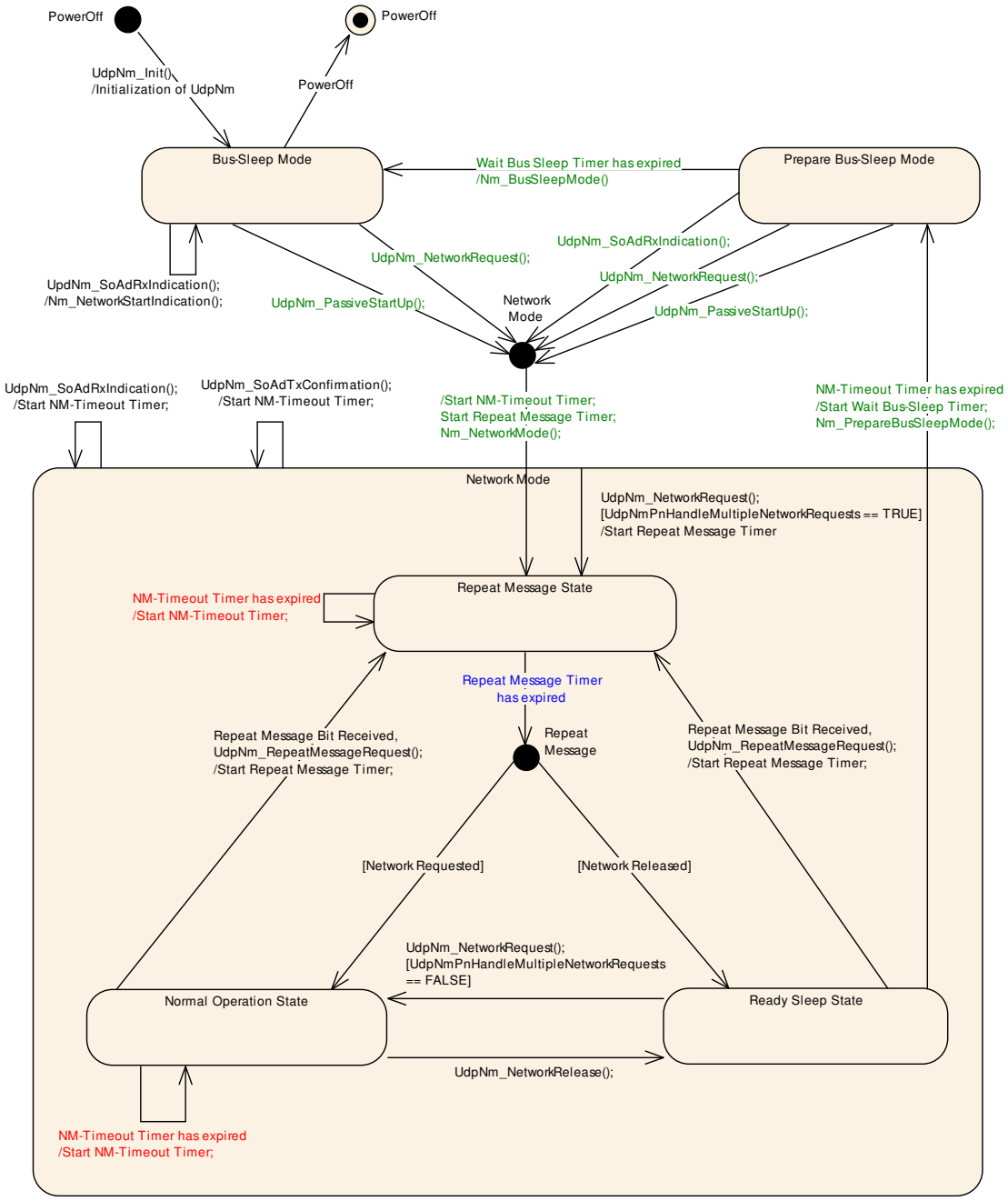

Figure 8.1: State chart diagram.

9 Sequence diagrams and Transition Tables

9.1 UdpNmTransmission

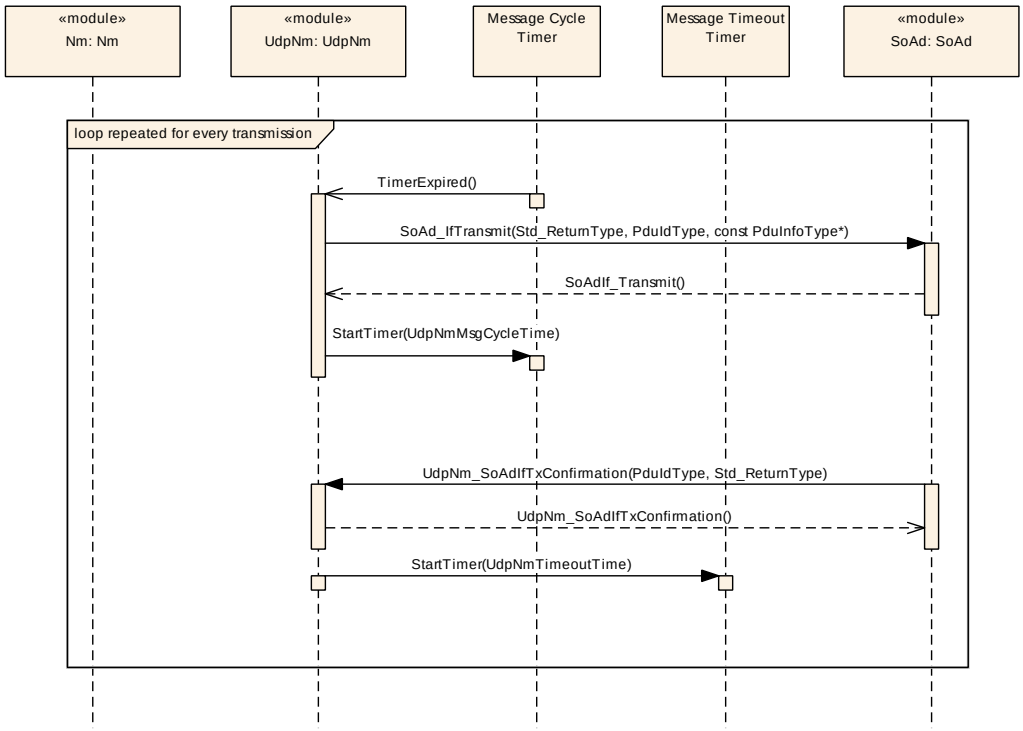

Figure 9.1: Sequence diagram - PDU transmission.

9.2 UdpNm Reception

Call direction	*Action/Decision*	*Description*
SoAd->UdpNm	UdpNm_SoAdIfRxIndication()	

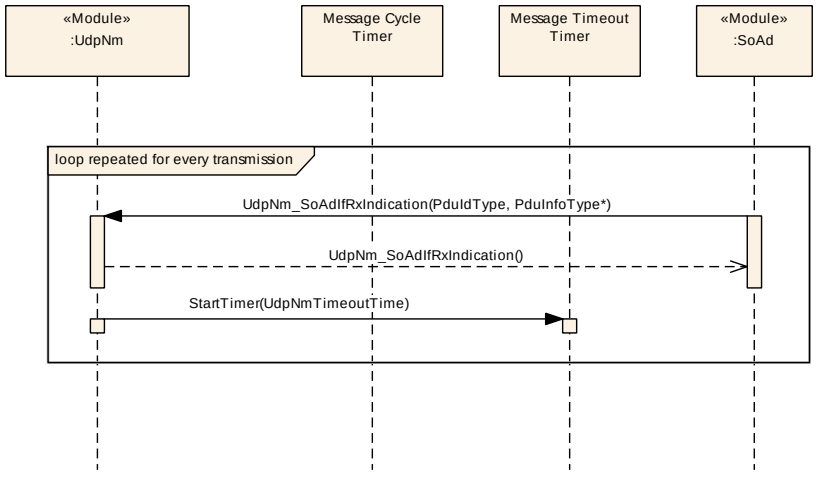

Figure 9.2: Sequence diagram - PDU reception.

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module UdpNm.

10.1 How to read this chapter

For details refer to [3] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The configuration parameters as defined in this chapter are used to create a data model for an AUTOSAR tool chain. The realization in the code is implementation specific.

The configuration parameters as defined in this chapter are used to create a data model for an AUTOSAR tool chain. The realization in the code is implementation specific.

The configuration parameters are divided into parameters used to enable features, parameters affecting all instances of the UdpNm and parameters affecting the respective instances of the UdpNm.

[SWS_UdpNm_00026]

Upstream requirements: SRS_BSW_00405

⌈All configuration items shall be located outside the kernel of the module.⌋

[SWS_UdpNm_00202] ⌈The container `UdpNm_ChannelConfig` specifies configuration parameter that shall be located in a data structure of type `UdpNm_ConfigType`.⌋

[SWS_UdpNm_00203] ⌈Runtime configurable parameters listed in container `UdpNm_ChannelConfig` shall be configurable for each NM-cluster separately.⌋

10.2.1 UdpNm

[ECUC_UdpNm_00088] Definition of EcucModuleDef UdpNm ⌈

Module Name	UdpNm
Description	Configuration of the UdpNm module.
Post-Build Selectable Support	true
Post-Build Loadable Support	false

Included Containers		
Container Name	Multiplicity	Description
UdpNmGlobalConfig	1	This container contains all global configuration parameters of UDP NM. The parameters and the parameters of the sub containers shall be mapped to the C data type UdpNm_Config Type (for parameters where it is possible) which is passed to the UdpNm_Init function.

