Document Title	Specification of UDP Network Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	414
Document Classification	Standard

Document Version	2.0.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	**Version**	**Changed by**	**Change Description**
09.12.2011	2.0.0	AUTOSAR Administration	• Support coordinated shutdown • New traceability mechanism
29.10.2010	1.1.0	AUTOSAR Administration	• ComStack Harmonization • Harmonization of NM interfaces
07.12.2009	1.0.0	AUTOSAR Administration	Initial Release

Table of Contents

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].

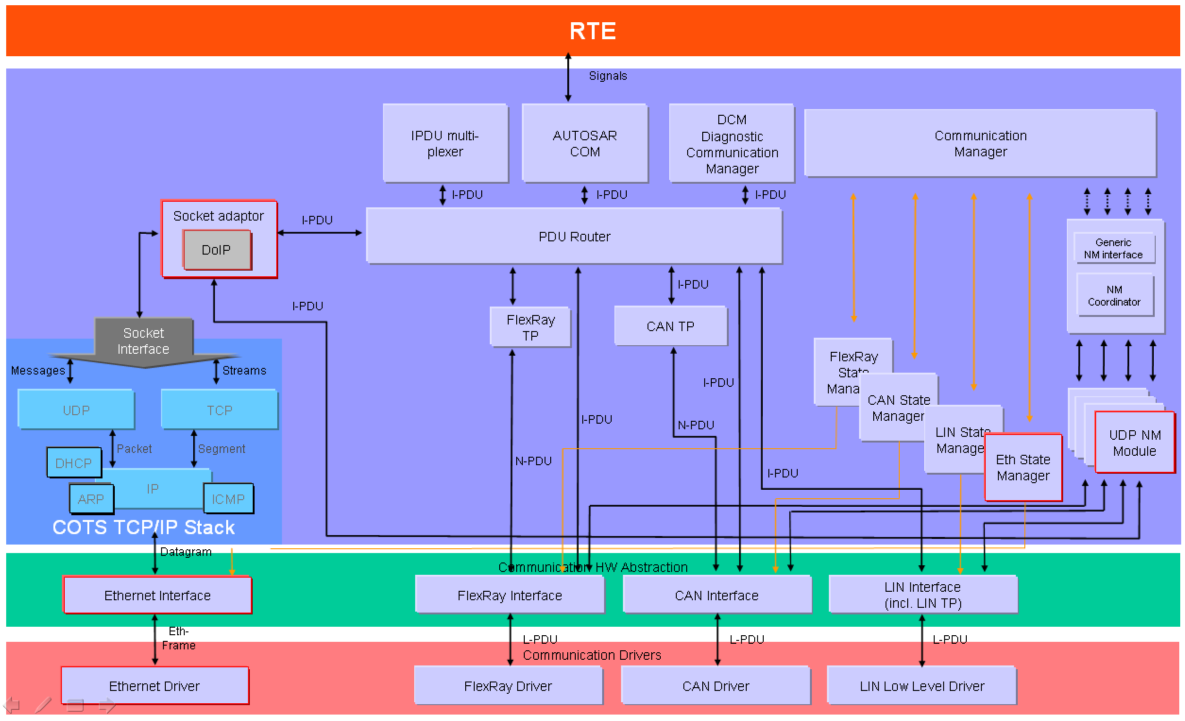


Figure 1: Extended AUTOSAR Communication Stack.

2 Acronyms and abbreviations

Acronym or Abbreviation:	***Description:***
API	Application Programming Interface
BSW	Basic Software
CanIf	CAN Interface
DEM	Diagnostic Event Manager
DET	Development Error Tracer
IP	Internet Protocol
NM	Network Management
PDU	Protocol Data Unit
SDU	Service Data Unit
TCP	Transmission Control Protocol
TCP/IP	A family of communication protocols used in computer networks
UDP	User Datagram Protocol
UdpNm	UDP Network Management

Term:	***Description:***
PDU transmission ability is disabled	This means that the NM message transmission has been disabled by the optional service UdpNm_DisableCommunication.
Repeat Message Request Bit Indication	UdpNm_SoAdIfRxIndication finds the Repeat Message Bit set in the Control Bit Vector of a received NM message.
NM PDU	Refers to the payload transmitted in a packet. It contains the NM User Data as well as the Control Bit Vector and the Source Node Identifier.
NM Packet	Refers to an Ethernet Frame containing an IP as well as a UDP header in addition to the data (PDU) transmitted by the NM in the payload section.
NM Message	Most abstract term referring to any single information item transferred within the methodology of the NM algorithm.
Bus-Off state	Refers to a situation where no cable is connected to the Ethernet HW.

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 CAN Interface
AUTOSAR_SWS_CANInterface.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_ECUStateManager.pdf

[12] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[13] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf

[14] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[15] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[16] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[17] Specification of Compiler Abstraction
AUTOSAR_SWS_CompilerAbstraction.pdf

[18] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[19] Specification of Socket Adaptor
AUTOSAR_SWS_SocketAdaptor.pdf

[20] Requirements on Ethernet
AUTOSAR_SRS_Ethernet.pdf

[21] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

3.2 Related standards and norms

[22] IEEE 1003.2
http://www.opengroup.org/onlinepubs/000095399/
[23] ISO 14229 Road Vehicles – Unified Diagnostic Services (UDS)

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.

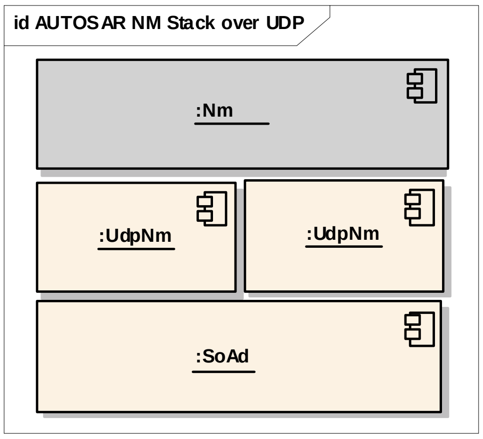


Figure 2: AUTOSAR NM stack within an example ECU belonging to two UDP NM-clusters

[UDPNM131]⌈ 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).

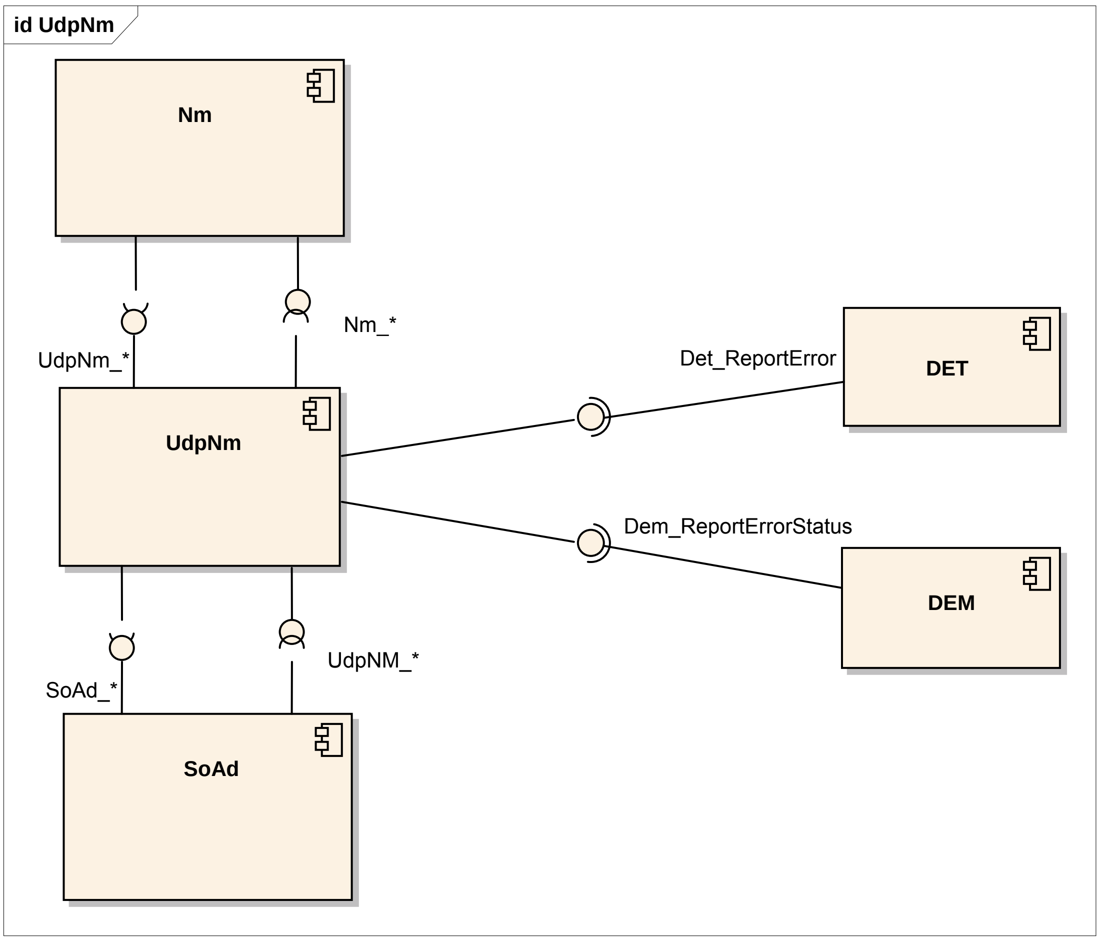


Figure 3: Dependencies on other modules.

5.1 File Structure

5.1.1 Code File Structure

[UDPNM081]⌈ 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. ⌋(BSW00419, BSW00346, BSW158, BSW00308)

5.1.2 Header File Structure

[UDPNM044]⌈ The UdpNm module shall provide the following H-files:

- `UdpNm.h` (for declaration of provided interface functions)
- `UdpNm_Cbk.h` (for declaration of provided call-back functions)
- `UdpNm_Cfg.h` (for pre-compile time configurable parameters) ⌋(BSW00345, BSW00380, BSW00381, BSW00412, BSW00346, BSW158, BSW00370, BSW00302)

[UDPNM082]⌈ The UdpNm module shall include the following H-files:
`ComStack_Types.h`

- Note: The following header files are indirectly included by `ComStack_Types.h`:
    - `Std_Types.h` (for AUTOSAR standard types)
    - `Platform_Types.h` (for platform specific types)
    - `Compiler.h` (for compiler specific language extensions)
- `UdpNm.h` (for declaration of provided interface functions)
- `Nm_Cbk.h` (for UdpNm specific call-backs to the Generic Network Management Interface)
- `Det.h` (for interface of DET – optional included only if DET is configured)
- `NmStack_Types.h` (for common network management types)
- `SchM_UdpNm.h` (for services of the Basic Software Scheduler)
- `MemMap.h` (for Memory Mapping) ⌋(BSW00348, BSW00353, BSW00361, BSW00301)

[UDPNM207] ⌈ The UdpNm module shall include the `Dem.h` file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in `Dem_IntErrId.h`. ⌋(BSW00409)

[UDPNM083] ⌈ The UdpNM module shall include the following header files containing configuration data:

- `SoAd_Cfg.h` (for the PDU IDs and socket connections)
- `Nm_Cfg.h` (for the derived configuration items from Nm) ⌋(BSW00383, BSW00301)

[UDPNM311] ⌈ The UdpNm module shall include `PduR_UdpNm.h` if `UdpNmComUserDataSupport` is enabled. ⌋()

6 Requirements traceability

Requirement	Satisfied by
-	UDPNM060
-	UDPNM119
-	UDPNM178
-	UDPNM246
-	UDPNM153
-	UDPNM088
-	UDPNM312
-	UDPNM163
-	UDPNM198
-	UDPNM154
-	UDPNM005
-	UDPNM098
-	UDPNM226
-	UDPNM061
-	UDPNM232
-	UDPNM101
-	UDPNM132
-	UDPNM077
-	UDPNM324
-	UDPNM233
-	UDPNM190
-	UDPNM109
-	UDPNM149
-	UDPNM099
-	UDPNM310
-	UDPNM121
-	UDPNM294
-	UDPNM191
-	UDPNM159
-	UDPNM013
-	UDPNM242
-	UDPNM258
-	UDPNM141
-	UDPNM316
-	UDPNM173
-	UDPNM019
-	UDPNM103
-	UDPNM170
-	UDPNM221

-	UDPNM100
-	UDPNM240
-	UDPNM180
-	UDPNM093
-	UDPNM306
-	UDPNM045
-	UDPNM085
-	UDPNM113
-	UDPNM172
-	UDPNM179
-	UDPNM144
-	UDPNM089
-	UDPNM228
-	UDPNM014
-	UDPNM108
-	UDPNM086
-	UDPNM175
-	UDPNM151
-	UDPNM115
-	UDPNM122
-	UDPNM106
-	UDPNM096
-	UDPNM127
-	UDPNM135
-	UDPNM104
-	UDPNM074
-	UDPNM164
-	UDPNM292
-	UDPNM169
-	UDPNM146
-	UDPNM290
-	UDPNM192
-	UDPNM075
-	UDPNM039
-	UDPNM035
-	UDPNM148
-	UDPNM133
-	UDPNM230
-	UDPNM143
-	UDPNM189
-	UDPNM289
-	UDPNM305

-	UDPNM248
-	UDPNM147
-	UDPNM161
-	UDPNM032
-	UDPNM319
-	UDPNM185
-	UDPNM165
-	UDPNM309
-	UDPNM160
-	UDPNM094
-	UDPNM287
-	UDPNM110
-	UDPNM128
-	UDPNM243
-	UDPNM219
-	UDPNM105
-	UDPNM210
-	UDPNM114
-	UDPNM181
-	UDPNM176
-	UDPNM139
-	UDPNM126
-	UDPNM217
-	UDPNM123
-	UDPNM152
-	UDPNM214
-	UDPNM255
-	UDPNM102
-	UDPNM168
-	UDPNM107
-	UDPNM199
-	UDPNM150
-	UDPNM138
-	UDPNM158
-	UDPNM092
-	UDPNM288
-	UDPNM197
-	UDPNM025
-	UDPNM234
-	UDPNM118
-	UDPNM187
-	UDPNM111

-	UDPNM112
-	UDPNM208
-	UDPNM237
-	UDPNM145
-	UDPNM218
-	UDPNM117
-	UDPNM244
-	UDPNM162
-	UDPNM130
-	UDPNM206
-	UDPNM087
-	UDPNM307
-	UDPNM018
-	UDPNM220
-	UDPNM116
-	UDPNM317
-	UDPNM193
-	UDPNM222
-	UDPNM213
-	UDPNM229
-	UDPNM174
-	UDPNM037
-	UDPNM040
-	UDPNM120
-	UDPNM051
-	UDPNM194
-	UDPNM239
-	UDPNM124
-	UDPNM311
-	UDPNM177
-	UDPNM315
-	UDPNM131
-	UDPNM249
-	UDPNM304
-	UDPNM308
-	UDPNM020
-	UDPNM095
-	UDPNM314
-	UDPNM224
-	UDPNM211
-	UDPNM072
-	UDPNM033

