

<b>Document Title</b>	Specification of UDP Network Management
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	414
<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R19-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Det error handling corrected</li> <li>• Harmonization of API</li> <li>• Minor corrections</li> <li>• Changed Document Status from Final to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Header file cleanup</li> <li>• Minor corrections</li> </ul>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Node Detection Configuration per channel</li> <li>• Det error handling corrected</li> <li>• Bug fixes and editorial changes</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Added Trigger Transmit feature</li> <li>• Car Wakeup support completed</li> <li>• Immediate TX Transmission corrected</li> <li>• Editorial changes</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Revised Error Classification</li> <li>• Added support for Car Wakeup</li> <li>• Bug fixes and editorial changes</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Harmonization of API description</li> <li>• Revised Partial Networking Requirements</li> <li>• Extended Production Errors</li> <li>• Editorial Changes</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Minor bug fixes</li> <li>• Editorial Changes</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• Revised Spontaneous Transmission</li><li>• Editorial changes</li><li>• Removed chapter(s) on change documentation</li></ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Added support for Partial Networking</li><li>• Added updated production errors</li><li>• Editorial changes</li></ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Support coordinated shutdown</li><li>• New traceability mechanism</li></ul>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"><li>• ComStack Harmonization</li><li>• Harmonization of NM interfaces</li></ul>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Initial Release</li></ul>

## 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 .....	7
2	Acronyms and abbreviations .....	8
3	Related documentation.....	9
3.1	Input documents.....	9
3.2	Related standards and norms .....	10
3.3	Related specification .....	10
4	Constraints and assumptions .....	11
4.1	Limitations .....	11
4.2	Applicability to car domains.....	11
5	Dependencies on other modules.....	12
5.1	File Structure .....	12
5.1.1	Code File Structure.....	12
6	Requirements traceability .....	13
7	Functional specification .....	17
7.1	Coordination algorithm .....	17
7.2	Operational Modes .....	18
7.2.1	Network Mode.....	18
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	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).....	29
7.7.5	Communication Control (optional).....	29
7.7.6	NM Coordinator synchronization support (optional) .....	30
7.8	Partial Networking .....	31
7.8.1	Rx Handling of NM PDUs .....	31
7.8.2	Tx Handling of NM PDUs.....	31
7.8.3	NM PDU Filter Algorithm.....	31
7.8.4	Aggregation of Internal and External Requested Partial Networks .....	32
7.8.5	Aggregation of External Requested Partial Networks .....	34
7.8.6	Spontaneous Transmission of NM-PDUs via UdpNm_NetworkRequest.....	35
7.9	Payload (PDU) Structure .....	36
7.10	Functional requirements on UdpNm API.....	37

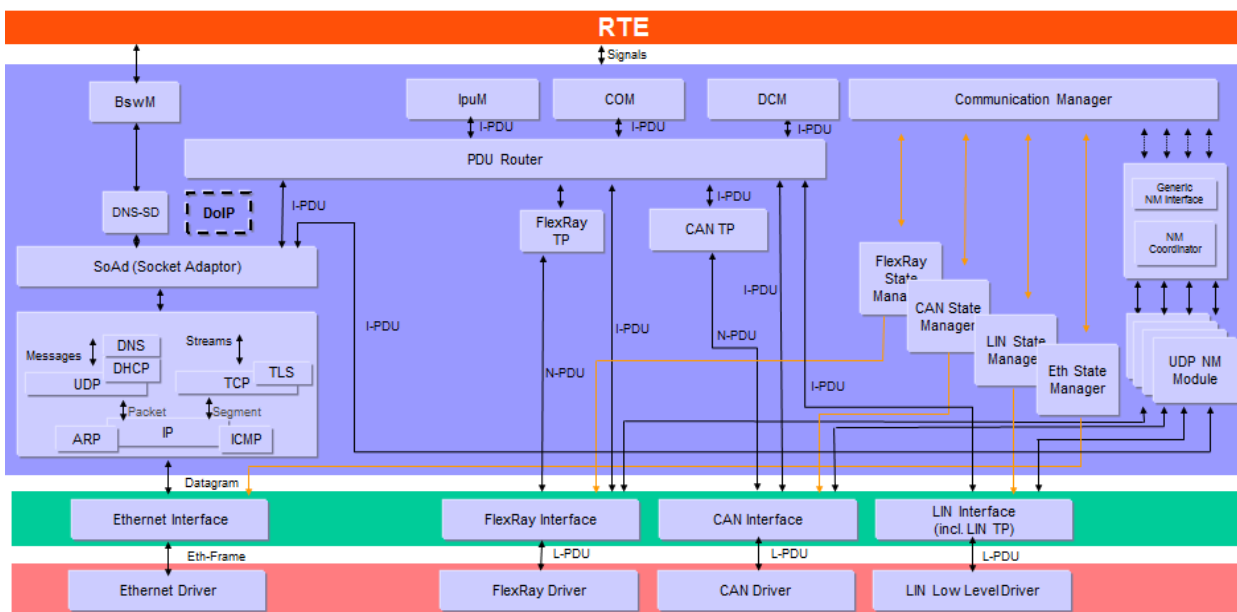
7.11	Car Wakeup .....	38
7.12	Error Classification .....	38
7.12.1	Development Errors.....	39
7.12.2	Run Time Errors .....	39
7.12.3	Transient Faults.....	39
7.12.4	Production Errors.....	39
7.12.5	Extended Production Errors.....	40
7.12.6	Error detection .....	40
7.12.7	Error notification .....	40
7.13	Scheduling of the main function .....	41
7.14	Application notes.....	41
7.14.1	Wakeup notification .....	41
7.14.2	Coordination of coupled networks .....	41
7.15	Version check.....	41
7.16	Parameter check .....	41
8	API specification.....	42
8.1	Imported Types .....	42
8.2	Type Definitions.....	42
8.2.1	UdpNm_ConfigType .....	42
8.2.2	UdpNm_PduPositionType.....	43
8.3	Function definitions.....	43
8.3.1	UdpNm_Init.....	43
8.3.2	UdpNm_PassiveStartUp .....	44
8.3.3	UdpNm_NetworkRequest .....	45
8.3.4	UdpNm_NetworkRelease .....	45
8.3.5	UdpNm_DisableCommunication .....	46
8.3.6	UdpNm_EnableCommunication.....	47
8.3.7	UdpNm_SetUserData .....	48
8.3.8	UdpNm_GetUserData.....	48
8.3.9	UdpNm_GetNodeIdentifier.....	49
8.3.10	UdpNm_GetLocalNodeIdentifier.....	50
8.3.11	UdpNm_RepeatMessageRequest .....	50
8.3.12	UdpNm_GetPduData.....	51
8.3.13	UdpNm_GetState .....	52
8.3.14	UdpNm_GetVersionInfo.....	53
8.3.15	UdpNm_RequestBusSynchronization.....	53
8.3.16	UdpNm_CheckRemoteSleepIndication .....	54
8.3.17	UdpNm_SetSleepReadyBit .....	55
8.3.18	UdpNm_Transmit .....	55
8.4	Call-back notifications.....	56
8.4.1	UdpNm_SoAdIfTxConfirmation.....	56
8.4.2	UdpNm_SoAdIfRxIndication .....	57
8.4.3	UdpNm_SoAdIfTriggerTransmit.....	58
8.5	Scheduled Functions.....	59
8.5.1	UdpNm_MainFunction_<Instance Id>.....	59
8.6	Expected Interfaces.....	59
8.6.1	Mandatory Interfaces .....	59
8.6.2	Optional Interfaces.....	61
8.6.3	Configurable interfaces.....	62

8.7	Service Interfaces.....	62
8.8	UML State chart diagram.....	63
9	Sequence diagrams and Transition Tables .....	64
9.1	UdpNmTransmission .....	64
9.2	UdpNm Reception .....	64
10	Configuration specification.....	66
10.1	How to read this chapter .....	66
10.2	Containers and configuration parameters .....	66
10.2.1	UdpNm .....	67
10.2.2	UdpNmGlobalConfig.....	67
10.2.3	UdpNmChannelConfig.....	74
10.2.4	UdpNmRxPdu.....	88
10.2.5	UdpNmTxPdu .....	89
10.2.6	UdpNmUserDataTxPdu .....	90
10.2.7	UdpNmPnInfo .....	91
10.2.8	UdpNmPnFilterMaskByte .....	93
10.3	Published parameters .....	93
11	Not applicable requirements .....	94

# 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). 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 [9].



**Figure 1: Extended AUTOSAR Communication Stack.**

## 2 Acronyms and abbreviations

<b>Acronym or Abbreviation:</b>	<b>Description:</b>
<b>API</b>	Application Programming Interface
<b>BSW</b>	Basic Software
<b>CWU</b>	Car Wakeup
<b>EthIf</b>	Ethernet Interface
<b>DET</b>	Default Error Tracer
<b>IP</b>	Internet Protocol
<b>NM</b>	Network Management
<b>PDU</b>	Protocol Data Unit
<b>SDU</b>	Service Data Unit
<b>TCP</b>	Transmission Control Protocol
<b>TCP/IP</b>	A family of communication protocols used in computer networks
<b>UDP</b>	User Datagram Protocol
<b>PNI</b>	Partial Network Information
<b>UdpNm</b>	UDP Network Management

<b>Term:</b>	<b>Description:</b>
<b>PDU transmission ability is disabled</b>	This means that the NM message transmission has been disabled by the optional service UdpNm_DisableCommunication.
<b>Repeat Message Request Bit Indication</b>	UdpNm_SoAdIfRxIndication finds the Repeat Message Bit set in the Control Bit Vector of a received NM message.
<b>NM PDU</b>	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.
<b>NM Packet</b>	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.
<b>NM Message</b>	Most abstract term referring to any single information item transferred within the methodology of the NM algorithm.
<b>Bus-Off state</b>	Refers to a situation where no cable is connected to the Ethernet HW.



### 3 Related documentation

#### 3.1 Input documents

- [1] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
  
- [2] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
  
- [3] Requirements on Network Management  
AUTOSAR\_SRS\_NetworkManagement.pdf
  
- [4] Specification of Ethernet Interface  
AUTOSAR\_SWS\_EthernetInterface.pdf
  
- [5] Specification of FlexRay Network Management  
AUTOSAR\_SWS\_FlexRayNetworkManagement.pdf
  
- [6] Specification of Communication Stack Types  
AUTOSAR\_SWS\_CommunicationStackTypes.pdf
  
- [7] Specification of ECU Configuration  
AUTOSAR\_TPS\_ECUConfiguration.pdf
  
- [8] Specification of BSW Scheduler  
AUTOSAR\_SWS\_BSW\_Scheduler.pdf
  
- [9] Specification of Generic Network Management Interface  
AUTOSAR\_SWS\_NetworkManagementInterface.pdf
  
- [10] Specification of Communication Manager  
AUTOSAR\_SWS\_ComManager.pdf
  
- [11] Specification of ECU State Manager  
AUTOSAR\_SWS\_ECUCStateManager.pdf
  
- [12] Specification of Operating System  
AUTOSAR\_SWS\_OS.pdf
  
- [13] Specification of Default Error Tracer  
AUTOSAR\_SWS\_Default ErrorTracer.pdf
  
- [14] Specification of Standard Types  
AUTOSAR\_SWS\_StandardTypes.pdf
  
- [15] Specification of Platform Types  
AUTOSAR\_SWS\_PlatformTypes.pdf

- [16] Specification of Compiler Abstraction  
AUTOSAR\_SWS\_CompilerAbstraction.pdf
- [17] Basic Software Module Description Template  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [18] Specification of Socket Adaptor  
AUTOSAR\_SWS\_SocketAdaptor.pdf
- [19] Requirements on Ethernet  
AUTOSAR\_SRS\_Ethernet.pdf
- [20] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList
- [21] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related standards and norms

- [22] IEEE  
<http://www.opengroup.org/onlinepubs/000095399/>
- [23] ISO 14229 Road Vehicles – Unified Diagnostic Services (UDS)

### 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [21] (SWS BSW General), which is also valid for UDP Network Management.

