Document Title	Specification of CAN Network Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	013
Document Classification	Standard

Document Version	3.5.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History			
Date	Version	Changed by	Change Description
28.02.2014	3.5.0	AUTOSAR Release Management	• Corrections for Partial Networking • Corrections when starting NM PDU sending • Harmonization with AR4 (Migration Support) • Editorial changes • Removed chapter(s) on change documentation
17.05.2012	3.4.0	AUTOSAR Administration	• Support of a coordinated shutdown with multiple connected gateways • Corrections within Timeout Handling for Partial Networking • Adapted restart handling to Repeat Message State
07.04.2011	3.3.0	AUTOSAR Administration	• Support for Partial Networking • Support for Car Wakeup • Immediate Transmission of NM PDUs
15.09.2010	3.2.0	AUTOSAR Administration	• Harmonization of CanNm_RxIndication signature • Legal disclaimer revised
28.01.2010	3.1.0	AUTOSAR Administration	• Improved configuration and interoperation of CanNm and CanIf • Legal disclaimer revised
23.06.2008	3.0.1	AUTOSAR Administration	• Legal disclaimer revised
18.12.2007	3.0.0	AUTOSAR Administration	• Merge CAN NM and Generic NM • Document meta information extended • Small layout adaptations made

Document Change History

Date	Version	Changed by	Change Description
31.01.07	2.0.0	AUTOSAR Administration	• Post build and link-time configuration variant introduced • Configurable NMPDU format introduced • Passive mode introduced • Legal disclaimer revised • Release Notes added • "Advice for users" revised • "Revision Information" added
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

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

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

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

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

1 Introduction and Functional Overview

This document describes the concept, core functionality, optional features, interfaces and configuration issues of the AUTOSAR CAN Network Management (CanNm).

The AUTOSAR CAN Network Management is a hardware independent protocol that can only be used on CAN (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 CAN Network Management (CanNm) function provides an adaptation between Network Management Interface (Nm) and CAN Interface (CanIf) module. For a general understanding of the AUTOSAR Network Management functionality please refer to [9].

2 Acronyms and abbreviations

Acronym / Abbreviation:	Description:
API	Application Programming Interface
BSW	Basic Software
CanIf	Abbreviation for the CAN Interface
CanNm	Abbreviation for CAN Network Management
DET	Development Error Tracer
DEM	Diagnostic Event Manager
NM	Network Management
PDU	Protocol Data Unit
PNI	Partial Network Information (Bit)
SDU	Service Data Unit

Term:	Description:
"PDU transmission ability is disabled"	This means that the NM PDU transmission has been disabled by the optional service CanNm_DisableCommunication.
"PN filter mask"	Vector of filter mask bytes defined by configuration container(s) CanNmPnFilterMaskByte
"Repeat Message Request Bit Indication"	CanNm_RxIndication finds the RptMsgRequest set in the Control Bit Vector of a received Network Management PDU.

3 Related documentation

3.1 Input documents

[1] AUTOSAR Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_General.pdf

[3] AUTOSAR Requirements on Network Management
AUTOSAR_SRS_NM.pdf

[4] AUTOSAR Specification of CAN Interface
AUTOSAR_SWS_CAN_Interface.pdf

[5] AUTOSAR Specification of FlexRay Network Management
AUTOSAR_SWS_FlexRay_NM.pdf

[6] AUTOSAR Specification of Communication Stack Types
AUTOSAR_SWS_ComStackTypes.pdf

[7] AUTOSAR Specification of ECU Configuration
AUTOSAR_ECU_Configuration.pdf

[8] AUTOSAR Specification of BSW Scheduler,
AUTOSAR_SWS_BSW_Scheduler.pdf

[9] AUTOSAR Specification of Generic Network Management Interface
AUTOSAR_SWS_NMInterface.pdf

[10] Specification of Communication Manager
AUTOSAR_SWS_ComManager.pdf

[11] Specification of ECU State Manager
AUTOSAR_SWS_ECU_StateManager.pdf

[12] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[13] Specification of Diagnostics Event Manager
AUTOSAR_SWS_DEM.pdf

[14] Specification of Development Error Tracer
AUTOSAR_SWS_DET.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] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

Not available.

4 Constraints and assumptions

4.1 Limitations

1. One channel of CanNm is associated with only one NM-Cluster in one network. One NM-Cluster can have only one channel of CanNm in one node.
2. One channel of CanNm is associated with only one network within the same ECU.
3. CanNm is only applicable for CAN systems.

The Figure 4-1 presents an AUTOSAR NM stack within an example ECU belonging to two CAN NM-clusters.
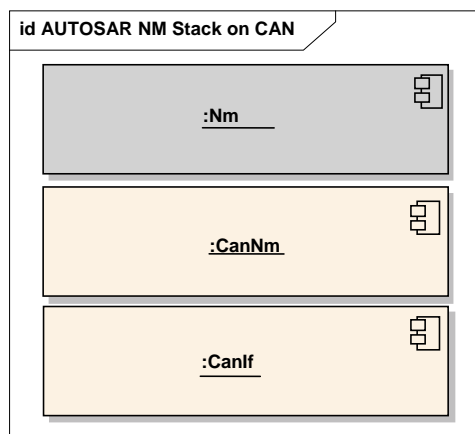
Figure 4-1

This module is only applicable for CAN systems.

CANNM131: The AUTOSAR CAN NM algorithm shall support up to 64 nodes per NM-Cluster.

Note: The AUTOSAR CAN NM algorithm can support an arbitrary number of nodes per NM-cluster (even more than default 64 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 CAN NM coordination algorithm.

4.2 Applicability to car domains

CanNm can be applied to any car domain under limitations provided above.

5 Dependencies to other modules

CAN Network Management (CanNm) uses services of CAN Interface (CanIf) and provides services to the Generic Network Management Interface (Nm).
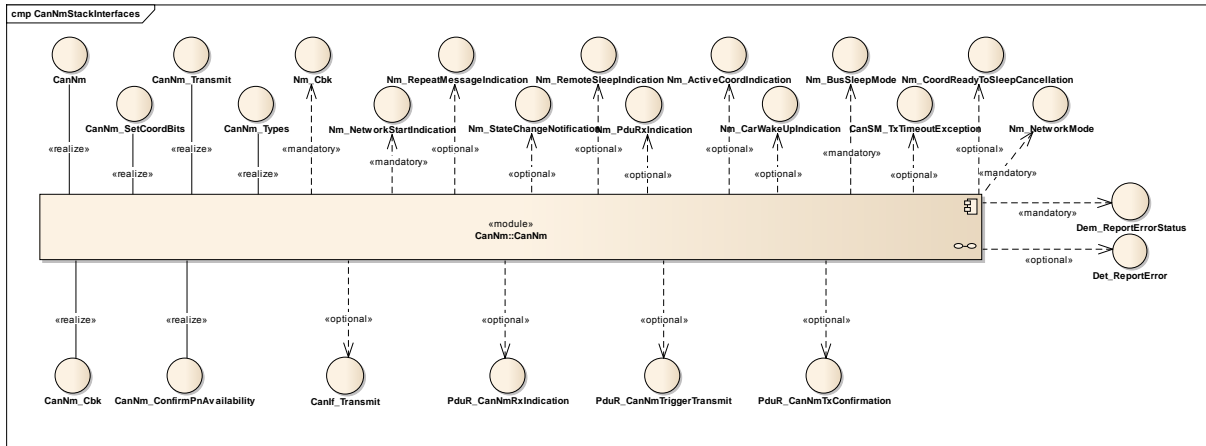
Figure 5-1 Dependencies to other modules

5.1 File Structure

5.1.1 Code File Structure

CANNM081: The following C-files shall be provided by the CanNm module.
- `CanNm.c` (for implementation of provided functionality)
- `CanNm_Cfg.c` (for pre-compile time configurable parameters)
- `CanNm_Lcfg.c` (for link time configurable parameters)
- `CanNm_PBcfg.c` (for post build time configurable parameters)

5.1.2 Header File Structure

CANNM044: The following H-files shall be provided by the CanNm module.
- `CanNm.h` (for declaration of provided interface functions)
- `CanNm_Cbk.h` (for declaration of provided call-back functions)
- `CanNm_Cfg.h` (for pre-compile time configurable parameters)

CANNM082: The following H-files shall be included by the CanNm module.
- `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)
- `CanNm.h` (for declaration of provided interface functions)
- `Nm_Cbk.h` (for CAN NM specific callbacks of Generic Generic Network Management Interface)
- `Det.h` (for interface of DET – optional, included only if DET is configured)
- `Dem.h` (for interface of DEM)
- `PduR_CanNm.h` (interface to PduR – optional)
- `CanIf.h` (interface to CanIf)
- `NmStack_Types.h` (for common NM types)
- `SchM_CanNm.h` (for services of the Basic Software Scheduler)
- `MemMap.h` (for Memory Mapping)

```
CanNm_Cfg.h      NmStack_Types.h      ComStack_Types.h


CanNm_Cbk.h              CanNm.h                      CanIf.h


Nm_Cbk.h                 CanNm.c                   PduR_CanNm.h


Nm_Cfg.h                                           SchM_CanNm.h


          MemMap.h      Det.h      Dem.h
```
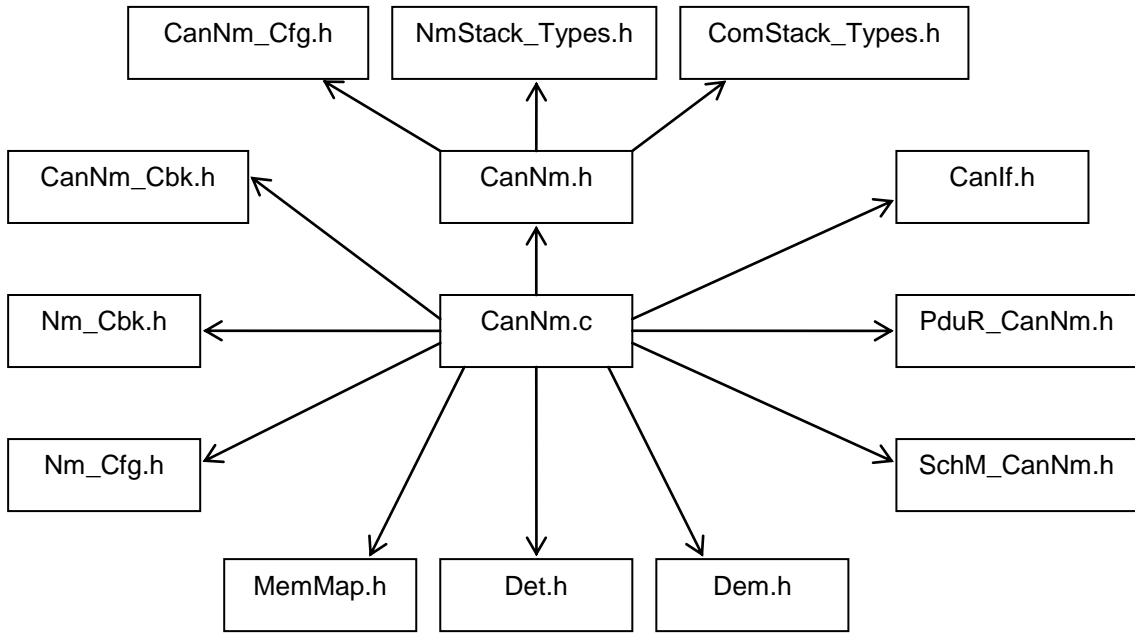
Figure 5-2 Header File Structure

6 Requirements traceability

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 CANNM044
[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 (CanNm is no SW-C)
[BSW171] Configurability of optional functionality	Ok; see Chapter 10.2
[BSW00380] Separate C-Files for configuration parameters	Ok; see CANNM044
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Ok; see CANNM081
[BSW00381] Separate configuration header file for pre-compile time parameters	Ok; see CANNM044
[BSW00412] Separate H-File for configuration parameters	Ok; see CANNM044
[BSW00383] List dependencies of configuration files	Ok; see CANNM083
[BSW00384] List dependencies to other modules	Ok;
[BSW00387] Specify the configuration class of callback function	n/a (Callback 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 0
[BSW00375] Notification of wake-up reason	n/a (CanNm 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 8
[BSW168] Diagnostic Interface of SW components	n/a (diagnostics for CanNm not required)
[BSW00407] Function to read out published parameters	Ok; see chapter 0
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	n/a (CanNm has no interface to the RTE)
[BSW00424] BSW main processing function task allocation	n/a (CanNm 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 7.15
[BSW00429] Restricted BSW OS functionality access	n/a (none of these services are used by the CanNm)
[BSW00431] The BSW Scheduler module implements task bodies	Ok; see chapter 7.15
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	n/a (transmission and reception is handled in CanNm_MainFunction)
[BSW00433] Calling of main processing functions	Ok; see chapter 7.15
[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 7.14
[BSW00338] Detection and Reporting of development errors	Ok; see chapter 7.14 and 10.2
[BSW00369] Do not return development error codes via API	Ok; see chapter 8
[BSW00339] Reporting of production relevant error status	Ok; see chapter 7.14
[BSW00417] Reporting of Error Events by Non-Basic Software	n/a (CanNm is no SW-C)
[BSW00323] API parameter checking	Ok; see CANNM016
[BSW004] Version check	Ok;
[BSW00409] Header files for production code error IDs	Ok; see CANNM082
[BSW00385] List possible error notifications	Ok; see 7.14
[BSW00386] Configuration for detecting an error	Ok: `CANNM_DEV_ERROR_DETECT`
[BSW161] Microcontroller abstraction	n/a (CanNm microcontroller independent)
[BSW162] ECU layout abstraction	n/a (CanNm is ECU hardware independent)
[BSW005] No hard coded horizontal interfaces within MCAL	n/a (CanNm 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; CanNm 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)