-	UDPNM196
-	UDPNM166
-	UDPNM227
-	UDPNM188
-	UDPNM231
-	UDPNM318
-	UDPNM137
-	UDPNM097
-	UDPNM247
-	UDPNM212
-	UDPNM223
-	UDPNM209
-	UDPNM313
-	UDPNM076
BSW	UDPNM999
BSW00301	UDPNM082, UDPNM083
BSW00302	UDPNM044
BSW00305	UDPNM999
BSW00306	UDPNM999
BSW00307	UDPNM999
BSW00308	UDPNM081
BSW00309	UDPNM999
BSW00312	UDPNM999
BSW00314	UDPNM999
BSW00321	UDPNM999
BSW00323	UDPNM241
BSW00325	UDPNM999
BSW00326	UDPNM999
BSW00328	UDPNM999
BSW00330	UDPNM999
BSW00331	UDPNM999
BSW00333	UDPNM999
BSW00334	UDPNM999
BSW00335	UDPNM999
BSW00336	UDPNM999
BSW00341	UDPNM999
BSW00345	UDPNM044
BSW00346	UDPNM081, UDPNM044
BSW00347	UDPNM999
BSW00348	UDPNM082
BSW00353	UDPNM082
BSW00361	UDPNM082

BSW00370	UDPNM044
BSW00375	UDPNM999
BSW00377	UDPNM999
BSW00380	UDPNM044
BSW00381	UDPNM044
BSW00383	UDPNM083
BSW00387	UDPNM999
BSW00409	UDPNM207
BSW00410	UDPNM999
BSW00412	UDPNM044
BSW00413	UDPNM999
BSW00415	UDPNM999
BSW00416	UDPNM999
BSW00417	UDPNM999
BSW00419	UDPNM081
BSW00423	UDPNM999
BSW00424	UDPNM999
BSW00425	UDPNM999
BSW00426	UDPNM999
BSW00427	UDPNM999
BSW00429	UDPNM999
BSW00432	UDPNM999
BSW00434	UDPNM999
BSW005	UDPNM999
BSW006	UDPNM999
BSW010	UDPNM999
BSW02509	UDPNM999
BSW02512	UDPNM216, UDPNM215
BSW046	UDPNM999
BSW050	UDPNM999
BSW052	UDPNM999
BSW054	UDPNM999
BSW136	UDPNM999
BSW139	UDPNM999
BSW140	UDPNM999
BSW142	UDPNM999
BSW144	UDPNM999
BSW147	UDPNM999
BSW151	UDPNM999
BSW153	UDPNM999
BSW154	UDPNM999
BSW158	UDPNM081, UDPNM044

BSW160	UDPNM999
BSW161	UDPNM999
BSW162	UDPNM999
BSW164	UDPNM999
BSW168	UDPNM999
BSW170	UDPNM999
BSW172	UDPNM999

Document: AUTOSAR General Requirements on Basic Software Modules [2].

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	Ok; see chapter 10.2
[BSW00404] Reference to post build time configuration	Ok; see Chapter 10.2
[BSW00405] Reference to multiple configuration sets	Ok; see Chapter 10.2
[BSW00345] Pre-compile-time configuration	Ok; see UDPNM044
[BSW159] Tool-based configuration	Ok; see Chapter 10.2
[BSW167] Static configuration checking	Ok; see Chapter 10.2
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	n/a (UdpNm is no SW-C)
[BSW171] Configurability of optional functionality	Ok; see Chapter 10.2
[BSW00380] Separate C-Files for configuration parameters	Ok; see UDPNM044
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Ok; see UDPNM081
[BSW00381] Separate configuration header file for pre-compile time parameters	Ok; see UDPNM044
[BSW00412] Separate H-File for configuration parameters	Ok; see UDPNM044
[BSW00383] List dependencies of configuration files	Ok; see UDPNM083
[BSW00384] List dependencies to other modules	Ok; see Figure 3
[BSW00387] Specify the configuration class of call-back function	n/a (Call-back functions are not configurable)
[BSW00388] Introduce containers	Ok; see Chapter 10.2
[BSW00389] Containers shall have names	Ok; see Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Ok; see Chapter 10.2
[BSW00391] Parameter shall have unique names	Ok; see Chapter 10.2
[BSW00392] Parameters shall have a type	Ok; see Chapter 10.2
[BSW00393] Parameters shall have a range	Ok; see Chapter 10.2
[BSW00394] Specify the scope of the parameters	Ok; see Chapter 10.2
[BSW00395] List the required parameters (per parameter)	Ok; see Chapter 10.2
[BSW00396] Configuration classes	Ok; see Chapter 10.2
[BSW00397] Pre-compile-time parameters	Ok; see Chapter 10.2
[BSW00398] Link-time parameters	Ok; see Chapter 10.2
[BSW00399] Loadable Post-build time parameters	Ok; see Chapter 10.2
[BSW00400] Selectable Post-build time parameters	Ok; see Chapter 10.2
[BSW00402] Published information	Ok; see Chapter 10.3
[BSW00375] Notification of wake-up reason	n/a (UdpNm does not wake-up an ECU)
[BSW101] Initialization interface	Ok; see chapter 8.3.1
[BSW00416] Sequence of Initialization	n/a (sequence is defined by ComM)
[BSW00406] Check module initialization	Ok; see chapter 7.12.3
[BSW168] Diagnostic Interface of SW components	n/a (diagnostics for UdpNm not required)
[BSW00407] Function to read out published parameters	Ok; see chapter 8.3.14
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	n/a (UdpNm has no interface to the RTE)
[BSW00424] BSW main processing function task allocation	n/a (UdpNm scheduled function is called by the BSW scheduler)
[BSW00425] Trigger conditions for schedulable objects	n/a (implementation specific)
[BSW00426] Exclusive areas in BSW modules	n/a (implementation specific)
[BSW00427] ISR description for BSW modules	n/a (implementation specific)
[BSW00428] Execution order dependencies of main processing functions	Ok; see chapter 7.11
[BSW00429] Restricted BSW OS functionality access	n/a (none of these services are used by UdpNm)
[BSW00431] The BSW Scheduler module implements task bodies	Ok; see chapter 7.11
[BSW00432] Modules should have separate main processing	n/a (transmission and reception is

functions for read/receive and write/transmit data path	handled in UdpNm_MainFunction)
[BSW00433] Calling of main processing functions	Ok; see chapter 7.11
[BSW00434] The Schedule Module shall provide an API for exclusive areas	n/a (implementation specific)
[BSW00336] Shutdown interface	n/a (no shutdown interface needed)
[BSW00337] Classification of errors	Ok; see chapter 7.10
[BSW00338] Detection and Reporting of development errors	Ok; see chapter 7.10 and 10.2
[BSW00369] Do not return development error codes via API	Ok; see chapter 7.12.3
[BSW00339] Reporting of production relevant error status	Ok; see chapter 7.10
[BSW00417] Reporting of Error Events by Non-Basic Software	n/a (UdpNm is no SW-C)
[BSW00323] API parameter checking	Ok; see UDPNM241
[BSW004] Version check	Ok; 7.13
[BSW00409] Header files for production code error IDs	Ok; see UDPNM207
[BSW00385] List possible error notifications	Ok; see 7.10
[BSW00386] Configuration for detecting an error	Ok: UDPNM_DEV_ERROR_DETECT
[BSW161] Microcontroller abstraction	n/a (UdpNm microcontroller independent)
[BSW162] ECU layout abstraction	n/a (UdpNm is ECU hardware independent)
[BSW005] No hard coded horizontal interfaces within MCAL	n/a (UdpNm is not part of the MCAL)
[BSW00415] User dependent include files	n/a (not flexible with respect to future extensions)
[BSW164] Implementation of interrupt service routines	n/a (no ISR provided)
[BSW00325] Runtime of interrupt service routines	n/a (no ISR provided)
[BSW00326] Transition from ISRs to OS tasks	n/a (no ISR provided)
[BSW00342] Usage of source code and object code	Ok; see chapter 10.2.1
[BSW00343] Specification and configuration of time	Ok; see chapter 10.2
[BSW160] Human-readable configuration data	n/a (implementation specific)
[BSW007] HIS MISRA C	Ok; all implementation related information
[BSW00300] Module naming convention	Ok; UdpNm prefix is used
[BSW00413] Accessing instances of BSW modules	n/a (implementation specific)
[BSW00347] Naming separation of different instances of BSW drivers	n/a (implementation specific)
[BSW00305] Self-defined data types naming convention	n/a (no self-defined data types used)
[BSW00307] Global variables naming convention	n/a (no global variables specified)
[BSW00310] API naming convention	Ok; see chapter 7.12.3
[BSW00373] Main processing function naming convention	Ok; see chapter 8.6.1
[BSW00327] Error values naming convention	Ok; see chapter 7.10
[BSW00335] Status values naming convention	n/a (no status values exported)
[BSW00350] Development error detection keyword	Ok; see chapter 8.8
[BSW00408] Configuration parameter naming convention	Ok; see chapter 10.2
[BSW00410] Compiler switches shall have defined values	n/a (UdpNm is compiler independent)
[BSW00411] Get version info keyword	Ok; see chapter 8.3.14
[BSW00346] Basic set of module files	Ok; see UDPNM081 and UDPNM044
[BSW158] Separation of configuration from implementation	Ok; see UDPNM081 and UDPNM044
[BSW00314] Separation of interrupt frames and service routines	n/a (UdpNm doesn't have interrupt frame definitions)
[BSW00370] Separation of call-back interface from API	Ok; see UDPNM044
[BSW00348] Standard type header	Ok; see UDPNM082
[BSW00353] Platform specific type header	Ok; see UDPNM082
[BSW00361] Compiler specific language extension header	Ok; see UDPNM082
[BSW00301] Limit imported information	Ok; see UDPNM082 and

	UDPNM083
[BSW00302] Limit exported information	Ok; see UDPNM044 and chapter 7.12.3
[BSW00328] Avoid duplication of code	n/a (implementation specific)
[BSW00312] Shared code shall be reentrant	n/a (implementation specific)
[BSW006] Platform independency	n/a (UdpNm is hardware independent)
[BSW00357] Standard API return type	Ok; see chapter 7.12.3
[BSW00377] Module specific API return types	n/a (UdpNm doesn't define own types)
[BSW00304] AUTOSAR integer data types	Ok; see chapter 7.12.3
[BSW00355] Do not redefine AUTOSAR integer data types	Ok; see chapter 7.12.3
[BSW00378] AUTOSAR boolean type	Ok; see chapter 7.12.3 and 10.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	n/a (implementation specific)
[BSW00308] Definition of global data	Ok; see UDPNM081
[BSW00309] Global data with read-only constraint	n/a (implementation specific)
[BSW00371] Do not pass function pointers via API	Ok; see chapter 7.12.3
[BSW00358] Return type of init() functions	Ok; see chapter 8.3.1
[BSW00414] Parameter of init function	Ok; see chapter 8.3.1
[BSW00376] Return type and parameters of main processing functions	Ok; see chapter 8.6.1
[BSW00359] Return type of call-back functions	Ok; see chapter 8.7
[BSW00360] Parameters of call-back functions	Ok; see chapter 8.7
[BSW00329] Avoidance of generic interfaces	Ok; see chapter 8
[BSW00330] Usage of macros / inline functions instead of functions	n/a (implementation specific)
[BSW00331] Separation of error and status values	n/a (UdpNm doesn't provide status information)
[BSW009] Module User Documentation	Ok; see whole document
[BSW00401] Documentation of multiple instances of configuration parameters	Ok; see chapter 10.2
[BSW172] Compatibility and documentation of scheduling strategy	n/a (implementation specific)
[BSW010] Memory resource documentation	n/a (implementation specific)
[BSW00333] Documentation of call-back function context	n/a (implementation specific)
[BSW00374] Module vendor identification	Ok; see chapter 10.30
[BSW00379] Module identification	Ok; see chapter 10.3
[BSW003] Version identification	Ok; see chapter 10.3
[BSW00318] Format of module version numbers	Ok; see chapter 10.3
[BSW00321] Enumeration of module version numbers	n/a (implementation specific)
[BSW00341] Microcontroller compatibility documentation	n/a (UdpNm is microcontroller independent)
[BSW00334] Provision of XML file	n/a (implementation specific)

Document: AUTOSAR Requirements on Basic Software, Module NM [3].

Requirement	***Satisfied by***
[BSW150] Configuration of functionality	Ok; see chapter 10.2
[BSW151] Integration into running NM cluster	n/a
[BSW043] Bus Traffic without NM Initialization	n/a (ComM is responsible to initialize the communication components)
[BSW044] Applicability to different types of communication systems	Ok; see chapter 4.2
[BSW045] NM-cluster Independent Shutdown Coordination	Ok; see chapter 7.12.3
[BSW046] Trigger of startup of all Nodes at any Point in Time	n/a (not in the responsibility of UdpNm)
[BSW047] Bus Keep Awake Services	Ok; see chapter 8.3.6
[BSW048] Bus Sleep Mode	n/a (not in the responsibility of UdpNm)
[BSW050] NM State Information	n/a (the application can determine the Nm states using ComM API)
[BSW051] NM State Change Indication	Ok; see chapter 8.7.1
[BSW052] Notification that all other ECUs are ready to sleep	n/a (not in the responsibility of UdpNm)
[BSW02509] Notification that at least one other node is not ready to sleep anymore	n/a (not in the responsibility of UdpNm)
[BSW02503] Sending user data	Ok; see chapter 8.3.7
[BSW02504] Receiving user data	Ok; see chapter 8.3.8
[BSW153] Detection of present nodes	n/a (not in the responsibility of UdpNm)
[BSW02508] Unambiguous node identification per bus	Ok; see chapter 8.3.10
[BSW02505] Sending node identifier	Ok; see chapter 7.7.7
[BSW02506] Receiving node identifier	Ok; see chapter 8.3.9
[BSW02511] Configurable Role in Cluster Shutdown	Ok; see 7.7.3
[BSW053] Deterministic Behavior in Case of Bus Unavailability	UdpNm interfaces to SoAd, unavailabilty is handled there.
[BSW137] Communication system error handling	UdpNm interfaces to SoAd, communication system errors are handled there.
[BSW136] Coordination of coupled networks	n/a (not in the scope of UdpNm)
[BSW140] Compliance with OSEK NM on a gateway	n/a (not in the scope of UdpNm)
[BSW054] Deterministic Time for Bus Sleep	n/a (not in the scope of UdpNm)
[BSW142] Limitation of NM bus load	n/a (not in the scope of UdpNm)
[BSW143] Predictable NM bus load	Ok; see 7.2
[BSW144] ECU cluster size	n/a (UdpNm hasn't any restriction concerning cluster size)
[BSW145] Robustness against NM message losses	UdpNm is robust against loss of NM messages for a time specified by `UDPNM_TIMEOUT_TIME`.
[BSW146] Robustness against NM message jitter	UdpNm is robust against jitter of NM messages for up to a time specified by `UDPNM_TIMEOUT_TIME`.
[BSW147] Processor independent algorithm	n/a (not in the responsibility of UdpNm)
[BSW149] Configurable Timing	Ok; see chapter 10.2
[BSW154] Bus independency of API	n/a (UdpNm has to be bus dependent)
[BSW148] Separation of Communication system dependent parts	Ok; see whole document
[BSW139] Compliance with OSEK NM on one cluster	n/a (not in the scope of UdpNm)
[BSW02510] Immediate Transmission Confirmation	Ok; see chapter 8.4.1
[BSW02512] CommunicationControl (`0x28`) service support	UDPNM215, UDPNM216

Document: AUTOSAR Requirements on Ethernet [20].