⌋

10.2.2 UdpNmGlobalConfig

[ECUC_UdpNm_00001] Definition of EcucParamConfContainerDef UdpNmGlobal Config ⌈

Container Name	UdpNmGlobalConfig
Parent Container	UdpNm
Upstream mapping defined	–
Description	This container contains all global configuration parameters of UDP NM. The parameters and the parameters of the sub containers shall be mapped to the C data type UdpNm_ConfigType (for parameters where it is possible) which is passed to the UdpNm_Init function.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
UdpNmBusSynchronizationEnabled	1	[ECUC_UdpNm_00006]
UdpNmComControlEnabled	1	[ECUC_UdpNm_00013]
UdpNmComUserDataSupport	1	[ECUC_UdpNm_00055]
UdpNmCoordinatorSyncSupport	1	[ECUC_UdpNm_00059]
UdpNmDevErrorDetect	1	[ECUC_UdpNm_00002]
UdpNmDynamicPncToChannelMappingSupport	1	[ECUC_UdpNm_00094]
UdpNmImmediateRestartEnabled	1	[ECUC_UdpNm_00009]
UdpNmNumberOfChannels	1	[ECUC_UdpNm_00014]
UdpNmPassiveModeEnabled	1	[ECUC_UdpNm_00010]
UdpNmPduRxIndicationEnabled	1	[ECUC_UdpNm_00011]
UdpNmRemoteSleepIndEnabled	1	[ECUC_UdpNm_00005]
UdpNmStateChangeIndEnabled	1	[ECUC_UdpNm_00012]
UdpNmUserDataEnabled	1	[ECUC_UdpNm_00004]

Included Containers		
Container Name	**Multiplicity**	**Description**
UdpNmChannelConfig	1..*	This container contains the channel-specific configuration parameters of the UdpNm.

⌋

[ECUC_UdpNm_00006] Definition of EcucBooleanParamDef UdpNmBusSynchronizationEnabled ⌈

Parameter Name	UdpNmBusSynchronizationEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling bus synchronization support. This feature is required for gateway nodes only. It must not be defined if UdpNm PassiveModeEnabled==true. This parameter shall be derived from NmBus SynchronizationEnabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00013] Definition of EcucBooleanParamDef UdpNmComControlEnabled ⌈

Parameter Name	UdpNmComControlEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling the Communication Control support. Dependency: calculationFormula = If (UdpNmPassiveModeEnabled == False) then Equal(NmComControlEnabled) else Equal(False)
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00055] Definition of EcucBooleanParamDef UdpNmComUserDataSupport ⌈

Parameter Name	UdpNmComUserDataSupport
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Enable/disable the user data support. Dependency: If UdpNmPassiveModeEnabled == True OR if all bytes of the NM PDU are used for NM System Bytes and for the PNC Request Bit Vector and no space is left for user data, then UdpNmComUserDataSupport shall be set to False .

▽

△

Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00059] Definition of EcucBooleanParamDef UdpNmCoordinatorSyncSupport ⌈

Parameter Name	UdpNmCoordinatorSyncSupport
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	–
Description	Enables/disables the coordinator synchronization support. Dependency: UdpNmCoordinatorSyncSupport has to be set to FALSE if UdpNm PassiveModeEnabled is set to TRUE.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00002] Definition of EcucBooleanParamDef UdpNmDevErrorDetect ⌈

Parameter Name	UdpNmDevErrorDetect
Parent Container	UdpNmGlobalConfig
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	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00094] Definition of EcucBooleanParamDef UdpNmDynamicPncToChannelMappingSupport

Status: DRAFT

⌈

Parameter Name	UdpNmDynamicPncToChannelMappingSupport
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	Precompile time switch to enable the dynamic PNC-to-channel-mapping handling. False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled **Tags:** atp.Status=draft Dependency: UdpNmDynamicPncToChannelMappingSupport == TRUE only allowed if UdpNmPnEnabled == true for at least one UdpNm Channel and UdpNmPassiveMode Enabled == FALSE
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00009] Definition of EcucBooleanParamDef UdpNmImmediate RestartEnabled ⌈

Parameter Name	UdpNmImmediateRestartEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling the immediate transmission of a NM PACKET upon bus-communication request in Prepare-Bus-Sleep mode. Must not be defined if UdpNmPassiveModeEnabled== true.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00014] Definition of EcucIntegerParamDef UdpNmNumberOf Channels ⌈

Parameter Name	UdpNmNumberOfChannels
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	–
Description	Number of NM channels allowed within one ECU.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 255
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00010] Definition of EcucBooleanParamDef UdpNmPassiveModeEnabled ⌈

Parameter Name	UdpNmPassiveModeEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling support of the Passive Mode.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00011] Definition of EcucBooleanParamDef UdpNmPduRxIndicationEnabled ⌈

Parameter Name	UdpNmPduRxIndicationEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling the PDU Rx Indication. This parameter shall be derived from NmPduRxIndicationEnabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00005] Definition of EcucBooleanParamDef UdpNmRemoteSleepIndEnabled ⌈

Parameter Name	UdpNmRemoteSleepIndEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling remote sleep indication support. This feature is required for gateway nodes only. It must not be defined if UdpNm PassiveModeEnabled==true. This parameter shall be derived from NmRemoteSleep IndEnabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00012] Definition of EcucBooleanParamDef UdpNmStateChangeIndEnabled ⌈

Parameter Name	UdpNmStateChangeIndEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling the UDP NM state change notification. This parameter shall be derived from NmStateChangeIndEnabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00004] Definition of EcucBooleanParamDef UdpNmUserDataEnabled ⌈

Parameter Name	UdpNmUserDataEnabled
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling user data support. This parameter shall be derived from NmUserDataEnabled. Dependency: UdpNmUserDataEnabled shall be set to FALSE, if all bytes of the NM PDU are used for NM System Bytes and for the PNC Request Bit Vector and no space is left for user data. Otherwise the parameter shall be set according the following formular: calculationFormula = Equal(NmUserDataEnabled).
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

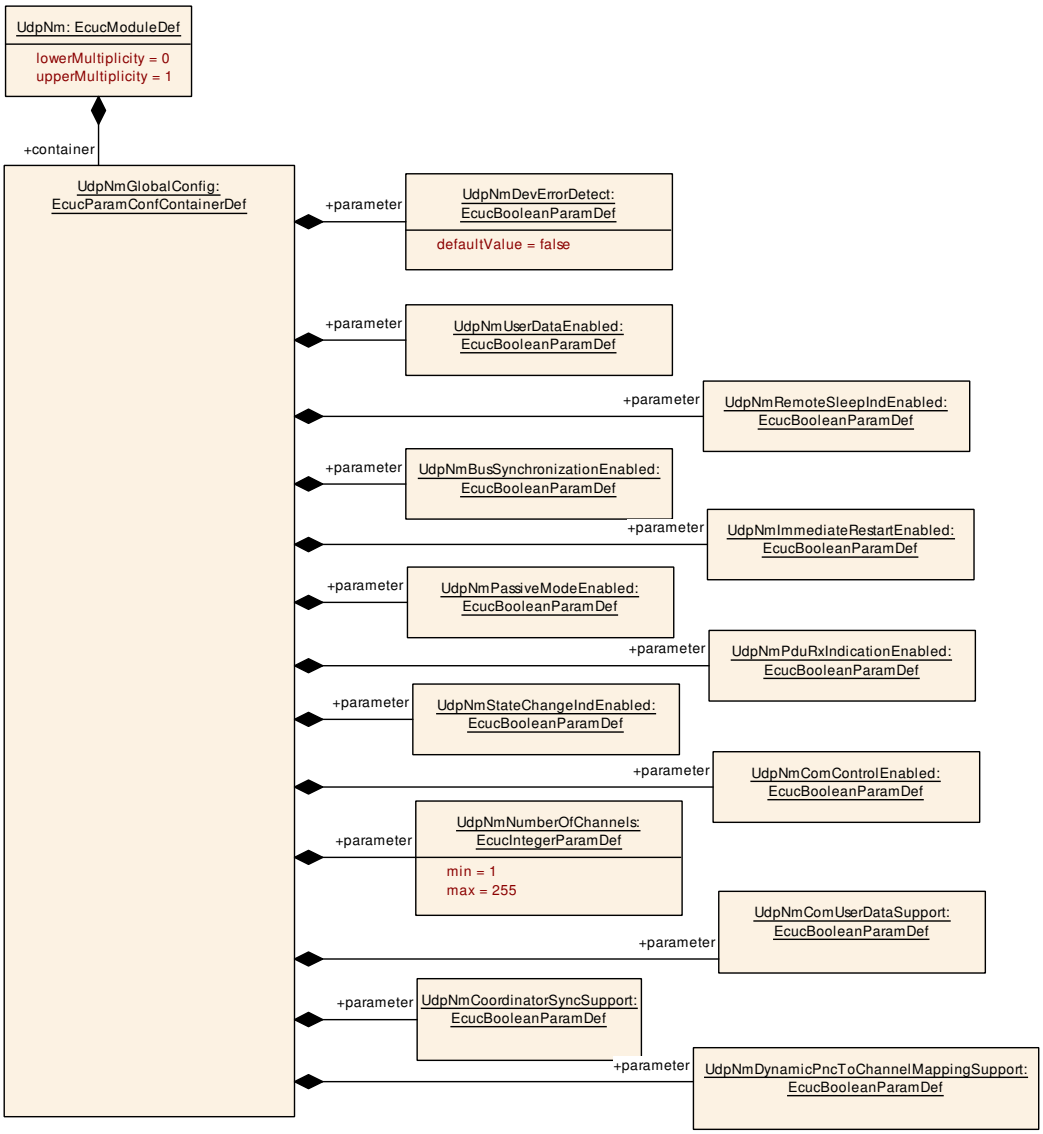

Figure 10.1: Diagram: UdpNmGlobalConfig.