Document ID 013:AUTOSAR_SWS_CAN_NM

[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 8
[BSW00373] Main processing function naming convention	Ok; see chapter 8.6.1
[BSW00327] Error values naming convention	Ok; see chapter 7.14
[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	Ok; see CANNM084
[BSW00411] Get version info keyword	Ok; see chapter 8.3.14
[BSW00346] Basic set of module files	Ok; see CANNM081 and CANNM044
[BSW158] Separation of configuration from implementation	Ok; see CANNM081 and CANNM044
[BSW00314] Separation of interrupt frames and service routines	n/a (CanNm doesn't have interrupt frame definitions)
[BSW00370] Separation of callback interface from API	Ok; see CANNM044
[BSW00348] Standard type header	Ok; see CANNM082
[BSW00353] Platform specific type header	Ok; see CANNM082
[BSW00361] Compiler specific language extension header	Ok; see CANNM082
[BSW00301] Limit imported information	Ok; see CANNM082 and CANNM083
[BSW00302] Limit exported information	Ok; see CANNM044 and chapter 8
[BSW00328] Avoid duplication of code	n/a (implementation specific)
[BSW00312] Shared code shall be reentrant	n/a (implementation specific)
[BSW006] Platform independency	n/a (CanNm is hardware independent)
[BSW00357] Standard API return type	Ok; see chapter 8
[BSW00377] Module specific API return types	n/a (CanNm doesn't define own types)
[BSW00304] AUTOSAR integer data types	Ok; see chapter 8
[BSW00355] Do not redefine AUTOSAR integer data types	Ok; see chapter 8
[BSW00378] AUTOSAR boolean type	Ok; see chapter 8 and 10.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	n/a (implementation specific)
[BSW00308] Definition of global data	Ok; see CANNM081
[BSW00309] Global data with read-only constraint	n/a (implementation specific)
[BSW00371] Do not pass function pointers via API	Ok; see chapter 8
[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 callback functions	Ok; see chapter 8.3.17
[BSW00360] Parameters of callback functions	Ok; see chapter 8.3.17
[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 (CanNm doesn't provide status information)
[BSW009] Module User Documentation	Ok; see whole document
[BSW00401] Documentation of multiple instances of	Ok; see chapter 10.2

configuration parameters	
[BSW172] Compatibility and documentation of scheduling strategy	n/a (implementation specific)
[BSW010] Memory resource documentation	n/a (implementation specific)
[BSW00333] Documentation of callback function context	n/a (implementation specific)
[BSW00374] Module vendor identification	Ok; see chapter 0
[BSW00379] Module identification	Ok; see chapter 0
[BSW003] Version identification	Ok; see chapter 0
[BSW00318] Format of module version numbers	Ok; see chapter 0
[BSW00321] Enumeration of module version numbers	n/a (implementation specific)
[BSW00341] Microcontroller compatibility documentation	n/a (CanNm 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 (The CAN bus satisfies already this requirement)
[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 8
[BSW046] Trigger of startup of all Nodes at any Point in Time	n/a (not in the responsibility of CanNm)
[BSW047] Bus Keep Awake Services	Ok; see chapter 8.3.6
[BSW048] Bus Sleep Mode	n/a (not in the responsibility of CanNm)
[BSW050] NM State Information	n/a (CanNm has no states)
[BSW051] NM State Change Indication	n/a (CanNm has no states)
[BSW052] Notification that all other ECUs are ready to sleep	n/a (not in the responsibility of CanNm)
[BSW02509] Notification that at least one other node is not ready to sleep anymore	n/a (not in the responsibility of CanNm)
[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 CanNm)
[BSW02508] Unambiguous node identification per bus	Ok; see chapter 8.3.10
[BSW02505] Sending node identifier	Ok; see chapter 7.12
[BSW02506] Receiving node identifier	Ok; see chapter 8.3.9
[BSW02511] Configurable Role in Cluster Shutdown	Ok; see 7.8.3
[BSW053] Deterministic Behavior in Case of Bus Unavailability	Ok; see chapter 0
[BSW137] Communication system error handling	Ok; see chapter 0
[BSW136] Coordination of coupled networks	n/a (not in the scope of CanNm)
[BSW140] Compliance with OSEK NM on a gateway	n/a

	(not in the scope of CanNm)
[BSW054] Deterministic Time for Bus Sleep	n/a (not in the scope of CanNm)
[BSW142] Limitation of NM bus load	Ok; see chapter 7.7
[BSW143] Predictable NM bus load	Ok; see chapter 7.6.1
[BSW144] ECU cluster size	n/a (CanNm hasn't any restriction concerning cluster size)
[BSW145] Robustness against NM message losses	n/a (not in the scope of CanNm)
[BSW146] Robustness against NM message jitter	n/a (not in the scope of CanNm)
[BSW147] Processor independent algorithm	n/a (not in the scope of CanNm)
[BSW149] Configurable Timing	Ok; see chapter 10.2
[BSW154] Bus independency of API	n/a (CanNm 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 CanNm)
[BSW02510] Immediate Transmission Confirmation	Ok; see chapter 8.4.1
[BSW02512] CommunicationControl (28 hex) service support	n/a (This feature is not supported currently; because of time constraints it is postponed to the next major release of AUTOSAR)

7 Functional specification

7.1 Coordination algorithm

The AUTOSAR CAN NM is based on decentralized direct network management strategy, which means that every network node performs activities self-sufficient depending on the NM PDUs only that are received or transmitted within the communication system.

The AUTOSAR CAN NM coordination algorithm is based on periodic NM PDUs, which are received by all nodes in the cluster via broadcast transmission. Reception of NM PDUs 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 PDUs, but as long as NM PDUs from other nodes are received, it postpones transition to the Bus Sleep Mode. Finally, if a dedicated timer elapses because no NM PDUs are received anymore, every node initiates transition to the Bus Sleep Mode.

If any node in the NM-cluster requires bus-communication, it can wake-up the NM-cluster from the Bus Sleep Mode by transmitting NM PDUs. For more details concerning wakeup procedure, please refer to the Software Specifications of the Mode Management.

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

CANNM087: Every network node shall transmit periodic NM PDUs as long as it requires bus-communication; otherwise it shall transmit no NM PDUs.

CANNM088: If bus communication is released and there are no NM PDUs on the bus for a configurable amount of time determined by `CANNM_TIMEOUT_TIME +` `CANNM_WAIT_BUS_SLEEP_TIME` (both configuration parameters) transition into the Bus Sleep Mode shall be performed.

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

CANNM089: The AUTOSAR CAN NM state machine shall contain states, transitions and triggers required for the AUTOSAR CAN NM coordination algorithm seen from point of view of one single node in the NM-cluster.

Note: Transitions in the AUTOSAR CAN NM state machine shall be triggered by calls of selected interface functions or by expiration of internal timers or counters.

Note: State transitions have to be performed latest within the next main function.

Note: An UML state chart of the AUTOSAR CanNm state machine from point of view of one single node in the network management cluster can be found in detail in the API specification chapter 8).

7.2 Operational Modes

In the following chapter operational modes of the AUTOSAR CAN NM coordination algorithm are described in detail.

CANNM091: The service call `CanNm_GetState` shall provide information about the current state and the current mode of the NM state machine.

CANNM092: The AUTOSAR CAN NM shall consist of three operational modes:
- Network Mode
- Prepare Bus Sleep Mode
- Bus Sleep Mode

CANNM093: Changes of the AUTOSAR CAN NM operational modes shall be notified to the upper layer by means of callback functions.

7.2.1 Network Mode

CANNM094: The Network Mode shall consist of three internal states:
- Repeat Message State
- Normal Operation State
- Ready Sleep State

CANNM095: When the Network Mode is entered from Bus Sleep Mode or Prepare Bus Sleep Mode, by default, the Repeat Message State shall be entered.

CANNM096: When the Network Mode is entered, the NM-Timeout Timer shall be started.

CANNM097: When the Network Mode is entered, the NM shall notify the upper layer by calling `Nm_NetworkMode`.

CANNM098: At successful reception of a NM PDU in the Network Mode, the NM-Timeout Timer shall be restarted.

CANNM099: At successful transmission of a NM PDU in the Network Mode, the NM-Timeout Timer shall be restarted.

CANNM206: 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 7.8.3) the Repeat Message State ensures, that any transition from Bus Sleep or Prepare Bus Sleep to the Network Mode becomes visible to the other nodes on the network. Additionally it

ensures that any node stays active for a minimum amount of time. Optionally it can be used for detection of present nodes.

CANNM100: When the Repeat Message State is entered from Bus Sleep Mode, Prepare-Bus Sleep Mode, Normal Operation State or Ready Sleep State, the CanNm module shall (re-)start transmission of Network Management PDUs unless passive mode is enabled and/or communication is disabled.

CANNM101: When the NM-Timeout Timer expires in the Repeat Message State, the NM-Timeout Timer shall be restarted.

CANNM102: The NM shall stay in the Repeat Message State for a configurable amount of time determined by the `CANNM_REPEAT_MESSAGE_TIME` (configuration parameter); after that time the Repeat Message State shall be left.

CANNM103: When Repeat Message State is left, the Normal Operation State shall be entered, if the network has been requested (see CANNM104).

CANNM106: When Repeat Message State is left, the Ready Sleep State shall be entered, if the network has been released (see CANNM105).

CANNM107: When Repeat Message State is left and the option `CANNM_NODE_DETECTION_ENABLED` is enabled, the Repeat Message Bit shall be cleared.

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 is requested.

CANNM116: 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 PDU transmission ability has been disabled.

CANNM117: When the NM-Timeout Timer expires in the Normal Operation State, the NM-Timeout Timer shall be restarted.

CANNM118: 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 CANNM105).

CANNM119: 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.

CANNM120: At Repeat Message Request (`CanNm_RepeatMessageRequest`) in the Normal Operation State, the Normal Operation State shall be left and the Repeat Message State shall be entered.

CANNM121: At Repeat Message Request 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.

CANNM108: When the Ready Sleep State is entered from Repeat Message State or Normal Operation State, transmission of NM PDUs shall be stopped unless passive mode is enabled.

CANNM109: 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.

CANNM110: 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 [CANNM104](#)).

CANNM111: 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.

CANNM112: At Repeat Message Request (`CanNm_RepeatMessageRequest`) in the Ready Sleep State, the Ready Sleep State shall be left and the Repeat Message State shall be entered.

CANNM113: At Repeat Message Request in Ready Sleep State the Repeat Message Bit shall be set.

7.2.2 Prepare Bus Sleep Mode

Bus activity is calmed down (i.e. queued messages are transmitted in order to make all Tx-buffers empty) and finally there is no activity on the bus in the Prepare Bus Sleep Mode.

CANNM114: When Prepare Bus Sleep Mode is entered, the NM shall notify the upper layer by calling `Nm_PrepareBusSleepMode`.

CANNM115: The NM shall stay in the Prepare Bus Sleep Mode for a configurable amount of time determined by the `CANNM_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.

CANNM122: When the network has been requested in the Prepare Bus Sleep Mode and Network Mode has been entered, a NM PDU shall be transmitted if `CANNM_IMMEDIATE_RESTART_ENABLED` (configuration parameter) is defined.

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 NM PDUs in CAN NM, the transmission of the first NM PDU in Repeat Message State can be delayed significantly. In order to avoid

a delayed re-start of the network the transmission of a NM PDU can be requested immediately.

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

CANNM123: 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 CANNM095).

CANNM124: At successful reception of a 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 CANNM095).

CANNM459 If `CanNm_PassiveStartUp` is called in the Prepare Bus Sleep Mode, the Prepare Bus Sleep Mode shall be left and the Network Mode shall be entered.

7.2.3 Bus Sleep Mode

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

CANNM125: If a configurable amount of time determined by the `CANNM_TIMEOUT_TIME + CANNM_WAIT_BUS_SLEEP_TIME` (both configuration parameters) is identically configured for all nodes in the NM-cluster, then all nodes in the NM-cluster that are coordinated with use of the AUTOSAR NM algorithm shall perform the transition into the Bus Sleep Mode at approximately the same time.

Note: The parameters `CANNM_TIMEOUT_TIME` and `CANNM_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 exactly or approximately at the same time; time jitter 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 PDU waiting time in the Tx-queue (if transmission confirmation is made immediately after transmit request).

In the best case only oscillator's drift shall be taken into account for a configurable amount of time determined by the value `CANNM_TIMEOUT_TIME + CANNM_WAIT_BUS_SLEEP_TIME` (both configuration parameters).

CANNM126: When Bus Sleep Mode is entered, the NM shall notify the upper layer by calling `Nm_BusSleepMode`; it shall not be however the case if Bus Sleep Mode is entered by default at initialization.

CANNM127: At successful reception of a NM PDU in the Bus Sleep Mode, the NM shall notify the upper layer by calling `Nm_NetworkStartIndication`.

CANNM413 When the CanNm module receives a NM PDU (call of `CanNm_RxIndication`) in the Bus Sleep Mode, the CanNm module shall report the error `CANNM_E_NET_START_IND` if development error tracing is enabled (`CANNM_DEV_ERROR_DETECT` is set to `TRUE`).

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

CANNM128: If `CanNm_PassiveStartUp` is called in the Bus Sleep Mode, the Bus Sleep Mode shall be left and the Network Mode shall be entered.

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

CANNM129: When the network is requested in Bus Sleep Mode, the Bus Sleep Mode shall be left and the Network Mode shall be entered; by default the Repeat Message State is entered (refer to [CANNM095](#) and [CANNM104](#)).

7.3 Network states

Network states (i.e. 'requested' and 'released') are two additional states of the AUTOSAR CAN NM 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.

CANNM104: The optional service call `CanNm_NetworkRequest` shall request the network. I.e. network state shall be changed to 'requested'.

CANNM105: The optional service call `CanNm_NetworkRelease` shall release the network. I.e. network state shall be changed to 'released'.

7.4 Initialization

CANNM140: After reset the Network Management state shall be set to `NM_STATE_UNINIT`.

CANNM141: After successful initialization the Network Management state shall be set to `NM_STATE_BUS_SLEEP`, otherwise it shall be set to `NM_STATE_UNINIT`.

CANNM041: The function `CanNm_Init` shall initialize the CanNm.

CANNM142: The NM shall be initialized after CanIf is initialized and before any other NM service is called.

CANNM143: If initialized, by default, the network state shall be set to 'released'

CANNM144: If initialized, by default, the Bus Sleep Mode shall be entered.

CANNM145: Bus traffic should not be prohibited if AUTOSAR CAN NM is not initialized.

CANNM147: If `CanNm_PassiveStartUp` is called in the Network Mode, the service shall not be executed and `NM_E_NOT_EXECUTED` shall be returned.

CANNM060: The function `CanNm_Init` shall select the active configuration set by means of a configuration pointer parameter being passed (see 8.3.1).