Requirement	***Satisfied by***
[BSW41900042] UdpNm Network abstraction	OK, see 8.3
[BSW41900037] UdpNm Network management information	OK, see 8.3

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:

[UDPNM087] ⌈ Every network node shall transmit periodic NM PDUs as long as it requires bus-communication; otherwise it shall not transmit NM PDUs. ⌋()

[UDPNM088] ⌈ If bus communication is released and there are no NM PDUs on the bus for a configurable amount of time, determined by `UDPNM_TIMEOUT_TIME + UDPNM_WAIT_BUS_SLEEP_TIME` (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:

[UDPNM089] ⌈ 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 (Figure 6).

7.2 Operational Modes

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

[UDPNM092] ⌈ The AUTOSAR UdpNm shall contain three operational modes visible at the modules interface:

- Network Mode
- Prepare Bus-Sleep Mode
- Bus-Sleep Mode ⌋()

[UDPNM093] ⌈ 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

[UDPNM094] ⌈ The Network Mode shall consist of three internal states:

- Repeat Message State
- Normal Operation State
- Ready Sleep State ⌋()

[UDPNM095] ⌈ When the Network Mode is entered from Bus-Sleep Mode or Prepare Bus-Sleep Mode, by default, the Repeat Message State shall be entered. ⌋()

[UDPNM096] ⌈ When the Network Mode is entered, the NM-Timeout Timer shall be started. ⌋()

[UDPNM097] ⌈ When the Network Mode is entered, the UdpNm shall notify the upper layer by calling `Nm_NetworkMode`. ⌋()

[UDPNM098] ⌈ Upon successful reception of an NM PDU (call of `UdpNm_SoAdIfRxIndication`) in Network Mode, the NM-Timeout Timer shall be restarted. ⌋()

[UDPNM099] ⌈ Upon transmission of an NM PDU (call of `UdpNm_SoAdIfTxConfirmation`) 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.

[UDPNM206] ⌈ 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 (`UDPNM_REPEAT_MESSAGE_TIME`). Optionally it can be used for detection of present nodes.

[UDPNM100] ⌈ 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 started unless passive mode is enabled. ⌋()

[UDPNM101] ⌈ When the NM-Timeout Timer expires in the Repeat Message State, the NM-Timeout Timer shall be restarted. ⌋()

[UDPNM102] ⌈ The NM shall stay in the Repeat Message State for a configurable amount of time determined by the `UDPNM_REPEAT_MESSAGE_TIME` (configuration parameter); after that time the Repeat Message State shall be left. ⌋()

[UDPNM103] ⌈ When Repeat Message State is left, the Normal Operation State shall be entered, if the network has been requested (see UDPNM104). ⌋()

[UDPNM106] ⌈ When Repeat Message State is left, the Ready Sleep State shall be entered, if the network has been released (see UDPNM105). ⌋()

[UDPNM107] ⌈ When Repeat Message State is left and the option `UDPNM_NODE_DETECTION_ENABLED` is enabled, the Repeat Message Bit shall be cleared. ⌋()

[UDPNM137] ⌈ 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 `NM_E_NOT_EXECUTED`. ⌋()

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.

[UDPNM116] ⌈ 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. ⌋()

[UDPNM117] ⌈ When the NM-Timeout Timer expires in the Normal Operation State, the NM-Timeout Timer shall be restarted. ⌋()

[UDPNM118] ⌈ 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 UDPNM105). ⌋()

[UDPNM119] ⌈ At Repeat Message Request Bit Indication in the Normal Operation State, the Normal Operation State shall be left and the Repeat Message State shall be entered. ⌋()

[UDPNM120] ⌈ At Repeat Message Request (`UdpNm_RepeatMessageRequest`) in the Normal Operation State, the Normal Operation State shall be left and the Repeat Message State shall be entered. ⌋()

[UDPNM121] ⌈ At Repeat Message Request (`UdpNm_RepeatMessageRequest`) in Normal Operation State the Repeat Message Bit shall be set. ⌋()

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.

[UDPNM108] ⌈ When the Ready Sleep State is entered from Repeat Message State or Normal Operation State, transmission of NM PDUs shall be stopped. ⌋()

Note: If passive mode is enabled no NM PDUs are transmited, no action is required.

[UDPNM109] ⌈ 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. ⌋()

[UDPNM110]⌈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 UDPNM104). ⌋()

[UDPNM111]⌈At Repeat Message Request Bit Indication in the Ready Sleep State, the Ready Sleep State shall be left and the Repeat Message State shall be entered. ⌋()

[UDPNM112]⌈At Repeat Message Request (`UdpNm_RepeatMessageRequest`) in the Ready Sleep State, the Ready Sleep State shall be left and the Repeat Message State shall be entered. ⌋()

[UDPNM113]⌈At Repeat Message Request (`UdpNm_RepeatMessageRequest`) in Ready Sleep State the Repeat Message Bit shall be set. ⌋()

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.

[UDPNM114]⌈When Prepare Bus-Sleep Mode is entered, the UdpNm shall notify the upper layer by calling `Nm_PrepareBusSleepMode`. ⌋()

[UDPNM115]⌈The NM shall stay in the Prepare Bus-Sleep Mode for a configurable amount of time determined by the `UDPNM_WAIT_BUS_SLEEP_TIME` (configuration parameter); after that time the Prepare Bus-Sleep Mode shall be left and the Bus-Sleep Mode shall be entered. ⌋()

[UDPNM124]⌈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 UDPNM095). ⌋()

[UDPNM123]⌈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 UDPNM095). ⌋()

[UDPNM122]⌈When the network has been requested in the Prepare Bus-Sleep Mode and the UdpNm module has entered Network Mode and if

UDPNM_IMMEDIATE_RESTART_ENABLED (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 UDPNM_IMMEDIATE_RESTART_ENABLED 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 a configurable amount of time determined by the UDPNM_TIMEOUT_TIME + UDPNM_WAIT_BUS_SLEEP_TIME (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 UDPNM_TIMEOUT_TIME and UDPNM_WAIT_BUS_SLEEP_TIME 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 UDPNM_TIMEOUT_TIME + UDPNM_WAIT_BUS_SLEEP_TIME (both configuration parameters).

[UDPNM126]⌈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. ⌋()

[UDPNM127] ⌈ When the UdpNm module receives successfully Network Management PDU in the Bus-Sleep Mode (call of `UdpNm_SoAdIfRxIndication`), the UdpNm module shall notify the upper layer by calling the callback function `Nm_NetworkStartIndication`. ⌋()

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

[UDPNM128]⌈If `UdpNm_PassiveStartUp` is called in the Bus-Sleep Mode, the UdpNm module shall enter the Network Mode; by default the Repeat Message State is entered (refer to UDPNM095 and UDPNM104). ⌋()

Note: In the Prepare Bus-Sleep Mode and Bus-Sleep Mode is assumed that the network is released, unless bus communication is explicitly requested.
UDPNM129: 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 UDPNM095 and UDPNM104).

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.

[UDPNM104]⌈ The function call `UdpNm_NetworkRequest` shall request the network. I.e. the UdpNm module shall change network state to 'requested'. ⌋()

[UDPNM105] ⌈ The function call `UdpNm_NetworkRelease` shall release the network. I.e. the UdpNm module shall change network state to 'released'. ⌋()

7.4 Initialization

[UDPNM141] ⌈ After successful initialization the Network Management state shall be set to `NM_STATE_BUS_SLEEP` ⌋()

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

[UDPNM143] ⌈ When initialized, by default, the UdpNm module shall set the network state to 'released'. ⌋()

[UDPNM144] ⌈ When initialized, by default, the UdpNm module shall enter the Bus-Sleep Mode. ⌋()

[UDPNM145] ⌈ If AUTOSAR UdpNm is not initialized it shall not prohibit bus traffic. ⌋()

[UDPNM147] ⌈ If `UdpNm_PassiveStartUp` is called in the Prepare Bus-Sleep Mode or Network Mode, the UdpNm module shall not execute this service and shall return `NM_E_NOT_EXECUTED`. ⌋()

[UDPNM060] ⌈ The function `UdpNm_Init` shall select the active configuration set by means of a configuration pointer parameter being passed (see 8.3.1). ⌋()

[UDPNM061] ⌈ After initialization the UdpNm Message Cycle Timer shall be stopped. ⌋()

Note: No timer (UdpNm Message Cycle Timer) is needed if `UDPNM_PASSIVE_MODE_ENABLED` is `TRUE`, because no NM messages are transmitted by such nodes.

[UDPNM033] ⌈ After initialization the transmission of NM messages shall be stopped. ⌋()

[UDPNM039] ⌈ If UdpNm is not initialized a call of any UdpNm function except `UdpNM_Init` shall be rejected and `NM_E_NOT_OK` shall be returned. If development error detection is enabled it shall report `UDPNM_E_NO_INIT` to the Development Error Tracer. ⌋()

[UDPNM025] ⌈ After initialization each byte of the user data bytes shall be set to `0xFF`. ⌋()

[UDPNM085] ⌈ After initialization the Control Bit Vector shall be set to `0x00`. ⌋()

[UDPNM148] ⌈ All instances of UDP NM on different ECUs in one NM cluster shall use the same UDP receive port (configuration parameter `UDPNM_PORT`). ⌋()

7.5 Execution

7.5.1 Processor architecture

[UDPNM146] ⌈ 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

[UDPNM246] ⌈ The configuration parameter `UDPNM_TIMEOUT_TIME` shall determine the AUTOSAR UdpNm timing parameter NM-Timeout Time. ⌋()

[UDPNM247] ⌈ The configuration parameter `UDPNM_REPEAT_MESSAGE_TIME` shall determine the AUTOSAR UdpNm timing parameter Repeat Message Time. ⌋()

[UDPNM248] ⌈ The configuration parameter `UDPNM_ WAIT_BUS_SLEEP_TIME` shall determine the AUTOSAR UdpNm timing parameter Wait Bus-Sleep Time. ⌋()

[UDPNM249] ⌈ The optional configuration parameter `UDPNM_REMOTE_SLEEP_IND_TIME` 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.

[UDPNM072] ⌈ The transmission of NM messages shall be configurable by means of `UDPNM_PASSIVE_MODE_ENABLED` (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 `UDPNM_PASSIVE_MODE_ENABLED` is `FALSE`.

[UDPNM237] ⌈ 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".

[UDPNM005] ⌈ If transmission of NM PDUs has been started, the UdpNm Message Cycle Timer shall be started with `UDPNM_MSG_CYCLE_OFFSET`. ⌋()

Note: This mechanism prevents bursts of NM messages.

[UDPNM032] ⌈ 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 `SoAdIf_Transmit`. ⌋()

[UDPNM040] ⌈ If the UdpNm Message Cycle Timer expires it shall be restarted with `UDPNM_MSG_CYCLE_TIME`. ⌋()

[UDPNM051] ⌈ If transmission of NM PDUs has been stopped the UdpNm Message Cycle Timer shall be canceled. ⌋()

7.6.2 Reception

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

[UDPNM035] ⌈ 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. ⌋()

[UDPNM037] ⌈ When an NM PDU has been received, the Nm function `Nm_PduRxIndication` shall be called, if `UDPNM_PDU_RX_INDICATION_ENABLED` (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.

[UDPNM149] ⌈ Detection of remote sleep indication shall be statically configurable with use of the `UDPNM_REMOTE_SLEEP_IND_ENABLED` switch (configuration parameter). ⌋()

[UDPNM150] ⌈ If no NM PDUs are received in the Normal Operation State for a configurable amount of time determined by the `UDPNM_REMOTE_SLEEP_IND_TIME` (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`. ⌋()

[UDPNM151] ⌈ 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_RemoteSleepCancelation`. ⌋()

[UDPNM152] ⌈ If Remote Sleep Indication has been previously detected and if Repeat Message State is entered from Normal Operation State, 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_RemoteSleepCancelation`. ⌋()

[UDPNM154] ⌈ 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 `NM_E_NOT_EXECUTED` shall be returned. ⌋()

7.7.2 User Data (optional)

[UDPNM158] ⌈ Support of NM user data shall be statically configurable using the `UDPNM_USER_DATA_ENABLED` switch (configuration parameter). ⌋()

[UDPNM159] ⌈ 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. ⌋()

[UDPNM160] ⌈ 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.

[UDPNM312] ⌈ If `UdpNmComUserDataSupport` is enabled the API `UdpNm_SetUserData` shall not be available. ⌋()

7.7.3 Passive Mode (optional)

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

[UDPNM161] ⌈ Passive Mode shall be statically configurable with use of the `UDPNM_PASSIVE_MODE_ENABLED` switch (configuration parameter). ⌋()

[UDPNM162] ⌈ Passive Mode shall be statically configured consistent for all instances within one ECU. ⌋()

[UDPNM163] ⌈ If Passive Mode is used (configuration parameter `UDPNM_PASSIVE_MODE_ENABLED`) the following options must not be used:

- Bus Synchronization (configuration parameter `UDPNM_BUS_SYNCHRONIZATION_ENABLED`)
- Remote Sleep Indication (configuration parameter `UDPNM_REMOTE_SLEEP_IND_ENABLED`)

- Node Detection (configuration parameter UDPNM_NODE_DETECTION_ENABLED) ⌋()

7.7.4 NM PDU Rx Indication (optional)

[UDPNM164] ⌈ At successful reception of a NM PDU the UdpNm shall notify the upper layer by calling Nm_PduRxIndication. ⌋()

Rationale: If any higher software layer needs to retrieve the NM PDU data of every NM PDU it is required to have an Rx Indication. Polling of the NM PDU data could result in loss of received NM PDU data in case of an NM PDU burst.

Note: UdpNm_SoAdIfRxIndication is called by SoAd upon NM PDU reception.