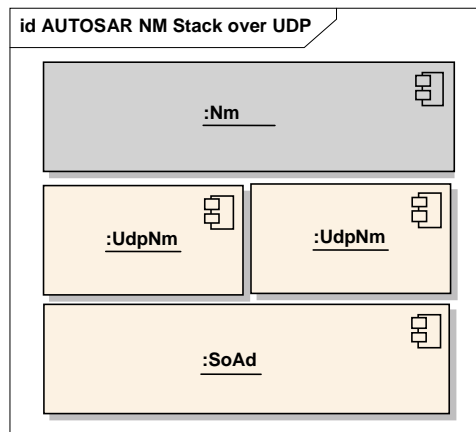
Thus, the specification SWS BSW General shall be considered as additional and required specification for UDP Network Management.

## 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 2 presents an AUTOSAR NM stack within an example ECU belonging to two UDP NM-clusters.



**Figure 2: AUTOSAR NM stack within an example ECU belonging to two UDP NM-clusters**

[SWS\_UdpNm\_00131] The AUTOSAR UdpNm algorithm shall support up to 250 nodes per NM-Cluster by default.

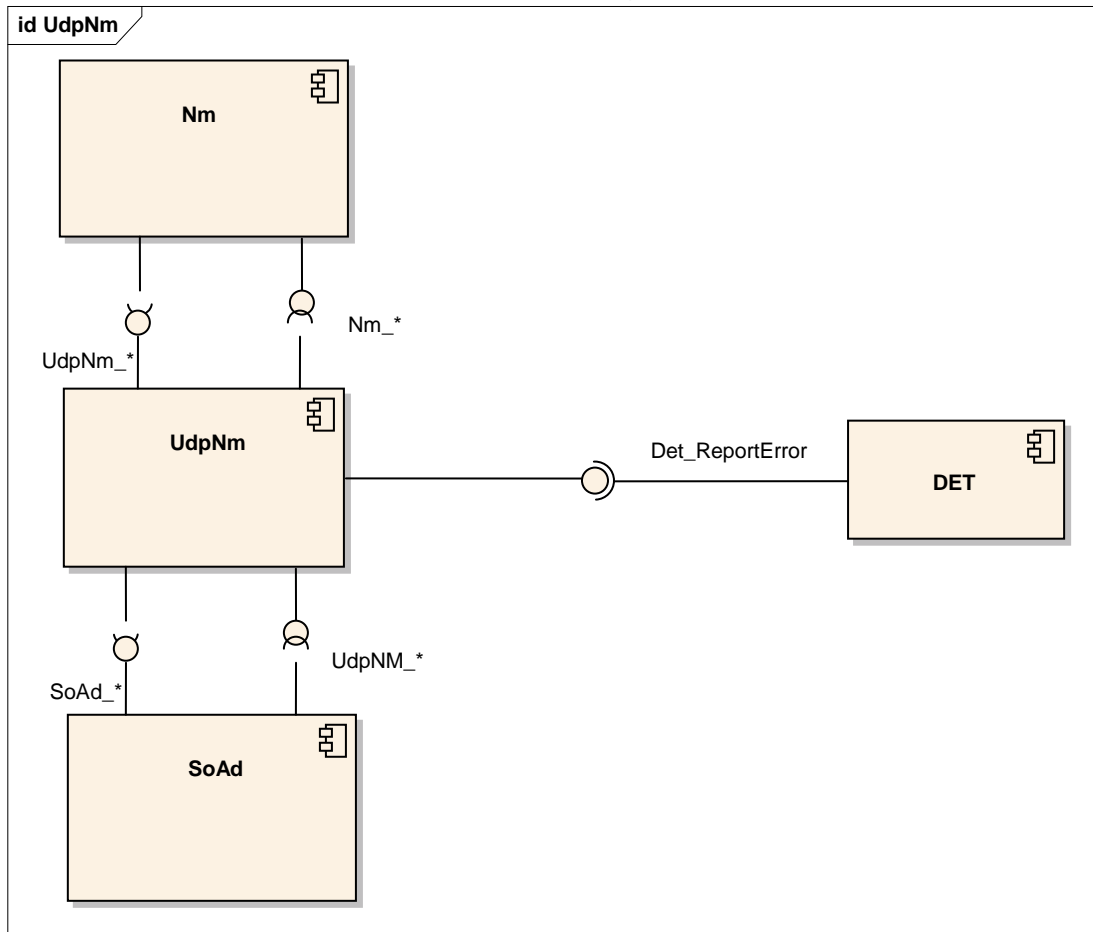
Note: The AUTOSAR UdpNm algorithm can support an arbitrary number of nodes per NM-cluster (even more than default 250 nodes per cluster, if necessary) – it is only a matter of configuration, since the upper limit is not fixed and depends on the trade off between response time, fault-tolerance and resulted bus load configured for the AUTOSAR UdpNm coordination algorithm. This might depend on the physical layer used. ] ( )

### 4.2 Applicability to car domains

N/A

## 5 Dependencies on other modules

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



**Figure 3: Dependencies on other modules.**

### 5.1 File Structure

#### 5.1.1 Code File Structure

[SWS\_UdpNm\_00081] 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.  
] (SRS\_BSW\_00419, SRS\_BSW\_00346, SRS\_BSW\_00158, SRS\_BSW\_00308)

## 6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00005	Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_UdpNm_NA_00999
SRS_BSW_00006	The source code of software modules above the $\mu$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_UdpNm_NA_00999
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_UdpNm_NA_00999
SRS_BSW_00158	-	SWS_UdpNm_00081
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_UdpNm_NA_00999
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_UdpNm_NA_00999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_UdpNm_NA_00999
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_UdpNm_NA_00999
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_UdpNm_NA_00999
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_UdpNm_NA_00999
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_UdpNm_NA_00999
SRS_BSW_00305	Data types naming convention	SWS_UdpNm_NA_00999
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_UdpNm_NA_00999
SRS_BSW_00307	Global variables naming convention	SWS_UdpNm_NA_00999

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_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_UdpNm_NA_00999
SRS_BSW_00312	Shared code shall be reentrant	SWS_UdpNm_NA_00999
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_UdpNm_NA_00999
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_UdpNm_NA_00999
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_UdpNm_NA_00999
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_UdpNm_NA_00999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_UdpNm_NA_00999
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_UdpNm_NA_00999
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_UdpNm_NA_00999
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_UdpNm_NA_00999
SRS_BSW_00335	Status values naming convention	SWS_UdpNm_NA_00999
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_UdpNm_NA_00999
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_UdpNm_NA_00999
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_UdpNm_00081
SRS_BSW_00347	A Naming separation of different instances of BSW drivers shall be in place	SWS_UdpNm_NA_00999
SRS_BSW_00375	Basic Software Modules shall	SWS_UdpNm_NA_00999

	report wake-up reasons	
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_UdpNm_NA_00999
SRS_BSW_00410	Compiler switches shall have defined values	SWS_UdpNm_NA_00999
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_UdpNm_NA_00999
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_UdpNm_NA_00999
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_UdpNm_NA_00999
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_UdpNm_NA_00999
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_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_UdpNm_NA_00999
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_UdpNm_NA_00999
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_UdpNm_NA_00999
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_UdpNm_NA_00999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_UdpNm_NA_00999
SRS_BSW_00429	Access to OS is restricted	SWS_UdpNm_NA_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_UdpNm_NA_00999
SRS_Nm_00046	It shall be possible to trigger the startup of all Nodes at any Point in Time.	SWS_UdpNm_NA_00999
SRS_Nm_00050	The NM shall provide the current state of NM	SWS_UdpNm_NA_00999
SRS_Nm_00052	The NM interface shall signal to	SWS_UdpNm_NA_00999

	the application that all other ECUs are ready to sleep.	
SRS_Nm_00054	There shall be a deterministic time from the point where all nodes agree to go to bus sleep to the point where bus is switched off.	SWS_UdpNm_NA_00999
SRS_Nm_00137	NM shall perform communication system error handling for errors that have impact on the NM behavior.	SWS_UdpNm_00379, SWS_UdpNm_00466, SWS_UdpNm_00467
SRS_Nm_00142	NM shall guarantee an upper limit for the bus load generated by NM itself.	SWS_UdpNm_NA_00999
SRS_Nm_00144	NM shall support communication clusters of up to 64 ECUs	SWS_UdpNm_NA_00999
SRS_Nm_00147	The NM algorithm shall be processor independent.	SWS_UdpNm_NA_00999
SRS_Nm_00151	The Network Management algorithm shall allow any node to integrate into an already running NM cluster	SWS_UdpNm_NA_00999
SRS_Nm_00153	The Network Management shall optionally provide a possibility to detect present nodes	SWS_UdpNm_00014, SWS_UdpNm_00111, SWS_UdpNm_00112, SWS_UdpNm_00113, SWS_UdpNm_00119, SWS_UdpNm_00120, SWS_UdpNm_00121, SWS_UdpNm_NA_00999
SRS_Nm_00154	The Network Management API shall be independent from the communication bus	SWS_UdpNm_NA_00999
SRS_Nm_02503	The NM API shall optionally give the possibility to send user data	SWS_UdpNm_00315, SWS_UdpNm_00317, SWS_UdpNm_00377, SWS_UdpNm_00464
SRS_Nm_02509	The NM interface shall signal to the application that at least one other ECUs is not ready to sleep anymore.	SWS_UdpNm_NA_00999
SRS_Nm_02512	The NM shall give the possibility to enable or disable the network management related communication configured for an active NM node	SWS_UdpNm_00178, SWS_UdpNm_00215, SWS_UdpNm_00216



## 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 broadcast 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 [10].

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

[SWS\_UdpNm\_00087] [ Every network node shall transmit periodic NM PDUs as long as it requires bus-communication; otherwise it shall not transmit NM PDUs. ] ()

[SWS\_UdpNm\_00088] [ If `UdpNmStayInPbsEnabled` is disabled and bus communication in a UdpNm cluster is released and there are no Network Management PDUs on the bus for a configurable amount of time determined by `UdpNmTimeoutTime + UdpNmWaitBusSleepTime` (both configuration parameters) transition into the Bus-Sleep Mode shall be performed. ] ()

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**

## 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. ] ()

### 7.2.1.1 Repeat Message State

For nodes that are not in passive mode (refer to chapter 7.7.3) 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\_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`. ] ()

### 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. ] (SRS\_Nm\_00153)

[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. ] (SRS\_Nm\_00153)

[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. ] (SRS\_Nm\_00153)

### 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 transmitted, no action is required.

[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. ] (SRS\_Nm\_00153)

[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. ] (SRS\_Nm\_00153)

[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. ] (SRS\_Nm\_00153)

## 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\_00115] [ If `UdpNmStayInPbsEnabled` is disabled `UdpNm` shall stay in the Prepare Bus-Sleep Mode for a configurable amount of time determined by the `UdpNmWaitBusSleepTime` (configuration parameter); after that time the Prepare Bus-Sleep Mode shall be left and the Bus-Sleep Mode shall be entered.] ()

**Note:** This requirement 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 (e.g. see following requirements).

[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 inconsistencies 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 UdpNm 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\_00145] [ If AUTOSAR UdpNm is not initialized it shall not prohibit bus traffic. ] ()

[SWS\_UdpNm\_00060] [ The function `UdpNm_Init` shall select the active configuration set by means of a configuration pointer parameter being passed (see [8.3.1](#)). ] ()

[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\_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 chapter [10.2](#)). ] ()

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`.

[SWS\_UdpNm\_00237] [ The `UdpNm` module shall provide the periodic transmission mode. In this transmission mode the `UdpNm` module shall send Network Management PDUs periodically. ] ()

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

[SWS\_UdpNm\_00005] [ If the Repeat Message State is not entered via `UdpNm_NetworkRequest` OR `UdpNmImmediateNmTransmissions` is zero the transmission of NM PDU shall be delayed by `UdpNmMsgCycleOffset` after entering the repeat message state. ] ()

Note: This requirement covers also the case if Repeat Message State is entered from Network Operation State or Ready Sleep State due to Repeat Message Request or Bit (see [SWS\\_UdpNm\\_00111](#), [SWS\\_UdpNm\\_00112](#), [SWS\\_UdpNm\\_00119](#), [SWS\\_UdpNm\\_00120](#)). This means that in this case the immediate transmission is not used (even if `UdpNmImmediateNmTransmissions` > 0 and independent from configuration of `UdpNmPnHandleMultipleNetworkRequests`) i.e. `UdpNmMsgCycleOffset` will always be applied. This mechanism prevents bursts of NM messages.