10.2.3 UdpNmChannelConfig

[ECUC_UdpNm_00017] Definition of EcucParamConfContainerDef UdpNmChannelConfig ⌈

Container Name	UdpNmChannelConfig
Parent Container	UdpNmGlobalConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container contains the channel-specific configuration parameters of the UdpNm.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
UdpNmActiveWakeupBitEnabled	0..1	[ECUC_UdpNm_00074]
UdpNmAllNmMessagesKeepAwake	0..1	[ECUC_UdpNm_00089]
UdpNmCarWakeUpBitPosition	0..1	[ECUC_UdpNm_00087]
UdpNmCarWakeUpBytePosition	0..1	[ECUC_UdpNm_00086]
UdpNmCarWakeUpFilterEnabled	0..1	[ECUC_UdpNm_00077]
UdpNmCarWakeUpFilterNodeId	0..1	[ECUC_UdpNm_00078]
UdpNmCarWakeUpRxEnabled	1	[ECUC_UdpNm_00076]
UdpNmDynamicPncToChannelMappingEnabled	0..1	[ECUC_UdpNm_00095]
UdpNmImmediateNmCycleTime	0..1	[ECUC_UdpNm_00079]
UdpNmImmediateNmTransmissions	1	[ECUC_UdpNm_00075]
UdpNmMainFunctionPeriod	1	[ECUC_UdpNm_00032]
UdpNmMsgCycleOffset	1	[ECUC_UdpNm_00029]
UdpNmMsgCycleTime	1	[ECUC_UdpNm_00028]
UdpNmNodeDetectionEnabled	1	[ECUC_UdpNm_00090]
UdpNmNodeId	0..1	[ECUC_UdpNm_00031]
UdpNmNodeIdEnabled	1	[ECUC_UdpNm_00091]
UdpNmPduCbvPosition	1	[ECUC_UdpNm_00026]
UdpNmPduNidPosition	1	[ECUC_UdpNm_00025]
UdpNmPnEnabled	0..1	[ECUC_UdpNm_00061]
UdpNmPnHandleMultipleNetworkRequests	0..1	[ECUC_UdpNm_00063]
UdpNmRemoteSleepIndTime	1	[ECUC_UdpNm_00023]
UdpNmRepeatMessageTime	1	[ECUC_UdpNm_00022]
UdpNmRepeatMsgIndEnabled	1	[ECUC_UdpNm_00092]
UdpNmRetryFirstMessageRequest	0..1	[ECUC_UdpNm_00085]
UdpNmStayInPbsEnabled	1	[ECUC_UdpNm_00093]
UdpNmSynchronizedPncShutdownEnabled	0..1	[ECUC_UdpNm_00097]
UdpNmTimeoutTime	1	[ECUC_UdpNm_00020]
UdpNmWaitBusSleepTime	0..1	[ECUC_UdpNm_00021]
UdpNmComMNetworkHandleRef	1	[ECUC_UdpNm_00018]

Included Containers		
Container Name	**Multiplicity**	**Description**
UdpNmRxPdu	1..*	This container describes the UdpNm RX PDU's.
UdpNmTxPdu	0..1	This container describes the UdpNm TX PDU's.
UdpNmUserDataTxPdu	0..1	Preprocessor switch for enabling the Tx path of Com User Data. Use case: Setting of NMUserData via SWC.

⌋

[ECUC_UdpNm_00074] Definition of EcucBooleanParamDef UdpNmActiveWakeupBitEnabled ⌈

Parameter Name	UdpNmActiveWakeupBitEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	–

▽

△

Description	Enables/Disables the handling of the Active Wakeup Bit in the UdpNm module. Dependency: This parameter is only valid if UdpNmPassiveModeEnabled is False.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00089] Definition of EcucBooleanParamDef UdpNmAllNmMessagesKeepAwake ⌈

Parameter Name	UdpNmAllNmMessagesKeepAwake
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Specifies if UdpNm drops irrelevant NM PDUs. false: Only NM PDUs with a PNI bit = true and containing a PN request for this ECU triggers the standard RX indication handling true: Every NM PDU triggers the standard RX indication handling Dependency: only valid if NmPnEiraCalcEnabled == true or NmPnEraCalcEnabled == true
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00087] Definition of EcucIntegerParamDef UdpNmCarWakeUpBitPosition ⌈

Parameter Name	UdpNmCarWakeUpBitPosition	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	–	
Description	Specifies the Bit position of the CWU within the NM PDU. Dependency: only available if UdpNmCarWakeUpRxEnabled == TRUE	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00086] Definition of EcucIntegerParamDef UdpNmCarWakeUpBytePosition ⌈

Parameter Name	UdpNmCarWakeUpBytePosition	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	–	
Description	Specifies the Byte position of the CWU within the NM PDU. Dependency: only available if UdpNmCarWakeUpRxEnabled == TRUE UdpNmCarWakeupBytePosition >= number of enabled system bytes (CBV, NID)	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00077] Definition of EcucBooleanParamDef UdpNmCarWakeUpFilterEnabled ⌈

Parameter Name	UdpNmCarWakeUpFilterEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	–
Description	If CWU filtering is supported, only the CWU bit within the NM PDU with source node identifier UdpNmCarWakeUpFilterNodeId is considered as CWU request. FALSE - CWU filtering is not supported TRUE - CWU filtering is supported. Dependency: only available if UdpNmCarWakeUpRxEnabled == TRUE
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00078] Definition of EcucIntegerParamDef UdpNmCarWakeUpFilterNodeId ⌈

Parameter Name	UdpNmCarWakeUpFilterNodeId	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	–	
Description	Source node identifier for CWU filtering. If CWU filtering is supported, only the CWU bit within the NM PDU with source node identifier UdpNmCarWakeUpFilterNodeId is considered as CWU request. Dependency: only available if UdpNmCarWakeUpFilterEnabled == TRUE	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00076] Definition of EcucBooleanParamDef UdpNmCarWakeUpRxEnabled ⌈

Parameter Name	UdpNmCarWakeUpRxEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	–
Description	Enables or disables support of CarWakeUp bit evaluation in received NM PDUs. FALSE - CarWakeUp not supported. TRUE - CarWakeUp supported.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00095] Definition of EcucBooleanParamDef UdpNmDynamicPncToChannelMappingEnabled

Status: DRAFT

⌈

Parameter Name	UdpNmDynamicPncToChannelMappingEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Channel-specific parameter to enable the dynamic PNC-to-channel-mapping feature. False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled **Tags:** atp.Status=draft Dependency: Shall only be TRUE if UdpNmDynamicPncToChannelMappingSupport is TRUE
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00079] Definition of EcucFloatParamDef UdpNmImmediateNmCycleTime ⌈

Parameter Name	UdpNmImmediateNmCycleTime	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Defines the immediate NM PDU cycle time in seconds which is used for UdpNm ImmediateNmTransmissions NM PDU transmissions. Dependency: This parameter is only valid if UdpNmImmediateNmTransmissions is greater one.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0.001 .. 65.535]	

▽

△

Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00075] Definition of EcucIntegerParamDef UdpNmImmediateNm Transmissions ⌈

Parameter Name	UdpNmImmediateNmTransmissions	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Defines the number of immediate NM PDUs which shall be transmitted. If the value is zero no immediate NM PDUs are transmitted. The cycle time of immeditate NM PDUs is defined by UdpNmImmediateNmCycleTime. Dependency: If UdpNmImmediateRestartEnabled = true then UdpNmImmediateNm Transmissions = 0 If UdpNmPnHandleMultipleNetworkRequests == True then UdpNm ImmediateNmTransmissions > 0	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00032] Definition of EcucFloatParamDef UdpNmMainFunction Period ⌈

Parameter Name	UdpNmMainFunctionPeriod	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	–	
Description	Call cycle of UdpNm_MainFunction_x for the respective instance in [s].	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00029] Definition of EcucFloatParamDef UdpNmMsgCycleOffset ⌈