[UDPNM165] ⌈ The optional service Nm_PduRxIndication shall be statically configurable. It shall be available if UDPNM_PDU_RX_INDICATION_ENABLED is TRUE. ⌋()

7.7.5 State change notification (optional)

[UDPNM166] ⌈ 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 UDPNM_STATE_CHANGE_IND_ENABLED is TRUE). ⌋()

7.7.6 Communication Control (optional)

[UDPNM168] ⌈ Communication Control shall be statically configurable with use of the UDPNM_COM_CONTROL_ENABLED switch (configuration parameter). ⌋()

[UDPNM169] ⌈ During initialization of the UdpNm module, the UdpNm module shall enable the Network Management PDU transmission (start the UdpNm Message Cycle Timer with UDPNM_MSG_CYCLE_OFFSET). ⌋()

[UDPNM170] ⌈ The optional service UdpNm_DisableCommunication shall disable the NM PDU transmission ability. ⌋()

[UDPNM172] ⌈ The optional service UdpNm_DisableCommunication shall return NM_E_NOT_EXECUTED, if the current mode is not Network Mode. ⌋()

[UDPNM173] ⌈ 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. ⌋()

[UDPNM174] ⌈ When the NM PDU transmission ability is disabled, the NM-Timeout Timer shall be stopped. ⌋()

[UDPNM175] ⌈ When the NM PDU transmission ability is disabled, the detection of Remote Sleep Indication Timer shall be suspended. ⌋()

[UDPNM178] ⌈ When the Network Management PDU transmission ability is enabled, the UdpNm module shall start the UdpNm Message Cycle Timer with `UDPNM_MSG_CYCLE_OFFSET` in order to start transmission of Network Management PDUs. ⌋()

[UDPNM179] ⌈ When the NM PDU transmission ability is enabled, the NM-Timeout Timer shall be restarted. ⌋()

[UDPNM180] ⌈ When the NM PDU transmission ability is enabled, the detection of Remote Sleep Indication Timer shall be resumed. ⌋()

[UDPNM181] ⌈ The optional service `UdpNm_RequestBusSynchronization` shall return `NM_E_NOT_EXECUTED` if the NM PDU transmission ability is disabled. ⌋()

7.7.7 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.

[UDPNM320]If the UdpNm receives a NM message with the NmCoordinatorSleepReady bit (see CBV) set it shall indicate this to the Nm by calling `Nm_CoordReadyToSleepIndication`.
[UDPNM321]The `NmCoordinatorSleepReady` bit in the CBV shall be set by the API `UdpNm_SetSleepReadyBit`.

[UDPNM322]This feature is optional and only available if `UdpNmCoordinatorSyncSupport` is set to `TRUE`.

7.8 Payload (PDU) Structure

The figure below shows the default format of the NM packet payload:

	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.

[UDPNM074] ⌈ 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). ⌋()

[UDPNM075] ⌈ 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). ⌋()

[UDPNM076] ⌈ The length of an NM packet shall not exceed the MTU 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***
Byte 1	Res	Res	Res	Res	NM Coordinator Sleep Ready	Res R3.2 NM Coordinator ID (High Bit)	Res R3.2 NM Coordinator ID (Low Bit)	RptMsgRequest

Figure 5: Control Bit Vector.

[UDPNM045] ⌈ 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 1,2,4..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 UDPNM085).

[UDPNM013] ⌈ The source node identifier shall be set with the configuration parameter `UDPNM_NODE_ID` unless `UDPNM_PDU_NID_POSITION` is set to off. ⌋()

[UDPNM135] ⌈ Support of Repeat Message Request Bit and Repeat Message State Request shall be statically configurable with use of the `UDPNM_NODE_DETECTION_ENABLED` switch (configuration parameter). ⌋()

[UDPNM138] ⌈ The optional service call `UdpNm_GetPduData` shall provide whole payload (Source Node ID, Control Bit Vector and User Data) of the most recently received UDP NM packet. ⌋()

[UDPNM139] ⌈ The optional service UdpNm_GetPduData shall be statically configurable. It shall be available if `UDPNM_NODE_ID_ENABLED` or `UDPNM_NODE_DETECTION_ENABLED` or `UDPNM_USER_DATA_ENABLED` is `TRUE`. ⌋()

7.9 Functional requirements on UdpNm API

[UDPNM014] ⌈ If the node detection functionality is enabled, the function `Nm_RepeatMessageIndication` shall be called upon every reception of the RepeatMessageRequest bit if `UDPNM_REPEAT_MSG_IND_ENABLED` is enabled. ⌋()

[UDPNM086] ⌈ If `UDPNM_USER_DATA_ENABLED` is enabled and `UDPNM_USER_DATA_LENGTH` is set to `0x00` an error during configuration or compilation time shall be raised. ⌋()

7.10 Error Handling

7.10.1 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 [3] 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.

[UDPNM223] ⌈ On errors and exceptions, the UdpNm module shall not modify its current module state but shall simply report the error event to the DEM. ⌋()

In case of production errors, the Diagnostic Event Manager module (via the Function Inhibition Manager) will perform the appropriate action (e.g. status modification of the calling module).

[UDPNM239] ⌈ Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file `Dem_IntErrId.h` and included via `Dem.h`. ⌋()

[UDPNM240] ⌈ Development error values are of type `uint8`. ⌋()

[UDPNM018] ⌈ The following errors shall be detectable by the UdpNm depending on its build version (development/production mode). ⌋()

Type or error	***Relevance***	***Related error code***	***Error Value***
API service used without module initialization	Development	`UDPNM_E_NO_INIT`	0x01
API service called with wrong channel handle	Development	`UDPNM_E_INVALID_CHANNEL`	0x02
API service called with wrong PDU ID.	Development	`UDP_E_INVALID_PDUID`	0x03
UdpNm initialization has failed, e.g. selected configuration set doesn't exist	Production	`UDPNM_E_INIT_FAILED`	Assigned by the DEM
A call to the TCP/IP stack has failed	Production	`UDPNM_E_TCPIP_TRANSMIT_ERROR`	Assigned by the DEM
NM-Timeout Timer has abnormally expired outside of the Ready Sleep State; it may happen: (1) because of Bus-Off state, (2) if some ECU requests bus communication or node detection shortly before the NM-Timeout Timer expires so that a NM message can not be transmitted in time; this race condition applies to event-triggered systems	Production	`UDPNM_E_NETWORK_TIMEOUT`	Assigned by DEM
Null pointer has been passed as an argument (Does not apply to function UdpNm_Init)	Development	`UDPNM_E_NULL_POINTER`	0x12

7.10.2 Error detection

[UDPNM188] ⌈ The detection of development errors is configurable at pre-compile time. The switch `UDPNM_DEV_ERROR_DETECT` shall activate or deactivate the detection of all development errors. ⌋()

[UDPNM241] ⌈ If the UDPNM_DEV_ERROR_DETECT switch is TRUE API parameter checking is enabled. ⌋(BSW00323)

[UDPNM242]⌈ The detection of production code errors cannot be switched off. ⌋()

7.10.3 Error notification

[UDPNM019] ⌈ Detected development errors shall be reported to the Det_ReportError service of the Development Error Tracer service (DET) [14] if the pre-processor switch UDPNM_DEV_ERROR_DETECT is set. ⌋()

[UDPNM189] ⌈ Development errors shall not be returned by API functions; in case of a development error, the respective API function will return NM_E_NOT_OK, if applicable. ⌋()

[UDPNM020] ⌈ Production errors shall be reported to Diagnostic Event Manager (DEM) [13]. ⌋()

[UDPNM190] ⌈ Production errors shall not be returned by API functions; in case of a production error, the respective API function will return NM_E_NOT_OK, if applicable. ⌋()

[UDPNM191] ⌈ If not initialized, the NM shall reject every API service apart from UdpNm_Init; the called function shall not be executed, but instead of that it shall report UDPNM_E_NO_INIT to the Development Error Tracer (if development error detection is enabled) and it shall return UDPNM_E_NOT_OK to the calling function ⌋()

[UDPNM192] ⌈ 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 Development Error Tracer (if development error detection is enabled; the value of the invalid network handle shall be passed to DET as instance ID) and it shall return UDPNM_E_NOT_OK to the calling function. ⌋()

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

[UDPNM292] ⌈ When the NULL Pointer is passed as an argument to a UdpNm service, the called function shall not be executed, but shall report UDPNM_E_NULL_POINTER to the Development Error Tracer instead. It shall return NM_E_NOT_OK to the calling function if development error detection is enabled (UDPNM_DEV_ERROR_DETECT is set to TRUE). ⌋()

[UDPNM193] ⌈ When the NM-Timeout Timer expires in the Repeat Message State, the NM shall report `UDPNM_E_NETWORK_TIMEOUT` to Diagnostic Event Manager. ⌋()

[UDPNM194] ⌈ When the NM-Timeout Timer expires in the Normal Operation State, the NM shall report `UDPNM_E_NETWORK_TIMEOUT` to Diagnostic Event Manager. ⌋()

[UDPNM314] ⌈ 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. ⌋()

7.11 Scheduling of the main function

[UDPNM077] ⌈ The `UdpNm_MainFunction_<Instance Id>` functions shall be scheduled by the BSW scheduler (see [8]). ⌋()

7.12 Application notes

7.12.1 Wakeup notification

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

7.12.2 Coordination of coupled networks

[UDPNM185] ⌈ Support of bus synchronization on demand shall be statically configurable with use of the `UDPNM_BUS_SYNCHRONIZATION_ENABLED` 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.

7.12.3 Debugging Concept

[UDPNM287] ⌈ Each variable that shall be accessible by AUTOSAR Debugging shall be defined as global variable. ⌋()

[UDPNM288] ⌈ All type definitions of variables which shall be debugged, shall be accessible by the header file `UdpNm.h`. ⌋()

[UDPNM289] ⌈ The declaration of variables in the header file shall be such that it is possible to calculate the size of the variables by C-"sizeof". ⌋()

[UDPNM290] ⌈ Variables available for debugging shall be described in the respective Basic Software Module Description. ⌋()

7.13 Version check

[UDPNM319] ⌈ The UdpNm module shall perform Inter Module Checks to avoid integration of incompatible files.
The imported included files shall be checked by preprocessing directives.
The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION
Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the UdpNm module.
If the values are not identical to the expected values, an error shall be reported. ⌋()

8 API specification

[UDPNM243] ⌈ The UdpNm module shall provide parameter value check only in "development mode". ⌋()