CANNM061: After initialization the CanNm Message Cycle Timer and NM Message Tx Timeout Timer shall be stopped.

Note: No timer (CanNm Message Cycle Timer and NM Message Tx Timeout Timer) is needed if `CANNM_PASSIVE_MODE_ENABLED` is enabled, because no NM PDUs are transmitted by such nodes.
Note: The NM Message Tx Timeout Timer is not needed in case of `CANNM_IMMEDIATE_TXCONF_ENABLED` is enabled.

CANNM023: After initialization the bus load reduction shall be deactivated if bus load reduction is enabled.

CANNM033: After initialization the transmission of NM PDUs shall be stopped.

CANNM039: If CanNm is not initialized a call of any CanNm function shall be rejected with the respective error code.

CANNM025: After initialization each byte of the user data bytes shall be set to `0xFF`.

CANNM085: After initialization the Control Bit Vector shall be set to `0x00`.

CANNM204: The NM State shall be initialized with `NM_STATE_UNINIT` for all CanNm channels that are configured to be inactive (configuration parameter `CANNM_CHANNEL_ACTIVE`).

7.5 Execution

7.5.1 Processor architecture

CANNM146: The AUTOSAR CAN NM coordination algorithm shall be processor independent, which means, it shall not rely on any processor specific hardware support and thus shall be realizable on any processor architecture that is in the scope of AUTOSAR.

7.5.2 Timing parameters

CANNM148: Timing parameters necessary for the AUTOSAR CAN NM shall be determined by the following configuration parameters:
- NM-Timeout Time: `CANNM_TIMEOUT_TIME`
- Repeat Message Time: `CANNM_REPEAT_MESSAGE_TIME`
- Wait Bus Sleep Time: `CANNM_WAIT_BUS_SLEEP_TIME`
- Remote Sleep Indication Time: `CANNM_REMOTE_SLEEP_IND_TIME` (optional)

7.6 Communication Scheduling

7.6.1 Transmission

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

CANNM072: The transmission of NM PDUs shall be configurable by means of `CANNM_PASSIVE_MODE_ENABLED` (see chapter 10.2).

Note: Passive nodes don't transmit NM PDUs i.e. they cannot actively influence the shutdown decision, but they do receive NM PDU in order to be able to shut down synchronous.

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

CANNM001: Two transmission modes should be provided by the CanNm:
1. Periodic transmission mode (mandatory). In this transmission mode the CanNm sends periodically NM PDUs.
2. Periodic transmission mode with bus load reduction (optional). In this transmission mode the CanNm transmits NM PDUs due to a specific algorithm. It ensures a reduced bus load.

Note: The periodic transmission mode is used in the "Repeat Message State" and "Normal Operation State" if the bus load reduction mechanism is disabled.
The periodic transmission mode with bus load reduction is only used, in the "Normal Operation State" if the bus load reduction mechanism is enabled.

CANNM071: The immediate transmission confirmation mechanism shall be configurable by means of the `CANNM_IMMEDIATE_TXCONF_ENABLED` (see 10.2).

Note: The immediate transmission confirmation mechanism is used for systems which don't want to use the actual confirmation from the CanIf.

Rationale: If the bus access is completely regulated through an offline system design tool, the actual transmit confirmation by CanIf can be regarded as redundant. Since the maximum arbitration time is assumed to be known it is acceptable to immediately raise the confirmation at the transmission request time.

[**CANNM005**] If the repeat message state is not entered due to a call to `CanNm_NetworkRequest` OR `CanNmImmediateNmTransmissions` is zero the transmission of NM PDU shall be delayed by `CANNM_MSG_CYCLE_OFFSET` after entering the repeat message state.

[**CANNM006**]⌈ If Normal Operation is entered from Ready Sleep State the

transmission of NM PDUs shall be started immediately.⌋()

CANNM032: If transmission of NM PDUs has been started and the CanNm Message Cycle Timer expires a NM PDU shall be transmitted by the CanIf function `CanIf_Transmit`.

Note: If the call of `CanIf_Transmit` fails the Nm is informed by the Transmission Error handling described in chapter 0.

CANNM040: If the CanNm Message Cycle Timer expires it shall be restarted with `CANNM_MSG_CYCLE_TIME`.

CANNM034: If a NM PDU has been successfully transmitted the function `CanNm_TxConfirmation` shall be called by CanIf if `CANNM_IMMEDIATE_TXCONF_ENABLED` (configuration parameter) is disabled.

CANNM052: The bus load reduction mechanism shall be statically configurable by means of the `CANNM_BUS_LOAD_REDUCTION_ENABLED` parameter (see 10.2).

CANNM051: If transmission of NM PDUs has been stopped the CanNm Message Cycle Timer shall be canceled.

CANNM130: The optional service call `CanNm_RequestBusSynchronization` shall trigger transmission of a single NM PDU independently of the normal periodic transmission if `CANNM_PASSIVE_MODE_ENABLED` (configuration parameter) is disabled.

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

[**CANNM007**]⌈If `CANNM_RETRY_FIRST_MESSAGE_REQUEST` is enabled and if the first transmit request after transition from Bus Sleep to Repeat Message State is not

accepted by CanIf, the message request shall be repeated in the next main function until one transmit request is accepted by CanIf. ⌋()

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

7.6.1.1 Immediate Transmission

CANNM406 When entering the Repeat Message state from Bus Sleep Mode or Prepare Bus Sleep Mode because of `CanNm_NetworkRequest` (active wakeup) and `CanNmImmediateNmTransmissions` is greater zero the NM PDUs shall be transmitted using `CanNmImmediateNmCycleTime` as cycle time. The transmission of the first NM PDU shall be triggered as soon as possible. After the transmission the Message Cycle Timer shall be reloaded with `CanNmImmediateNmCycleTime`. The `CanNmMsgCycleOffset` shall not be applied in this case.

CANNM454 If `CanNmPnHandleMultipleNetworkRequests` is set to `TRUE` and `CanNm_NetworkRequest` triggers a state transition from Network Mode to Repeat Message state and `CanNmImmediateNmTransmissions` is greater zero the NM PDUs shall be transmitted using `CanNmImmediateNmCycleTime` as cycle time. The transmission of the first NM PDU shall be triggered as soon as possible. After the transmission the Message Cycle Timer shall be reloaded with `CanNmImmediateNmCycleTime`. The `CanNmMsgCycleOffset` shall not be applied in this case.

CANNM407 The number of NM PDUs transmitted with the cycle time `CanNmImmediateNmCycleTime` is defined by `CanNmImmediateNmTransmissions` (counter). After all immediate NM PDUs have been transmitted the CanNm shall continue the transmission using the `CanNmMsgCycleTime`.

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

Note: CANNM102 is not is not influenced by `CanNmImmediateNmCycleTime`.

7.6.2 Reception

CANNM035: If a NM PDU has been successfully received the function `CanNm_RxIndication` shall be called by CanIf. The NM PDU shall be copied to an internal buffer.

CANNM037: If a NM PDU has been successfully received the Nm function `Nm_PduRxIndication` shall be called (triggered by the function `CanNm_RxIndication`) if `CANNM_PDU_RX_INDICATION_ENABLED` (configuration parameter) is defined.

7.7 Bus Load Reduction Mechanism (optional)

The transmission period of NM PDUs is usually determined by the timing parameter CANNM_MSG_CYCLE_TIME. This parameter has to be equal for all NM nodes which belong to a NM cluster. Without any action this would lead to a bus load which depends on the amount of members of the NM cluster. Even if bursts are prevented through a node specific timing parameter called CANNM_MSG_OFFSET_TIME a mechanism is necessary which reduces the bus load independently of the size of the NM cluster.

In order to achieve that the following two aspects have to be considered:

1. If a NM PDU has been successfully received the CanNm Message Cycle Timer is reloaded with the node specific timing parameter CANNM_MSG_REDUCED_TIME. The node specific time CANNM_MSG_REDUCED_TIME shall be greater than ½ CANNM_MSG_CYCLE_TIME and less than CANNM_MSG_CYCLE_TIME.

2. If a NM PDU has been successfully transmitted the CanNm Message Cycle Timer is reloaded with the NM cluster specific timing parameter CANNM_MSG_CYCLE_TIME.

This leads to the following behavior:
Only the two nodes with the smallest CANNM_MSG_REDUCED_TIME time transmit alternating NM PDUs on the network. If one of the nodes stops transmission, the node with the next smallest CANNM_MSG_REDUCED_TIME time will start to transmit NM PDUs. If there is only one node on the network that requires bus communication, one NM PDU per CANNM_MSG_CYCLE_TIME is transmitted.

The algorithm ensures that the bus load is limited to maximum two NM PDUs per CANNM_MSG_CYCLE_TIME.

An example can be found in chapter 11.

CANNM155: Busload reduction shall be statically configurable with use of the CANNM_BUS_LOAD_REDUCTION_ENABLED switch (configuration parameter).

CANNM156: When the Repeat Message State is entered from Bus Sleep Mode, Prepare Bus Sleep Mode, Normal Operation or Ready Sleep State the busload reduction shall be deactivated.

CANNM157: When the Normal Operation State is entered from Repeat Message State or Ready Sleep State the busload reduction shall be activated.

CANNM069: If the bus load reduction mechanism is globally enabled, for a particular channel activated and a NM PDU has been successfully received the CanNm Message Cycle Timer shall be restarted with the node specific time CANNM_MSG_REDUCED_TIME.

7.8 Additional features

7.8.1 Detection of Remote Sleep Indication (optional)

The so-called "Remote Sleep Indication" denotes a situation, where all nodes in the cluster are ready to sleep apart from one node, which still keeps the bus awake.

CANNM149: Detection of remote sleep indication shall be statically configurable with use of the `CANNM_REMOTE_SLEEP_IND_ENABLED` switch (configuration parameter).

CANNM150: If no NM PDUs are received in the Normal Operation State for a configurable amount of time determined by the `CANNM_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`.

CANNM151: If Remote Sleep Indication has been previously detected and if a 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`.

CANNM152: 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`.

CANNM153: Service call `CanNm_CheckRemoteSleepIndication` shall provide the information about current status of Remote Sleep Indication (i.e. already detected or not).

CANNM154: 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.

7.8.2 User Data (optional)

CANNM158: Support of NM user data shall be statically configurable with use of the `CANNM_USER_DATA_ENABLED` switch (configuration parameter).

CANNM159: When `CanNm_SetUserData` is called the NM user data for NM PDUs transmitted next on the bus shall be set; operation of setting the NM user data shall guarantee data consistency.

CANNM160: When `CanNm_GetUserData` is called the NM user data out of the most recently received NM PDU shall be provided; operation of providing the NM user data shall guarantee data consistency.

Note: If user data is configured it will be sent for sure in the Repeat Message State. In the Normal Operation State it is up to the configuration of busload reduction. In the Ready Sleep State the user data will not be sent.

7.8.2.1 COM User Data

Alternatively to the usage of the CanNm APIs to set and get user data, CanNm may use the COM to retrieve its user data.

CANNM400 If `CanNmComUserDataSupport` is enabled the API `CanNm_SetUserData` shall not be available.

CANNM401 If `CanNmComUserDataSupport` is enabled the CanNm shall collect the NM User Data from the referenced NM I-PDU by calling `PduR_CanNmTriggerTransmit` and combine the user data with the further NM bytes each time before it requests the transmission of the corresponding NM PDU.

CANNM402 If `CanNmComUserDataSupport` is enabled the CanNm shall call `PduR_CanNmTxConfirmation` within the message transmission confirmation function `CanNm_TxConfirmation` called by the CanIf.

CANNM403 If `CanNmComUserDataSupport` is enabled and the CanNm User Data length does not match with the length of the referenced I-PDU an error shall be reported at generation time.

7.8.3 Passive Mode (optional)

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

CANNM161: Passive Mode shall be statically configurable with use of the `CANNM_PASSIVE_MODE_ENABLED` switch (configuration parameter).

CANNM162: Passive Mode shall be statically configured consistent for all channels within one ECU.

CANNM163: If Passive Mode is used (configuration parameter `CANNM_PASSIVE_MODE_ENABLED`) the following options must not be used:
- Bus Synchronization
  (configuration parameter `CANNM_BUS_SYNCHRONIZATION_ENABLED`)
- Bus Load Reduction
  (configuration parameter `CANNM_BUS_LOAD_REDUCTION_ENABLED`)
- Remote Sleep Indication
  (configuration parameter `CANNM_REMOTE_SLEEP_IND_ENABLED`)
- Node Detection
  (configuration parameter `CANNM_NODE_DETECTION_ENABLED`)
- Communication Control
  (configuration parameter `CANNM_COM_CONTROL_ENABLED`)

7.8.4 NM PDU Rx Indication (optional)

CANNM164: At successful reception of a NM PDU the NM 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 a Rx Indication. Polling of the NM PDU data could result in loss of received NM PDU data in case of a NM PDU burst.

CANNM165: The optional service `Nm_PduRxIndication` shall be statically configurable. It shall be available if `CANNM_PDU_RX_INDICATION_ENABLED` is defined.

7.8.5 State change notification (optional)

CANNM166: All changes of the AUTOSAR CAN NM states shall be notified to the upper layer by calling `Nm_StateChangeNotification`.

CANNM167: The optional service `Nm_StateChangeNotification` shall be statically configurable. It shall be available if `CANNM_STATE_CHANGE_IND_ENABLED` is defined.

7.8.6 Communication Control (optional)

CANNM168: Communication Control shall be statically configurable with use of the `CANNM_COM_CONTROL_ENABLED` switch (configuration parameter).

CANNM169: If initialized, by default, the network NM PDU transmission ability shall be enabled.

CANNM170: The optional service `CanNm_DisableCommunication` shall disable the NM PDU transmission ability.

Note: It is dangerous to use this service, since the NM PDUs are turned off and they need to be turned on after diagnostic session. ComM ensures that the ECU is not shutdown as long as the NM PDU transmission ability is disabled.

CANNM172: The optional service `CanNm_DisableCommunication` shall return `NM_E_NOT_EXECUTED`, if the current mode is not Network Mode.

CANNM173: When the NM PDU transmission ability is disabled, the transmission of NM PDUs shall be stopped.