[SWS\_UdpNm\_00334] [ When entering the Repeat Message State from Bus Sleep Mode or Prepare Bus Sleep Mode because of `UdpNm_NetworkRequest()` (active wakeup) and if `UdpNmImmediateNmTransmissions` is greater zero, the NM PDUs shall be transmitted using `UdpNmImmediateNmCycleTime` as cycle time. The transmission of the first NM PDU shall be triggered as soon as possible. After the transmission the Message Cycle Timer shall be reloaded with `UdpNmImmediateNmCycleTime`. The `UdpNmMsgCycleOffset` shall not be 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\_00454] [ If `UdpNmPnHandleMultipleNetworkRequests` is set to TRUE `UdpNm_NetworkRequest` shall trigger a state transition from Network Mode to Repeat Message state. If PDU transmission ability is enabled the NM PDUs shall be transmitted using `UdpNmImmediateNmCycleTime` as cycle time. The transmission of the first NM PDU shall be triggered as soon as possible. After the transmission the Message Cycle Timer shall be reloaded with `UdpNmImmediateNmCycleTime`. The `UdpNmMsgCycleOffset` shall not be applied in this case. ] ()

Note: `UdpNmImmediateNmTransmissions` has to be greater zero in this case due to [ECUC UdpNm 00075](#).

[SWS\_UdpNm\_00330] [ If NM PDUs shall be transmitted with `UdpNmImmediateNmCycleTime` (See [SWS UdpNm 00334](#) and [SWS UdpNm 00454](#)), `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\_00032] [ If transmission of NM PDUs has been started and the `UdpNm` Message Cycle Timer expires an NM PDU shall be transmitted through the SoAd by calling `SoAd_IfTransmit`. ] ()

[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`. ] (SRS\_Nm\_00137)

## 7.6.2 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. ] (SRS\_Nm\_02503)

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 [8.4.3 UdpNm\\_SoAdIfTriggerTransmit](#)).

[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 user data. ] (SRS\_Nm\_02503)

### 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 (configuration parameter <code>UdpNmBusSynchronizationEnabled</code> )	Synchronization
Remote (configuration parameter <code>UdpNmRemoteSleepIndEnabled</code> )	Sleep Indication
Node (configuration parameter <code>UdpNmNodeDetectionEnabled</code> ) ] ()	Detection

### 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. ] (`SRS_Nm_02512`)

[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 `UdpNmCoordinatorSyncSupport` 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

### 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 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\_00331] [ If `UdpNmPnEnabled` is `TRUE` and the PNI bit in the received NM-PDU is 1, `UdpNm` module shall process the Partial Networking Information of the NM-PDU as described in chapter 7.8.3 to 7.8.5.] ()

### 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.] ()

### 7.8.3 NM PDU Filter Algorithm

[SWS\_UdpNm\_00335] [ The range (in bytes) that contains the PN request information (PN Info Range) in the received NM-PDU is defined by `UdpNmPnInfoOffset` (in bytes) starting from byte 0 and `UdpNmPnInfoLength` (in bytes). This range is called PN Info Range.] ()

**Example:**

- UdpNmPnInfoOffset = 3
- UdpNmPnInfoLength = 2

Only Byte 3 and Byte 4 of the NM message contains PN request information

[SWS\_UdpNm\_00336] [ Every bit of the PN Info Range represents one Partial Network. If the bit is set to 1 the Partial Network is requested. If the bit is set to 0 there is no request for this PN.] ()

[SWS\_UdpNm\_00337] [ By means of the configuration parameter `UdpNmPnFilterMaskByte` the UdpNm is able to detect which PN is relevant for the ECU and which not.

Each bit of `UdpNmPnFilterMaskByte` has the following meaning:

- 0 The PN request is irrelevant for the ECU. The communication stack of the ECU is not kept awake if this bit is set in a received NM-PDU.
- 1 The PN request is relevant for the ECU. The communication stack of the ECU is kept awake if this bit is set in a received NM-PDU.] ()

[SWS\_UdpNm\_00338] [ Each PN filter mask byte shall be mapped (bitwise AND) to the corresponding byte in the PN info range of the NM message. ] ()

[SWS\_UdpNm\_00339] [ If at least one bit within the PN Info Range of the received NM-PDU matches with a bit in the NM filter mask the PN request information is relevant for the ECU] ()

[SWS\_UdpNm\_00460] [ If no relevant PN is requested in the received NM-PDU and `UdpNmAllNmMessagesKeepAwake` is `FALSE` the PDU shall be dropped from further processing. ] ()

[SWS\_UdpNm\_00461] [ If no relevant PN is requested in the received NM-PDU and `UdpNmAllNmMessagesKeepAwake` is `TRUE` the PDU shall not be dropped from further Rx Indication handling. ] ()

#### 7.8.4 Aggregation of Internal and External Requested Partial Networks

**Note:** This feature is used by every ECU that has to switch I-PDU-Groups because of the activity of partial networks. (e.g. to prevent false timeouts) I-PDU-Groups shall be switched on if the corresponding PN is requested internally or externally. I-PDU-Groups shall not be switched off until all internal and external requests for the corresponding PN are released.

The logic for switching the IPDU-Groups is implemented by ComM. The UdpNm only provides the information if a PN is requested or not. The COM module is used to transfer the data to the upper layers.

To switch the I-PDU-Groups synchronously on all direct connected ECUs, UdpNm shall provide the information of a request change to the upper layer at (almost) the



same time on every ECU. This is why the reset timer is restarted on every received and every sent NM message (see below).

The aggregated state of the internal/external requested PNs is called External Internal Requests Aggregated (EIRA).

[SWS\_UdpNm\_00344] [ If `UdpNmPnEiraCalcEnabled` is TRUE, the `UdpNm` shall provide the possibility to store external and internal requested PNs combined over all relevant channels (all `UdpNm` Channels where `UdpNmPnEnabled` is TRUE). At initialization the values of all PNs shall be set to 0 (not requested) ] ()

[SWS\_UdpNm\_00347] [ If

- `UdpNmPnEiraCalcEnabled` is TRUE
- a NM-PDU is received
- PNs are requested within this message (bits are set to 1)
- And the requested PNs are set to 1 within the [configured PN filter mask] then `UdpNm` shall store the request information (value 1) for these PNs ] ()

[SWS\_UdpNm\_00348] [ If

- `UdpNmPnEiraCalcEnabled` is TRUE
- NM-PDU is being requested to send by `UdpNm`
- PNs are requested within this message(bits are set to 1)
- And the requested PNs are set to 1 within the [configured PN filter mask] then `UdpNm` shall store the request information (value 1) for these PNs.] ()

[SWS\_UdpNm\_00345] [ If `UdpNmPnEiraCalcEnabled` is TRUE, the `UdpNm` module shall provide a possibility to monitor each PN, if this PN is still externally or internally requested on at least one of the relevant channels.] ()

**Note:** This means, only one timer is required to handle one PN on multiple connected physical channels. For example: only 8 EIRA reset timers are required to handle the requests of a Gateway with 6 physical channels and 8 partial networks. This is possible because the switch of PN PDU-Groups is done global for the ECU and not dependent of the physical channel.

[SWS\_UdpNm\_00349] [ If `UdpNmPnEiraCalcEnabled` is TRUE and a PN is requested by message reception or sending (see [SWS\\_UdpNm\\_00347](#) and [SWS\\_UdpNm\\_00348](#)) the monitoring for this PN shall be restarted with respect to `UdpNmPnResetTime`.] ()

**Note:** `UdpNmPnResetTime` shall be configured to a value greater than `UdpNmMsgCycleTime`. If `UdpNmPnResetTime` is configured to a value smaller than

`UdpNmMsgCycleTime` and only one ECU requests the PN, the request state toggles in the EIRA because request state is reset before the requesting ECU is able to send the next NM message.

**Note:** `UdpNmPnResetTime` shall be configured to a value smaller than `UdpNmTimeoutTime` to avoid that the timer could elapse after NM already changed to Prepare Bus Sleep.

[SWS\_UdpNm\_00351] [ If `UdpNmPnEiraCalcEnabled` is TRUE and a PN is not requested again within `UdpNmPnResetTime` the corresponding stored value for this PN shall be set to 0 (not requested) ] ()

[SWS\_UdpNm\_00352] [ If `UdpNmPnEiraCalcEnabled` is TRUE and the stored value for a PN is set to requested or back to not requested (see [SWS UdpNm 00347](#), [SWS UdpNm 00348](#) and [SWS UdpNm 00351](#)) `UdpNm` shall inform upper layers by calling `PduR_UdpNmRxIndication()` for the configured EIRA PDU (i.e changed EIRA information shall be passed to COM). ] ()

[SWS\_UdpNm\_00372] [ If `UdpNmPnEiraCalcEnabled` is TRUE and `UdpNmPnEraCalcEnabled` is TRUE, the PN status information shall be stored separately for both, the EIRA and ERA information (compare to [SWS UdpNm 00344](#) and [SWS UdpNm 00355](#)). ]()

## 7.8.5 Aggregation of External Requested Partial Networks

**Note:** This feature is used by the Gateways to collect only the external PN requests. The external PN requests are mirrored back to the requesting bus and provided to other (required) physical channels of a central gateway.

In case of a sub gateway the requests bit must not be mirrored back to the requesting physical channel in order to avoid static waking between central- and sub gateways. This logic shall be implemented by the ComM.

The `UdpNm` module provides the information if the PN is externally requested or not. The COM module is used for data transmission to the upper layer. The aggregated state of the external requested PNs is called “External Requests Aggregated” (ERA).

[SWS\_UdpNm\_00355] [ If `UdpNmPnEraCalcEnabled` is TRUE, the `UdpNm` shall provide the possibility to store external requested PNs on each relevant channel. At initialization the values of all PNs shall be set to 0 (not requested) ] ()

[SWS\_UdpNm\_00357] [ If

- `UdpNmPnEraCalcEnabled` is TRUE
- a NM-PDU is received
- PNs are requested within this message (bits are set to 1)
- and the requested PNs are set to 1 within the [configured PN filter mask] then `UdpNm` shall store the request information (value 1) for these PNs] ()

[SWS\_UdpNm\_00358] [ If `UdpNmPnEraCalcEnabled` is `TRUE`, the `UdpNm` module shall provide a possibility to monitor each relevant channel and for each PN if this PN is still externally requested.] ()

**Note:** This means, a separate timer is required to handle one PN on multiple physical channels.

For example: 48 ERA reset timers are required to handle the requests of a gateway with 6 physical channels and 8 partial networks. It is not possible to combine the reset timer like EIRA timers, because the external request mustn't be mirrored back to the requesting bus by a sub gateway. Thus it is required to detect the physical channel that is the source of the request bit.

[SWS\_UdpNm\_00359] [ If `UdpNmPnEraCalcEnabled` is `TRUE` and a PN is requested by message reception ( see [SWS UdpNM 00357](#)) the monitoring for this PN shall be restarted with respect to the `UdpNmPnResetTime`.] ()

**Note:** `UdpNmPnResetTime` shall be configured to a value greater than `UdpNmMsgCycleTime`. If `UdpNmPnResetTime` is configured to a value smaller than `UdpNmMsgCycleTime` and only one ECU requests the PN, the request state toggles in the ERA because request state is reset before the requesting ECU is able to send the next NM-PDU.

**Note:** `UdpNmPnResetTime` shall be configured to a value smaller than `UdpNmTimeoutTime` to avoid that the timer could elapse after NM already changed to Prepare Bus Sleep.

[SWS\_UdpNm\_00360] [ If `UdpNmPnEraCalcEnabled` is `TRUE` and PN is not requested again within `UdpNmPnResetTime` then the corresponding stored value for this PN shall be set to not requested (value 0)] ()

[SWS\_UdpNm\_00361] [ If `UdpNmPnEraCalcEnabled` is `TRUE` and the stored value for a PN changes to requested or back to not requested (see [SWS UdpNm 00357](#) and [SWS UdpNm 00360](#)), the `UdpNm` module shall inform the upper layers by calling `PduR_UdpNmRxIndication()` for the configured ERA PDU (i.e. changed ERA information shall be passed to the COM module).] ()

**Note:** In case both `UdpNmPnEraCalcEnabled` and `UdpNmPnEraCalcEnabled` are set to `TRUE`, the PN status information is handled as specified in [SWS UdpNm 00372](#).

## 7.8.6 Spontaneous Transmission of NM-PDUs via `UdpNm_NetworkRequest`

[SWS\_UdpNm\_00362] [ If `UdpNm_NetworkRequest` is called, `UdpNmPnHandleMultipleNetworkRequests` is set to `TRUE` and `UdpNm` is in Ready Sleep State, Normal Operation State or Repeat Message State, `UdpNm` shall change to or restart the Repeat Message State] () .