[UDPNM244] ⌈ 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
```

Module	***Imported Type***
ComStack_Types	PduIdType
	PduInfoType
	NetworkHandleType
Dem	Dem_EventIdType
	Dem_EventStatusType
Nm	Nm_ModeType
	Nm_StateType
Std_Types	Std_ReturnType
	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.

[UDPNM308] ⌈

Name:	`UdpNm_ConfigType`		
Type:	`Structure`		
Element:	`void`	`implementation specific`	This type shall contain the parameters of the container UdpNm_GlobalConfig and its sub containers.
Description:	--		

⌋()

8.2.2 UdpNm_PduPositionType

[UDPNM304] ⌈

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

⌋()

8.3 UdpNm Functions called by the Nm

8.3.1 UdpNm_Init

[UDPNM208] ⌈

Service name:	UdpNm_Init	

Syntax:	`void UdpNm_Init(`
	`    const UdpNm_ConfigType* UdpNmConfigPtr`
	`)`	

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

⌋()

[UDPNM209] ⌈ If a NULL pointer is passed as an argument to this function the default configuration shall be used. ⌋()

[UDPNM210] ⌈ If an error has to be indicated to the DET the value `0x00` shall be used as the instance id. ⌋()

8.3.2 UdpNm_PassiveStartUp

[UDPNM211] ⌈

Service name:	UdpNm_PassiveStartUp	

Syntax:	`Std_ReturnType UdpNm_PassiveStartUp(`
	`    const NetworkHandleType nmChannelHandle`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Passive startup of network management has failed
Description:	Passive startup of the AUTOSAR UdpNm. It triggers the transition from Bus-Sleep Mode to the Network Mode in Repeat Message State. Caveats: UdpNm is initialized correctly. Configuration: Mandatory Configuration: Mandatory	

⌋()

[UDPNM212] ⌈ This service has no effect if the current state is not equal to Bus-Sleep Mode. In that case `NM_E_NOT_EXECUTED` is returned. ⌋()

8.3.3 UdpNm_NetworkRequest

[UDPNM213] ⌈

Service name:	UdpNm_NetworkRequest	

Syntax:	`Std_ReturnType UdpNm_NetworkRequest(`
	`    const NetworkHandleType nmChannelHandle`
	`)`	

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

⌋()

[UDPNM255] ⌈ The function `UdpNm_NetworkRequest` shall change the Network state to 'requested'. ⌋()

8.3.4 UdpNm_NetworkRelease

[UDPNM214] ⌈

Service name:	UdpNm_NetworkRelease	

Syntax:	`Std_ReturnType UdpNm_NetworkRelease(`
	`    const NetworkHandleType nmChannelHandle`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Releasing of network has failed
Description:	Release the network, since ECU doesn't have to communicate on the bus. Network state shall be changed to 'released'. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UDPNM_PASSIVE_MODE_ENABLED is FALSE)	

⌋()

[UDPNM258] ⌈ The function `UdpNm_NetworkRelease` shall change the Network state to 'released'. ⌋()

[UDPNM294] ⌈ If the Network Management PDU transmission ability of UdpNm has been disabled by calling `UdpNm_DisableCommunication`, then the function `UdpNm_NetworkRelease` shall have not be executed and `NM_E_NOT_EXECUTED` shall be returned. ⌋()

8.3.5 UdpNm_DisableCommunication

[UDPNM215] ⌈

Service name:	UdpNm_DisableCommunication	

Syntax:	`Std_ReturnType UdpNm_DisableCommunication(`
	`    const NetworkHandleType nmChannelHandle`
	`)`	

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

⌋(BSW02512)

[UDPNM307] ⌈ If the module operates in passive mode (CANNM_PASSIVE_MODE_ENABLED) the service CanNm_DisableCommunication shall have no effects and shall directly return NM_E_NOT_EXECUTED. ⌋()

8.3.6 UdpNm_EnableCommunication

[UDPNM216] ⌈

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

⌋(BSW02512)

[UDPNM176] ⌈ The optional service `UdpNm_EnableCommunication` shall enable the NM PDU transmission ability if the NM PDU transmission ability is disabled. ⌋()

[UDPNM177] ⌈ The optional service UdpNm_EnableCommunication shall return `NM_E_NOT_EXECUTED` if the NM PDU transmission ability is already enabled when the service is called. ⌋()

[UDPNM305] ⌈ The service UdpNm_EnableCommunication shall return NM_E_NOT_EXECUTED, if the current mode is not Network Mode. ⌋()

[UDPNM306] ⌈ If the module operates in passive mode (UDPNM_PASSIVE_MODE_ENABLED is TRUE) the service UdpNm_EnableCommunication shall have no effects and shall directly return NM_E_NOT_EXECUTED. ⌋()

8.3.7 UdpNm_SetUserData

[UDPNM217] ⌈

Service name:	UdpNm_SetUserData	

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

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

⌋()

8.3.8 UdpNm_GetUserData

[UDPNM218] ⌈

Service name:	UdpNm_GetUserData	

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

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

⌋()

8.3.9 UdpNm_GetNodeIdentifier

[UDPNM219] ⌈

Service name:	UdpNm_GetNodeIdentifier	
Syntax:	`Std_ReturnType UdpNm_GetNodeIdentifier( const NetworkHandleType nmChannelHandle, uint8* const nmNodeIdPtr )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer where the source node identifier from the most recently received NM PDU shall be copied to.
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Getting of the source node identifier from the most recently received NM PDU has failed
Description:	Get node identifier from the most recently received NM PDU. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UDPNM_NODE_ID_ENABLED is TRUE).	

⌋()

[UDPNM132] ⌈ The optional service call `UdpNm_GetNodeIdentifier` shall provide the source node identifier contained in the most recently received NM packet. ⌋()

8.3.10 UdpNm_GetLocalNodeIdentifier

[UDPNM220] ⌈

Service name:	UdpNm_GetLocalNodeIdentifier	
Syntax:	`Std_ReturnType UdpNm_GetLocalNodeIdentifier( const NetworkHandleType nmChannelHandle, uint8* const nmNodeIdPtr )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer where node identifier of the local node shall be copied to.
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Getting of the node identifier of the local node has failed
Description:	Get node identifier configured for the local node. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UDPNM_NODE_ID_ENABLED is TRUE).	

⌋()

[UDPNM133] ⌈ The optional service call `UdpNm_GetLocalNodeIdentifier` shall provide the node identifier configured for the local host node. ⌋()

8.3.11 UdpNm_RepeatMessageRequest

[UDPNM221] ⌈

Service name:	UdpNm_RepeatMessageRequest	

Syntax:	`Std_ReturnType UdpNm_RepeatMessageRequest(`
	`    const NetworkHandleType nmChannelHandle`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Setting of Repeat Message Request Bit has failed
Description:	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. Configuration: Configuration of UdpNm_RepeatMessageRequest: Optional (Only available if UDPNM_NODE_DETECTION_ENABLED is TRUE).	

⌋()

8.3.12 UdpNm_GetPduData

[UDPNM309] ⌈

Service name:	UdpNm_GetPduData	

Syntax:	`Std_ReturnType UdpNm_GetPduData(`
	`    const NetworkHandleType nmChannelHandle,`
	`    uint8* const nmPduDataPtr`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmPduDataPtr	Pointer where NM PDU shall be copied to.
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Getting of NM PDU data has failed
Description:	Get the whole PDU data out of the most recently received NM message. Caveats: UdpNm is initialized correctly. Configuration: Optional (Only available if UDPNM_NODE_ID_ENABLED or UDPNM_NODE_DETECTION_ENABLED or UDPNM_USER_DATA_ENABLED is TRUE).	

⌋()

8.3.13 UdpNm_GetState

[UDPNM310] ⌈

Service name:	UdpNm_GetState	

Syntax:	`Std_ReturnType UdpNm_GetState(`
	`    const NetworkHandleType nmChannelHandle,`
	`    Nm_StateType* const nmStatePtr,`
	`    Nm_ModeType* const nmModePtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmStatePtr	Pointer where state of the network management shall be copied to.
	nmModePtr	Pointer where the mode of the network management shall be copied to.
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Getting of NM state has failed
Description:	Returns the state and the mode of the network management. Caveats: UdpNm is initialized correctly. Configuration: Mandatory	

⌋()

8.3.14 UdpNm_GetVersionInfo

[UDPNM224] ⌈

Service name:	UdpNm_GetVersionInfo	

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

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). Note: This function can be called even if UdpNm is not initialized. Configuration: Optional (only available if UDPNM_VERSION_INFO_API is TRUE).	

⌋()

[UDPNM318] ⌈ If DET is enabled for the UdpNm module, the function `UdpNm_GetVersionInfo` shall raise `UDPNM_E_NULL_POINTER`, if the argument versioninfo is a NULL pointer and return without any action. ⌋()

8.3.15 UdpNm_RequestBusSynchronization

[UDPNM226] ⌈

Service name:	UdpNm_RequestBusSynchronization	

Syntax:	`Std_ReturnType UdpNm_RequestBusSynchronization(`
	`    const NetworkHandleType nmChannelHandle`
	`)`	

Service ID[hex]:	0x14	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Requesting of bus synchronization has failed
Description:	Request bus synchronization. Caveats: UdpNm is initialized correctly. Configuration: Optional (only available if UDPNM_BUS_SYNCHRONIZATION_ENABLED is defined and UDPNM_PASSIVE_MODE_ENABLED is not defined).	

⌋()

[UDPNM130] ⌈ The service call `UdpNm_RequestBusSynchronization` shall trigger transmission of a single Network Management PDU if `UDPNM_PASSIVE_MODE_ENABLED` (configuration parameter) is `FALSE`. ⌋()

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

[UDPNM187] ⌈ 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 `NM_E_NOT_EXECUTED`. ⌋()

8.3.16 UdpNm_CheckRemoteSleepIndication

[UDPNM227] ⌈

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

⌋()

[UDPNM153] ⌈ 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_SetCoordBits

[UDPNM222] ⌈

Service name:	UdpNm_SetCoordBits	
Syntax:	`Std_ReturnType UdpNm_SetCoordBits( const NetworkHandleType nmChannelHandle, const uint8 nmCoordBits )`	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	nmCoordBits	2 bit value to set the NM coordinator ID in the control bit vector of each NM message (coding as depicted in Figure "Control Bit Vector".)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error E_NOT_OK: Setting the coordinator ID bits has failed
Description:	Sets the NM coordinator ID in the control bit vector of each NM message.	

⌋()

8.3.18 UdpNm_SetSleepReadyBit

[UDPNM324] ⌈

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

8.4 UdpNm functions called by the SoAd

8.4.1 UdpNm_SoAdIfTxConfirmation

[UDPNM228] ⌈

Service name:	UdpNm_SoAdIfTxConfirmation	
Syntax:	`void UdpNm_SoAdIfTxConfirmation( PduIdType UdpNmTxPduId )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not within the same channel)	
Parameters (in):	UdpNmTxPduId	Identification of the network through PDU-ID
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service confirms a previous successfully processed transmit request. Caveats: - The call context is either on interrupt level (interrupt mode) or on task level (polling mode). - The UdpNm module is initialized correctly.	

⌋()

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

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

[UDPNM229] ⌈ The callback function `UdpNm_SoAdIfTxConfirmation` shall inform the DET (if enabled), if the function call has failed because of the following reasons:

- Invalid channel handle (`UDPNM_E_INVALID_CHANNEL`)