CANNM174: When the NM PDU transmission ability is disabled, the NM-Timeout Timer shall be stopped.

CANNM175: When the Network Management PDU transmission ability is disabled, the CanNm module shall stop the Remote Sleep Indication Timer.

CANNM176: The optional service `CanNm_EnableCommunication` shall enable the NM PDU transmission ability if the NM PDU transmission ability is disabled.

CANNM177: The optional service `CanNm_EnableCommunication` shall return `NM_E_NOT_EXECUTED` if the NM PDU transmission ability is enabled.

CANNM178: When the Network Management PDU transmission ability is enabled, the transmission of NM PDUs shall be started latest within the next NM main function.

CANNM179: When the NM PDU transmission ability is enabled, the NM-Timeout Timer shall be restarted.

CANNM180: When the Network Management PDU transmission ability is enabled, the CanNm module shall re-start the Remote Sleep Indication Timer.

CANNM181: The optional service `CanNm_RequestBusSynchronization` shall return `NM_E_NOT_EXECUTED` if the NM PDU transmission ability is disabled.

7.8.7 NM coordinator synchronization support

When having more than one coordinator connected to the same bus additional bits in the control bit vector are used to indicate the ID of a coordinator and if the main coordinator requests to start shutdown sequence. The main functionality is described in the Nm module.

CANNM347 The Coordinator ID bits in the CBV shall be set by the API `CanNm_SetCoordBits`.

CANNM409 If the CaNm receives a NM PDU with set NM coordinator ID (see CBV) it shall indicate this to the NmIf by calling `Nm_ActiveCoordIndication`.

Rational: To determine whether a coordinator should be active or passive it needs to know if there are other coordinators with higher priority on the bus. If so it will act as passive.

CANNM341 If CanNm has entered Network Mode or called `Nm_CoordReadyToSleepCancellation` before it shall notify the Nm by calling `Nm_CoordReadyToSleepIndication` on the first reception of a NM PDU with the *NmCoordinatorSleepReady* bit (see CBV) set to 1.

CANNM348 If CanNm called `Nm_CoordReadyToSleepIndication` and is still in Network Mode it shall notify the Nm by calling `Nm_CoordReadyToSleepCancellation` on the first reception of a NM PDU with the NmCoordinatorSleepReady bit (see CBV) set to 0.

CANNM342 The NM Coordinator Sleep Ready bit in the CBV shall be set by the API `CanNm_SetSleepReadyBit`.

CANNM343 This feature is optional and only available if `CanNmCoordinatorSyncSupport` is set to TRUE.

7.9 Car Wakeup (optional)

CANNM414 The position of the Car Wakeup bit in the NM PDU is defined by the configuration parameters `CanNmCarWakeUpBytePosition` and `CanNmCarWakeUpBitPosition`.

7.9.1 Rx Path

CANNM410 If the car wakeup bit within any received NM PDU is `1`, `CanNmCarWakeUpRxEnabled` is `TRUE`, and `CanNmCarWakeUpFilterEnabled` is `FALSE` CanNm shall call `Nm_CarWakeUpIndication` and perform the standard Rx indication handling.

CANNM411 If `CanNm_GetPduData` is called in the context of `Nm_CarWakeUpIndication`, CanNm shall return the PDU data of the PDU that causes the call of `Nm_CarWakeUpIndication`.

Note: This is required to enable the ECU to identify detail about the sender of the car wakeup request.

CANNM412 If `CanNmCarWakeUpFilterEnabled` is `TRUE`, the car wakeup bit within any received NM PDU is `1`, `CanNmCarWakeUpRxEnabled` is `TRUE` and the Node ID in the received NM PDU is equal to `CanNmCarWakeUpFilterNodeId` the CanNm module shall call `Nm_CarWakeUpIndication` and perform the standard Rx Indication handling.

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

7.9.2 Tx Path

The transmission of the car wakeup bit shall be handled by the application using the NM user data mechanism provided by the CanNm module.

7.10 Extensions for Partial Networking

7.10.1 Rx Handling of NM PDUs

CANNM415 If the `CanNmPnEnabled` is `FALSE`, the CanNm shall not drop NM PDUs from further Rx handling and the partial networking extensions shall be disabled.

CANNM416 If `CanNmPnEnabled` is `TRUE`, the PNI bit in the received NM PDU is 0 and `CanNmAllNmMessagesKeepAwake` is `TRUE`, the CanNm module shall not drop NM PDUs from further Rx handling omitting the extensions for partial networking.

Note: This is required to enable the Gateway to stay awake on any kind of NM PDU.

CANNM417 If `CanNmPnEnabled` is `TRUE`, the PNI bit in the received NM PDU is 0 and `CanNmAllNmMessagesKeepAwake` is `FALSE`, the CanNm module shall ignore the received NM PDU.

CANNM418 If `CanNmPnEnabled` is `TRUE` and the PNI bit in the received NM PDU is 1, CanNm module shall process the Partial Networking Information of the NM PDU as described in chapter 7.10.3 NM PDU Filter Algorithm.

7.10.2 Tx Handling of NM PDUs

CANNM419 If `CanNmPnEnabled` is `TRUE` the CanNm module shall set the value of the transmitted PNI bit to 1.

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

CANNM420 If `CanNmPnEnabled` is `FALSE` the CanNm module shall set the value of the transmitted PNI bit always to 0.

7.10.3 NM PDU Filter Algorithm

The intention of the NM PDU filter algorithm is to drop all received NM PDUs that are not relevant for the ECU. If there is no NM PDU on the network, that is relevant for the receiving ECU, the NM Timeout Timer is no longer restarted and the CanNm module changes to Prepare Bus Sleep Mode during active bus communication.
In order to distinguish between NM PDUs that are relevant for the ECU and PDUs that are not relevant, the CanNm evaluates the NM User Data that contains the PN requests provided by requesting ECU. Every bit of the PN request information represents one PN.
It is statically configured if the ECU (CanNm) is part of one specific partial network or not. The NM PDUs are ignored if the ECU is not part of the requested partial networks.

CANNM456 During initialization CanNm shall disable the NM PDU filter algorithm on all networks where PN is enabled. When the NM PDU filter algorithm is disabled all received NM PDUs shall not be dropped from further Rx handling.

CANNM457 If the CanSm calls CanNm_ConfirmPnAvailability the NM PDU filter algorithm shall be enabled on the indicated channel.

Rationale: This is required to allow a malfunctioning PN transceiver to shut down synchronously with the remaining network.

Note: If the NM PDU filter algorithm is not enabled (e.g. due to malfunctioning PN transceiver) the CanNm restarts the NM-Timeout Timer when receiving a NM PDU. Therefore normal shutdown behaviour is performed.

CANNM421 The NM PDU filter algorithm shall evaluate the bytes of the received NM PDU defined by `CanNmPnInfoOffset` (in bytes) starting from byte 0 and `CanNmPnInfoLength` (in bytes). This range is called PN Info Range.

CANNM422 Every bit of the PN Info Range represents one Partial Network. If the bit is set to 1 the Partial Network is requested. If the bit is set to 0 there is no request for this PN.

CANNM423 The filter algorithm shall compare (bitwise AND) the received PN information with the PN filter mask to detect if a relevant PN is requested or not. Each bit of `CanNmPnFilterMaskByte` shall have the following meaning:
0   The PN request is irrelevant for the ECU. The communication stack of the ECU is not kept awake if this bit is set in a received NM PDU.
1   The PN request is relevant for the ECU. The communication stack of the ECU is kept awake if this bit is set in a received NM PDU.

CANNM424 If at least one relevant PN is requested in the received NM PDU the PDU shall not be dropped from further Rx handling

CANNM425 If no relevant PN is requested in the received NM PDU and `CanNmAllNmMessagesKeepAwake` is `FALSE` the PDU shall be dropped from further Rx handling.

CANNM427 If no relevant PN is requested in the received NM PDU and `CanNmAllNmMessagesKeepAwake` is `TRUE` the PDU shall not be dropped from further Rx handling.

Note: This is required to enable the Gateway to stay awake on any kind of NM PDU.

Example:
  -   CanNmPnInfoOffset   = 4
  -   CanNmPnInfoLength   = 2
Only Byte 4 and Byte 5 of the NM PDU contains PN information:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CBV	NID	User Data		PN Info		User Data	
0x40	0x00	0xFF	0xFF	0x12	0x8E	0xFF	0xFF

Figure 7-1 Example NM PDU containing PN information

For this example two CanNmPnFilterMaskBytes would be defined, e.g
  -   CanNmPnFilterMaskByteIndex = 0 with CanNmPnFilterMaskByteValue = 0x01
  -   CanNmPnFilterMaskByteIndex = 1 with CanNmPnFilterMaskByteValue = 0x97

The filter algorithm actions and result would then be:

PN Filter Mask Value (Byte)	Compared to received PN info	Resulting in

0x01 (Byte 0)	0x12 (NM PDU Byte 4)	0x00 (no relevant PN information)
0x97 (Byte 1)	0x8E (NM PDU Byte 5)	0x86 (relevant PN information)

Figure 7-2 Example PN Filter Algorithm

As one byte contains relevant information this NM PDU would not be dropped from further Rx handling.

7.10.4 Aggregation of internal and external requested Partial Networks

Note: This feature is used by every ECU that has to switch I-PDU-Groups because of the activity of partial networks. (e.g. to prevent false timeouts) I-PDU-Groups shall be switched on if the corresponding PN is requested internally or externally. I-PDU-Groups shall be switched off not until all internal and external requests for the corresponding PN are released.
The logic for switching the IPDU-Groups is implemented by ComM. The CanNm only provides the information if a PN is requested or not. The COM module is used to transfer the data to the upper layers.
To switch the I-PDU-Groups synchronously on all direct connected ECUs, CanNm shall provide the information of a request change to the upper layer at (almost) the same time on every ECU. This is why the reset timer is restarted on every received and every sent NM PDU (see below).
The aggregated state of the internal/external requested PNs is called External Internal Requests Aggregated (EIRA).

CANNM428 If `CanNmPnEiraCalcEnabled` is `FALSE` the CanNm module shall skip the aggregation of external and internal PN requests information.

CANNM429 If `CanNmPnEiraCalcEnabled` is `TRUE` the CanNm module shall calculate the aggregation of external and internal PN requests over all CanNm channels.

CANNM431 The EIRA shall have the size of `CanNmNmPnInfoLength` and shall be initialized with value 0 (not requested) for every external and internal PN request.

CANNM432 The CanNm shall only consider the PN request bits that are relevant for the ECU (defined by PN filter mask). All other PN request bits are ignored. Thus the EIRA only contains those PN requests, which are relevant for the ECU.

CANNM430 If a NM PDU is received the CanNm shall set every requested and filtered (relevant) PN request bit within this received NM PDU in the EIRA to 1.

CANNM433 If a NM PDU is send by the CanNm, the CanNm module shall set every PN request bit in the EIRA to 1 that has been requested by the PN request bits in the transmitted NM PDU.

CANNM434 The CanNm module shall provide an EIRA reset timer for every PN request bit together for all physical CAN channels.

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

CANNM435 If an NM PDU is received the CanNm module shall restart the EIRA reset timer for every PN request bit that has been requested in the received NM PDU with `CanNmPnResetTime`.

CANNM436 If a NM PDU is send by the CanNm, the CanNm module shall restart the EIRA reset timer for every PN request bit that has been requested in the NM PDU with `CanNmPnResetTime`.

Note: `CanNmPnResetTime` shall be configured to a value greater than `CanNmMsgCycleTime`. If `CanNmPnResetTime` is configured to a value smaller than `CanNmMsgCycleTime` and only one ECU requests the PN, the request state toggles in the EIRA because request state is rested before the requesting ECU is able to send the next NM PDU.

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

CANNM437 If one of the EIRA reset timers expires, the CanNm module shall set PN request bit for the corresponding PN in the EIRA to 0.

CANNM438 If content of EIRA changes (any bit changes from 1 to 0 or from 0 to 1) because of a received or transmitted NM PDU or the EIRA reset timer expiration, the CanNm shall inform the upper layers by calling `PduR_CanNmRxIndication()`. By means of the Rx Indication function the EIRA data shall be provided to the COM module.

7.10.5 Aggregation of external requested Partial Networks

Note: This feature is used by the Gateways to collect only the external PN requests. The external PN requests are mirrored back to the requesting bus and provided to other (required) physical channels of a central gateway. In case of a sub gateway the requests bit must not be mirrored back to the requesting physical channel in order to avoid static waking between central- and sub gateways. This logic shall be implemented by the ComM. The CanNm module provides the information if the PN is externally requested or not. The COM module is used for data transmission to the upper layer.
The aggregated state of the external requested PNs is called "External Requests Aggregated" (ERA).

CANNM439 If `CanNmPnEraCalcEnabled` is `FALSE` the CanNm module shall skip the aggregation of external PN requests.

CANNM440 If `CanNmPnEraCalcEnabled` is `TRUE` the CanNm module shall calculate the aggregation of external PN requests for each CanNm channel separately.

CANNM441 The ERA shall have a size of `CanNmNmPnInfoLength` and shall be initialized with value 0 (no PN requested).

CANNM442 The CanNm shall only consider the PN request bits in the NM PDU that are relevant for the ECU (defined by PN filter mask). All other PN request bits are ignored. Thus the ERA only contains those PN requests, which are relevant for the ECU.

CANNM443 If a NM PDU is received the CanNm module shall set every PN request bit in the ERA to 1 that has been requested by the PN request bits of the received NM PDU.

CANNM444 The CanNm module shall provide an ERA reset timer for every PN request bit for every physical CAN channel.

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

CANNM445 If a NM PDU is received the CanNm module shall restart the ERA reset timer for every PN request bit that is requested in the NM PDU with `CanNmPnResetTime`.

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

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

CANNM446 If one of the ERA reset timers expires, the CanNm module shall set the PN request bit of the corresponding PN in the ERA to 0.

CANNM447 If content of ERA changes (any bit changes from 1 to 0 or from 0 to 1) because of a received NM PDU or the ERA reset timer expiration the CanNm module shall inform the upper layers by calling `PduR_CanNmRxIndication()`. By means of the Rx Indication function the ERA data shall be provided to the COM module.