Parameter Name	UdpNmMsgCycleOffset
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	Time offset in the periodic transmission node. It determines the start delay of the transmission. < UdpNmMsgCycleTime This parameter is only valid if UdpNmPassiveModeEnabled is disabled.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Post-Build Value	-	Selectable

⌋

[ECUC_UdpNm_00028] Definition of EcucFloatParamDef UdpNmMsgCycleTime ⌈

Parameter Name	UdpNmMsgCycleTime	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Period of a NM-message. It determines the periodic rate and is the basis for transmit scheduling. NmTimeoutTime = n * UdpNmMsgCycleTime This parameter is only valid if UdpNmPassiveModeEnabled is disabled.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0.001 .. 65.535]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00090] Definition of EcucBooleanParamDef UdpNmNodeDetectionEnabled ⌈

Parameter Name	UdpNmNodeDetectionEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling the node detection support. This parameter shall be derived from NmNodeDetectionEnabled. This parameter shall only be enabled if UdpNmNodeIdEnabled == true. If(UdpNmPduCbvPosition != UDPNM_PDU_OFF) then Equal(NmNodeDetection Enabled) else Equal(False). Dependency: Not available if UdpNmPassiveModeEnabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00031] Definition of EcucIntegerParamDef UdpNmNodeId ⌈

Parameter Name	UdpNmNodeId	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Node identifier of local node. Dependency: This parameter is only relevant if UdpNmNodeIdEnabled == True.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00091] Definition of EcucBooleanParamDef UdpNmNodeIdEnabled ⌈

Parameter Name	UdpNmNodeIdEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Pre-processor switch for enabling the source node identifier. This parameter shall be derived from NmNodeIdEnabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00026] Definition of EcucEnumerationParamDef UdpNmPduCbvPosition ⌈

Parameter Name	UdpNmPduCbvPosition	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	

Description	Defines the position of the control bit vector within the NM PACKET.
	The value of the parameter represents the location of the control bit vector in the NM PACKET (UDPNM_PDU_BYTE_0 means byte 0, UDPNM_PDU_BYTE_1 means byte 1, UDPNM_PDU_OFF means the control bit vector is not part of the NM PACKET)
	See also UdpNmPduNidPosition
	if (UdpNmPduCbvPosition != UDPNM_PDU_OFF && UdpNmPduNidPosition != UDPNM_PDU_OFF) then UdpNmPduCbvPosition != UdpNmPduNidPosition
	if (UdpNmPduCbvPosition != UDPNM_PDU_OFF && UdpNmPduNidPosition == UDPNM_PDU_OFF) then UdpNmPduCbvPosition = UDPNM_PDU_BYTE0	

Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	UDPNM_PDU_BYTE_0	Byte 0 is used.
	UDPNM_PDU_BYTE_1	Byte 1 is used.
	UDPNM_PDU_OFF	Control bit vector is not used.

▽

△

Pre-Compile only	true

⌋

[ECUC_UdpNm_00025] Definition of EcucEnumerationParamDef UdpNmPduNidPosition ⌈

Parameter Name	UdpNmPduNidPosition	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Defines the position of the source node identifier within the NM PACKET. ImplementationType: UdpNm_PduPositionType The value of the parameter represents the location of the source node identifier in the NM PACKET (UDPNM_PDU_BYTE_0 means byte 0, UDPNM_PDU_BYTE_1 means byte 1, UDPNM_PDU_OFF means source node identifier is not part of the NM PACKET) See also UdpNmPduCbvPosition if (UdpNmPduNidPosition!= UDPNM_PDU_OFF && UdpNmPduCbvPosition != UDPNM_PDU_OFF) then UdpNmPduNidPosition != UdpNmPduCbvPosition if (UdpNmPduNidPosition != UDPNM_PDU_OFF && UdpNmPduCbvPosition == UDPNM_PDU_OFF) then UdpNmPduNidPosition = UDPNM_PDU_BYTE0	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	UDPNM_PDU_BYTE_0	Byte 0 is used.
	UDPNM_PDU_BYTE_1	Byte 1 is used.
	UDPNM_PDU_OFF	Node Identification is not used.
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00061] Definition of EcucBooleanParamDef UdpNmPnEnabled ⌈

Parameter Name	UdpNmPnEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Enables or disables support of partial networking. false: Partial networking Range not supported true: Partial networking supported
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00063] Definition of EcucBooleanParamDef UdpNmPnHandleMultipleNetworkRequests ⌈

Parameter Name	UdpNmPnHandleMultipleNetworkRequests
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	false: UdpNm_NetworkRequest is ignored in NO. true: UdpNm_NetworkRequest triggers a change from NO to RM. Dependency: only available if UdpNmPnEnabled == true
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00023] Definition of EcucFloatParamDef UdpNmRemoteSleepIndTime ⌈

Parameter Name	UdpNmRemoteSleepIndTime
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Timeout for Remote Sleep Indication. It defines the time in [s] how long it shall take to recognize that all other nodes are ready to sleep. Typically it should be equal to: n * UdpNmMsgCycleTime, where n denotes the number of NM packets that are normally sent before Remote Sleep Indication is detected. The value of n decremented by one determines the amount of lost NM packets that can be tolerated by the Remote Sleep Indication procedure.
Multiplicity	1
Type	EcucFloatParamDef
Range	[0.001 .. 65.535]
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00022] Definition of EcucFloatParamDef UdpNmRepeatMessageTime ⌈

Parameter Name	UdpNmRepeatMessageTime
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Timeout for Repeat Message State. It defines the time in seconds how long the NM shall stay in the Repeat Message State. Dependency: UdpNmRepeatMessageTime = n * UdpNmMsgCycleTime; UdpNm RepeatMessageTime > UdpNmImmediateNmTransmissions * UdpNmImmediateNm CycleTime Typically it should be equal to: n * UdpNmMsgCycleTime, where n denotes the number of NM PDUs that are normally sent in the Repeat Message State. The value of n decremented by one determines the amount of lost NM PDUs that can be tolerated by the node detection procedure. The value 0 denotes that no Repeat Message State is configured. It means that Repeat Message State is transient what implicates that it is left immediately after entrance and in result no start-up stability is guaranteed and no node detection procedure is possible.
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. 65.535]
Default value	–

▽

△

Pre-Compile only	true

⌋

[ECUC_UdpNm_00092] Definition of EcucBooleanParamDef UdpNmRepeatMsgIndEnabled ⌈

Parameter Name	UdpNmRepeatMsgIndEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Enable/disable the notification that a RepeatMessageRequest bit has been received. Dependency: UdpNmRepeatMsgIndEnabled = FALSE if UdpNmPassiveModeEnabled == TRUE or (UdpNmNodeDetectionEnabled == FALSE && UdpNmDynamicPncTo ChannelMappingEnabled == FALSE). UdpNmRepeatMsgIndEnabled = TRUE if Udp NmDynamicPncToChannelMappingEnabled == TRUE.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00085] Definition of EcucBooleanParamDef UdpNmRetryFirstMessageRequest ⌈

Parameter Name	UdpNmRetryFirstMessageRequest
Parent Container	UdpNmChannelConfig
Upstream mapping defined	–
Description	Specifies if first message request in UdpNm is repeated until accepted by SoAd. Dependency: UdpNmRetryFirstMessageRequest = false if UdpNmPassiveMode Enabled == true
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_UdpNm_00093] Definition of EcucBooleanParamDef UdpNmStayInPbsEnabled ⌈

Parameter Name	UdpNmStayInPbsEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	–
Description	If this parameter is disabled Prepare Bus-Sleep Mode is left after UdpNmWaitBusSleep Time. If this parameter is enabled Prepare Bus-Sleep Mode can only be left if ECU is powered off or any restart reason applies.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00097] Definition of EcucBooleanParamDef UdpNmSynchronizedPncShutdownEnabled ⌈

Parameter Name	UdpNmSynchronizedPncShutdownEnabled
Parent Container	UdpNmChannelConfig
Upstream mapping defined	–
Description	Specifies if UdpNm handle PN shutdown messages to support a synchronized PNC shutdown across a PN topology. This is only used for ECUs in the role of a top-level PNC coordinator or intermediate PNC coordinator. Thus, the PNC gateway functionality is enabled and therefore ERA calculation is used. FALSE: synchronized PNC shutdown is disabled TRUE: synchronized PNC shutdown is enabled Dependency: Only available if UdpNmPnEnabled == TRUE and NmPnEraCalcEnabled == TRUE.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_UdpNm_00020] Definition of EcucFloatParamDef UdpNmTimeoutTime ⌈

Parameter Name	UdpNmTimeoutTime	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Network Timeout for NM packets. It denotes the time in [s] how long the NM shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place. It shall be equal for all nodes in the cluster. It shall be greater than UdpNmMsgCycle Time. Typically, it should be equal to: x * UdpNmMsgCycleTime, where n denotes the number of NM PACKET cycle times in the Ready Sleep State before transition into the Bus-Sleep Mode is initiated. The value of n decremented by one determines the amount of lost NM packets that can be tolerated by the coordination algorithm.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0.002 .. 65.535]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00021] Definition of EcucFloatParamDef UdpNmWaitBusSleepTime ⌈