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

Note: The PN Control Module (e.G. ComM) is responsible to call `UdpNm_NetworkRequest` if the PN request bits changes.

## 7.9 Payload (PDU) Structure

The figure below shows an example for n bytes PDU length:

	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	User data 2							
Byte 5	User data 3							
...	...							
Byte n	User data n-2							

**Figure 4: NM packet payload (NM PDU) default format.**

### Note:

The length of the Network Management PDU is defined by the `PduLength` parameter in the "global" ECUC module ([`EcuC003_Conf`], see Ecu Configuration specification). The difference between number of enabled system bytes and length is the amount of user data bytes.

[SWS\_UdpNm\_00074] [ The location of the source node identifier shall be configurable by means of `UDPNM_PDU_NID_POSITION` to Byte 0, Byte 1, or off (default: Byte 0). ] ()

[SWS\_UdpNm\_00075] [ The location of the control Bit vector shall be configurable by means of `UDPNM_PDU_CBV_POSITION` to Byte 0, Byte 1, or off (default: Byte 1). ] ()

[SWS\_UdpNm\_00076] [ The length of an NM packet shall not exceed the MTU(Maximum Transmission Unit)of the underlying physical transport layer. ] ()

The figure below describes the format of the Control Bit Vector:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CBV	Res	PNL Bit	Res	Active Wakeup Bit	NM Coordinator Sleep Ready	Res R3.2 NM Coordinator ID (High Bit)	Res R3.2 NM Coordinator ID (Low Bit)	Repeat Message Request

**Figure 5: Control Bit Vector.**

[SWS\_UdpNm\_00045] [ The Control Bit Vector shall consist of:

Bit 0: Repeat Message Request

0: Repeat Message State not requested

1: Repeat Message State requested

Bit 3 :NM Coordinator Sleep Bit

0: Start of synchronized shutdown is not requested by main coordinator

1: Start of synchronized shutdown is requested by main coordinator

Bit 4 Active Wakeup Bit

0: Node has not woken up the network (passive wakeup)

1: Node has woken up the network (active Wakeup)

- Bit 6 Partial Network Information Bit (PNI)

0: NM message contains no Partial Network request information

1: NM message contains Partial Network request information

Bit 1,2,5,7 are reserved for future extensions

0 : Disabled / Reserved for future usage ] ( )

Note: The Control Bit Vector is initialized with 0x00 during initialization (also refer to [SWS UdpNm\\_00085](#)).

[SWS\_UdpNm\_00013] [ The source node identifier shall be set with the configuration parameter `UDPNM_NODE_ID` unless `UDPNM_PDU_NID_POSITION` 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 `ActiveWakeupBit` in the CBV.] ( )

## 7.10 Functional requirements on UdpNm API

[SWS\_UdpNm\_00014] [ If `UdpNmNodeDetectionEnabled` and `UdpNmRepeatMsgIndEnabled` are set to `TRUE` and Repeat Message Request bit is received, UdpNm module shall call the callback function `Nm_RepeatMessageIndication.` ] (SRS\_Nm\_00153)

## 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 [2] 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.

[SWS\_UdpNm\_00223] [ On errors and exceptions, the UdpNm module shall not modify its current module state. ] ()

### 7.12.1 Development Errors

[SWS\_UdpNm\_00018] [ The following errors shall be detectable by the UdpNm depending on its build version (development/production mode).

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

] ()

### 7.12.2 Run Time Errors

[SWS\_UdpNm\_00465] [

<i>Type or error</i>	<i>Related error code</i>	<i>Error Value</i>
NM-Timeout timer has expired outside Ready Sleep State (either in Repeat Message state or in Normal Operation state)	UDPNM_E_NETWORK_TIMEOUT	0x11

] ()

### 7.12.3 Transient Faults

The UdpNm module, currently does not specify any Transient Faults

### 7.12.4 Production Errors

The UdpNm module, currently does not specify any Production Errors

### 7.12.5 Extended Production Errors

The UdpNm module, currently does not specify any Extended Production Errors

### 7.12.6 Error detection

For details refer to the chapter 7.3 “Error Detection” in *SWS\_BSWGeneral*.

### 7.12.7 Error notification

[SWS\_UdpNm\_00189] [ 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. ] ()

[SWS\_UdpNm\_00190] [ 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] [ 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\_00463] [ 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\_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. ] (SRS\_Nm\_00137)

[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. ] (SRS\_Nm\_00137)

[SWS\_UdpNm\_00314] [ 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 [\[22\]](#)



## 7.13 Scheduling of the main function

For details refer to the chapter 8.5 “Scheduled functions” in *SWS\_BSWGeneral*.

## 7.14 Application notes

### 7.14.1 Wakeup notification

Wakeup notification is defined in detail in the ECU State Manager specification [11].

### 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 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

## 7.16 Parameter check

[SWS\_UdpNm\_00196] [ 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] [ 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] [ 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] [ 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. ] ()

## 8 API specification

[SWS\_UdpNm\_00244] [ 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.

### 8.1 Imported Types

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

```
boolean
uint8
uint16
uint32
```

[[[

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
ComStack_Types	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
	Std_Types.h	Std_VersionInfoType

]()

### 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]

<b>Name</b>	UdpNm_ConfigType
<b>Kind</b>	Structure

<b>Elements</b>	implementation specific	
	<b>Type</b>	--
	<b>Comment</b>	This type shall contain the parameters of the container UdpNm_Global Config and its sub containers.
<b>Description</b>	--	
<b>Available via</b>	UdpNm.h	

]()

### 8.2.2 UdpNm\_PduPositionType

[SWS\_UdpNm\_00304]

<b>Name</b>	UdpNm_PduPositionType		
<b>Kind</b>	Enumeration		
<b>Range</b>	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
<b>Description</b>	Used to define the position of the control bit vector within the NM PACKET.		
<b>Available via</b>	UdpNm.h		

]()

## 8.3 Function definitions

### 8.3.1 UdpNm\_Init

[SWS\_UdpNm\_00208]

<b>Service Name</b>	UdpNm_Init	
<b>Syntax</b>	<pre>void UdpNm_Init (     const UdpNm_ConfigType* UdpNmConfigPtr )</pre>	
<b>Service ID [hex]</b>	0x01	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	UdpNmConfigPtr	Pointer to a selected configuration structure

<b>Parameters (inout)</b>	None
<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	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
<b>Available via</b>	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]

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

<b>Service Name</b>	UdpNm_NetworkRequest	
<b>Syntax</b>	Std_ReturnType UdpNm_NetworkRequest ( NetworkHandleType nmChannelHandle )	
<b>Service ID [hex]</b>	0x02	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant (but not for the same NM-Channel)	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Requesting of network has failed
<b>Description</b>	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)	
<b>Available via</b>	UdpNm.h	

]()

### 8.3.4 UdpNm\_NetworkRelease

[SWS\_UdpNm\_00214]

<b>Service Name</b>	UdpNm_NetworkRelease	
<b>Syntax</b>	Std_ReturnType UdpNm_NetworkRelease ( NetworkHandleType nmChannelHandle )	
<b>Service ID [hex]</b>	0x03	

<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant (but not for the same NM-Channel)	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Releasing of network has failed
<b>Description</b>	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)	
<b>Available via</b>	UdpNm.h	

]()

### 8.3.5 UdpNm\_DisableCommunication

#### [SWS\_UdpNm\_00215]

<b>Service Name</b>	UdpNm_DisableCommunication	
<b>Syntax</b>	Std_ReturnType UdpNm_DisableCommunication ( NetworkHandleType nmChannelHandle )	
<b>Service ID [hex]</b>	0x0c	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant (but not for the same NM-Channel)	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Disabling of NM PDU transmission ability has failed
<b>Description</b>	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)	

<b>Available via</b>	UdpNm.h
----------------------	---------

](SRS\_Nm\_02512)  
[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]

<b>Service Name</b>	UdpNm_EnableCommunication	
<b>Syntax</b>	Std_ReturnType UdpNm_EnableCommunication ( NetworkHandleType nmChannelHandle )	
<b>Service ID [hex]</b>	0x0d	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant (but not for the same NM-Channel)	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Enabling of NM PDU transmission ability has failed
<b>Description</b>	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).	
<b>Available via</b>	UdpNm.h	

](SRS\_Nm\_02512)  
[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]

<b>Service Name</b>	UdpNm_SetUserData	
<b>Syntax</b>	<pre>Std_ReturnType UdpNm_SetUserData (     NetworkHandleType nmChannelHandle,     const uint8* nmUserDataPtr )</pre>	
<b>Service ID [hex]</b>	0x04	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
	nmUserDataPtr	Pointer where the user data for the next transmitted NM message shall be copied from.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_Return-Type	E_OK: No error E_NOT_OK: Setting of user data has failed
<b>Description</b>	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 UdpNmPassiveModeEnabled==false).	
<b>Available via</b>	UdpNm.h	

()

### 8.3.8 UdpNm\_GetUserData

[SWS\_UdpNm\_00218]

<b>Service Name</b>	UdpNm_GetUserData	
<b>Syntax</b>	<pre>Std_ReturnType UdpNm_GetUserData (     NetworkHandleType nmChannelHandle,     uint8* nmUserDataPtr )</pre>	



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

()

### 8.3.9 UdpNm\_GetNodeIdentifier

[SWS\_UdpNm\_00219]

<b>Service Name</b>	UdpNm_GetNodeIdentifier	
<b>Syntax</b>	Std_ReturnType UdpNm_GetNodeIdentifier ( NetworkHandleType nmChannelHandle, uint8* nmNodeIdPtr )	
<b>Service ID [hex]</b>	0x06	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	nmChannel Handle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	nmNodeId Ptr	Pointer where the source node identifier from the most recently received NM PDU shall be copied to.
<b>Return value</b>	Std_Return-Type	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.

<b>Description</b>	Get node identifier from the most recently received NM PDU. Caveats: UdpNm is initialized correctly.
<b>Available via</b>	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]

<b>Service Name</b>	UdpNm_GetLocalNodeIdentifier	
<b>Syntax</b>	<pre>Std_ReturnType UdpNm_GetLocalNodeIdentifier (     NetworkHandleType nmChannelHandle,     uint8* nmNodeIdPtr )</pre>	
<b>Service ID [hex]</b>	0x07	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	nmNodeIdPtr	Pointer where node identifier of the local node shall be copied to.
<b>Return value</b>	Std_Return-Type	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.
<b>Description</b>	Get node identifier configured for the local node. Caveats: UdpNm is initialized correctly.	
<b>Available via</b>	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]

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

()

### 8.3.12 UdpNm\_GetPduData

[SWS\_UdpNm\_00309]

<b>Service Name</b>	UdpNm_GetPduData	
<b>Syntax</b>	Std_ReturnType UdpNm_GetPduData ( NetworkHandleType nmChannelHandle, uint8* nmPduDataPtr )	
<b>Service ID [hex]</b>	0x0a	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	nmChannel Handle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	nmPduDataPtr	Pointer where NM PDU shall be copied to.