- UdpNm was not initialized (`UDPNM_E_NO_INIT`) ⌋()

[UDPNM230] ⌈ If an error has to be indicated to the DET, the callback function `UdpNm_SoAdIfTxConfirmation` shall use the value of UdpNm channel handle as the instance id. ⌋()

[UDPNM316] ⌈ If `UdpNmComUserDataSupport` is enabled the UdpNm shall call `PduR_UdpNmTxConfirmation` within the message transmission confirmation function `UdpNm_SoAdIfTxConfirmation` called by the SoAd. ⌋()

8.4.2 UdpNm_SoAdIfRxIndication

[UDPNM231] ⌈

Service name:	UdpNm_SoAdIfRxIndication	

Syntax:	`void UdpNm_SoAdIfRxIndication(`
	`    PduIdType udpNmRxPduId,`
	`    const uint8* udpSduPtr`
	`)`	

Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not within the same channel)	
Parameters (in):	udpNmRxPduId	Identification of the network through PDU-ID
	udpSduPtr	Pointer to received SDU
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service indicates a successful reception of a received NM message to the UdpNm after passing all filters and validation checks. Caveats: - Until this service returns the SoAd will not access udpSduPtr. The udpSduPtr is only valid and can be used by upper layers until the indication returns. SoAd guarantees that the number of configured bytes for this udpNmRxPduId is valid. The call context is either on interrupt level (interrupt mode) or on task level (polling mode). - The UdpNm module is initialized correctly.	

⌋()

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

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

[UDPNM232] ⌈ The callback function UdpNm_SoAdIfRxIndication shall inform the DET (if enabled), if function call has failed because of the following reasons:

- Invalid channel handle (`UDPNM_E_INVALID_CHANNEL`)
- UdpNm was not initialized (`UDPNM_E_NO_INIT`)
- udpSduPtr equals NULL_PTR (`UDPNM_E_NULL_POINTER`)
- Invalid PDU ID (`UDPNM_E_INVALID_PDUID`) ⌋()

[UDPNM233] ⌈ If an error has to be indicated to the DET, the callback function `UdpNm_SoAdIfRxIndication` shall use the value of UdpNm channel handle as the instance id. ⌋()

8.5 UdpNm functions called by the PDU-Router

8.5.1 UdpNm_Transmit

[UDPNM313] ⌈

Service name:	UdpNm_Transmit	

Syntax:	`Std_ReturnType UdpNm_Transmit(`
	`    PduIdType UdpNmSrcPduId,`
	`    const PduInfoType* UdpNmSrcPduInfoPtr`
	`)`	

Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	UdpNmSrcPduId	This parameter contains a unique identifier referencing to the PDU Routing Table and therby specifiying the socket to be used for tranmission of the data.
	UdpNmSrcPduInfoPtr	A pointer to a structure with socket related data: data length and pointer to a data buffer.

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted, e.g. due to a still ongoing transmission in the corresponding socket or the to be transmitted message is too long.
Description:	UdpNm_Transmit is implemented as an empty function and shall always return E_OK. The function UdpNm_Transmit is only available if the configuration switch UdpNmComUserDataSupport is enabled.	

⌋()

[UDPNM315] ⌈ If `UdpNmComUserDataSupport` is enabled the UdpNm implementation shall provide an API `UdpNm_Transmit`. This API shall never be called by PduR as the UdpNm will always query the data by means of `PduR_UdpNmTriggerTransmit`. `UdpNm_Transmit` is an empty function returning `E_NOT_OK` at any time. This requirement is relevant to avoid linker errors as PduR expects this API to be provided. ⌋()

8.6 Scheduled Functions

8.6.1 UdpNm_MainFunction_<Instance Id>

[UDPNM234] ⌈

Service name:	UdpNm_MainFunction<Instance_Id>

Syntax:	`void UdpNm_MainFunction<Instance_Id>(`
	`    void`
	`)`

Service ID[hex]:	0x13
Timing:	FIXED_CYCLIC
Description:	Main function of the UdpNm which processes the algorithm describes in that document. E.g.: UdpNm_MainFunction_0() represents the UdpNm instance for the UDP channel 0 UdpNm_MainFunction_1() represents the UdpNm instance for the UDP channel 1 ... Inform the DET (if enabled) if function call has failed because of the following reasons: UdpNm was not initialized (UDPNM_E_NO_INIT) If an error has to be indicated to the DET the <Instance Id> shall be used as the instance id. Caveats: UdpNm is initialized correctly, i.e. the function shall be robust if one or more channels are not initialized Configuration: Mandatory

⌋()

8.7 Expected Interfaces

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

8.7.1 Mandatory Interfaces

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

API function	***Description***
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.
Nm_BusSleepMode	Notification that the network management has entered Bus-Sleep Mode.
Nm_NetworkMode	Notification that the network management has entered Network Mode.
Nm_NetworkStartIndication	Notification that a NM-message has been received in the Bus-Sleep Mode, what indicates that some nodes in the network have already entered the Network Mode.
Nm_PrepareBusSleepMode	Notification that the network management has entered Prepare Bus-Sleep Mode.
SoAdIf_Transmit	This service initiates a request for transmission of the L-PDU specified by the SoAdSrcPduId. The corresponding socket has to be resolved by the SoAdSrcPduId. This call is used to mimic the call to an IF in AUTOSAR. Development errors: Invalid values of SoAdSrcPduId or SoAdSrcPduInfoPtr will be reported to the development error tracer (SOAD_E_INVALID_TXPDUID or SOAD_E_PARAM_POINTER).

8.7.2 Optional Interfaces

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

API function	Description
Det_ReportError	Service to report development errors.
Nm_CoordReadyToSleepIndication	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set
Nm_PduRxIndication	Notification that a NM message has been received.
Nm_RemoteSleepCancellation	Notification that the network management has detected that not all other nodes on the network are longer ready to enter Bus-Sleep Mode.
Nm_RemoteSleepIndication	Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode.
Nm_RepeatMessageIndication	Service to indicate that an NM message with set Repeat Message Request Bit has been received.
Nm_StateChangeNotification	Notification that the state of the lower layer <BusNm> has changed.
Nm_TxTimeoutException	Service to indicate that an attempt to send an NM message failed.
PduR_UdpNmTriggerTransmit	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.
PduR_UdpNmTxConfirmation	The lower layer communication module confirms the transmission of an I-PDU.

8.7.2.1 Functions of PDU Router

[UDPNM317]⌈ 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 each time before it requests the transmission of the corresponding NM message. ⌋()

8.7.3 Configurable interfaces

Not applicable

8.7.4 Job End Notification

Not applicable

8.8 Parameter check

[UDPNM196]⌈ 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. Exception: The NULL Pointer check of input parameters shall not be done for `UdpNm_Init`. ⌋()

[UDPNM197]⌈ 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. ⌋()

[UDPNM198]⌈ 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. ⌋()

[UDPNM199]⌈ 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. ⌋()

8.9 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.

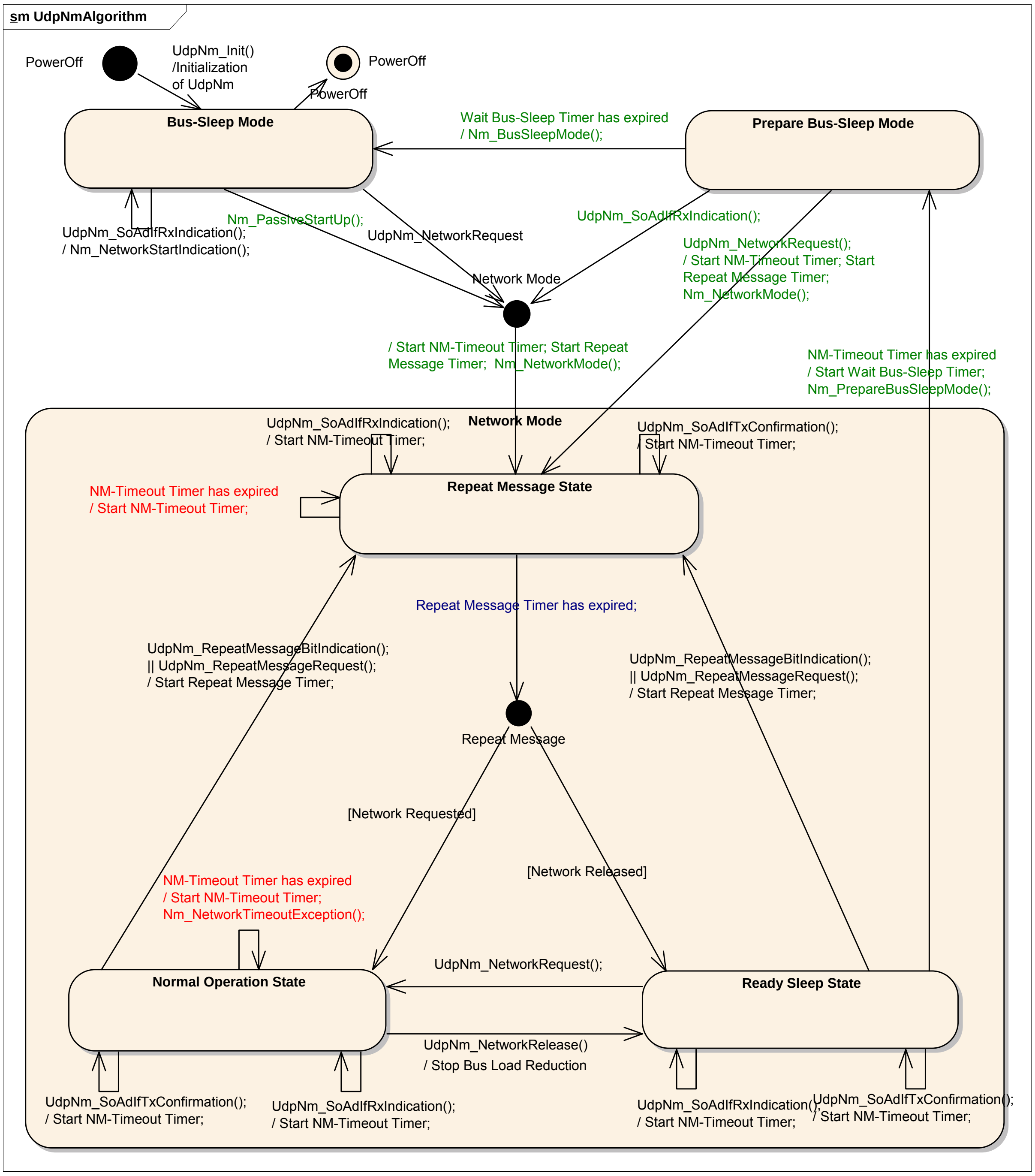

sm UdpNmAlgorithm
PowerOff
UdpNm_Init() /Initialization of UdpNm
PowerOff
PowerOff
Bus-Sleep Mode
Wait Bus-Sleep Timer has expired / Nm_BusSleepMode();
Prepare Bus-Sleep Mode
UdpNm_SoAdIfRxIndication(); / Nm_NetworkStartIndication();
Nm_PassiveStartUp();
UdpNm_NetworkRequest
UdpNm_SoAdIfRxIndication();
UdpNm_NetworkRequest(); / Start NM-Timeout Timer; Start Repeat Message Timer; Nm_NetworkMode();
Network Mode
/ Start NM-Timeout Timer; Start Repeat Message Timer; Nm_NetworkMode();
NM-Timeout Timer has expired / Start Wait Bus-Sleep Timer; Nm_PrepareBusSleepMode();
Network Mode
UdpNm_SoAdIfRxIndication(); / Start NM-Timeout Timer;
UdpNm_SoAdIfTxConfirmation(); / Start NM-Timeout Timer;
NM-Timeout Timer has expired / Start NM-Timeout Timer;
Repeat Message State
Repeat Message Timer has expired;
UdpNm_RepeatMessageBitIndication(); || UdpNm_RepeatMessageRequest(); / Start Repeat Message Timer;
UdpNm_RepeatMessageBitIndication(); || UdpNm_RepeatMessageRequest(); / Start Repeat Message Timer;
Repeat Message
[Network Requested]
[Network Released]
NM-Timeout Timer has expired / Start NM-Timeout Timer; Nm_NetworkTimeoutException();
UdpNm_NetworkRequest();
Normal Operation State
Ready Sleep State
UdpNm_NetworkRelease() / Stop Bus Load Reduction
UdpNm_SoAdIfTxConfirmation(); / Start NM-Timeout Timer;
UdpNm_SoAdIfRxIndication(); / Start NM-Timeout Timer;
UdpNm_SoAdIfRxIndication(); / Start NM-Timeout Timer;
UdpNm_SoAdIfTxConfirmation(); / Start NM-Timeout Timer;

Figure 6: State chart diagram.

9 Sequence diagrams and Transition Tables

9.1 UdpNm Transmission

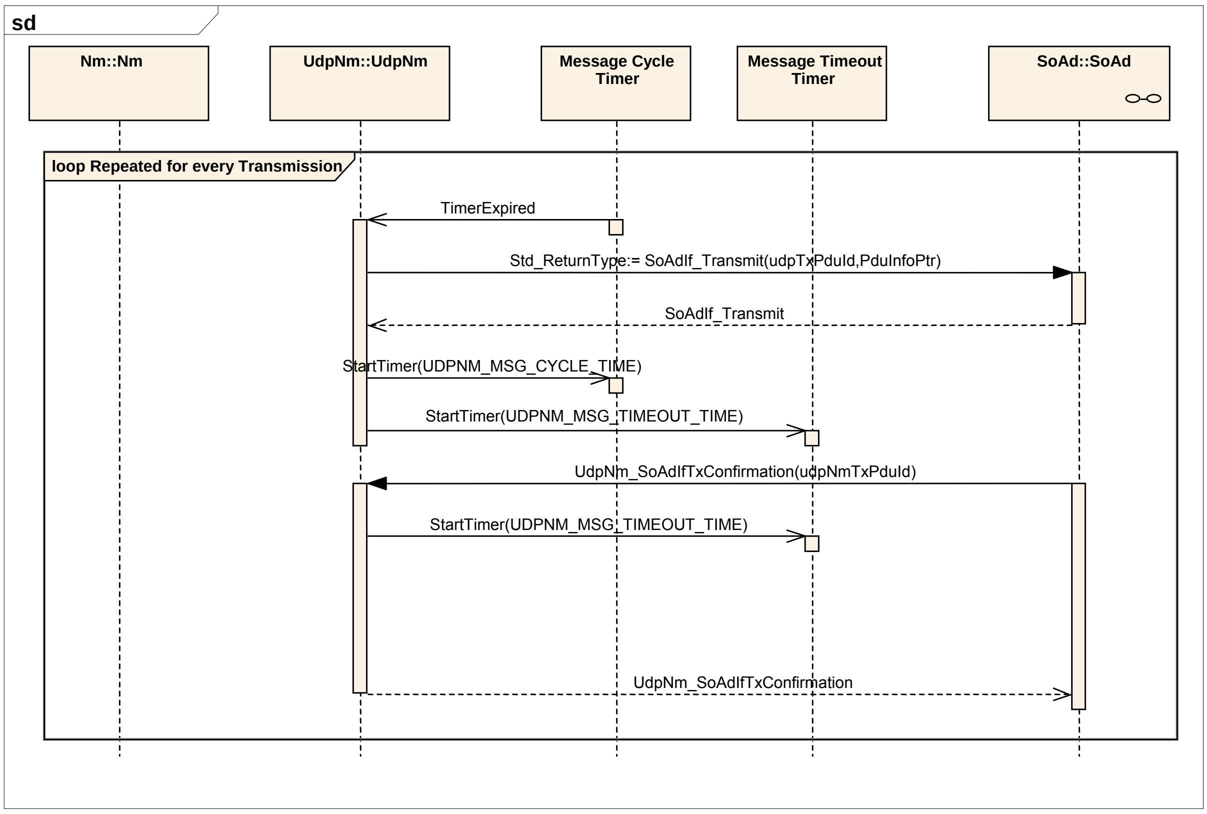

sd
Nm::Nm
UdpNm::UdpNm
Message Cycle Timer
Message Timeout Timer
SoAd::SoAd
loop Repeated for every Transmission
TimerExpired
Std_ReturnType:= SoAdIf_Transmit(udpTxPduId,PduInfoPtr)
SoAdIf_Transmit
StartTimer(UDPNM_MSG_CYCLE_TIME)
StartTimer(UDPNM_MSG_TIMEOUT_TIME)
UdpNm_SoAdIfTxConfirmation(udpNmTxPduId)
StartTimer(UDPNM_MSG_TIMEOUT_TIME)
UdpNm_SoAdIfTxConfirmation

Figure 7: Sequence diagram – PDU transmission.

9.2 UdpNm Reception

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

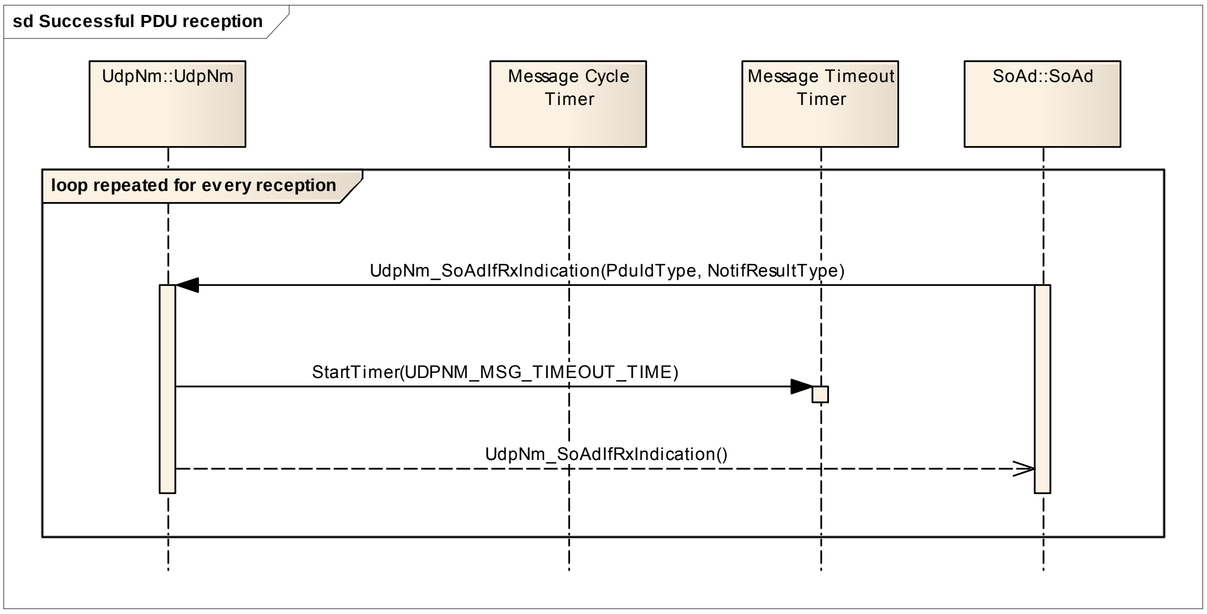

sd Successful PDU reception
UdpNm::UdpNm
Message Cycle Timer
Message Timeout Timer
SoAd::SoAd
loop repeated for every reception
UdpNm_SoAdIfRxIndication(PduIdType, NotifResultType)
StartTimer(UDPNM_MSG_TIMEOUT_TIME)
UdpNm_SoAdIfRxIndication()

Figure 8: Sequence diagram – PDU transmission.

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

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [7]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration meta model in detail.

The following is just a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.
The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- all configuration parameters are kept in containers.

- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class Pre-compile time or not

Label	***Description***
X	The configuration parameter shall be of configuration class Pre-compile time.
--	The configuration parameter shall never be of configuration class Pre-compile time.

Link time - specifies whether the configuration parameter shall be of configuration class Link time or not

Label	***Description***
X	The configuration parameter shall be of configuration class Link time.
--	The configuration parameter shall never be of configuration class Link time.

Post Build - specifies whether the configuration parameter shall be of configuration class Post Build or not

Label	***Description***
X	The configuration parameter shall be of configuration class Post Build and no specific implementation is required.
L	Loadable - the configuration parameter shall be of configuration class Post Build and only one configuration parameter set resides in the ECU.
M	Multiple - the configuration parameter shall be of configuration class Post Build and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class Post Build.

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.