Parameter Name	UdpNmWaitBusSleepTime	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Timeout for bus calm down phase. It denotes the time in [s] how long the NM shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place. It shall be equal for all nodes in the cluster. It shall be long enough to empty all Tx-buffer empty. Dependency: In case UdpNmStayInPbsEnabled is disabled this parameter shall be mandatory.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0.001 .. 65.535]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00018] Definition of EcucReferenceDef UdpNmComMNetworkHandleRef ⌈

Parameter Name	UdpNmComMNetworkHandleRef	
Parent Container	UdpNmChannelConfig	
Upstream mapping defined	–	
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId.	
Multiplicity	1	
Type	Symbolic name reference to ComMChannel	
Post-Build Value	-	Selectable

⌋

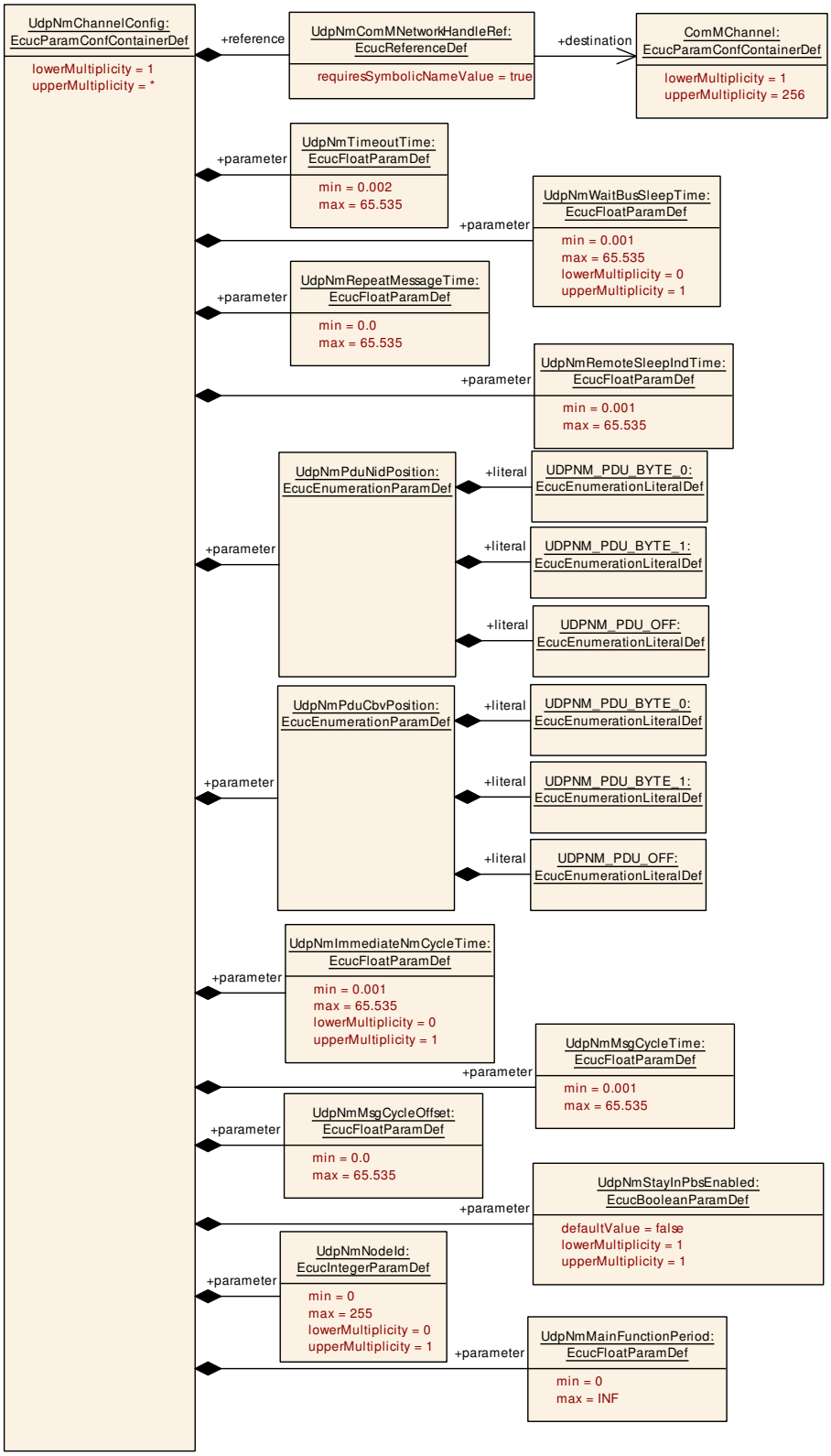

Figure 10.2: UdpNmChannelConfig - part 1

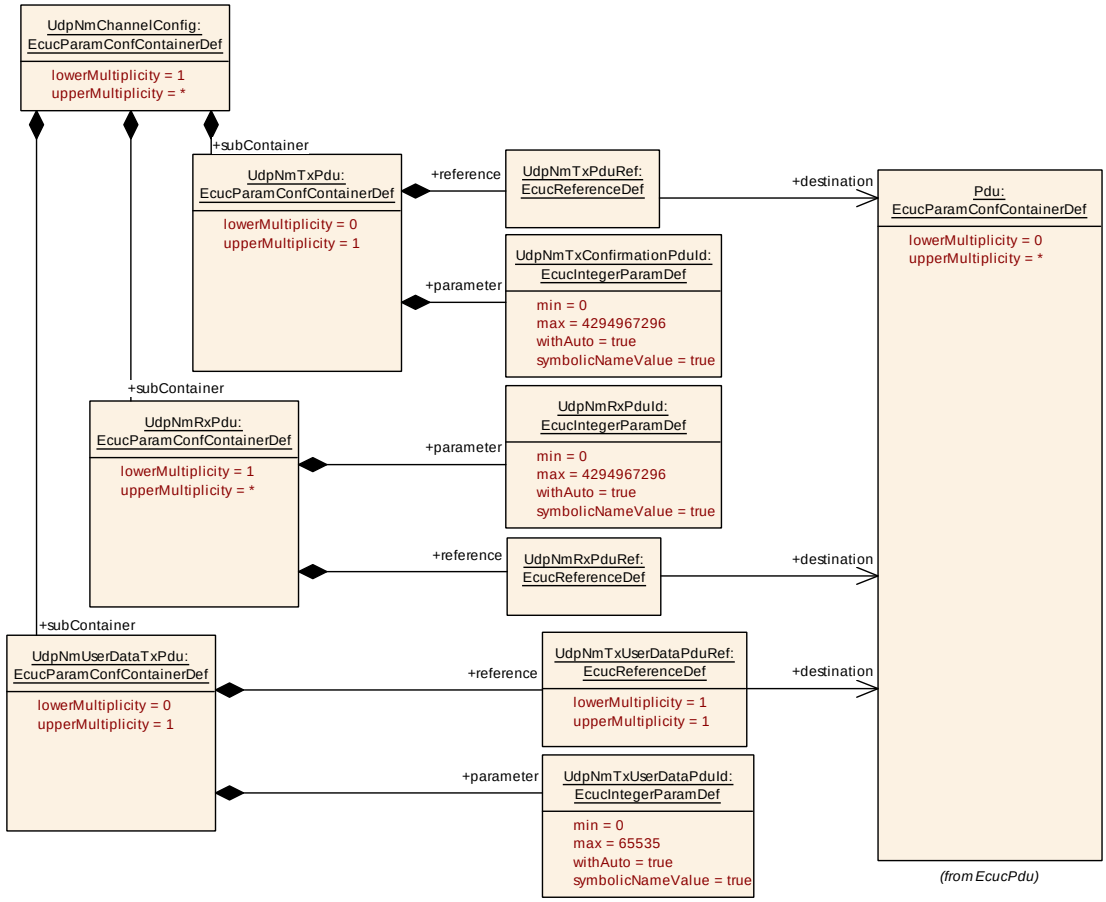

Figure 10.3: UdpNmChannelConfig - part 2

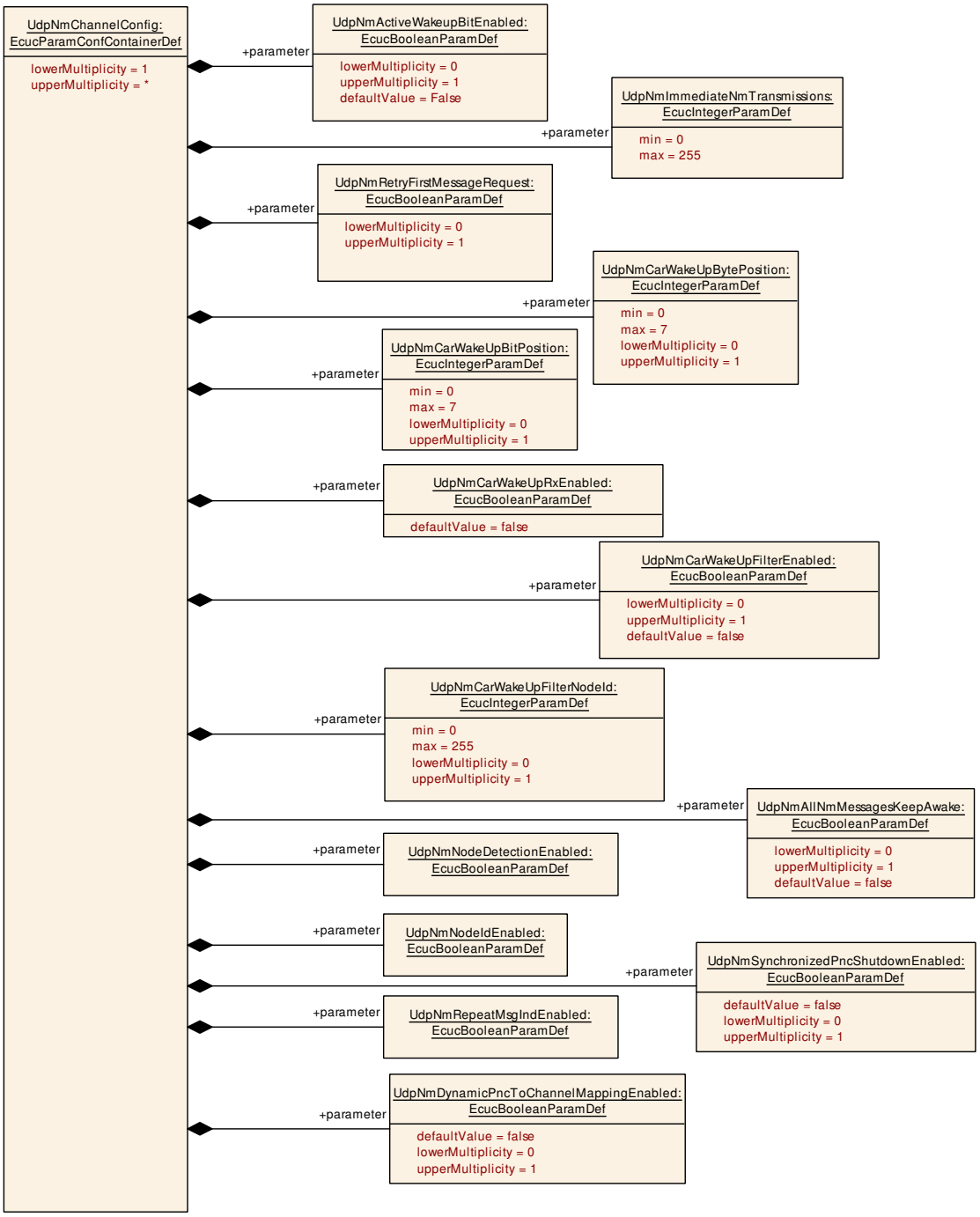

Figure 10.4: UdpNmChannelConfig - part 3

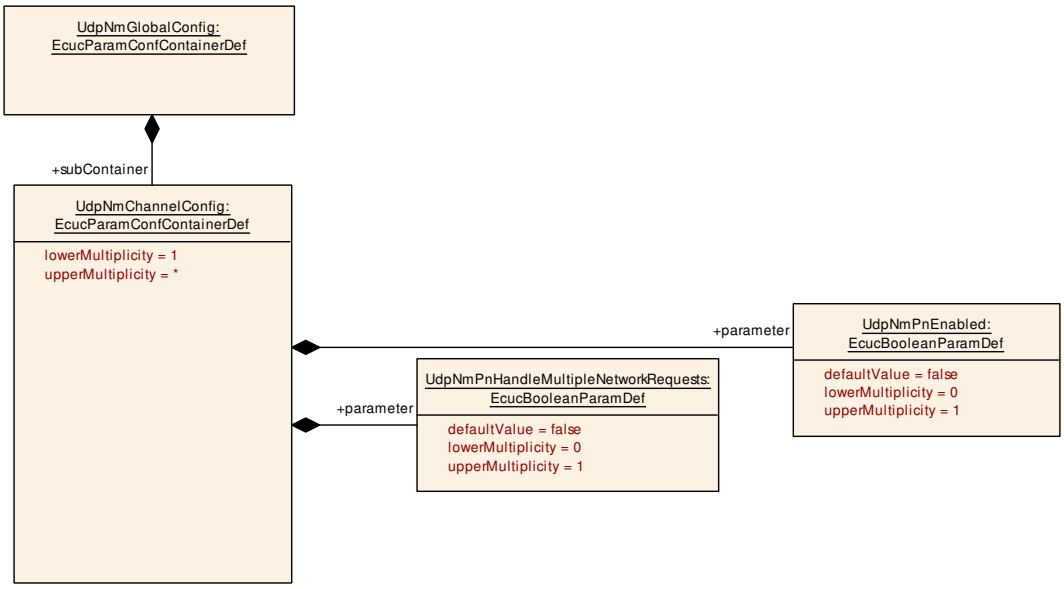

Figure 10.5: UdpNmPnConfig

10.2.4 UdpNmRxPdu

[ECUC_UdpNm_00038] Definition of EcucParamConfContainerDef UdpNmRxPdu ⌈

Container Name	UdpNmRxPdu
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes the UdpNm RX PDU's.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
UdpNmRxPduId	1	[ECUC_UdpNm_00043]
UdpNmRxPduRef	1	[ECUC_UdpNm_00039]

No Included Containers

⌋

[ECUC_UdpNm_00043] Definition of EcucIntegerParamDef UdpNmRxPduId ⌈

Parameter Name	UdpNmRxPduId
Parent Container	UdpNmRxPdu
Upstream mapping defined	–

▽

△

Description	ID of the RxPdu that will be used by a RxIndication of the lower layer.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 4294967296	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00039] Definition of EcucReferenceDef UdpNmRxPduRef ⌈

Parameter Name	UdpNmRxPduRef
Parent Container	UdpNmRxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the UdpNm module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.2.5 UdpNmTxPdu

[ECUC_UdpNm_00036] Definition of EcucParamConfContainerDef UdpNmTxPdu ⌈

Container Name	UdpNmTxPdu
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This container describes the UdpNm TX PDU's.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
UdpNmTxConfirmationPduId	1	[ECUC_UdpNm_00042]
UdpNmTxPduRef	1	[ECUC_UdpNm_00037]

No Included Containers

⌋

[ECUC_UdpNm_00042] Definition of EcucIntegerParamDef UdpNmTxConfirmationPduId ⌈