7.10.6 Spontaneous Transmission of NM PDUs via CanNm_NetworkRequest

CANNM448 If `CanNm_NetworkRequest` is called, `CanNmPnHandleMultipleNetworkRequest` is set to `TRUE` and CanNm is in Ready Sleep State, Normal Operation State or Repeat Message State, CanNm shall change to or restart the Repeat Message State.

CANNM449 If `CanNmPnHandleMultipleNetworkRequests` is set to `TRUE` the CanNm feature 'Immediate Transmission' is mandatory. It shall be ensured that `CanNmImmediateNmTransmissions > 0` is given.

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

7.11 Transmission Error Handling

CANNM073 The transmission error handling shall be inactive if `CANNM_PASSIVE_MODE_ENABLED` is enabled (see [CANNM72](#)) or `CANNM_IMMEDIATE_TXCONF_ENABLED` is enabled.

Rationale: Transmission error handling makes only sense if a node is allowed to transmit NM PDUs and the real confirmation from the CanIf is evaluated.

CANNM064 The NM Message Tx Timeout Timer shall be started with `CANNM_MSG_TIMEOUT_TIME` when the transmission of a NM PDU is requested.

CANNM065 The NM Message Tx Timeout Timer shall be stopped when `CanNm_TxConfirmation` is called by the CanIf.

CANNM205 The NM Message Tx Timeout Timer shall be stopped as soon as possible after each NM PDU transmit request if `CANNM_IMMEDIATE_TXCONF_ENABLED` is defined.

CANNM066 The function `Nm_TxTimeoutException` shall be called only once when the NM Message Tx Timeout Timer expires.

CANNM458 If `CanNmPnEnabled` is set to `TRUE` the function CanSM_TxTimeoutException shall be called once when the NM Message Tx Timeout Timer expires.

7.12 Message Structure

The figure below shows the default format of the NM PDU:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Byte 7	User data 5
Byte 6	User data 4
Byte 5	User data 3
Byte 4	User data 2
Byte 3	User data 1
Byte 2	User data 0
Byte 1	Control Bit Vector
Byte 0	Source Node Identifier

Figure 7-3   NM PDU Default Format

CANNM074: The location of the source node identifier shall be configurable by means of CANNM_PDU_NID_POSITION to Byte 0, Byte 1, or off (default: Byte 0).

CANNM075: The location of the control Bit vector shall be configurable by means of CANNM_PDU_CBV_POSITION to Byte 0, Byte 1, or off (default: Byte 1).

Note that when Source Node Identifier and/or Control Bit Vector are configured to off then they are not part of the NM PDU. This means that then one/two more byte(s) for user data are available.

CANNM076: The length of the NM PDU shall be configurable to any integer value between (and including) 0 and 8 by means of CANNM_PDU_LENGTH, the difference between number of enabled system bytes and length is user data.

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	Partial Network Information Bit	Res	Active Wakeup Bit	NM Coordinator Sleep Ready Bit	NM Coordinator ID (high bit)	NM Coordinator ID (low bit)	Repeat Message Request

Figure 7-4 Control Bit Vector

CANNM045: The Control Bit Vector shall consist of
Bit 0: Repeat Message Request
    0: Repeat Message State not requested
    1: Repeat Message State requested
Bit 1, 2: NM Coordinator ID (represents the priority of the coordinator)
    1: low
    2: medium
    3: high
Bit 3: NM Coordinator Sleep Ready Bit
    0: Start of synchronized shutdown is not requested by main coordinator
    1: Start of synchronized shutdown is requested by main coordinator
Bit 4 Active Wakeup Bit
    0: Node has not woken up the network (passive wakeup)
    1: Node has woken up the network (active Wakeup)
Bit 6 Partial Network Information Bit (PNI)
    0: NM PDU contains no Partial Network request information
    1: NM PDU contains Partial Network request information

Bit 3, 5, 7 are reserved for future extensions
0: Disabled / Reserved for future usage

Note: The Control Bit Vector is initialized with `0x00` during initialization (also refer to CANNM085).

CANNM013: The source node identifier shall be set with the configuration parameter `CANNM_NODE_ID` unless `CANNM_PDU_NID_POSITION` is set to off.

CANNM132: The optional service call `CanNm_GetNodeIdentifier` shall provide the node identifier out of the most recently received NM PDU.

CANNM133: The optional service call `CanNm_GetLocalNodeIdentifier` shall provide the node identifier configured for the local host node.

CANNM135: Support of Repeat Message Request Bit and Repeat Message State Request shall be statically configurable with use of the `CANNM_NODE_DETECTION_ENABLED` switch (configuration parameter).

CANNM136: The optional service call `CanNm_RepeatMessageRequest` shall be statically configured with use of the `CANNM_NODE_DETECTION_ENABLED` switch (configuration parameter).

CANNM137: If the optional service `CanNm_RepeatMessageRequest` is called in Repeat Message State, Prepare Bus Sleep Mode or Bus Sleep Mode, the service shall not be executed and `NM_E_NOT_EXECUTED` shall be returned.

CANNM138: The optional service call `CanNm_GetPduData` shall provide whole PDU data (Node ID, Control Bit Vector and User Data) of the most recently received NM PDU.

CANNM139: The optional service `CanNm_GetPduData` shall be statically configurable. It shall be available if `CANNM_NODE_ID_ENABLED` or `CANNM_NODE_DETECTION_ENABLED` or `CANNM_USER_DATA_ENABLED` is defined.

CANNM408 If the CanNm performs a state change from Bus Sleep Mode or Prepare Bus Sleep Mode to Network Mode due to a call to `CanNm_NetworkRequest` (i.e. due to an active wakeup) and `CanNmActiveWakeupBitEnabled` is `TRUE`, the CanNm shall set the ActiveWakeupBit in the CBV.

CANNM455 If the CanNm module leaves the Network Mode and CanNmActiveWakeupBitEnabled is TRUE, the CanNm module shall clear the ActiveWakeupBit in the CBV.

7.13 Functional requirements on CanNm API

CANNM014: If the node detection functionality is enabled, the function `Nm_RepeatMessageIndication` shall be called upon every reception of the RepeatMessageRequest bit if `CANNM_REPEAT_MSG_IND_ENABLED` is enabled.

CANNM086: If `CANNM_USER_DATA_ENABLED` is enabled and the available user data length (difference between CanNmPduLength and number of enabled system bytes) is `0` an error during configuration or compilation time shall be raised.

7.14 Error classification

CANNM018: The following errors shall be detectable by the CanNm depending on its build version (development/production mode).

Type or error	Relevance	Related error code	Error Value
API service used without module initialization	Development	CANNM_E_NO_INIT	0x01
API service called with wrong channel handle	Development	CANNM_E_INVALID_CHANNEL	0x02
CanNm initialization has failed, e.g. selected configuration set doesn't exist	Development	CANNM_E_INIT_FAILED	0x03
Reception of NM PDUs in Bus Sleep Mode.	Development	CANNM_E_NET_START_IND	0x04
Call of CanIf function `CanIf_Transmit` has failed	Development	CANNM_E_CANIF_TRANSMIT_ERROR	0x05
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 PDU cannot be transmitted in time; this race condition applies to event-triggered systems	Production	CANNM_E_NETWORK_TIMEOUT_CHANNEL_<X>	Assigned by 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	Development	CANNM_E_DEV_NETWORK_TIMEOUT	0x11

detection shortly before the NM-Timeout Timer expires so that a NM PDU cannot be transmitted in time; this race condition applies to event-triggered systems			
Null pointer has been passed as an argument (Does not apply to function `CanNm_Init`)	Development	`NM_E_NULL_POINTER`	`0x12`

CANNM019: Development errors shall be reported to the DET.

CANNM188: Reporting of development errors shall be statically configurable with use of the `CANNM_DEV_ERROR_DETECT` switch (configuration parameter).

CANNM189: 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.

CANNM020: Production errors shall be reported to the DEM.

CANNM190: 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.

CANNM191: If not initialized, the NM shall reject every API service apart from `CanNm_Init`; the called function shall not be executed, but instead of that it shall report `NM_E_NO_INIT` to the Development Error Tracer and it shall return `NM_E_NOT_OK` to the calling function

CANNM192: 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 `NM_E_INVALID_CHANNEL` to the Development Error Tracer and it shall return `NM_E_NOT_OK` to the calling function.

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

CANNM193: When the NM-Timeout Timer expires in the Repeat Message State, the NM shall report `NM_E_NETWORK_TIMEOUT` to Diagnostic Event Manager

CANNM194: When the NM-Timeout Timer expires in the Normal Operation State, the NM shall report `NM_E_NETWORK_TIMEOUT` to Diagnostic Event Manager

7.15 Scheduling of the main function

CANNM077: The `CanNm_MainFunction_<CanNm Channel ID>` functions shall be scheduled by the BSW scheduler (see [8]).

7.16 Application notes

7.16.1 Wakeup notification

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

7.16.2 Coordination of coupled networks

The AUTOSAR CAN NM supports coordination of coupled networks with the following optional services:

- Bus synchronization on demand: Bus synchronization on demand allows synchronization of a NM-cluster for an arbitrary point of time; in result NM-Timeout Timers in all nodes of the NM-cluster are restarted.

CANNM185: Support of bus synchronization on demand shall be statically configurable with use of the `CANNM_BUS_SYNCHRONIZATION_ENABLED` switch (configuration parameter).

CANNM187: If `CanNm_RequestBusSynchronization` is called in Bus Sleep Mode and Prepare Bus Sleep Mode the service shall not be executed and `NM_E_NOT_EXECUTED` shall be returned.

7.16.3 Interoperability with direct OSEK NM

Interoperability with direct OSEK NM is defined in detail in the network management interface specification.

8 API specification

CANNM016: Parameter value check shall be provided only in "development mode". The execution of a service shall be rejected and the API shall inform the DET.

CANNM195: AUTOSAR CAN NM API consists of services, which are CAN specific and can be called whenever they are required; each service apart from CanNm_Init refers to one NM channel only.

8.1 Imported Types

Module	Imported Type
ComStack_Types	NetworkHandleType
	PduIdType
	PduInfoType
Dem	Dem_EventIdType
Nm	Nm_ModeType
	Nm_ReturnType
	Nm_StateType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type Definitions

8.2.1 CanNm_ConfigType

Name:	CanNm_ConfigType
Type:	Structure
Range:	Implementation specific.
Description:	This type shall contain at least all parameters that are post-build able according to chapter 10.

8.3 CanNm Functions

8.3.1 CanNm_Init

Service name:	CanNm_Init	

Syntax:	`void CanNm_Init(`
	`    const CanNm_ConfigType* const cannmConfigPtr`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	cannmConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	

Description:	Initialize the complete CanNm module, i.e. all channels which are activated (see also configuration parameter CANNM_CHANNEL_ACTIVE) at configuration time are initialized.
	If a NULL pointer is passed as an argument to this function the default configuration shall be used.
	Caveats: This function has to be called after initialization of the CanIf.
	Configuration: Mandatory

8.3.2 CanNm_PassiveStartUp

Service name:	CanNm_PassiveStartUp	
Syntax:	Nm_ReturnType CanNm_PassiveStartUp(const NetworkHandleType nmChannelHandle)	
Service ID[hex]:	0x01	
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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Passive startup of network management has failed NM_E_NOT_EXECUTED: Passive startup of network management is currently not executed
Description:	Passive startup of the AUTOSAR CAN NM. It triggers the transition from Bus-Sleep Mode or Prepare Bus Sleep Mode to the Network Mode in Repeat Message State. This service has no effect if the current state is not equal to Bus-Sleep Mode or Prepare Bus Sleep Mode. In that case NM_E_NOT_EXECUTED is returned. Caveats: CanNm is initialized correctly. Configuration: Mandatory	

8.3.3 CanNm_NetworkRequest

Service name:	CanNm_NetworkRequest	
Syntax:	Nm_ReturnType CanNm_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:	Nm_ReturnType	NM_E_OK: No error NM_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: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_PASSIVE_MODE_ENABLED is not defined).

8.3.4 CanNm_NetworkRelease

Service name:	CanNm_NetworkRelease	

Syntax:	`Nm_ReturnType CanNm_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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Releasing of network has failed NM_E_NOT_EXECUTED: Releasing of network is currently not executed.
Description:	Release the network, since ECU doesn't have to communicate on the bus. Network state shall be changed to 'released'. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_PASSIVE_MODE_ENABLED is not defined).	

8.3.5 CanNm_DisableCommunication

Service name:	CanNm_DisableCommunication

Syntax:	`Nm_ReturnType CanNm_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:	Nm_ReturnType · NM_E_OK: No error NM_E_NOT_OK: Disabling of NM PDU transmission ability has failed. NM_E_NOT_EXECUTED: Disabling of NM PDU transmission ability is not executed.
Description:	Disable the NM PDU transmission ability. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_COM_CONTROL_ENABLED is defined).

8.3.6 CanNm_EnableCommunication

Service name:	CanNm_EnableCommunication

Syntax:	`Nm_ReturnType CanNm_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:	Nm_ReturnType · NM_E_OK: No error NM_E_NOT_OK: Enabling of NM PDU transmission ability has failed. NM_E_NOT_EXECUTED: Enabling of NM PDU transmission ability is not executed.
Description:	Enable the NM PDU transmission ability. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_COM_CONTROL_ENABLED is defined).

8.3.7 CanNm_SetUserData

Service name:	CanNm_SetUserData

Syntax:	`Nm_ReturnType CanNm_SetUserData(`
	`    const NetworkHandleType nmChannelHandle,`
	`    const uint8* const 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 PDU shall be copied from
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Setting of user data has failed
Description:	Set user data for NM PDUs transmitted next on the bus. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_USER_DATA_ENABLED is defined and CANNM_PASSIVE_MODE_ENABLED is not defined).	

8.3.8 CanNm_GetUserData

Service name:	CanNm_GetUserData	

Syntax:	`Nm_ReturnType CanNm_GetUserData(`
	`    const NetworkHandleType nmChannelHandle,`
	`    uint8* const nmUserDataPtr`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	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 PDU shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of user data has failed
Description:	Get user data out of the most recently received NM PDU. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_USER_DATA_ENABLED is defined).	

8.3.9 CanNm_GetNodeIdentifier

Service name:	CanNm_GetNodeIdentifier	

Syntax:	`Nm_ReturnType CanNm_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 node identifier out of the most recently received NM PDU shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of the node identifier out of the most recently received NM PDU has failed
Description:	Get node identifier out of the most recently received NM PDU. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_ID_ENABLED is defined).	