<b>Return value</b>	Std_Return-Type	E_OK: No error E_NOT_OK: Getting of NM PDU Data has failed or is not configured for this network handle.
<b>Description</b>	Get the whole PDU data out of the most recently received NM message. Caveats: UdpNm is initialized correctly.	
<b>Available via</b>	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]

<b>Service Name</b>	UdpNm_GetState	
<b>Syntax</b>	<pre>Std_ReturnType UdpNm_GetState (     NetworkHandleType nmChannelHandle,     Nm_StateType* nmStatePtr,     Nm_ModeType* nmModePtr )</pre>	
<b>Service ID [hex]</b>	0x0b	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	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.
<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Getting of NM state has failed
<b>Description</b>	Returns the state and the mode of the network management. Caveats: UdpNm is initialized correctly. Configuration: Mandatory	
<b>Available via</b>	UdpNm.h	

]()

### 8.3.14 UdpNm\_GetVersionInfo

#### [SWS\_UdpNm\_00224]

<b>Service Name</b>	UdpNm_GetVersionInfo	
<b>Syntax</b>	<pre>void UdpNm_GetVersionInfo (     Std_VersionInfoType* versioninfo )</pre>	
<b>Service ID [hex]</b>	0x09	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	versioninfo	Pointer to where to store the version information of this module.
<b>Return value</b>	None	
<b>Description</b>	This service returns the version information of this module.	
<b>Available via</b>	UdpNm.h	

]()

[SWS\_UdpNm\_00318] [ If DET is enabled for the UdpNm module, the function UdpNm\_GetVersionInfo shall raise UDPNM\_E\_PARAM\_POINTER, if the argument versioninfo is a NULL pointer and return without any action. ] ()

### 8.3.15 UdpNm\_RequestBusSynchronization

#### [SWS\_UdpNm\_00226]

<b>Service Name</b>	UdpNm_RequestBusSynchronization	
<b>Syntax</b>	<pre>Std_ReturnType UdpNm_RequestBusSynchronization (     NetworkHandleType nmChannelHandle )</pre>	
<b>Service ID [hex]</b>	0x14	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	

<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Requesting of bus synchronization has failed
<b>Description</b>	Request bus synchronization. Caveats: UdpNm is initialized correctly. Configuration: Optional (only available if UdpNmBusSynchronizationEnabled==true and UdpNmPassiveModeEnabled==false).	
<b>Available via</b>	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. ] ()

### 8.3.16 UdpNm\_CheckRemoteSleepIndication

[SWS\_UdpNm\_00227]

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

### 8.3.17 UdpNm\_SetSleepReadyBit

[SWS\_UdpNm\_00324]

<b>Service Name</b>	UdpNm_SetSleepReadyBit	
<b>Syntax</b>	Std_ReturnType UdpNm_SetSleepReadyBit ( NetworkHandleType nmChannelHandle, boolean nmSleepReadyBit )	
<b>Service ID [hex]</b>	0x16	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	nmChannelHandle	Identification of the NM-channel
	nmSleepReadyBit	Value written to ReadySleep Bit in CBV
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: No error E_NOT_OK: Writing of remote sleep indication bit has failed
<b>Description</b>	Set the NM Coordinator Sleep Ready bit in the Control Bit Vector	
<b>Available via</b>	UdpNm.h	

]()

### 8.3.18 UdpNm\_Transmit

[SWS\_UdpNm\_00313]

<b>Service Name</b>	UdpNm_Transmit	
<b>Syntax</b>	Std_ReturnType UdpNm_Transmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )	
<b>Service ID [hex]</b>	0x49	

<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in)</b>	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to Meta Data.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_Return-Type	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
<b>Description</b>	Requests transmission of a PDU.	
<b>Available via</b>	UdpNm.h	

]()  
 [SWS\_UdpNm\_00315] [ If UdpNmComUserDataSupport or UdpNmPnEnabled is enabled the UdpNm implementation shall provide an API UdpNm\_Transmit. ]  
 (SRS\_Nm\_02503)

## 8.4 Call-back notifications

### 8.4.1 UdpNm\_SoAdIfTxConfirmation

[SWS\_UdpNm\_00228]

<b>Service Name</b>	UdpNm_SoAdIfTxConfirmation	
<b>Syntax</b>	<pre>void UdpNm_SoAdIfTxConfirmation (     PduIdType TxPduId,     Std_ReturnType result )</pre>	
<b>Service ID [hex]</b>	0x40	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in)</b>	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	



<b>Return value</b>	None
<b>Description</b>	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
<b>Available via</b>	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 ] ()

#### 8.4.2 UdpNm\_SoAdIfRxIndication

[SWS\_UdpNm\_00231]

<b>Service Name</b>	UdpNm_SoAdIfRxIndication	
<b>Syntax</b>	<pre>void UdpNm_SoAdIfRxIndication (     PduIdType RxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID [hex]</b>	0x42	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different Pdulds. Non reentrant for the same PduId.	
<b>Parameters (in)</b>	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length ( <code>SduLength</code> ) of the received PDU, a pointer to a buffer ( <code>SduDataPtr</code> ) containing the PDU, and the <code>MetaData</code> related to this PDU.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	

<b>Return value</b>	None
<b>Description</b>	Indication of a received PDU from a lower layer communication interface module.
<b>Available via</b>	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 `udpNmRxPduld` shall refer to the UdpNm channel handle, i.e. a mapping from Pduld to UdpNm channel handle is not necessary.

### 8.4.3 UdpNm\_SoAdIfTriggerTransmit

[SWS\_UdpNm\_91001]

<b>Service Name</b>	UdpNm_SoAdIfTriggerTransmit	
<b>Syntax</b>	<pre>Std_ReturnType UdpNm_SoAdIfTriggerTransmit (     PduIdType TxPduld,     PduInfoType* PduInfoPtr )</pre>	
<b>Service ID [hex]</b>	0x41	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different Pdulds. Non reentrant for the same Pduld.	
<b>Parameters (in)</b>	TxPduld	ID of the SDU that is requested to be transmitted.
<b>Parameters (inout)</b>	PduInfoPtr	Contains a pointer to a buffer ( <code>SduDataPtr</code> ) to where the SDU data shall be copied, and the available buffer size in <code>SduLength</code> . On return, the service will indicate the length of the copied SDU data in <code>SduLength</code> .
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: SDU has been copied and <code>SduLength</code> indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. <code>PduInfoPtr</code> must not be used since it may contain a NULL pointer or point to invalid data.
<b>Description</b>	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by <code>PduInfoPtr-&gt;SduLength</code> . If it fits, it shall copy its data into the buffer provided by <code>PduInfoPtr-&gt;SduDataPtr</code> and update the length of the actual copied data in <code>PduInfoPtr-&gt;SduLength</code> . If not, it returns E_NOT_OK without changing <code>PduInfoPtr</code> .	
<b>Available via</b>	UdpNm.h	

]()

[SWS\_UdpNm\_00377] [ If `UdpNmComUserDataSupport` is enabled 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 within the call of `UdpNm_SoAdIfTriggerTransmit`. ] (SRS\_Nm\_02503)

[SWS\_UdpNm\_00378] [ The function `UdpNm_SoAdIfTriggerTransmit` shall copy the NM PDU data of the according NM PDU requested by `TxPduld`. ] ()

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

## 8.5 Scheduled Functions

### 8.5.1 UdpNm\_MainFunction\_<Instance Id>

[SWS\_UdpNm\_00234]

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

]()

## 8.6 Expected Interfaces

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

### 8.6.1 Mandatory Interfaces

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

[]

<b>API Function</b>	<b>Header</b>	<b>Description</b>
---------------------	---------------	--------------------

	<i>File</i>	
Nm_BusSleep-Mode	Nm.h	Notification that the network management has entered Bus-Sleep Mode.
Nm_Network-Mode	Nm.h	Notification that the network management has entered Network Mode.
Nm_Network-StartIndication	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_Prepare-BusSleepMode	Nm.h	Notification that the network management has entered Prepare Bus-Sleep Mode.
SoAd_If-Transmit	SoAd.h	Requests transmission of a PDU.

J()

## 8.6.2 Optional Interfaces

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

[[

<b>API Function</b>	<b>Header File</b>	<b>Description</b>
Det_ReportError	Det.h	Service to report development errors.
Nm_CarWake-UpIndication	Nm.h	This function is called by a <Bus>Nm to indicate reception of a CWU request.
Nm_Coord-ReadyToSleep-Cancellation	Nm.h	Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0.
Nm_Coord-ReadyToSleep-Indication	Nm.h	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set
Nm_PduRx-Indication	Nm.h	Notification that a NM message has been received.
Nm_Remote-Sleep-Cancellation	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_Remote-SleepIndication	Nm.h	Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode.
Nm_Repeat-Message-Indication	Nm.h	Service to indicate that an NM message with set Repeat Message Request Bit has been received.
Nm_State-Change-Notification	Nm.h	Notification that the state of the lower layer <BusNm> has changed.
Nm_TxTimeout-Exception	Nm.h	Service to indicate that an attempt to send an NM message failed.
PduR_UdpNm-RxIndication	PduR_UdpNm.h	Indication of a received PDU from a lower layer communication interface module.
PduR_UdpNm-TriggerTransmit	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->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.
PduR_UdpNm-TxConfirmation	PduR_UdpNm.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.

10)