Parameter Name	UdpNmTxConfirmationPduId	
Parent Container	UdpNmTxPdu	
Upstream mapping defined	–	
Description	Id of the TxPdu that will be used by a TxConfirmation from the lower layer.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 4294967296	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00037] Definition of EcucReferenceDef UdpNmTxPduRef ⌈

Parameter Name	UdpNmTxPduRef	
Parent Container	UdpNmTxPdu	
Upstream mapping defined	–	
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the UdpNm module to derive the PDU Id.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	-	Selectable

⌋

10.2.6 UdpNmUserDataTxPdu

[ECUC_UdpNm_00056] Definition of EcucParamConfContainerDef UdpNmUserDataTxPdu ⌈

Container Name	UdpNmUserDataTxPdu
Parent Container	UdpNmChannelConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Preprocessor switch for enabling the Tx path of Com User Data. Use case: Setting of NMUserData via SWC.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
UdpNmTxUserDataPduId	1	[ECUC_UdpNm_00058]
UdpNmTxUserDataPduRef	1	[ECUC_UdpNm_00057]

No Included Containers

⌋

[ECUC_UdpNm_00058] Definition of EcucIntegerParamDef UdpNmTxUserDataPduId ⌈

Parameter Name	UdpNmTxUserDataPduId	
Parent Container	UdpNmUserDataTxPdu	
Upstream mapping defined	–	
Description	This parameter defines the Handle ID of the NM User Data I-PDU.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_UdpNm_00057] Definition of EcucReferenceDef UdpNmTxUserDataPduRef ⌈

Parameter Name	UdpNmTxUserDataPduRef
Parent Container	UdpNmUserDataTxPdu
Upstream mapping defined	–
Description	Reference to the NM User Data I-PDU in the global PDU collection.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.3 Published Information

For details refer to [3] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[SWS_UdpNm_NA_00999]

Upstream requirements: SRS_BSW_00170, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00336, SRS_BSW_00417

⌈This specification item references requirements that are not applicable to this specification.⌋

[SWS_UdpNm_NA_01000]

Upstream requirements: SRS_BSW_00386, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472

⌈This specification item references requirements that are not applicable, because Udp Nm does not report any DEM errors.⌋

[SWS_UdpNm_NA_01001]

Upstream requirements: SRS_BSW_00490, SRS_BSW_00491

⌈This specification item references requirements that are not applicable, because Udp Nm does not report any security events.⌋

[SWS_UdpNm_NA_01002]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00422, SRS_BSW_00478