8.3.10 CanNm_GetLocalNodeIdentifier

Service name:	CanNm_GetLocalNodeIdentifier	

Syntax:	`Nm_ReturnType CanNm_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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of the node identifier of the local node has failed
Description:	Get node identifier configured for the local node. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_ID_ENABLED is defined).	

8.3.11 CanNm_RepeatMessageRequest

Service name:	CanNm_RepeatMessageRequest	

Syntax:	`Nm_ReturnType CanNm_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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Setting of Repeat Message Request Bit has failed NM_E_NOT_EXECUTED: Repeat Message Request is currently not executed.
Description:	Set Repeat Message Request Bit for NM PDUs transmitted next on the bus.	

	Caveats: CanNm is initialized correctly.
	Configuration: Optional (Only available if CANNM_NODE_DETECTION_ENABLED is defined).

8.3.12 CanNm_GetPduData

Service name:	CanNm_GetPduData	

Syntax:	Nm_ReturnType CanNm_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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK:Getting of NM PDU data has failed
Description:	Get the whole PDU data out of the most recently received NM PDU. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_ID_ENABLED or CANNM_NODE_DETECTION_ENABLED or CANNM_USER_DATA_ENABLED is defined).	

8.3.13 CanNm_GetState

Service name:	CanNm_GetState	

Syntax:	Nm_ReturnType CanNm_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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of NM state has failed
Description:	Returns the state and the mode of the network management. Caveats: CanNm is initialized correctly. Configuration: Mandatory	

8.3.14 CanNm_GetVersionInfo

Service name:	CanNm_GetVersionInfo

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

Service ID[hex]:	0xf1
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 CanNm is not inialized. Configuration: Optional (only available if CANNM_VERSION_INFO_API is defined).

8.3.15 CanNm_RequestBusSynchronization

Service name:	CanNm_RequestBusSynchronization

Syntax:	`Nm_ReturnType CanNm_RequestBusSynchronization(`
	`    const NetworkHandleType nmChannelHandle`
	`)`

Service ID[hex]:	0xc0
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	nmChannelHandle — Identification of the NM-channel
Parameters (inout):	None
Parameters (out):	None
Return value:	Nm_ReturnType — NM_E_OK : No error NM_E_NOT_OK: Requesting of bus synchronization has failed NM_E_NOT_EXECUTED: Bus synchronization is currently not executed.
Description:	Request bus synchronization. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_BUS_SYNCHRONIZATION_ENABLED is defined) and CANNM_PASSIVE_MODE_ENABLED is not defined.

8.3.16 CanNm_CheckRemoteSleepIndication

Service name:	CanNm_CheckRemoteSleepIndication
Syntax:	`Nm_ReturnType CanNm_CheckRemoteSleepIndication(`

	const NetworkHandleType nmChannelHandle, boolean* const nmRemoteSleepIndPtr)	
Service ID[hex]:	0xd0	
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:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Checking of remote sleep indication bits has failed NM_E_NOT_EXECUTED: Checking of Remote Sleep Indication is currently not executed.
Description:	Check if remote sleep indication takes place or not. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_REMOTE_SLEEP_INDICATION_ENABLED is defined).	

8.3.17 CanNm_Transmit

Service name:	CanNm_Transmit	

Syntax:	Std_ReturnType CanNm_Transmit(
	    PduIdType CanNmTxPduId,
	    const PduInfoType* PduInfoPtr
	)	

Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	CanNmTxPduId	Handle of the CanNm user data PDU.
	PduInfoPtr	Pointer to a structure containing the pointer to the NM user data buffer and the related DLC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK Transmit request has been accepted E_NOT_OK Transmit request has not been accepted (CanNm is not in RM or NO)
Description:	This function is used by the PduR to trigger a spontaneous transmission of an NM PDU with the provided NM User Data. **Configuration:** Triggers the transmission of a CanNm PDU	

CANNM404 If `CanNmComUserDataSupport` is enabled the CanNm implementation shall provide an API `CanNm_Transmit`.

CANNM453 Spontaneous transmission of a NM PDU via `CanNm_Transmit` is required in the following configuration setups:

- `CanNmPnEnabled` set to `TRUE` and `CanNmPnHandleMultipleNetworkRequests` is set to `FALSE`. (PN feature enabled but spontaneous transmission of NM PDUs not done via `CanNm_NetworkRequest`).

- `CanNmCarWakeUpRxEnabled` set to `TRUE` (Car WakeUp feature enabled)

CANNM405 `CanNm_Transmit` is an empty function returning `E_OK` at any time if the spontaneous transmission of a NM PDUs via `CanNm_Transmit` is not required. This requirement is relevant to avoid linker errors as PduR expects this API to be provided.

CANNM450 If `CanNm_Transmit` is called and CanNm is in Normal Operation or Repeat Message State, CanNm shall return `E_OK` and send a spontaneous NM PDU with the next CanNm main function if spontaneous transmission of NM PDUs via `CanNm_Transmit` is required.

CANNM451 After the transmission of the spontaneous NM PDU, the CanNm module shall restart the Message Cycle Timer with the corresponding timer value (e.g. `CanNmMsgCycleTime`).

CANNM452 If `CanNm_Transmit` is called, the spontaneous transmission of NM PDUs via `CanNm_Transmit` is required and CanNm is not in Normal Operation or Repeat Message State, CanNm shall ignore the request and return `E_NOT_OK`.

8.3.18 CanNm_SetCoordBits

Service name:	CanNm_SetCoordBits

Syntax:	`Nm_ReturnType CanNm_SetCoordBits(`
	`    const NetworkHandleType nmChannelHandle,`
	`    const uint8 nmCoordBits`
	`)`

Service ID[hex]:	0x15
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	nmChannelHandle / Identification of the NM-channel
	nmCoordBits / Priority of the NM coordinator
Parameters (inout):	None
Parameters (out):	None
Return value:	Nm_ReturnType / NM_E_OK: No error NM_E_NOT_OK: Checking of remote sleep indication bits has failed NM_E_NOT_EXECUTED: Checking of Remote Sleep Indication is currently not executed.
Description:	Set the two bits in the Control Bit Vector which represent the NM coordinator ID.

8.3.19 CanNm_SetSleepReadyBit

Service name:	CanNm_SetSleepReadyBit

Syntax:	`Nm_ReturnType CanNm_SetSleepReadyBit(`
	`    const NetworkHandleType nmChannelHandle,`
	`    const boolean nmSleepReadyBit`
	`)`

Service ID[hex]:	0x17
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:	Nm_ReturnType	NM_E_OK: No error NM_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 CanNm functions called by the CanIf

8.4.1 CanNm_TxConfirmation

Service name:	CanNm_TxConfirmation

Syntax:	`void CanNm_TxConfirmation(`
	`    PduIdType canNmTxPduId`
	`)`

Service ID[hex]:	0x0f
Sync/Async:	Synchronous
Reentrancy:	Reentrant (but not within the same channel)
Parameters (in):	canNmTxPduId	Identification of the network through PDU-ID
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This service confirms a previous successfully processed CAN transmit request. This callback service is called by the CanIf and implemented by the CanNm. The value passed to CanNm via the API parameter canNmTxPduId shall refer to the NM channel handle, i.e. a mapping from PduId to NM channel handle is not necessary. Inform the DET (if enabled) if function call has failed because of the following reasons: - Invalid channel handle (CANNM_E_INVALID_CHANNEL) - CanNm was not initialized (CANNM_E_NO_INIT) Caveats: The call context is either on interrupt level (interrupt mode) or on task level (polling mode). This callback service is re-entrant for multiple CAN controller usage. The CanNm module is initialized correctly. Configuration: Optional (Only available if CANNM_IMMEDIATE_TXCONF_ENABLED is defined).

8.4.2 CanNm_RxIndication

Service name:	CanNm_RxIndication

Syntax:	`void CanNm_RxIndication(`
	`    PduIdType canNmRxPduId,`
	`    const PduInfoType* PduInfoPtr`
	`)`

Service ID[hex]:	0x10
Sync/Async:	Synchronous
Reentrancy:	Reentrant (but not within the same channel)

	canNmRxPduId	Identification of the network through PDU-ID
Parameters (in):	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service indicates a successful reception of a received NM PDU to the CanNm after passing all filters and validation checks. This callback service is called by the CAN Interface and implemented by the CanNm. It is called in case of a receive indication event (i.e. ISR is triggered) of the CAN driver. The value passed to CanNm via the API parameter canNmRxPduId shall refer to the NM channel handle, i.e. a mapping from PduId to NM channel handle is not necessary. Inform the DET (if enabled) if function call has failed because of the following reasons: - Invalid channel handle (CANNM_E_INVALID_CHANNEL) - CanNm was not initialized (CANNM_E_NO_INIT) Caveats: Until this service returns the CAN Interface will not access PduInfoPtr. PduInfoPtr is only valid and can be used by upper layers until the indication returns. CAN Interface guarantees that the number of configured bytes for this canNmRxPduId is valid. The call context is either on interrupt level (interrupt mode) or on task level (polling mode). This callback service is re-entrant for multiple CAN controller usage. The CanNm module is initialized correctly. Configuration: Mandatory

8.5 Called by CanSm

8.5.1 CanNm_ConfirmPnAvailability

Service name:	CanNm_ConfirmPnAvailability
Syntax:	```void CanNm_ConfirmPnAvailability(const NetworkHandleType nmChannelHandle)```
Service ID[hex]:	0x16
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):	None	
Return value:	None	
Description:	Enables the PN filter functionality on the indicated NM channel. Availability: The API is only available if CanNmPnEnabled is TRUE.

8.6 Scheduled Functions

8.6.1 CanNm_MainFunction_<CanNm Channel ID>

Service name:	CanNm_MainFunction_<CanNm Channel ID>
Syntax:	`void CanNm_MainFunction_<CanNm Channel ID>(` `)`
Service ID[hex]:	0x13
Timing:	FIXED_CYCLIC
Description:	Main function of the CanNm which processes the algorithm describes in that document.

e.g.
CanNm_MainFunction_0() is the main function for CanNm channel 0
CanNm_MainFunction_1() is the main function for CanNm channel 1
...

Inform the DET (if enabled) if function call has failed because of the following reasons:
- CanNm was not initialized (CANNM_E_INVALID_CHANNEL)

Caveats: CanNm 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.

CANNM324:

API function	Description
Dem_ReportErrorStatus	Reports errors to the DEM.
Nm_BusSleepMode	Notification that the network management has entered Bus-Sleep Mode.
Nm_NetworkMode	Notification that the network management has entered Network Mode. The callback function shall enable transmission of application messages.
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. The callback function shall start the network management state machine.
Nm_PrepareBusSleepMode	Notification that the network management has entered Prepare Bus-Sleep Mode. The callback function shall disable transmission of application messages.

8.7.2 Optional Interfaces

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

CANNM325:

API function	Description
CanIf_Transmit	CANIF005: This service initiates a request for transmission of the CAN L-PDU specified by the CanTxPduId and CAN related data in the L-PDU structure. The corresponding CAN controller and HTH have to be resolved by the CanTxPduId. A transmit request has not been accepted, if the controller mode is not STARTED and/or the channel mode at least for the transmit path is not online or offline active. One call of this function results in one call of Can_Write(Hth, *PduInfo). Development errors: Invalid values of CanTxPduId or PduInfoPtr will be reported to the development error tracer (CANIF_E_INVALID_TXPDUID or CANIF_E_PARAM_POINTER). If the CAN Interface was not initialized before, the call of this function will be reported to the development error tracer (CANIF_E_UNINIT). The function returns with E_NOT_OK. Caveats: During the call of this API the buffer of PduInfoPtr is controlled by the CAN Interface may not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer. The CAN Interface must be initialized after Power ON.
CanSM_TxTimeoutException	This function shall notify the CanSM module, that the CanNm has detected for the affected partial CAN network a tx timeout exception, which shall be recovered by the CanSM module with a transition to no communication and back to the requested communication mode again.
Det_ReportError	Service to report development errors.
Nm_ActiveCoordIndication	Set the two bits in the Control Bit Vector which represent the NM coordinator ID.
Nm_CarWakeUpIndication	This function is called by a <Bus>Nm to indicate reception of a CWU request.
Nm_CoordReadyToSleepCancellation	Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0.
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 any longer ready to enter Bus-Sleep Mode.
Nm_RemoteSleepIndication	Notification that the network management has detected that all other nodes are ready to sleep. The NM gateway shall check if the Bus is still required.
Nm_RepeatMessageIndication	Service to indicate that a 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_CanNmRxIndication	Reports the EIRA and ERA to the COM
PduR_CanNmTriggerTransmit	Triggers the transmission of a CanNm Message. Request for the

	User Data content of the NM message be sent.
PduR_CanNmTxConfirmation	Tx confirmation for the CanNm

8.8 Parameter check

CANNM196: If detection of development errors is enabled by
`CANNM_DEV_ERROR_DETECT` (configuration parameter), then for all CAN NM API
services validity check of input parameters shall be made.
Exception: The NULL Pointer check of input parameters shall not be done for
`CanNm_Init`

CANNM197: Parameter type checking shall be made at compile time; if types do not
fit the compilation process shall be stopped and respective compilation warnings or
errors shall be returned as far as supported by the compiler.

CANNM198: Parameter value check (for parameters of the constant value) shall be
made at configuration time; if the value is invalid, the configuration process shall be
stopped and respective configuration error shall be reported.

CANNM199: Parameter value check (for parameters of the variable value) shall be
made at execution time; if the value is invalid, execution of a service shall be rejected
and respective development error shall be reported.

8.9 Version check

CANNM200: NM shall check at pre-compile time if the version numbers of C- and H-
files are identical.