### **8.6.3 Configurable interfaces**

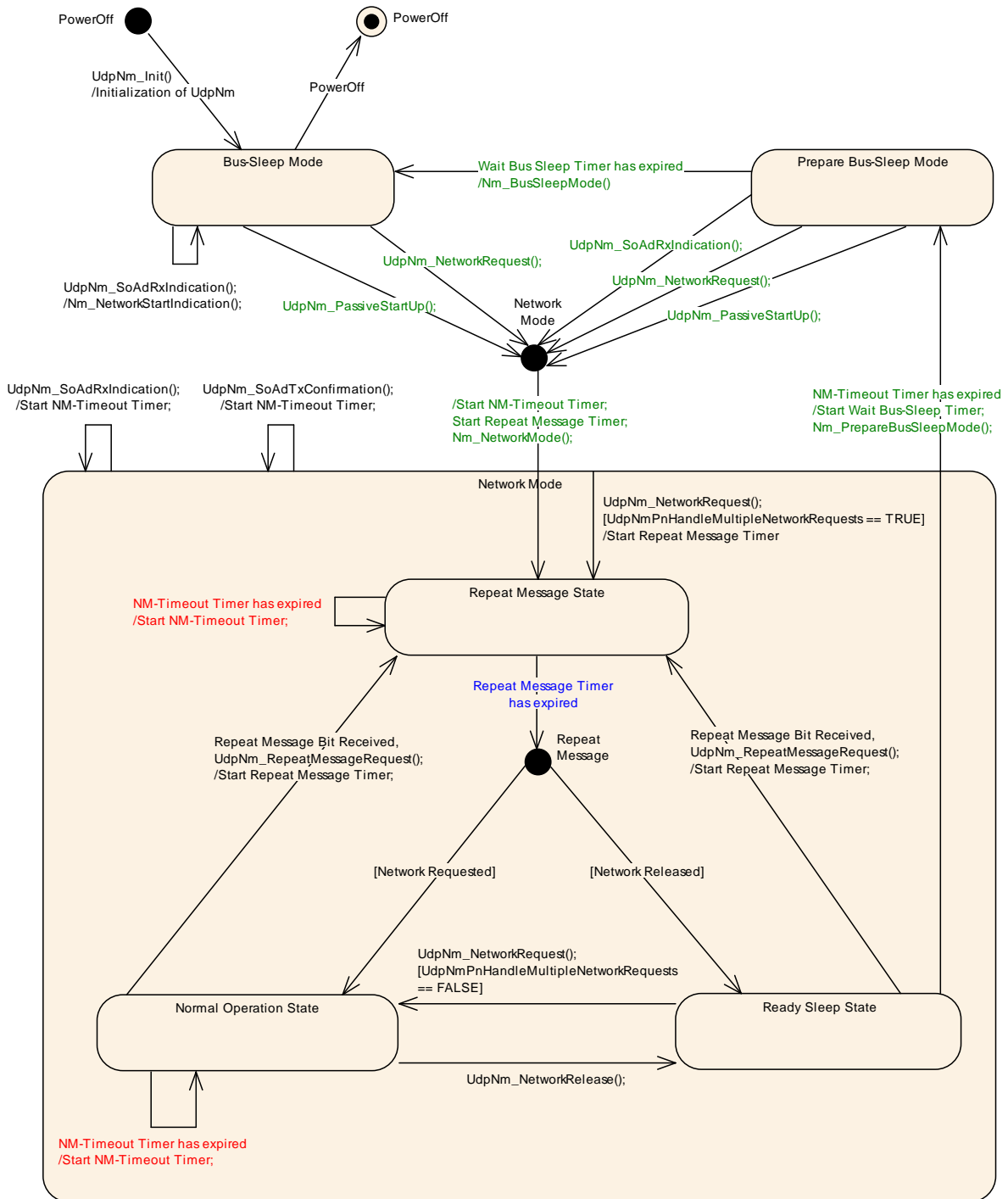
Not applicable

### **8.7 Service Interfaces**

Not applicable

### 8.8 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 6: State chart diagram.**

## 9 Sequence diagrams and Transition Tables

### 9.1 UdpNmTransmission

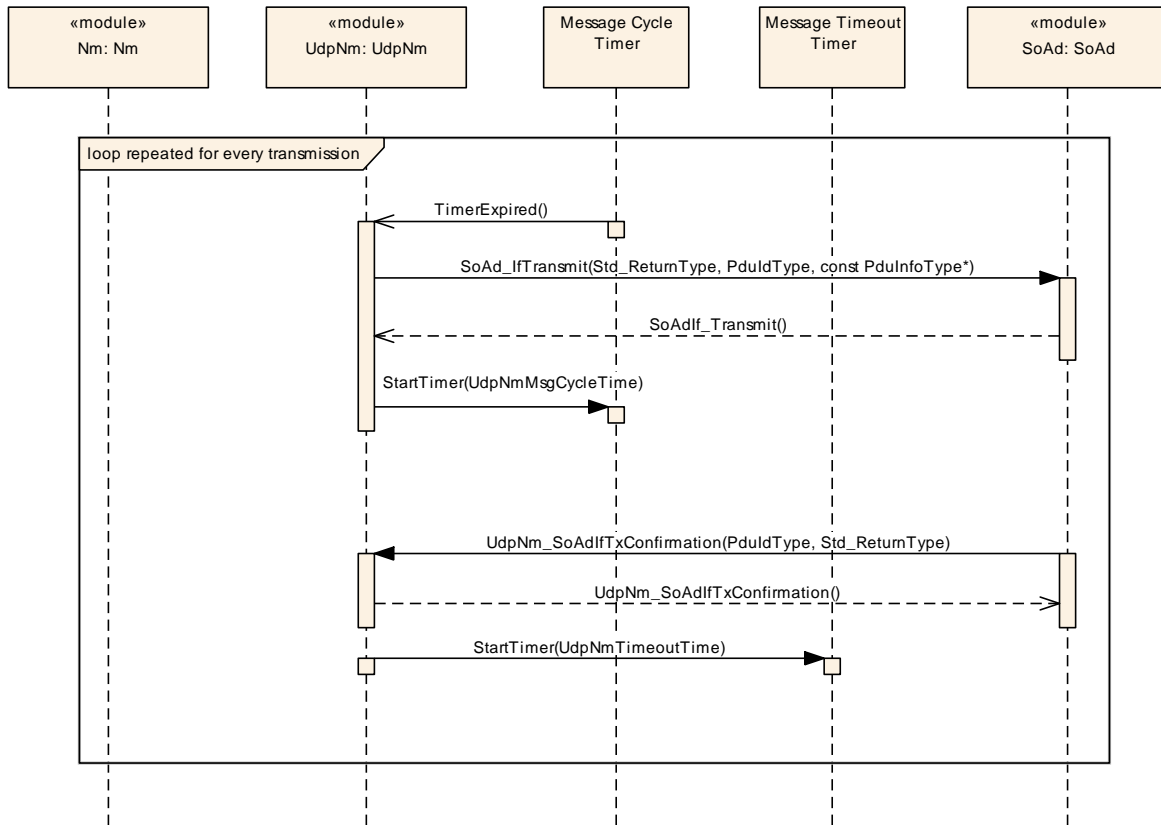
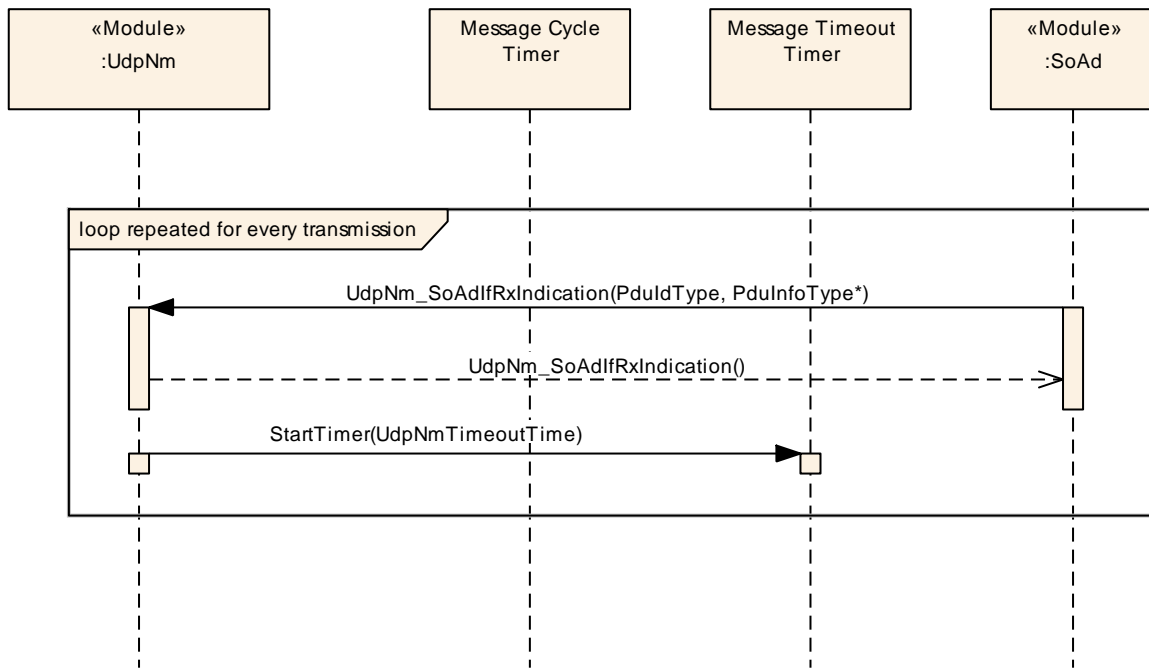


Figure 7: Sequence diagram – PDU transmission.

### 9.2 UdpNm Reception

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





**Figure 8: 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) to be use for the parameter specification. Chapter 10.1 is intended to remain in the specification document to ensure comprehensiveness.

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

Chapter 10.3 specifies published information of module UdpNm.

### 10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS\_BSWGeneral*.

### 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] [ 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

<b>SWS Item</b>	<b>ECUC_UdpNm_00088 :</b>
<b>Module Name</b>	UdpNm
<b>Module Description</b>	--
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-PRE-COMPILE

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
UdpNmGlobalConfig	1	This container contains all global configuration parameters of UDP NM configured from the CanTrcv Module perspective.

### 10.2.2 UdpNmGlobalConfig

<b>SWS Item</b>	<b>ECUC_UdpNm_00001 :</b>
<b>Container Name</b>	UdpNmGlobalConfig
<b>Parent Container</b>	UdpNm
<b>Description</b>	This container contains all global configuration parameters of UDP NM configured from the CanTrcv Module perspective.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_UdpNm_00006 :</b>		
<b>Name</b>	UdpNmBusSynchronizationEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling bus synchronization support. This feature is required for gateway nodes only. It must not be defined if UdpNmPassiveModeEnabled==true. This parameter shall be derived from NmBusSynchronizationEnabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00013 :</b>		
<b>Name</b>	UdpNmComControlEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling the Communication Control support.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: calculationFormula = If (UdpNmPassiveModeEnabled == False) then Equal(NmComControlEnabled) else Equal(False)
---------------------------	--

<b>SWS Item</b>	<b>ECUC_UdpNm_00055 :</b>		
<b>Name</b>	UdpNmComUserDataSupport		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Enable/disable the user data support.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: If UdpNmPassiveModeEnabled == True then UdpNmComUserDataSupport = False		

<b>SWS Item</b>	<b>ECUC_UdpNm_00059 :</b>		
<b>Name</b>	UdpNmCoordinatorSyncSupport		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Enables/disables the coordinator synchronization support.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: UdpNmCoordinatorSyncSupport has to be set to FALSE if UdpNmPassiveModeEnabled is set to TRUE.		

<b>SWS Item</b>	<b>ECUC_UdpNm_00002 :</b>		
<b>Name</b>	UdpNmDevErrorDetect		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Switches the development error detection and notification on or off.  <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>SWS Item</b>	<b>ECUC_UdpNm_00009 :</b>		
<b>Name</b>	UdpNmImmediateRestartEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00014 :</b>		
<b>Name</b>	UdpNmNumberOfChannels		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Number of NM channels allowed within one ECU.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00010 :</b>		
<b>Name</b>	UdpNmPassiveModeEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling support of the Passive Mode.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00011 :</b>		
<b>Name</b>	UdpNmPduRxIndicationEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling the PDU Rx Indication. This parameter shall be derived from NmPduRxIndicationEnabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local		
<b>SWS Item</b>	<b>ECUC_UdpNm_00066 :</b>		
<b>Name</b>	UdpNmPnEiraCalcEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Specifies if UdpNm calculates the PN request information for internal and external requests. (EIRA) true: PN request are calculated false: PN request are not calculated		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel		

<b>SWS Item</b>	<b>ECUC_UdpNm_00065 :</b>		
<b>Name</b>	UdpNmPnResetTime		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA. The value shall be the same for every channel. Thus it is a global config parameter.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0.001 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel.
---------------------------	--

<b>SWS Item</b>	<b>ECUC_UdpNm_00005 :</b>		
<b>Name</b>	UdpNmRemoteSleepIndEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling remote sleep indication support. This feature is required for gateway nodes only. It must not be defined if UdpNmPassiveModeEnabled==true. This parameter shall be derived from NmRemoteSleepIndEnabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00012 :</b>		
<b>Name</b>	UdpNmStateChangeIndEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling the UDP NM state change notification. This parameter shall be derived from NmStateChangeIndEnabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00004 :</b>		
<b>Name</b>	UdpNmUserDataEnabled		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling user data support. This parameter shall be derived from NmUserDataEnabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

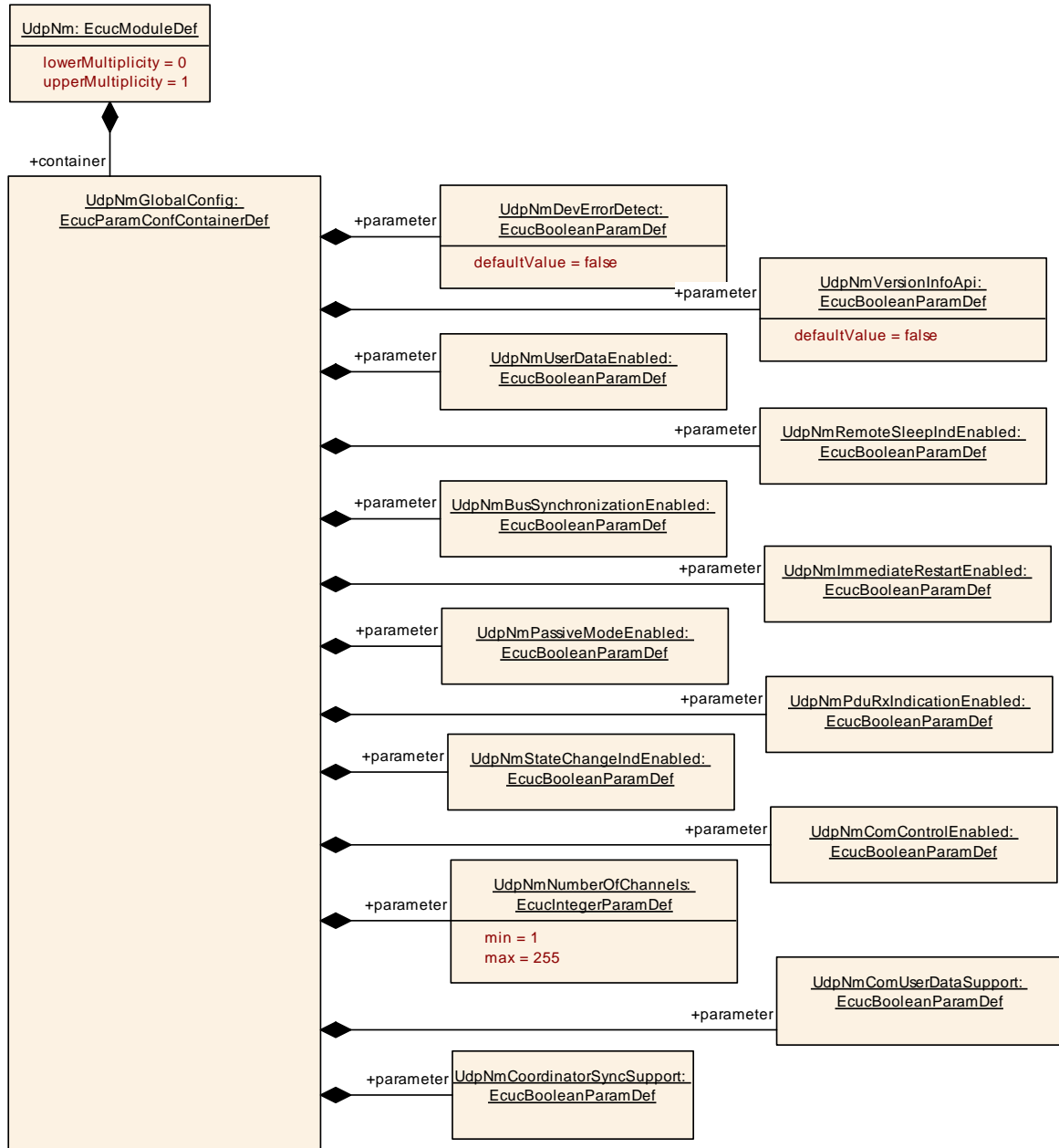
<b>SWS Item</b>	<b>ECUC_UdpNm_00003 :</b>		
<b>Name</b>	UdpNmVersionInfoApi		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Pre-processor switch for enabling version info API support.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00062 :</b>		
<b>Name</b>	UdpNmPnEiraRxNSduRef		
<b>Parent Container</b>	UdpNmGlobalConfig		
<b>Description</b>	Reference to a Pdu in the COM-Stack. Only one SduRef is required for UdpNm because the EIRA is the aggregation over all Ethernet Channels.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	