⌈This specification item references requirements that are not applicable, because it is no requirement against UdpNm SWS or only against ECUC elements.⌋

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R22-11

B.1.1 Added Advisories in R22-11

none

B.1.2 Changed Advisories in R22-11

none

B.1.3 Deleted Advisories in R22-11

none

B.1.4 Added Constraints in R22-11

none

B.1.5 Changed Constraints in R22-11

none

B.1.6 Deleted Constraints in R22-11

none

B.1.7 Added Specification Items in R22-11

Number	Heading
[SWS_UdpNm_00005]	
[SWS_UdpNm_00006]	
[SWS_UdpNm_00007]	
[SWS_UdpNm_00013]	
[SWS_UdpNm_00014]	
[SWS_UdpNm_00018]	
[SWS_UdpNm_00025]	
[SWS_UdpNm_00026]	
[SWS_UdpNm_00033]	
[SWS_UdpNm_00035]	
[SWS_UdpNm_00037]	
[SWS_UdpNm_00040]	
[SWS_UdpNm_00045]	
[SWS_UdpNm_00051]	
[SWS_UdpNm_00060]	
[SWS_UdpNm_00072]	
[SWS_UdpNm_00074]	
[SWS_UdpNm_00075]	
[SWS_UdpNm_00076]	
[SWS_UdpNm_00081]	

▽

△

Number	Heading
[SWS_UdpNm_00085]	
[SWS_UdpNm_00087]	
[SWS_UdpNm_00088]	
[SWS_UdpNm_00089]	
[SWS_UdpNm_00092]	
[SWS_UdpNm_00093]	
[SWS_UdpNm_00094]	
[SWS_UdpNm_00095]	
[SWS_UdpNm_00096]	
[SWS_UdpNm_00097]	
[SWS_UdpNm_00098]	
[SWS_UdpNm_00099]	
[SWS_UdpNm_00100]	
[SWS_UdpNm_00101]	
[SWS_UdpNm_00102]	
[SWS_UdpNm_00103]	
[SWS_UdpNm_00104]	
[SWS_UdpNm_00105]	
[SWS_UdpNm_00106]	
[SWS_UdpNm_00107]	
[SWS_UdpNm_00108]	

▽

△

Number	Heading
[SWS_UdpNm_00109]	
[SWS_UdpNm_00110]	
[SWS_UdpNm_00111]	
[SWS_UdpNm_00112]	
[SWS_UdpNm_00113]	
[SWS_UdpNm_00114]	
[SWS_UdpNm_00115]	
[SWS_UdpNm_00116]	
[SWS_UdpNm_00117]	
[SWS_UdpNm_00118]	
[SWS_UdpNm_00119]	
[SWS_UdpNm_00120]	
[SWS_UdpNm_00121]	
[SWS_UdpNm_00122]	
[SWS_UdpNm_00123]	
[SWS_UdpNm_00124]	
[SWS_UdpNm_00126]	
[SWS_UdpNm_00127]	
[SWS_UdpNm_00128]	
[SWS_UdpNm_00129]	:
[SWS_UdpNm_00130]	

▽

△

Number	Heading
[SWS_UdpNm_00131]	
[SWS_UdpNm_00132]	
[SWS_UdpNm_00133]	
[SWS_UdpNm_00137]	
[SWS_UdpNm_00138]	
[SWS_UdpNm_00141]	
[SWS_UdpNm_00143]	
[SWS_UdpNm_00144]	
[SWS_UdpNm_00145]	
[SWS_UdpNm_00146]	
[SWS_UdpNm_00147]	
[SWS_UdpNm_00148]	
[SWS_UdpNm_00149]	
[SWS_UdpNm_00150]	
[SWS_UdpNm_00151]	
[SWS_UdpNm_00152]	
[SWS_UdpNm_00153]	
[SWS_UdpNm_00154]	
[SWS_UdpNm_00158]	
[SWS_UdpNm_00159]	
[SWS_UdpNm_00160]	

▽

△

Number	Heading
[SWS_UdpNm_00161]	
[SWS_UdpNm_00162]	
[SWS_UdpNm_00163]	
[SWS_UdpNm_00166]	
[SWS_UdpNm_00168]	
[SWS_UdpNm_00170]	
[SWS_UdpNm_00172]	
[SWS_UdpNm_00173]	
[SWS_UdpNm_00174]	
[SWS_UdpNm_00175]	
[SWS_UdpNm_00176]	
[SWS_UdpNm_00177]	
[SWS_UdpNm_00178]	
[SWS_UdpNm_00179]	
[SWS_UdpNm_00180]	
[SWS_UdpNm_00181]	
[SWS_UdpNm_00185]	
[SWS_UdpNm_00187]	
[SWS_UdpNm_00189]	
[SWS_UdpNm_00190]	
[SWS_UdpNm_00192]	

▽

△

Number	Heading
[SWS_UdpNm_00196]	
[SWS_UdpNm_00197]	
[SWS_UdpNm_00198]	
[SWS_UdpNm_00199]	
[SWS_UdpNm_00202]	
[SWS_UdpNm_00203]	
[SWS_UdpNm_00206]	
[SWS_UdpNm_00208]	
[SWS_UdpNm_00210]	
[SWS_UdpNm_00211]	
[SWS_UdpNm_00213]	
[SWS_UdpNm_00214]	
[SWS_UdpNm_00215]	
[SWS_UdpNm_00216]	
[SWS_UdpNm_00217]	
[SWS_UdpNm_00218]	
[SWS_UdpNm_00219]	
[SWS_UdpNm_00220]	
[SWS_UdpNm_00221]	
[SWS_UdpNm_00223]	
[SWS_UdpNm_00224]	

▽

△

Number	Heading
[SWS_UdpNm_00226]	
[SWS_UdpNm_00227]	
[SWS_UdpNm_00228]	
[SWS_UdpNm_00231]	
[SWS_UdpNm_00234]	
[SWS_UdpNm_00237]	
[SWS_UdpNm_00244]	
[SWS_UdpNm_00246]	
[SWS_UdpNm_00247]	
[SWS_UdpNm_00248]	
[SWS_UdpNm_00249]	
[SWS_UdpNm_00304]	
[SWS_UdpNm_00305]	
[SWS_UdpNm_00306]	
[SWS_UdpNm_00307]	
[SWS_UdpNm_00308]	
[SWS_UdpNm_00309]	
[SWS_UdpNm_00310]	
[SWS_UdpNm_00312]	
[SWS_UdpNm_00313]	
[SWS_UdpNm_00314]	

▽

△

Number	Heading
[SWS_UdpNm_00315]	
[SWS_UdpNm_00316]	
[SWS_UdpNm_00317]	
[SWS_UdpNm_00318]	
[SWS_UdpNm_00320]	
[SWS_UdpNm_00321]	
[SWS_UdpNm_00322]	
[SWS_UdpNm_00324]	
[SWS_UdpNm_00328]	
[SWS_UdpNm_00329]	
[SWS_UdpNm_00330]	
[SWS_UdpNm_00332]	
[SWS_UdpNm_00333]	
[SWS_UdpNm_00334]	
[SWS_UdpNm_00362]	
[SWS_UdpNm_00364]	
[SWS_UdpNm_00366]	
[SWS_UdpNm_00367]	
[SWS_UdpNm_00373]	
[SWS_UdpNm_00374]	
[SWS_UdpNm_00375]	

▽

△

Number	Heading
[SWS_UdpNm_00376]	
[SWS_UdpNm_00378]	
[SWS_UdpNm_00379]	
[SWS_UdpNm_00454]	
[SWS_UdpNm_00462]	
[SWS_UdpNm_00463]	
[SWS_UdpNm_00464]	
[SWS_UdpNm_00465]	
[SWS_UdpNm_00466]	
[SWS_UdpNm_00467]	
[SWS_UdpNm_00468]	
[SWS_UdpNm_00469]	
[SWS_UdpNm_00470]	
[SWS_UdpNm_00471]	
[SWS_UdpNm_00473]	
[SWS_UdpNm_00485]	
[SWS_UdpNm_00486]	
[SWS_UdpNm_00487]	
[SWS_UdpNm_00488]	
[SWS_UdpNm_00491]	
[SWS_UdpNm_00492]	

▽

△

Number	Heading
[SWS_UdpNm_00495]	
[SWS_UdpNm_00496]	
[SWS_UdpNm_00497]	
[SWS_UdpNm_00498]	
[SWS_UdpNm_00499]	
[SWS_UdpNm_00500]	
[SWS_UdpNm_00501]	
[SWS_UdpNm_00502]	
[SWS_UdpNm_00503]	
[SWS_UdpNm_00504]	
[SWS_UdpNm_00505]	
[SWS_UdpNm_00506]	
[SWS_UdpNm_00507]	
[SWS_UdpNm_00508]	
[SWS_UdpNm_91001]	
[SWS_UdpNm_91004]	
[SWS_UdpNm_91006]	
[SWS_UdpNm_91007]	
[SWS_UdpNm_91009]	
[SWS_UdpNm_91010]	
[SWS_UdpNm_NA_00999]	