8.10 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. Additionally it is assumed that optional busload reduction functionality is enabled.
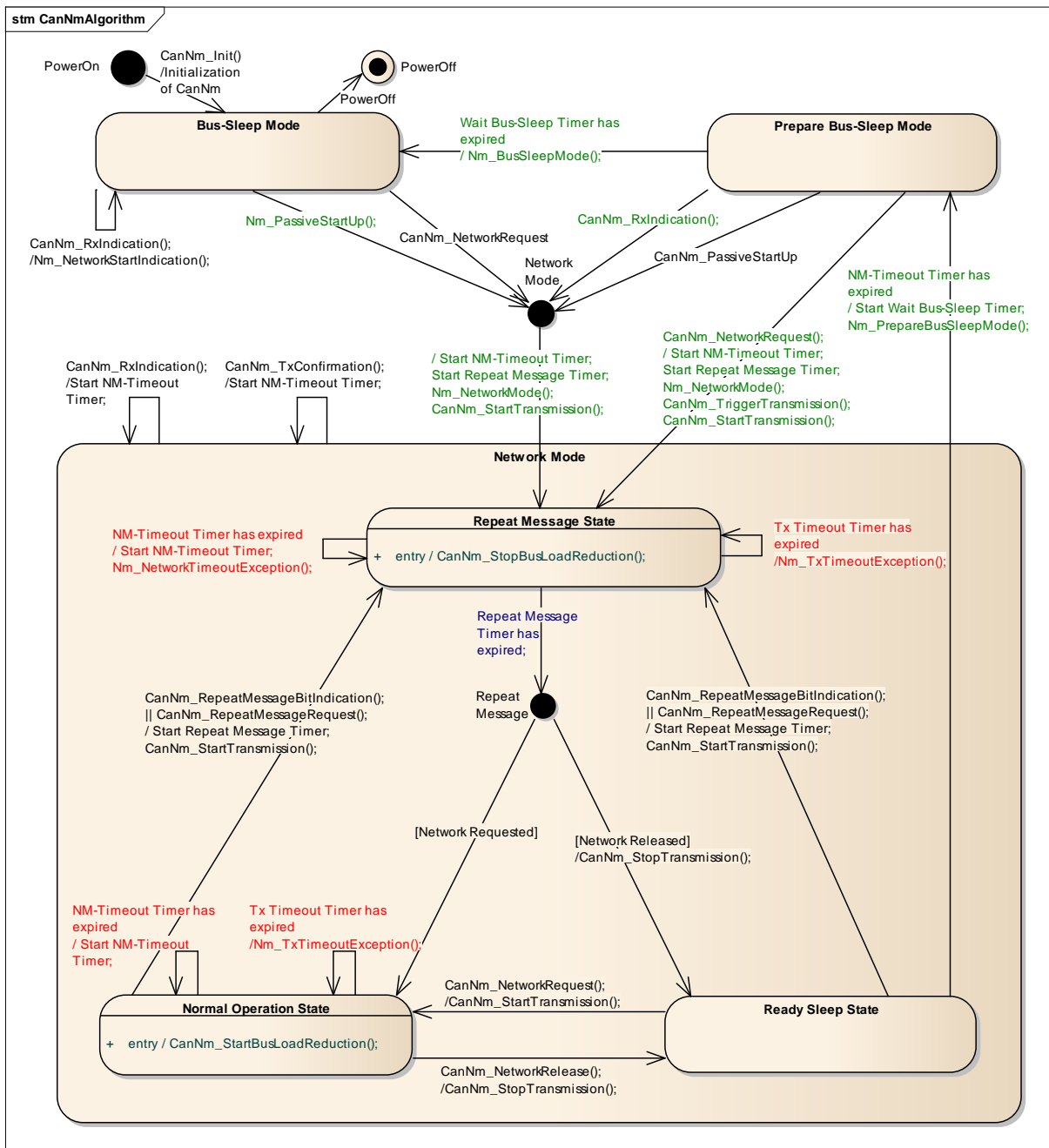
Figure 8-1 CanNm Algorithm

9 Sequence diagrams

9.1 CanNm Transmission
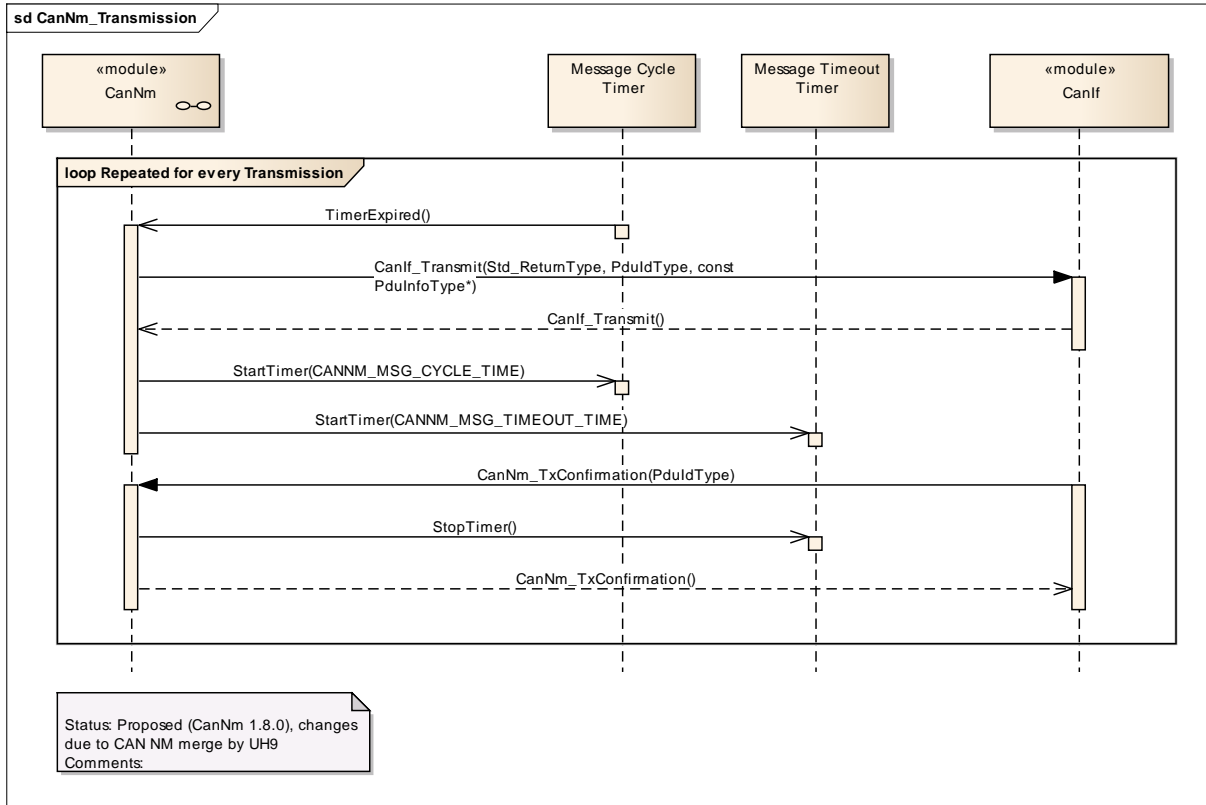
9.2 CanNm Reception
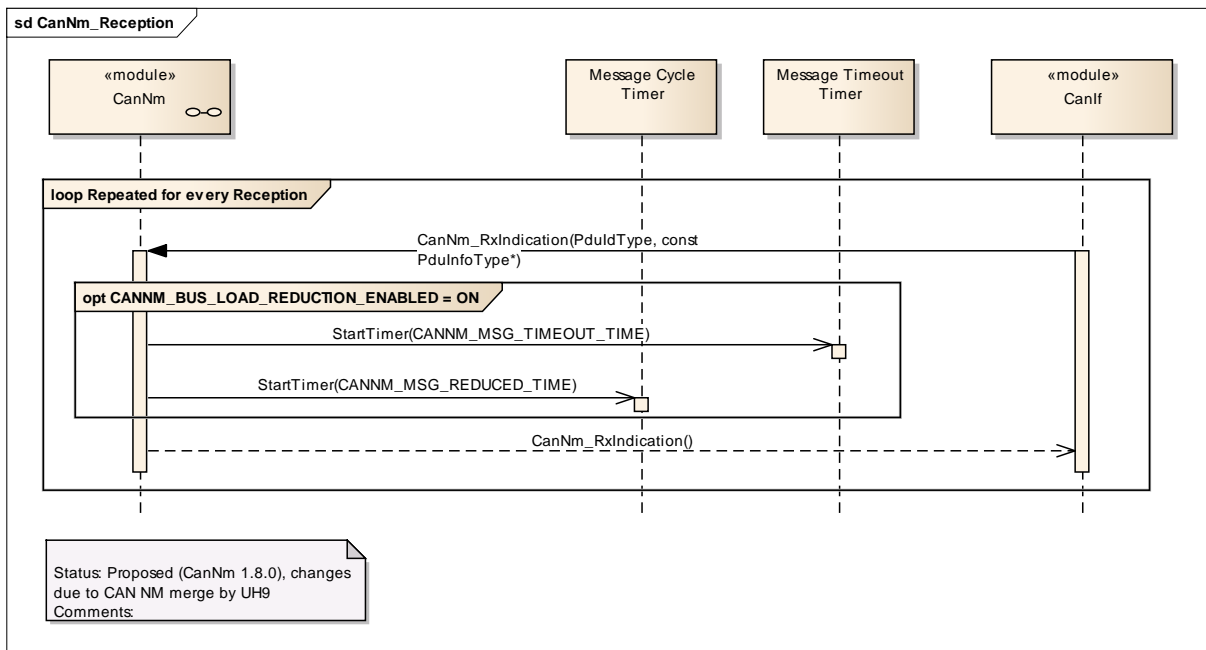
9.3 Car Wakeup Handling

act CAN CWU indication handling

CanNm_RxIndication ●

copy data to internal buffer

CanNmCwuRxEnabled

[false]

[true]

CWU bit in NM message == 1

[false]

[true]

CanNmCarWakeUpFilterNodeId
== received Node ID

CanNmCarWakeUpFilterEnabled

[true]

[false]

[false]

[true]

call Nm_CarWakeUpIndication()

Standard RX indication handling

end

Document ID 013:AUTOSAR_SWS_CAN_NM
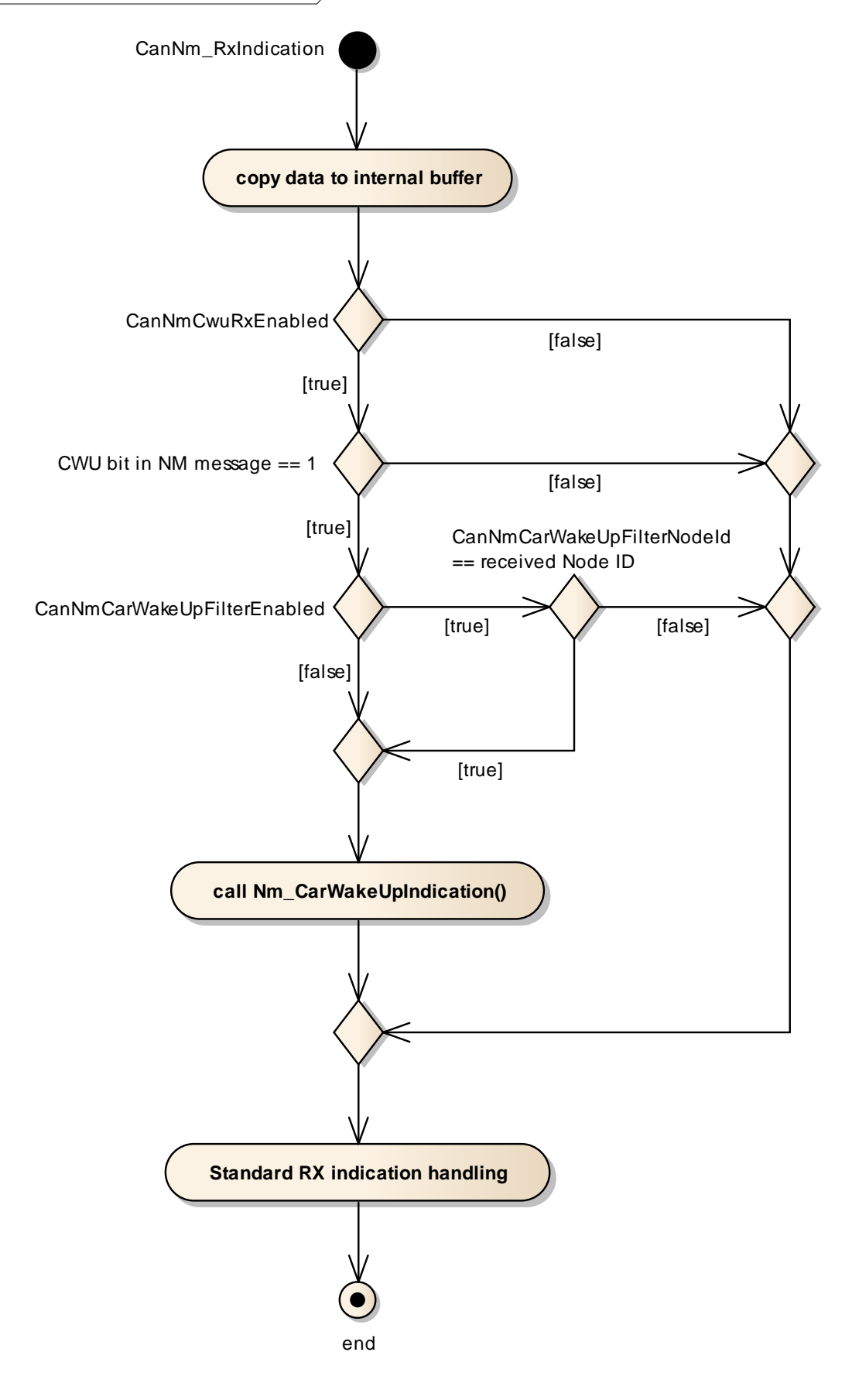
10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module CanNm.

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 metamodel in detail.

The following is only 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 channels 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 in parameters which are used to enable features, parameters which affect all channels of the CanNm and parameters which affect the respective channels of the CanNm.

CANNM026: All configuration items shall be located outside the kernel of the module.

10.2.1 Variants

Variant 1: All configuration parameters shall be configurable at pre-compile time.
Use case: Source code optimizations

Variant 2: All configuration parameters of the container `CanNm_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 `CanNm_ChannelConfig` are configurable at post-build time. The parameters contained in `CanNm_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 only explicitly mentioned for Variant 3, but from a technical perspective it is also possible to provide this configuration variant for variant 1 and 2.