<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel
---------------------------	---

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
UdpNmChannelConfig	1..*	This container contains the channel-specific configuration parameters of the UdpNm.
UdpNmPnInfo	0..1	PN information configuration



**Figure 9: Diagram: UdpNmGlobalConfig**

### 10.2.3 UdpNmChannelConfig

<b>SWS Item</b>	<b>ECUC_UdpNm_00017 :</b>
<b>Container Name</b>	UdpNmChannelConfig
<b>Parent Container</b>	UdpNmGlobalConfig
<b>Description</b>	This container contains the channel-specific configuration parameters of the UdpNm.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_UdpNm_00074 :</b>		
<b>Name</b>	UdpNmActiveWakeupBitEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Enables/Disables the handling of the Active Wakeup Bit in the UdpNm module.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	false	
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: This parameter is only valid if UdpNmPassiveModeEnabled is False.		

<b>SWS Item</b>	<b>ECUC_UdpNm_00089 :</b>		
<b>Name</b>	UdpNmAllNmMessagesKeepAwake		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	false	
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true
---------------------------	--

<b>SWS Item</b>	<b>ECUC_UdpNm_00087 :</b>		
<b>Name</b>	UdpNmCarWakeUpBitPosition		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Specifies the Bit position of the CWU within the NM PDU.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 7		
<b>Default value</b>	--		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	false	
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmCarWakeUpRxEnabled == TRUE		

<b>SWS Item</b>	<b>ECUC_UdpNm_00086 :</b>		
<b>Name</b>	UdpNmCarWakeUpBytePosition		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Specifies the Byte position of the CWU within the NM PDU.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 7		
<b>Default value</b>	--		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	false	
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmCarWakeUpRxEnabled == TRUE UdpNmCarWakeUpBytePosition ≥ number of enabled system bytes (CBV, NID)
---------------------------	---

<b>SWS Item</b>	<b>ECUC_UdpNm_00077 :</b>		
<b>Name</b>	UdpNmCarWakeUpFilterEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmCarWakeUpRxEnabled == TRUE		

<b>SWS Item</b>	<b>ECUC_UdpNm_00078 :</b>		
<b>Name</b>	UdpNmCarWakeUpFilterNodeId		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmCarWakeUpFilterEnabled == TRUE
---------------------------	---

<b>SWS Item</b>	<b>ECUC_UdpNm_00076 :</b>		
<b>Name</b>	UdpNmCarWakeUpRxEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Enables or disables support of CarWakeUp bit evaluation in received NM PDUs. FALSE - CarWakeUp not supported. TRUE - CarWakeUp supported.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_UdpNm_00079 :</b>		
<b>Name</b>	UdpNmImmediateNmCycleTime		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Defines the immediate NM PDU cycle time in seconds which is used for UdpNmImmediateNmTransmissions NM PDU transmissions.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0.001 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: This parameter is only valid if UdpNmImmediateNmTransmissions is greater one.		

<b>SWS Item</b>	<b>ECUC_UdpNm_00075 :</b>		
<b>Name</b>	UdpNmImmediateNmTransmissions		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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 immediate NM PDUs is defined by UdpNmImmediateNmCycleTime.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: If UdpNmImmediateRestartEnabled = true then UdpNmImmediateNmTransmissions = 0 If UdpNmPnHandleMultipleNetworkRequests == True then UdpNmImmediateNmTransmissions > 0
---------------------------	---

<b>SWS Item</b>	<b>ECUC_UdpNm_00032 :</b>		
<b>Name</b>	UdpNmMainFunctionPeriod		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Call cycle of UdpNm_MainFunction_x for the respective instance in [s].		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	]0 .. INF[		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00029 :</b>		
<b>Name</b>	UdpNmMsgCycleOffset		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	]0 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00028 :</b>		
<b>Name</b>	UdpNmMsgCycleTime		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	]0.001 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local		
<b>SWS Item</b>	<b>ECUC_UdpNm_00090 :</b>		
<b>Name</b>	UdpNmNodeDetectionEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	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(NmNodeDetectionEnabled) else Equal(False).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: Not available if UdpNmPassiveModeEnabled		

<b>SWS Item</b>	<b>ECUC_UdpNm_00031 :</b>		
<b>Name</b>	UdpNmNodeId		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Node identifier of local node. This parameter is only valid if UdpNmPassiveModeEnabled is set to OFF and UdpNmNodeDetectionEnabled is set to ON.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00091 :</b>		
<b>Name</b>	UdpNmNodeIdEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Pre-processor switch for enabling the source node identifier. This parameter shall be derived from NmNodeIdEnabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>		scope: local	
<b>SWS Item</b>	<b>ECUC_UdpNm_00026 :</b>		
<b>Name</b>	UdpNmPduCbvPosition		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	<p>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)</p> <p>See also UdpNmPduNidPosition</p> <p>if (UdpNmPduCbvPosition != UDPNM_PDU_OFF &amp;&amp; UdpNmPduNidPosition != UDPNM_PDU_OFF) then UdpNmPduCbvPosition != UdpNmPduNidPosition</p> <p>if (UdpNmPduCbvPosition != UDPNM_PDU_OFF &amp;&amp; UdpNmPduNidPosition == UDPNM_PDU_OFF) then UdpNmPduCbvPosition = UDPNM_PDU_BYTE0</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	UDPNM_PDU_BYTE_0		--
	UDPNM_PDU_BYTE_1		--
	UDPNM_PDU_OFF		--
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00025 :</b>		
<b>Name</b>	UdpNmPduNidPosition		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	<p>Defines the position of the source node identifier within the NM PACKET. ImplementationType: UdpNm_PduPositionType</p> <p>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)</p> <p>See also UdpNmPduCbvPosition</p> <p>if (UDPNM_PDU_NID_POSITION != UDPNM_PDU_OFF &amp;&amp; UDPNM_PDU_CBV_POSITION != UDPNM_PDU_OFF) then UDPNM_PDU_NID_POSITION != UDPNM_PDU_CBV_POSITION</p> <p>if (UDPNM_PDU_NID_POSITION != UDPNM_PDU_OFF &amp;&amp; UDPNM_PDU_CBV_POSITION == UDPNM_PDU_OFF) then UDPNM_PDU_IND_POSITION = UDPNM_PDU_BYTE0</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	UDPNM_PDU_BYTE_0		Byte 0 is used.
	UDPNM_PDU_BYTE_1		Byte 1 is used.
	UDPNM_PDU_OFF		Node Identification is not used.



<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00061 :</b>		
<b>Name</b>	UdpNmPnEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Enables or disables support of partial networking. false: Partial networking Range not supported true: Partial networking supported		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true		

<b>SWS Item</b>	<b>ECUC_UdpNm_00060 :</b>		
<b>Name</b>	UdpNmPnEraCalcEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Specifies if UdpNm calculates the PN request information for external requests. (ERA) false: PN request are not calculated true: PN request are calculated.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true
---------------------------	--

<b>SWS Item</b>	<b>ECUC_UdpNm_00063 :</b>		
<b>Name</b>	UdpNmPnHandleMultipleNetworkRequests		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	false: UdpNm_NetworkRequest is ignored in NO. true: UdpNm_NetworkRequest triggers a change from NO to RM.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true		

<b>SWS Item</b>	<b>ECUC_UdpNm_00023 :</b>		
<b>Name</b>	UdpNmRemoteSleepIndTime		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	<p>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.</p> <p>Typically it should be equal to: <math>n * \text{UdpNmMsgCycleTime}</math>, where n denotes the number of NM packets that are normally sent before Remote Sleep Indication is detected.</p> <p>The value of n decremented by one determines the amount of lost NM packets that can be tolerated by the Remote Sleep Indication procedure.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0.001 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local		
<b>SWS Item</b>	<b>ECUC_UdpNm_00022 :</b>		
<b>Name</b>	UdpNmRepeatMessageTime		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Timeout for Repeat Message State. It defines the time in seconds how long the NM shall stay in the Repeat Message State.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	<p>scope: local            dependency: <math>UdpNmRepeatMessageTime = n * UdpNmMsgCycleTime;</math>  <math>UdpNmRepeatMessageTime &gt; UdpNmImmediateNmTransmissions * UdpNmImmediateNmCycleTime</math></p> <p>Typically it should be equal to: <math>n * UdpNmMsgCycleTime</math>, where n denotes the number of NM PDUs that are normally sent in the Repeat Message State.</p> <p>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.</p>		

<b>SWS Item</b>	<b>ECUC_UdpNm_00092 :</b>		
<b>Name</b>	UdpNmRepeatMsgIndEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Enable/disable the notification that a RepeatMessageRequest bit has been received.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: calculationFormula = If (UdpNmPassiveModeEnabled == False) then Equal(NmRepeatMsgIndEnabled) else Equal(False)
---------------------------	--

<b>SWS Item</b>	<b>ECUC UdpNm_00085 :</b>		
<b>Name</b>	UdpNmRetryFirstMessageRequest		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Specifies if first message request in UdpNm is repeated until accepted by SoAd.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: UdpNmRetryFirstMessageRequest = false if UdpNmPassiveModeEnabled == true		

<b>SWS Item</b>	<b>ECUC UdpNm_00093 :</b>		
<b>Name</b>	UdpNmStayInPbsEnabled		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	If this parameter is disabled Prepare Bus-Sleep Mode is left after UdpNmWaitBusSleepTime. If this parameter is enabled Prepare Bus-Sleep Mode can only be left if ECU is powered off or any restart reason applies.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local		
<b>SWS Item</b>	<b>ECUC_UdpNm_00020 :</b>		
<b>Name</b>	UdpNmTimeoutTime		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	<p>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.</p> <p>It shall be equal for all nodes in the cluster. It shall be greater than UdpNmMsgCycleTime. Typically, it should be equal to: <math>x * \text{UdpNmMsgCycleTime}</math>, where <math>n</math> 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 <math>n</math> decremented by one determines the amount of lost NM packets that can be tolerated by the coordination algorithm.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0.002 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00021 :</b>		
<b>Name</b>	UdpNmWaitBusSleepTime		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	<p>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.</p> <p>It shall be equal for all nodes in the cluster. It shall be long enough to empty all Tx-buffer empty.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0.001 .. 65.535]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local dependency: In case UdpNmStayInPbsEnabled is disabled this parameter shall be mandatory.
---------------------------	--

<b>SWS Item</b>	<b>ECUC UdpNm_00018 :</b>		
<b>Name</b>	UdpNmComMNetworkHandleRef		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId.		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to [ ComMChannel ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_UdpNm_00073 :</b>		
<b>Name</b>	UdpNmPnEraRxNSduRef		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Reference to a Pdu in the COM-Stack. The SduRef is required for every UdpNm Channel, because ERA is reported per channel.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
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.