[UDPNM026] ⌈ All configuration items shall be located outside the kernel of the module. ⌋()

[UDPNM201] ⌈ The Global Scope specifies configuration parameter that shall be defined in the module's configuration header file `UdpNm_Cfg.h`. ⌋()

[UDPNM202] ⌈ The container `UdpNm_ChannelConfig` specifies configuration parameter that shall be located in a data structure of type `UdpNm_ConfigType`. ⌋()

[UDPNM203] ⌈ Runtime configurable parameters listed in container `UdpNm_ChannelConfig` shall be configurable for each NM-cluster separately. ⌋()

10.2.1 Variants

Variant 1: All configuration parameters shall be configurable at pre-compile time. Use case: Source code optimization.
Variant 2: All configuration parameters of the container UdpNm_GlobalConfig related to enable or disable an optional feature shall be configurable at pre-compile time; the remaining configuration parameters shall be configurable at link time.

Use case: Object code.
Variant 3: The parameters contained in UdpNm_ChannelConfig are configurable at post-build time. The parameters contained in UdpNm_GlobalConfig are configurable at pre-compile time

Use case: ECU configuration can be flashed (L) and selected during initialization phase (M).
Note:
The possibility to select a configuration (post-build time type L) is explicitly mentioned for Variant 3 only, but from a technical perspective it is also possible to provide this configuration variant for variant 1 and 2.

10.2.2 UdpNm

Module Name	*UdpNm*
Module Description	--

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

10.2.3 UdpNmGlobalConfig

SWS Item	**UDPNM001_Conf :**
Container Name	UdpNmGlobalConfig{UdpNm_GlobalConfig} [Multi Config Container]
Description	This container contains all global configuration parameters of UDP NM configured from the NM Module perspective.
Configuration Parameters	

SWS Item	**UDPNM006_Conf :**		
Name	UdpNmBusSynchronizationEnabled {UDPNM_BUS_SYNCHRONIZATION_ENABLED}		
Description	Pre-processor switch for enabling bus synchronization support. This feature is required for gateway nodes only. It must not be defined if UDPNM_PASSIVE_MODE_ENABLED is defined. This parameter shall be derived from NM_BUS_SYNCHRONIZATION_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM013_Conf :**		
Name	UdpNmComControl_Enabled {UDPNM_COM_CONTROL_ENABLED}		
Description	Pre-processor switch for enabling the Communication Control support. This parameter shall be derived from NM_COM_CONTROL_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM055_Conf :**		
Name	UdpNmComUserDataSupport {UDPNM_COM_USER_DATA_SUPPORT}		
Description	Enable/disable the user data support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**UDPNM040_Conf :**		
Name	UdpNmCoordinatorEnabled {UDPNM_COORDINATOR_ENABLED}		
Description	Enable/disable the NM Coordination algorithm to being able to initiate the synchronization algorithm. TRUE: Option is enabled FALSE: The parameter shall be FALSE by default and shall only be allowed to be TRUE if the parameter UDPNM_REMOTE_SLEEP_IND_ENABLED is TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	--	
	Link time	X	All Variants
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM041_Conf :**		
Name	UdpNmCoordinatorId {UDPNM_COORDINATOR_ID}		
Description	Set the NM coordination ID for this gateway. 0x00: passive coordinator only 0x01 - 0x03: coordinator priority Only valid, if UDPNM_COORDINATOR_ENABLED is TRUE.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 3		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM059_Conf :**		
Name	UdpNmCoordinatorSyncSupport {UDPNM_COORDINATOR_SYNC_SUPPORT}		
Description	Enables/disables the coordinator synchronisation support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM002_Conf :**		
Name	UdpNmDevErrorDetect {UDPNM_DEV_ERROR_DETECT}		
Description	Pre-processor switch for enabling development error detection support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM009_Conf :**		
Name	UdpNmImmediateRestartEnabled {UDPNM_IMMEDIATE_RESTART_ENABLED}		
Description	Pre-processor switch for enabling the asynchronous transmission of a NM PACKET upon bus-communication request in Prepare-Bus-Sleep mode. Must not be defined if UDPNM_PASSIVE_MODE_ENABLED is defined.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM007_Conf :**		
Name	UdpNmNodeDetectionEnabled {UDPNM_NODE_DETECTION_ENABLED}		
Description	Pre-processor switch for enabling the node detection support. This parameter shall be derived from NM_NODE_DETECTION_ENABLED. This parameter shall only be enabled if UDPNM_NODE_ID_ENABLED is defined. If(UdpNmPduCbvPosition != UDPNM_PDU_OFF) then Equal(NmNodeDetectionEnabled) else Equal(False).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM008_Conf :**		
Name	UdpNmNodeIdEnabled {UDPNM_NODE_ID_ENABLED}		
Description	Pre-processor switch for enabling the source node identifier. This parameter shall be derived from NM_NODE_ID_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM014_Conf :**		
Name	UdpNmNumberOfChannels {UDPNM_NUMBER_OF_CHANNELS}		
Description	Number of NM channels allowed within one ECU.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM010_Conf :**		
Name	UdpNmPassiveModeEnabled {UDPNM_PASSIVE_MODE_ENABLED}		
Description	Pre-processor switch for enabling support of the Passive Mode. This parameter shall be derived from NM_PASSIVE_MODE_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM011_Conf :**		
Name	UdpNmPduRxIndicationEnabled {UDPNM_PDU_RX_INDICATION_ENABLED}		
Description	Pre-processor switch for enabling the PDU Rx Indication. This parameter shall be derived from NM_PDU_RX_INDICATION_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM005_Conf :**		
Name	UdpNmRemoteSleepIndEnabled {UDPNM_REMOTE_SLEEP_IND_ENABLED}		
Description	Pre-processor switch for enabling remote sleep indication support. This feature is required for gateway nodes only. It must not be defined if UDPNM_PASSIVE_MODE_ENABLED is defined. This parameter shall be derived from NM_REMOTE_SLEEP_IND_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM015_Conf :**		
Name	UdpNmRepeatMsgIndEnabled {UDPNM_REPEAT_MSG_IND_ENABLED}		
Description	Enable/disable the notification that a RepeatMessageRequest bit has been received. This parameter shall be derived from NM_REPEAT_MSG_IND_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

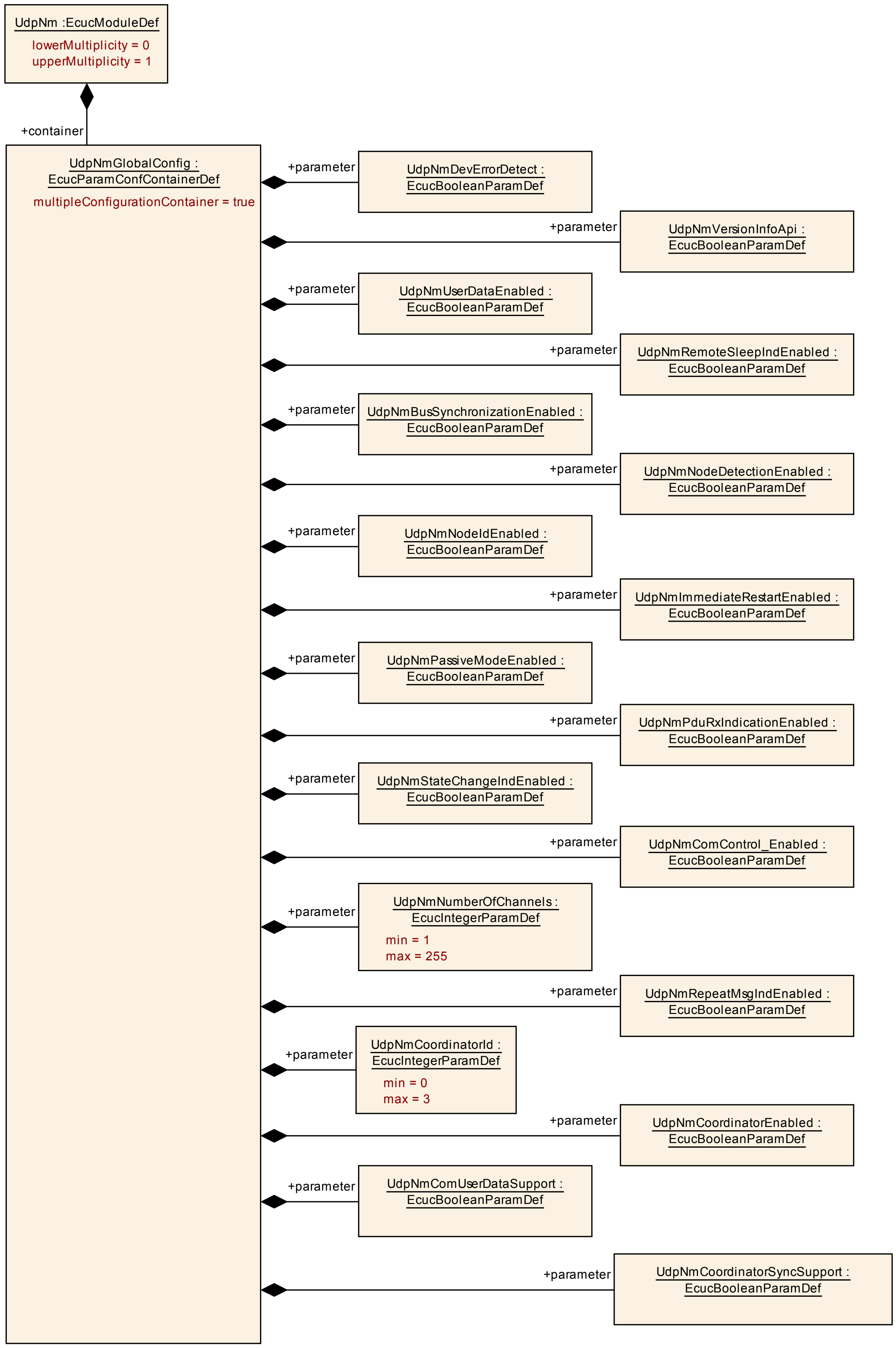
SWS Item	**UDPNM012_Conf :**		
Name	UdpNmStateChangeIndEnabled {UDPNM_STATE_CHANGE_IND_ENABLED}		
Description	Pre-processor switch for enabling the UDP NM state change notification. This parameter shall be derived from NM_STATE_CHANGE_ID_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM004_Conf :**		
Name	UdpNmUserDataEnabled {UDPNM_USER_DATA_ENABLED}		
Description	Pre-processor switch for enabling user data support. This parameter shall be derived from NM_USER_DATA_ENABLED.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM003_Conf :**		
Name	UdpNmVersionInfoApi {UDPNM_VERSION_INFO_API}		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
UdpNmChannelConfig	1..*	This container contains the channel-specific configuration parameters of the UdpNm.
UdpNmDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.


UdpNm :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
UdpNmGlobalConfig : EcucParamConfContainerDef
multipleConfigurationContainer = true
+parameter
UdpNmDevErrorDetect : EcucBooleanParamDef
UdpNmVersionInfoApi : EcucBooleanParamDef
UdpNmUserDataEnabled : EcucBooleanParamDef
UdpNmRemoteSleepIndEnabled : EcucBooleanParamDef
UdpNmBusSynchronizationEnabled : EcucBooleanParamDef
UdpNmNodeDetectionEnabled : EcucBooleanParamDef
UdpNmNodeIdEnabled : EcucBooleanParamDef
UdpNmImmediateRestartEnabled : EcucBooleanParamDef
UdpNmPassiveModeEnabled : EcucBooleanParamDef
UdpNmPduRxIndicationEnabled : EcucBooleanParamDef
UdpNmStateChangeIndEnabled : EcucBooleanParamDef
UdpNmComControl_Enabled : EcucBooleanParamDef
UdpNmNumberOfChannels : EcucIntegerParamDef
min = 1
max = 255
UdpNmRepeatMsgIndEnabled : EcucBooleanParamDef
UdpNmCoordinatorId : EcucIntegerParamDef
min = 0
max = 3
UdpNmCoordinatorEnabled : EcucBooleanParamDef
UdpNmComUserDataSupport : EcucBooleanParamDef
UdpNmCoordinatorSyncSupport : EcucBooleanParamDef

10.2.4 UdpNmChannelConfig

SWS Item	**UDPNM017_Conf :**
Container Name	UdpNmChannelConfig{UdpNm_ChannelConfig}
Description	This container contains the channel-specific configuration parameters of the UdpNm.
Configuration Parameters	

SWS Item	**UDPNM031_Conf :**		
Name	UdpNmNodeId {UDPNM_NODE_ID}		
Description	Node identifier of local node. This parameter is only valid if UDPNM_PASSIVE_MODE_ENABLED is set to OFF and UDPNM_NODE_DETECTION_ENABLED is set to ON.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM026_Conf :**		
Name	UdpNmPduCbvPosition {UDPNM_PDU_CBV_POSITION}		
Description	Defines the position of the control bit vector within the NM PACKET. The value of the parameter represents the location of the control bit vector in the NM PACKET (UDPNM_PDU_BYTE_0 means byte 0, UDPNM_PDU_BYTE_1 means byte 1, UDPNM_PDU_OFF means the control bit vector is not part of the NM PACKET) See also UDPNM_PDU_NID_POSITION if (UDPNM_PDU_CBV_POSITION != UDPNM_PDU_OFF && UDPNM_PDU_NID_POSITION != UDPNM_PDU_OFF) then UDPNM_PDU_CBV_POSITION != UDPNM_PDU_NID_POSITION if (UDPNM_PDU_CBV_POSITION != UDPNM_PDU_OFF && UDPNM_PDU_NID_POSITION == UDPNM_PDU_OFF) then UDPNM_PDU_CBV_POSITION = UDPNM_PDU_BYTE0		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	UDPNM_PDU_BYTE_0	--	
	UDPNM_PDU_BYTE_1	--	
	UDPNM_PDU_OFF	--	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM024_Conf :**		
Name	UdpNmPduLength {UDPNM_PDU_LENGTH}		
Description	Defines the length of the NM PACKET in bytes. Valid values are within the range 0 ≤ UDPNM_PDU_LENGTH ≤ 8.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 8		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM025_Conf :**		
Name	UdpNmPduNidPosition {UDPNM_PDU_NID_POSITION}		
Description	Defines the position of the source node identifier within the NM PACKET. ImplementationType: UdpNm_PduPositionType The value of the parameter represents the location of the source node identifier in the NM PACKET (UDPNM_PDU_BYTE_0 means byte 0, UDPNM_PDU_BYTE_1 means byte 1, UDPNM_PDU_OFF means source node identifier is not part of the NM PACKET) See also UDPNM_PDU_CBV_POSITION if (UDPNM_PDU_NID_POSITION != UDPNM_PDU_OFF && UDPNM_PDU_CBV_POSITION != UDPNM_PDU_OFF) then UDPNM_PDU_NID_POSITION != UDPNM_PDU_CBV_POSITION if (UDPNM_PDU_NID_POSITION != UDPNM_PDU_OFF && UDPNM_PDU_CBV_POSITION == UDPNM_PDU_OFF) then UDPNM_PDU_IND_POSITION = UDPNM_PDU_BYTE0		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	UDPNM_PDU_BYTE_0	Byte 0 is used.	
	UDPNM_PDU_BYTE_1	Byte 1 is used.	
	UDPNM_PDU_OFF	Node Identification is not used.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM027_Conf :**		
Name	UdpNmUserDataLength {UDPNM_USER_DATA_LENGTH}		
Description	Defines the length of the user data contained in the NM PACKET. The difference between UDPNM_PDU_LENGTH and applied standardized bytes (source node identifier and control bit vector) within the NM PACKET. Valid values are 0x00..0x08.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 8		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM032_Conf :**		