Table B.1: Added Specification Items in R22-11

B.1.8 Changed Specification Items in R22-11

none

B.1.9 Deleted Specification Items in R22-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R23-11

B.2.1 Added Specification Items in R23-11

Number	Heading
[SWS_UdpNm_00509]	
[SWS_UdpNm_91008]	Definition of callback function UdpNm_RepeatMessageIndication
[SWS_UdpNm_91011]	Definition of imported datatypes of module UdpNm

Table B.2: Added Specification Items in R23-11

B.2.2 Changed Specification Items in R23-11

Number	Heading
[SWS_UdpNm_00013]	
[SWS_UdpNm_00074]	
[SWS_UdpNm_00075]	
[SWS_UdpNm_00088]	
[SWS_UdpNm_00470]	
[SWS_UdpNm_91006]	Definition of optional interfaces in module UdpNm

Table B.3: Changed Specification Items in R23-11

B.2.3 Deleted Specification Items in R23-11

Number	Heading
[SWS_UdpNm_ 00045]	
[SWS_UdpNm_ 00076]	
[SWS_UdpNm_ 00087]	
[SWS_UdpNm_ 00115]	
[SWS_UdpNm_ 00131]	
[SWS_UdpNm_ 00145]	
[SWS_UdpNm_ 00223]	

Table B.4: Deleted Specification Items in R23-11

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

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

none

B.3.3 Deleted Specification Items in R24-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R25-11

B.4.1 Added Specification Items in R25-11

none

B.4.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_UdpNm_00042]	Definition of EcucIntegerParamDef UdpNmTxConfirmationPduId
[ECUC_UdpNm_00043]	Definition of EcucIntegerParamDef UdpNmRxPduId
[SWS_UdpNm_00500]	
[SWS_UdpNm_00501]	
[SWS_UdpNm_00502]	

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

B.4.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_UdpNm_00005]	
[SWS_UdpNm_00237]	
[SWS_UdpNm_00334]	
[SWS_UdpNm_00362]	
[SWS_UdpNm_00454]	

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

B.5 Traceable item history of this document according to AUTOSAR Release R26-11

B.5.1 Added Specification Items in R26-11

none

B.5.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_UdpNm_00001]	Definition of EcucParamConfContainerDef UdpNmGlobalConfig
[ECUC_UdpNm_00002]	Definition of EcucBooleanParamDef UdpNmDevErrorDetect
[ECUC_UdpNm_00004]	Definition of EcucBooleanParamDef UdpNmUserDataEnabled
[ECUC_UdpNm_00005]	Definition of EcucBooleanParamDef UdpNmRemoteSleepIndEnabled
[ECUC_UdpNm_00006]	Definition of EcucBooleanParamDef UdpNmBusSynchronizationEnabled
[ECUC_UdpNm_00009]	Definition of EcucBooleanParamDef UdpNmImmediateRestartEnabled
[ECUC_UdpNm_00010]	Definition of EcucBooleanParamDef UdpNmPassiveModeEnabled
[ECUC_UdpNm_00011]	Definition of EcucBooleanParamDef UdpNmPduRxIndicationEnabled
[ECUC_UdpNm_00012]	Definition of EcucBooleanParamDef UdpNmStateChangeIndEnabled
[ECUC_UdpNm_00013]	Definition of EcucBooleanParamDef UdpNmComControlEnabled
[ECUC_UdpNm_00014]	Definition of EcucIntegerParamDef UdpNmNumberOfChannels
[ECUC_UdpNm_00017]	Definition of EcucParamConfContainerDef UdpNmChannelConfig
[ECUC_UdpNm_00018]	Definition of EcucReferenceDef UdpNmComMNetworkHandleRef
[ECUC_UdpNm_00020]	Definition of EcucFloatParamDef UdpNmTimeoutTime
[ECUC_UdpNm_00021]	Definition of EcucFloatParamDef UdpNmWaitBusSleepTime
[ECUC_UdpNm_00022]	Definition of EcucFloatParamDef UdpNmRepeatMessageTime
[ECUC_UdpNm_00023]	Definition of EcucFloatParamDef UdpNmRemoteSleepIndTime
[ECUC_UdpNm_00025]	Definition of EcucEnumerationParamDef UdpNmPduNidPosition
[ECUC_UdpNm_00026]	Definition of EcucEnumerationParamDef UdpNmPduCbvPosition
[ECUC_UdpNm_00028]	Definition of EcucFloatParamDef UdpNmMsgCycleTime

▽

△

Number	Heading
[ECUC_UdpNm_00029]	Definition of EcucFloatParamDef UdpNmMsgCycleOffset
[ECUC_UdpNm_00031]	Definition of EcucIntegerParamDef UdpNmNodeId
[ECUC_UdpNm_00032]	Definition of EcucFloatParamDef UdpNmMainFunctionPeriod
[ECUC_UdpNm_00036]	Definition of EcucParamConfContainerDef UdpNmTxPdu
[ECUC_UdpNm_00037]	Definition of EcucReferenceDef UdpNmTxPduRef
[ECUC_UdpNm_00038]	Definition of EcucParamConfContainerDef UdpNmRxPdu
[ECUC_UdpNm_00039]	Definition of EcucReferenceDef UdpNmRxPduRef
[ECUC_UdpNm_00042]	Definition of EcucIntegerParamDef UdpNmTxConfirmationPduId
[ECUC_UdpNm_00043]	Definition of EcucIntegerParamDef UdpNmRxPduId
[ECUC_UdpNm_00055]	Definition of EcucBooleanParamDef UdpNmComUserDataSupport
[ECUC_UdpNm_00056]	Definition of EcucParamConfContainerDef UdpNmUserDataTxPdu
[ECUC_UdpNm_00057]	Definition of EcucReferenceDef UdpNmTxUserDataPduRef
[ECUC_UdpNm_00058]	Definition of EcucIntegerParamDef UdpNmTxUserDataPduId
[ECUC_UdpNm_00059]	Definition of EcucBooleanParamDef UdpNmCoordinatorSyncSupport
[ECUC_UdpNm_00061]	Definition of EcucBooleanParamDef UdpNmPnEnabled
[ECUC_UdpNm_00063]	Definition of EcucBooleanParamDef UdpNmPnHandleMultipleNetwork Requests
[ECUC_UdpNm_00074]	Definition of EcucBooleanParamDef UdpNmActiveWakeupBitEnabled
[ECUC_UdpNm_00075]	Definition of EcucIntegerParamDef UdpNmImmediateNmTransmissions
[ECUC_UdpNm_00076]	Definition of EcucBooleanParamDef UdpNmCarWakeUpRxEnabled
[ECUC_UdpNm_00077]	Definition of EcucBooleanParamDef UdpNmCarWakeUpFilterEnabled
[ECUC_UdpNm_00078]	Definition of EcucIntegerParamDef UdpNmCarWakeUpFilterNodeId
[ECUC_UdpNm_00079]	Definition of EcucFloatParamDef UdpNmImmediateNmCycleTime

▽

△

Number	Heading
[ECUC_UdpNm_00085]	Definition of EcucBooleanParamDef UdpNmRetryFirstMessageRequest
[ECUC_UdpNm_00086]	Definition of EcucIntegerParamDef UdpNmCarWakeUpBytePosition
[ECUC_UdpNm_00087]	Definition of EcucIntegerParamDef UdpNmCarWakeUpBitPosition
[ECUC_UdpNm_00088]	Definition of EcucModuleDef UdpNm
[ECUC_UdpNm_00089]	Definition of EcucBooleanParamDef UdpNmAllNmMessagesKeepAwake
[ECUC_UdpNm_00090]	Definition of EcucBooleanParamDef UdpNmNodeDetectionEnabled
[ECUC_UdpNm_00091]	Definition of EcucBooleanParamDef UdpNmNodeIdEnabled
[ECUC_UdpNm_00092]	Definition of EcucBooleanParamDef UdpNmRepeatMsgIndEnabled
[ECUC_UdpNm_00093]	Definition of EcucBooleanParamDef UdpNmStayInPbsEnabled
[ECUC_UdpNm_00094]	Definition of EcucBooleanParamDef UdpNmDynamicPncToChannelMapping Support
[ECUC_UdpNm_00095]	Definition of EcucBooleanParamDef UdpNmDynamicPncToChannelMapping Enabled
[ECUC_UdpNm_00097]	Definition of EcucBooleanParamDef UdpNmSynchronizedPncShutdown Enabled
[SWS_UdpNm_00228]	Definition of callback function UdpNm_SoAdIfTxConfirmation
[SWS_UdpNm_91006]	Definition of optional interfaces requested by module UdpNm
[SWS_UdpNm_91011]	Definition of imported datatypes of module UdpNm

Table B.7: Changed Specification Items in R26-11

B.5.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_UdpNm_00003]	Definition of EcucBooleanParamDef UdpNmVersionInfoApi
[SWS_UdpNm_00224]	Definition of API function UdpNm_GetVersionInfo
[SWS_UdpNm_00318]	

▽

△

Number	Heading
[SWS_UdpNm_ 91008]	Definition of callback function UdpNm_RepeatMessageIndication

Table B.8: Deleted Specification Items in R26-11