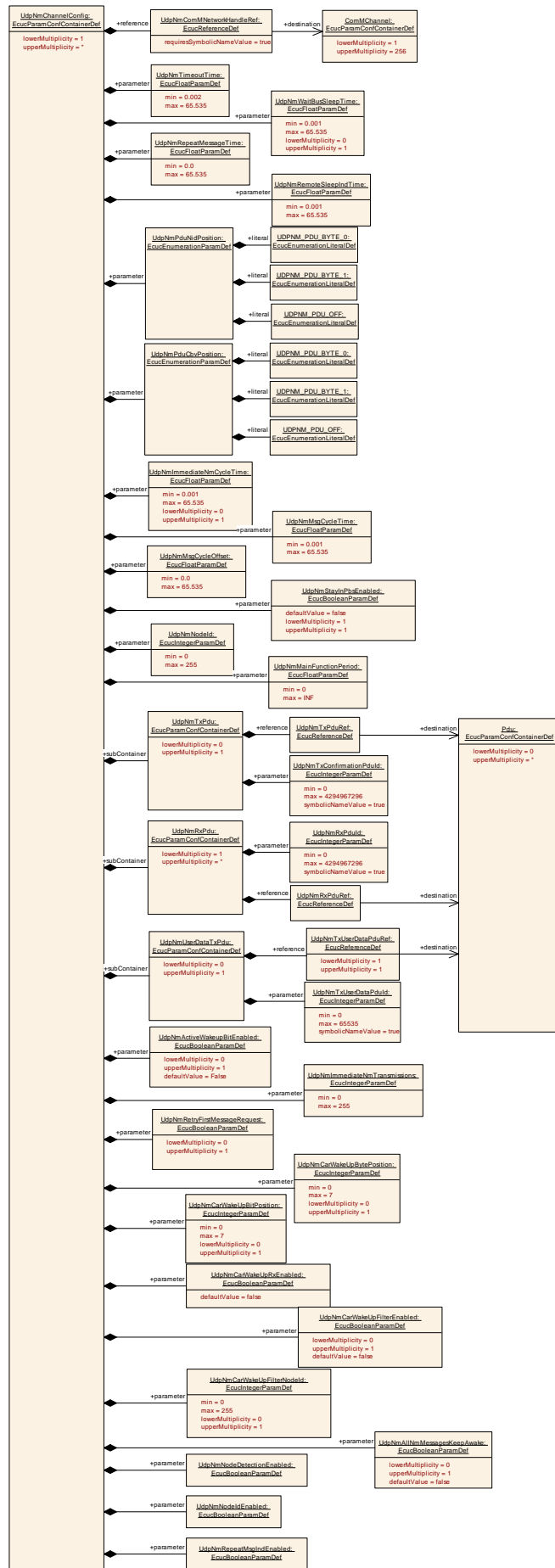


Figure 10: UdpNmChannelConfig

### 10.2.4 UdpNmRxPdu

<b>SWS Item</b>	<b>ECUC_UdpNm_00038 :</b>		
<b>Container Name</b>	UdpNmRxPdu		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	This container describes the UdpNm RX PDU's.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_UdpNm_00043 :</b>		
<b>Name</b>	UdpNmRxPduId		
<b>Parent Container</b>	UdpNmRxPdu		
<b>Description</b>	ID of the RxPdu that will be used by a RxIndication of the lower layer.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 4294967296		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00039 :</b>		
<b>Name</b>	UdpNmRxPduRef		
<b>Parent Container</b>	UdpNmRxPdu		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>No Included Containers</b>
-------------------------------



### 10.2.5 UdpNmTxPdu

<b>SWS Item</b>	<b>ECUC_UdpNm_00036 :</b>
<b>Container Name</b>	UdpNmTxPdu
<b>Parent Container</b>	UdpNmChannelConfig
<b>Description</b>	This container describes the UdpNm TX PDU's.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_UdpNm_00042 :</b>		
<b>Name</b>	UdpNmTxConfirmationPduId		
<b>Parent Container</b>	UdpNmTxPdu		
<b>Description</b>	Id of the TxPdu that will be used by a TxConfirmation from the lower layer.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 4294967296		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00037 :</b>		
<b>Name</b>	UdpNmTxPduRef		
<b>Parent Container</b>	UdpNmTxPdu		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>No Included Containers</b>
-------------------------------

### 10.2.6 UdpNmUserDataTxPdu

<b>SWS Item</b>	<b>ECUC_UdpNm_00056 :</b>		
<b>Container Name</b>	UdpNmUserDataTxPdu		
<b>Parent Container</b>	UdpNmChannelConfig		
<b>Description</b>	Preprocessor switch for enabling the Tx path of Com User Data. Use case: Setting of NMUserData via SWC.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_UdpNm_00058 :</b>		
<b>Name</b>	UdpNmTxUserDataPduId		
<b>Parent Container</b>	UdpNmUserDataTxPdu		
<b>Description</b>	This parameter defines the Handle ID of the NM User Data I-PDU.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_UdpNm_00057 :</b>		
<b>Name</b>	UdpNmTxUserDataPduRef		
<b>Parent Container</b>	UdpNmUserDataTxPdu		
<b>Description</b>	Reference to the NM User Data I-PDU in the global PDU collection.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>No Included Containers</b>
-------------------------------

### 10.2.7 UdpNmPnInfo

<b>SWS Item</b>	<b>ECUC_UdpNm_00067 :</b>	
<b>Container Name</b>	UdpNmPnInfo	
<b>Parent Container</b>	UdpNmGlobalConfig	
<b>Description</b>	PN information configuration	
<b>Configuration Parameters</b>		

<b>SWS Item</b>	<b>ECUC_UdpNm_00069 :</b>		
<b>Name</b>	UdpNmPnInfoLength		
<b>Parent Container</b>	UdpNmPnInfo		
<b>Description</b>	Specifies the length of the PN request information in the NM message.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 7		
<b>Default value</b>	1		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel.		

<b>SWS Item</b>	<b>ECUC_UdpNm_00068 :</b>		
<b>Name</b>	UdpNmPnInfoOffset		
<b>Parent Container</b>	UdpNmPnInfo		
<b>Description</b>	Specifies the offset of the PN request information in the NM message.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 7		
<b>Default value</b>	1		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel.		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
UdpNmPnFilterMaskByte	0..7	PN information configuration

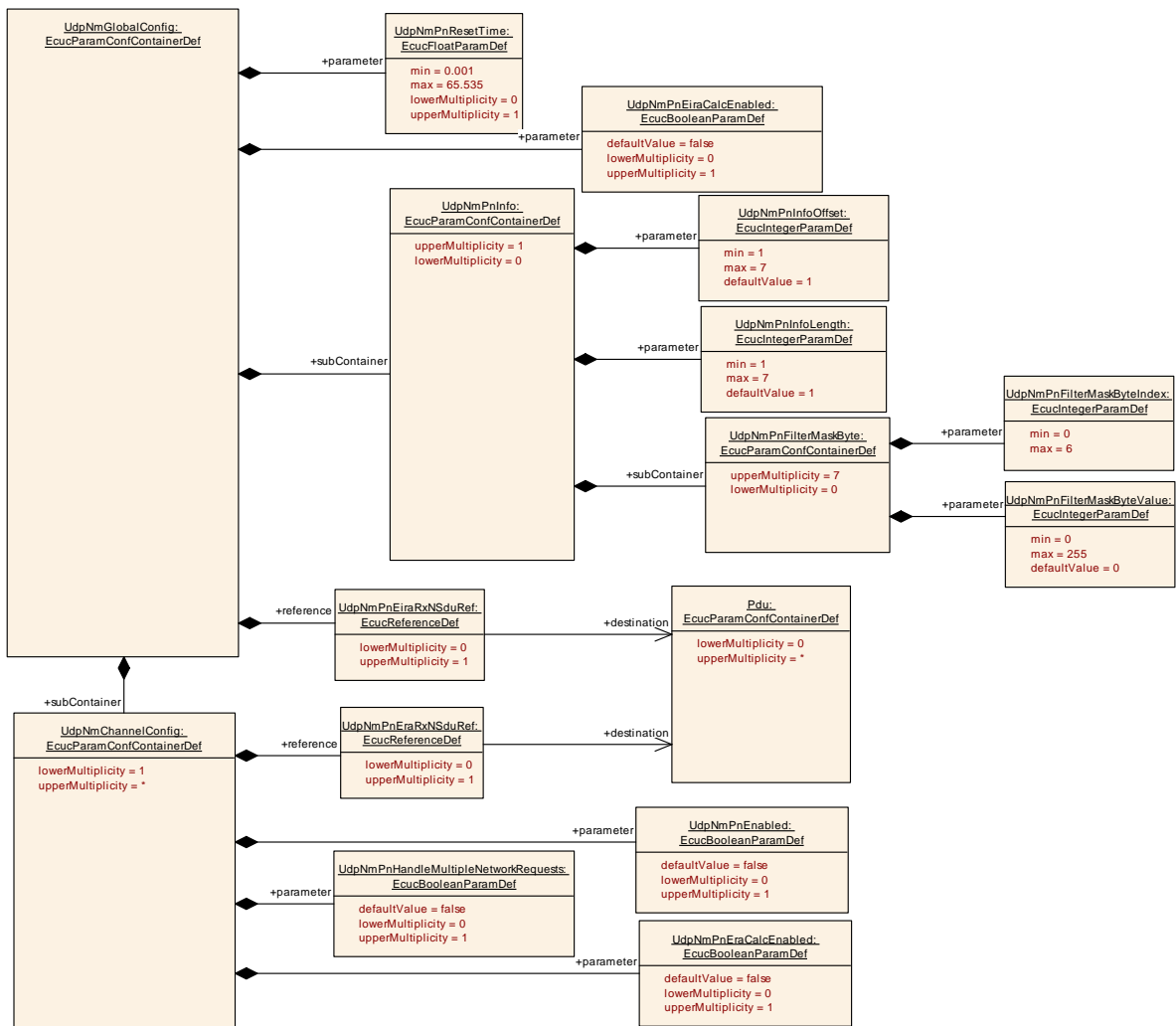


Figure 12: Diagram: UdpNmPNConfig

### 10.2.8 UdpNmPnFilterMaskByte

<b>SWS Item</b>	<b>ECUC_UdpNm_00070 :</b>		
<b>Container Name</b>	UdpNmPnFilterMaskByte		
<b>Parent Container</b>	UdpNmPnInfo		
<b>Description</b>	PN information configuration		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_UdpNm_00071 :</b>		
<b>Name</b>	UdpNmPnFilterMaskByteIndex		
<b>Parent Container</b>	UdpNmPnFilterMaskByte		
<b>Description</b>	Index of the filter mask byte. Specifies the position within the filter mask byte array.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 6		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel. UdpNmPnFilterMaskByteIndex < UdpNmPnInfoLength		

<b>SWS Item</b>	<b>ECUC_UdpNm_00072 :</b>		
<b>Name</b>	UdpNmPnFilterMaskByteValue		
<b>Parent Container</b>	UdpNmPnFilterMaskByte		
<b>Description</b>	Parameter to configure the filter mask byte.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	0		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: only available if UdpNmPnEnabled == true for at least one UdpNm Channel; UdpNmPnFilterMaskByteIndex < UdpNmPnInfoLength		

**No Included Containers**

### 10.3 Published parameters

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.

## 11 Not applicable requirements

[SWS\_UdpNm\_NA\_00999] [ This specification item references requirements that are not applicable to this specification. ] (SRS\_BSW\_00170, SRS\_BSW\_00375, SRS\_BSW\_00416, SRS\_BSW\_00168, SRS\_BSW\_00423, SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_BSW\_00426, SRS\_BSW\_00427, SRS\_BSW\_00429, SRS\_BSW\_00432, SRS\_BSW\_00336, SRS\_BSW\_00417, SRS\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00005, SRS\_BSW\_00415, SRS\_BSW\_00164, SRS\_BSW\_00325, SRS\_BSW\_00160, SRS\_BSW\_00413, SRS\_BSW\_00347, SRS\_BSW\_00305, SRS\_BSW\_00307, SRS\_BSW\_00335, SRS\_BSW\_00410, SRS\_BSW\_00314, SRS\_BSW\_00328, SRS\_BSW\_00312, SRS\_BSW\_00006, SRS\_BSW\_00377, SRS\_BSW\_00306, SRS\_BSW\_00309, SRS\_BSW\_00330, SRS\_BSW\_00331, SRS\_BSW\_00172, SRS\_BSW\_00010, SRS\_BSW\_00333, SRS\_BSW\_00321, SRS\_BSW\_00341, SRS\_BSW\_00334, SRS\_Nm\_00151, SRS\_Nm\_00046, SRS\_Nm\_00050, SRS\_Nm\_00052, SRS\_Nm\_02509, SRS\_Nm\_00153, SRS\_Nm\_00054, SRS\_Nm\_00142, SRS\_Nm\_00144, SRS\_Nm\_00147, SRS\_Nm\_00154)