Name	UpdNmMainFunctionPeriod {UDPNM_MAIN_FUNCTION_PERIOD}		
Description	Call cycle of UdpNm_MainFunction_x for the respective instance in [s].		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.001 .. 0.255`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM029_Conf :**		
Name	UpdNmMsgCycleOffset {UDPNM_MSG_CYCLE_OFFSET}		
Description	Time offset in the periodic transmission node. It determines the start delay of the transmission. < UDPNM_MSG_CYCLE_TIME This parameter is only valid if UDPNM_PASSIVE_MODE_ENABLED is disabled.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency			

SWS Item	**UDPNM028_Conf :**		
Name	UpdNmMsgCycleTime {UDPNM_MSG_CYCLE_TIME}		
Description	Period of a NM-message. It determines the periodic rate in the "periodic transmission mode with bus load reduction" and is the basis for transmit scheduling in the "periodic transmission mode without bus load reduction". NM_TIMEOUT_TIME = n * UDPNM_MSG_CYCLE_TIME This parameter is only valid if UDPNM_PASSIVE_MODE_ENABLED is disabled.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.001 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM030_Conf :**		
Name	UpdNmMsgTimeoutTime {UDPNM_MSG_TIMEOUT_TIME}		
Description	Transmission Timout of NM-message. If there is no transmission confirmation by the UDP Interface within this timeout, the UDPNM module shall gibe an error notification. This parameter is only valid if UDPNM_PASSIVE_MODE_ENABLED is disabled. UDPNM_MSG_TIMEOUT_TIME should be a multiple of UDPNM_MSG_CYCLE_TIME.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.001 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM023_Conf :**		
Name	UpdNmRemoteSleepIndTime {UDPNM_REMOTE_SLEEP_IND_TIME}		
Description	Timeout for Remote Sleep Indication. It defines the time in [s] how long it shall take to recognize that all other nodes are ready to sleep. Typically it should be equal to: n * UDPNM_MSG_CYCLE_TIME, where n denotes the number of NM packets that are normally sent before Remote Sleep Indication is detected. The value of n decremented by one determines the amount of lost NM packets that can be tolerated by the Remote Sleep Indication procedure.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.001 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM022_Conf :**		
Name	UpdNmRepeatMessageTime {UDPNM_REPEAT_MESSAGE_TIME}		
Description	Timeout for Repeat Message State. It defines the time in [s] how long the NM shall stay in the Repeat Message State. Typically it should be equal to: n * UDPNM_MSG_CYCLE_TIME, where n denotes the number of NM packets that are normally sent in the Repeat Message State. The value of n decremented by one determines the amount of lost NM packets that can be tolerated by the node detection procedure. The value 0 denotes that no Repeat Message State is configured. It means that Repeat Message State is transient what implicates that it is left immediately after entrance and in result no start-up stability is guaranteed and no node detection procedure is possible.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

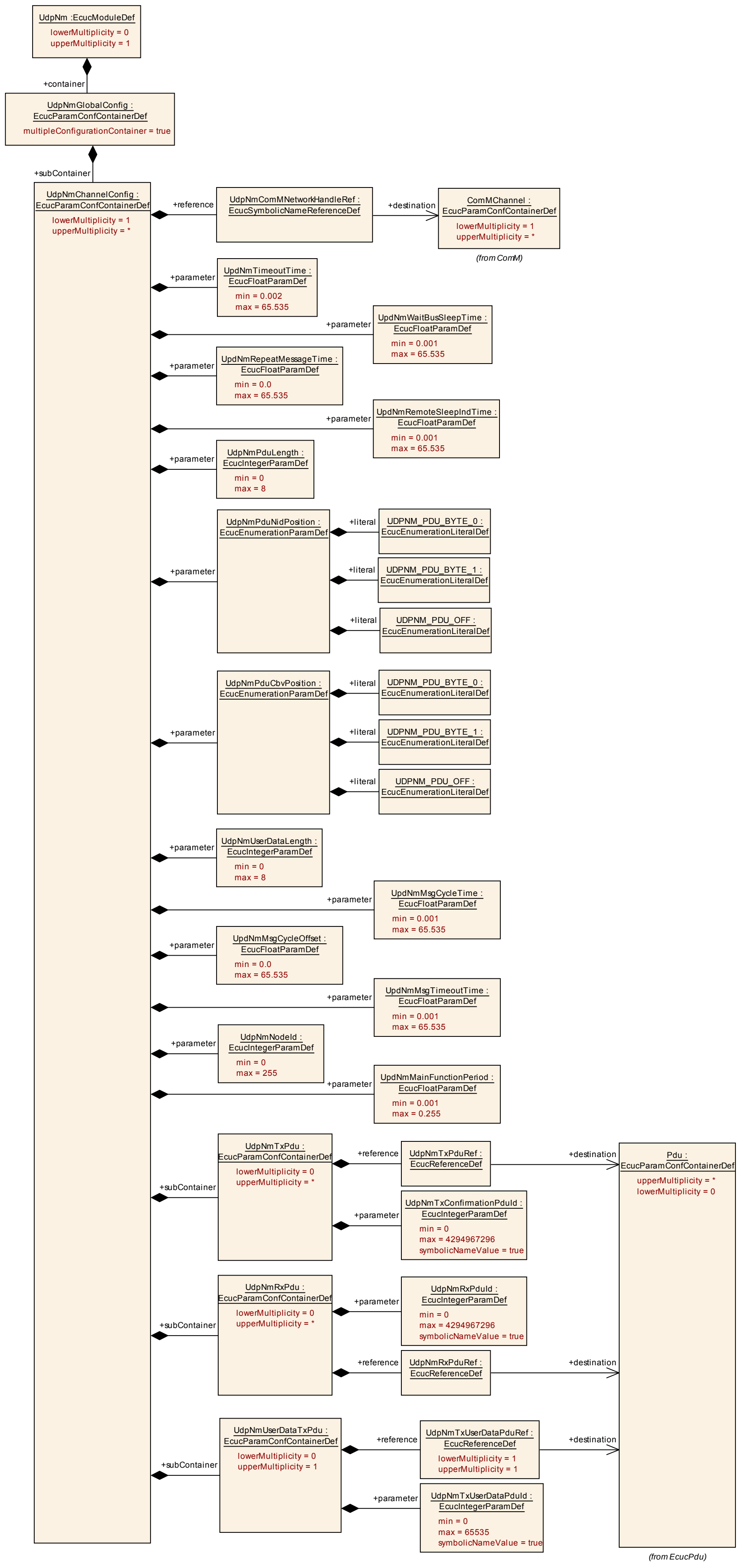
SWS Item	**UDPNM020_Conf :**		
Name	UpdNmTimeoutTime {UDPNM_TIMEOUT_TIME}		
Description	Network Timeout for NM packets. It denotes the time in [s] how long the NM shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place. It shall be equal for all nodes in the cluster. It shall be greater than UDPNM_MSG_CYCLE_TIME. Typically, it should be equal to: x * UDPNM_MSG_CYCLE_TIME, where n denotes the number of NM PACKET cycle times in the Ready Sleep State before transition into the Bus-Sleep Mode is initiated. The value of n decremented by one determines the amount of lost NM packets that can be tolerated by the coordination algorithm.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.002 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency	

SWS Item	**UDPNM021_Conf :**		
Name	UpdNmWaitBusSleepTime {UDPNM_WAIT_BUS_SLEEP_TIME}		
Description	Timeout for bus calm down phase. It denotes the time in [s] how long the NM shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place. It shall be equal for all nodes in the cluster. It shall be long enough to empty all Tx-buffer empty.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.001 .. 65.535`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM018_Conf :**		
Name	UdpNmComMNetworkHandleRef {UDPNM_CHANNEL_ID}		
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
UdpNmRxPdu	0..*	This container describes the UdpNm RX PDU's.
UdpNmTxPdu	0..*	This container describes the UdpNm TX PDU's.
UdpNmUserDataTxPdu	0..1	This optional container is used to configure the UserNm PDU. This container is only available if UdpNmComUserDataSupport is enabled.


UdpNm :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
UdpNmGlobalConfig : EcucParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
UdpNmChannelConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
UdpNmComMNetworkHandleRef : EcucSymbolicNameReferenceDef
+destination
ComMChannel : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from ComM)
+parameter
UpdNmTimeoutTime : EcucFloatParamDef
min = 0.002
max = 65.535
UpdNmWaitBusSleepTime : EcucFloatParamDef
min = 0.001
max = 65.535
UpdNmRepeatMessageTime : EcucFloatParamDef
min = 0.0
max = 65.535
UpdNmRemoteSleepIndTime : EcucFloatParamDef
min = 0.001
max = 65.535
UdpNmPduLength : EcucIntegerParamDef
min = 0
max = 8
UdpNmPduNidPosition : EcucEnumerationParamDef
+literal
UDPNM_PDU_BYTE_0 : EcucEnumerationLiteralDef
UDPNM_PDU_BYTE_1 : EcucEnumerationLiteralDef
UDPNM_PDU_OFF : EcucEnumerationLiteralDef
UdpNmPduCbvPosition : EcucEnumerationParamDef
UDPNM_PDU_BYTE_0 : EcucEnumerationLiteralDef
UDPNM_PDU_BYTE_1 : EcucEnumerationLiteralDef
UDPNM_PDU_OFF : EcucEnumerationLiteralDef
UdpNmUserDataLength : EcucIntegerParamDef
min = 0
max = 8
UpdNmMsgCycleTime : EcucFloatParamDef
min = 0.001
max = 65.535
UpdNmMsgCycleOffset : EcucFloatParamDef
min = 0.0
max = 65.535
UpdNmMsgTimeoutTime : EcucFloatParamDef
min = 0.001
max = 65.535
UdpNmNodeId : EcucIntegerParamDef
min = 0
max = 255
UpdNmMainFunctionPeriod : EcucFloatParamDef
min = 0.001
max = 0.255
UdpNmTxPdu : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
UdpNmTxPduRef : EcucReferenceDef
UdpNmTxConfirmationPduId : EcucIntegerParamDef
min = 0
max = 4294967296
symbolicNameValue = true
Pdu : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
(from EcucPdu)
UdpNmRxPdu : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
UdpNmRxPduId : EcucIntegerParamDef
min = 0
max = 4294967296
symbolicNameValue = true
UdpNmRxPduRef : EcucReferenceDef
UdpNmUserDataTxPdu : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
UdpNmTxUserDataPduRef : EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
UdpNmTxUserDataPduId : EcucIntegerParamDef
min = 0
max = 65535
symbolicNameValue = true

10.2.5 UdpNmRxPdu

SWS Item	**UDPNM038_Conf :**
Container Name	UdpNmRxPdu
Description	This container describes the UdpNm RX PDU's.
Configuration Parameters	

SWS Item	**UDPNM043_Conf :**		
Name	UdpNmRxPduId		
Description	ID of the RxPdu that will be used by a RxIndication of the lower layer.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967296		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM039_Conf :**		
Name	UdpNmRxPduRef		
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the UdpNm module to derive the PDU Id.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.6 UdpNmTxPdu

SWS Item	**UDPNM036_Conf :**
Container Name	UdpNmTxPdu
Description	This container describes the UdpNm TX PDU's.
Configuration Parameters	

SWS Item	**UDPNM042_Conf :**		
Name	UdpNmTxConfirmationPduId		
Description	Id of the TxPdu that will be used by a TxConfirmation from the lower layer.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967296		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM037_Conf :**		
Name	UdpNmTxPduRef		
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the UdpNm module to derive the PDU Id.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.7 UdpNmUserDataTxPdu

SWS Item	**UDPNM056_Conf :**
Container Name	UdpNmUserDataTxPdu
Description	This optional container is used to configure the UserNm PDU. This container is only available if UdpNmComUserDataSupport is enabled.
Configuration Parameters	

SWS Item	**UDPNM058_Conf :**		
Name	UdpNmTxUserDataPduId {UDPNM_TX_USER_DATA_PDU_ID}		
Description	This parameter defines the Handle ID of the NM User Data I-PDU.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM057_Conf :**		
Name	UdpNmTxUserDataPduRef		
Description	Reference to the NM User Data I-PDU in the global PDU collection.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.8 UdpNmDemEventParameterRefs

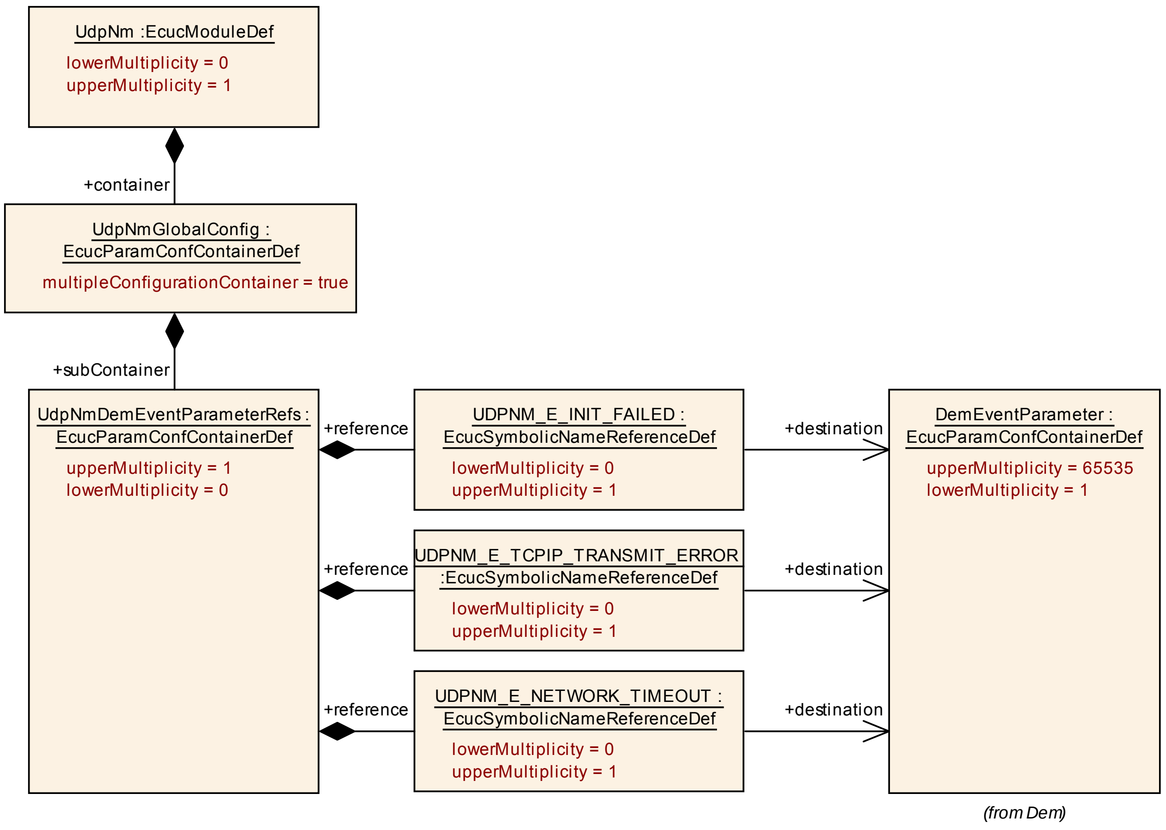
SWS Item	**UDPNM050_Conf :**
Container Name	UdpNmDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**UDPNM051_Conf :**		
Name	UDPNM_E_INIT_FAILED		
Description	Reference to the DemEventParameter which shall be issued when the error "UdpNm initialization has failed, e.g. selected configuration set doesn't exist" has occured.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM053_Conf :**		
Name	UDPNM_E_NETWORK_TIMEOUT		
Description	Reference to the DemEventParameter which shall be issued when the error "NM-Timeout Timer has abnormally expired outside of the Ready Sleep State" has occured.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**UDPNM052_Conf :**		
Name	UDPNM_E_TCPIP_TRANSMIT_ERROR		
Description	Reference to the DemEventParameter which shall be issued when the error "A call to the TCP/IP stack has failedA call to the TCP/IP stack has failed" has occured.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers


UdpNm :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
UdpNmGlobalConfig : EcucParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
UdpNmDemEventParameterRefs : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+reference
UDPNM_E_INIT_FAILED : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
UDPNM_E_TCPIP_TRANSMIT_ERROR :EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
UDPNM_E_NETWORK_TIMEOUT : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+destination
+destination
DemEventParameter : EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 1
(from Dem)

10.3 Published parameters

[UDPNM021] ⌈ The standardized common published parameters as required by BSW00402 in the SRS General on Basic Software Modules [2] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [21]. ⌋()

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[UDPNM999] ⌈ These requirements are not applicable to this specification. ⌋

(BSW170, BSW00387, BSW00375, BSW00416, BSW168, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00429, BSW00432, BSW00434, BSW00336, BSW00417, BSW161, BSW162, BSW005, BSW00415, BSW164, BSW00325, BSW00326, BSW160, BSW00413, BSW00347, BSW00305, BSW00307, BSW00335, BSW00410, BSW00314, BSW00328, BSW00312, BSW006, BSW00377, BSW00306, BSW00309, BSW00330, BSW00331, BSW172, BSW010, BSW00333, BSW00321, BSW00341, BSW00334, BSW151, BSW046, BSW050, BSW052, BSW02509, BSW153, BSW136, BSW140, BSW054, BSW142, BSW144, BSW147, BSW154, BSW139, BSW)