10.3.2 CanNm_GlobalConfig General

The Global Scope specifies configuration parameter that is defined in the module's configuration header file CanNm_Cfg.h.

10.3.3 CanNmGlobalConfig

SWS Item	:
Container Name	CanNmGlobalConfig{CanNm_GlobalConfig} [Multi Config Container]
Description	This container contains the global configuration parameter of the CanNm. The parameters and the parameters of the sub containers shall be mapped to the C data type CanNm_ConfigType (for parameters where it is possible) which is passed to the CanNm_Init function. This container is a MultipleConfigurationContainer (only for variant 3), i.e. this container and its sub-containers exit once per configuration set.
Configuration Parameters	

SWS Item	:		
Name	CanNmActiveWakeupBitEnabled {CANNM_ACTIVE_WAKEUP_BIT_ENABLED}		
Description	Please note that this parameter is obsolete and replaced by the CanNmActiveWakeupBitEnabled parameter in the CanNmChannelConfig container. Enables/Disables the handling of the Active Wakeup Bit in the CanNm module.		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	CanNmBusLoadReductionEnabled {CANNM_BUS_LOAD_REDUCTION_ENABLED}		
Description	Pre-processor switch for enabling busload reduction support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: CanNmBusLoadReductionEnabled = false if CanNmPassiveModeEnabled == true or CanNmPnEnabled == true for at least one CanNm Channel.		

SWS Item	:
Name	CanNmBusSynchronizationEnabled {CANNM_BUS_SYNCHRONIZATION_ENABLED}
Description	Pre-processor switch for enabling bus synchronization support.
Multiplicity	1
Type	DerivedBooleanParamDef
Default value	--

ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: This feature is required for gateway nodes only. calculationFormula = If (CanNmPassiveModeEnabled == False) then Equal(NmBusSynchronizationEnabled) else Equal(False)		

SWS Item	:		
Name	CanNmComControlEnabled {CANNM_COM_CONTROL_ENABLED}		
Description	Pre-processor switch for enabling the Communication Control support.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: calculationformula = Equal(NmComControlEnabled)		

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

SWS Item	:		
Name	CanNmConfigPtr {CANNM_CANNM_CONFIG_PTR}		
Description	Pointer to configuration of CanNm. This parameter is set to obsolete and will be removed in future.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:
Name	CanNmCoordinatorSyncSupport {CANNM_COORDINATOR_SYNC_SUPPORT}
Description	Enables/disables the coordinator synchronisation support.
Multiplicity	1
Type	BooleanParamDef

Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

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

SWS Item	:		
Name	CanNmImmediateRestartEnabled {CANNM_IMMEDIATE_RESTART_ENABLED}		
Description	Pre-processor switch for enabling the asynchronous transmission of a NM PDU upon bus-communication request in Prepare-Bus-Sleep mode.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Must not be defined if CanNmPassiveModeEnabled is defined.		

SWS Item	:		
Name	CanNmImmediateTxconfEnabled {CANNM_IMMEDIATE_TXCONF_ENABLED}		
Description	Enable/disable the immediate tx confirmation.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: CanNmImmediateTxconfEnabled shall not be enabled if CanNmPasiveModeEnabled is enabled.		

SWS Item	:		
Name	CanNmNodeDetectionEnabled {CANNM_NODE_DETECTION_ENABLED}		
Description	Pre-processor switch for enabling the node detection support.		
Multiplicity	1		

Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: If(CanNmNodeIdEnabled == True) then Equal(NmNodeDetectionEnabled) else Equal(False)		

SWS Item	:		
Name	CanNmNodeIdEnabled {CANNM_NODE_ID_ENABLED}		
Description	Pre-processor switch for enabling the source node identifier.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Network dependency: calculationFormula = Equal(NmNodeIdEnabled)		

SWS Item	:		
Name	CanNmNumberOfChannels {CANNM_NUMBER_OF_CHANNELS}		
Description	Number of Can NM channels allowed within one ECU. Please note that this parameter is deprecated and will be removed in future.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	CanNmPassiveModeEnabled {CANNM_PASSIVE_MODE_ENABLED}		
Description	Pre-processor switch for enabling support of the Passive Mode.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: calculationFormula = Equal(NmPassiveModeEnabled)		

SWS Item	:		
Name	CanNmPduRxIndicationEnabled {CANNM_PDU_RX_INDICATION_ENABLED}		
Description	Pre-processor switch for enabling the PDU Rx Indication.		
Multiplicity	1		

Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: calculationFormula = Equal(NmPduRxIndicationEnabled)		

SWS Item	CANNM070_Conf :		
Name	CanNmPnEiraCalcEnabled		
Description	Specifies if CanNm calculates the PN request information for internal an external requests. (EIRA) true: PN request are calculated false: PN request are not calculated		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel		

SWS Item	CANNM059_Conf :		
Name	CanNmPnResetTime		
Description	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA. The value shall be the same for every channel. Thus it is a global config parameter.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	0.001 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel.		

SWS Item	:		
Name	CanNmRemoteSleepIndEnabled {CANNM_REMOTE_SLEEP_IND_ENABLED}		
Description	Pre-processor switch for enabling remote sleep indication support.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: This feature is required for gateway nodes only.		

	calculationFormula = If (CanNmPassiveModeEnabled == False) then Equal(NmRemoteSleepIndEnabled) else Equal(False)

SWS Item	:		
Name	CanNmRepeatMsgIndEnabled {CANNM_REPEAT_MSG_IND_ENABLED}		
Description	Enable/disable the notification that a RepeatMessageRequest bit has been received.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: calculationformula = Equal(NmRepeatMsgIndEnabled)		

SWS Item	:		
Name	CanNmStateChangeIndEnabled {CANNM_STATE_CHANGE_IND_ENABLED}		
Description	Pre-processor switch for enabling the CAN NM state change notification.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: calculationFormula = Equal(NmStateChangeIdEnabled)		

SWS Item	:		
Name	CanNmUserDataEnabled {CANNM_USER_DATA_ENABLED}		
Description	Pre-processor switch for enabling user data support.		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Network dependency: calculationFormula = Equal(NmUserDataEnabled)		

SWS Item	:		
Name	CanNmVersionInfoApi {CANNM_VERSION_INFO_API}		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	

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

SWS Item	CANNM058_Conf :		
Name	CanNmPnEiraRxNSduRef		
Description	Reference to a Pdu in the COM-Stack. Only one SduRef is required for CanNm because the EIRA is the aggregation over all Can Channels.		
Multiplicity	0..1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel.		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
CanNmChannelConfig	1..*	This container contains the channel specific configuration parameter of the CanNm.
CanNmPnInfo	0..1	PN information configuration

10.3.4 CanNm_ChannelConfig general

CANNM202: The container `CanNm_ChannelConfig` specifies configuration parameter that shall be located in a data structure of type `CanNm_ConfigType`.

CANNM203: Runtime configurable parameters listed below shall be configurable for each NM-cluster separately.

10.3.5 CanNmChannelConfig

SWS Item	:
Container Name	CanNmChannelConfig{CanNm_ChannelConfig}
Description	This container contains the channel specific configuration parameter of the CanNm.
Configuration Parameters	

SWS Item	:		
Name	CanNmActiveWakeupBitEnabled {CANNM_ACTIVE_WAKEUP_BIT_ENABLED}		
Description	Enables/Disables the handling of the Active Wakeup Bit in the CanNm module.		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local dependency: This parameter is only valid if CanNmPassiveModeEnabled is False.		

SWS Item	**CANNM068_Conf :**		
Name	CanNmAllNmMessagesKeepAwake		
Description	Specifies if CanNm drops irrelevant NM PDUs. false: Only NM PDUs with an with PNI bit = true and containing an PN request for this ECU triggers the standard RX indication handling true: Every NM PDU triggers the standard RX indication handling		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true		

SWS Item	:
Name	CanNmBusLoadReductionActive {CANNM_BUS_LOAD_REDUCTION_ACTIVE}
Description	This parameter defines if bus load reduction for the respective NM channel is active or not.
Multiplicity	1
Type	BooleanParamDef

Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel dependency: Shall only be True if CanNmBusLoadReductionEnabled is True.		

SWS Item	:		
Name	CanNmCarWakeUpBitPosition {CANNM_CAR_WAKE_UP_BIT_POSITION}		
Description	Specifies the Bit position of the CWU within the NM PDU.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 7		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmCarWakeUpRxEnabled == TRUE		

SWS Item	:		
Name	CanNmCarWakeUpBytePosition {CANNM_CAR_WAKE_UP_BYTE_POSITION}		
Description	Specifies the Byte position of the CWU within the NM PDU.		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 7		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmCarWakeUpRxEnabled == TRUE		

SWS Item	:		
Name	CanNmCarWakeUpFilterEnabled {CANNM_CAR_WAKE_UP_FILTER_ENABLED}		
Description	If CWU filtering is supported, only the CWU bit within the NM PDU with source node identifier CanNmCarWakeUpFilterNodeId is considered as CWU request. FALSE - CWU filtering is not supported TRUE - CWU filtering is supported		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmCarWakeUpRxEnabled == TRUE		

SWS Item	:

Name	CanNmCarWakeUpFilterNodeId {CANNM_CAR_WAKE_UP_FILTER_NODE_ID}		
Description	Source node identifier for CWU filtering. If CWU filtering is supported, only the CWU bit within the NM PDU with source node identifier CanNmCarWakeUpFilterNodeId is considered as CWU request.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmCarWakeUpRxEnabled == TRUE		

SWS Item	:		
Name	CanNmCarWakeUpRxEnabled {CANNM_CAR_WAKE_UP_RX_ENABLED}		
Description	Enables or disables support of CarWakeUp bit evaluation in received NM PDUs. FALSE - CarWakeUp not supported TRUE - CarWakeUp supported		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	CanNmChannelActive {CANNM_CHANNEL_ACTIVE}		
Description	Please note that this parameter is obsolete and will be removed in future revisions. It determines if the respective NM channel is active or not. Indicates whether a particular NM-channel shall be initialized (TRUE) or not (FALSE). If this parameter is set to FALSE the respective CanNm channel shall not be used during runtime.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel		

SWS Item	:		
Name	CanNmImmediateNmCycleTime {CANNM_IMMEDIATE_NM_CYCLETIME}		
Description	Defines the immediate NM PDU cycle time in seconds which is used for CanNmImmediateNmTransmissions NM PDU transmissions.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	0.001 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Network dependency: This parameter is only valid if CanNmImmediateNmTransmissions is greater one.		

SWS Item	:		
Name	CanNmImmediateNmTransmissions {CANNM_IMMEDIATE_NM_TRANSMISSIONS}		
Description	Defines the number of immediate NM PDUs which shall be transmitted. If the value is zero no immediate NM PDUs are transmitted. The cycle time of immeditate NM PDUs is defined by CanNmImmediateNmCycleTime.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Network		

SWS Item	:		
Name	CanNmMainFunctionPeriod {CANNM_MAIN_FUNCTION_PERIOD}		
Description	Call cycle in seconds of CanNm_MainFunction_x for the respective CanNm channel.		
Multiplicity	1		
Type	FloatParamDef		
Range	0.001 .. 0.255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel		

SWS Item	:		
Name	CanNmMsgCycleOffset {CANNM_MSG_CYCLE_OFFSET}		
Description	Time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel dependency: This parameter is only valid if CanNmPassiveModeEnabled is False. Parameter value < CanMsgCycleTime		

SWS Item	:		
Name	CanNmMsgCycleTime {CANNM_MSG_CYCLE_TIME}		

Description	Period of a NM PDU in seconds. 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".		
Multiplicity	1		
Type	FloatParamDef		
Range	0.001 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Network dependency: This parameter is only valid if CanNmPassiveModeEnabled is False. NmTimeoutTime = n*CanNmMsgCycleTime		

SWS Item	:		
Name	CanNmMsgReducedtime {CANNM_MSG_REDUCED_TIME}		
Description	Node specific bus cycle time in the periodic transmission mode with bus load reduction. Specified in seconds.		
Multiplicity	1		
Type	FloatParamDef		
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	scope: Channel dependency: This parameter is only valid if CanNmBusLoadReductionEnabled == True and CanNmBusLoadReductionActive == True and CanNmPassiveModeEnabled == False Otherwise this parameter is not used. 0,5 * CanNmMsgCycleTime ≤ CanNmMsgReducedTime < CanNmMsgCycleTime		

SWS Item	:		
Name	CanNmMsgTimeoutTime {CANNM_MSG_TIMEOUT_TIME}		
Description	Transmission Timeout of NM PDU.		
Multiplicity	1		
Type	FloatParamDef		
Range	1 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel dependency: This parameter is only valid if CANNM_PASSIVE_MODE_ENABLED is disabled. If there is no transmission confirmation by the CAN Interface within this timeout, the CANNM module shall give an error notification. CanNmMsgTimeoutTime = n*CanNmMsgCycleTime		

SWS Item	:			
Name	CanNmNodeId {CANNM_NODE_ID}			
Description	Node identifier of local node			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 255			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: Channel dependency: This parameter is only valid if CanNmNodeIdEnabled == True			

SWS Item	:			
Name	CanNmPduCbvPosition {CANNM_PDU_CBV_POSITION}			
Description	Defines the position of the control bit vector within the NM PDU. The value of the parameter represents the location of the control bit vector in the NM PDU (CanNmPduByte0 means byte 0, CanNmPduByte1 means byte 1, CanNmPduOff means source node identifier is not part of the NM PDU). ImplementationType: CanNm_PduPositionType			
Multiplicity	1			
Type	EnumerationParamDef			
Range	CANNM_PDU_BYTE_0	Byte 0 is used		
	CANNM_PDU_BYTE_1	Byte 1 is used		
	CANNM_PDU_OFF	Control Bit Vector is not used		
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: ECU dependency: if(CANNM_PDU_CBV_POSITION != CANNM_PDU_OFF && CANNM_PDU_NID_POSITION != CANNM_PDU_OFF) then CANNM_PDU_CBV_POSITION != CANNM_PDU_NID_POSITION if(CANNM_PDU_CBV_POSITION != CANNM_PDU_OFF && CANNM_PDU_NID_POSITION == CANNM_PDU_OFF) then CANNM_PDU_CBV_POSITION = CANNM_PDU_BYTE0 CanNmPduNidPosition; If CanNmNodeDetectionEnabled == true then CanNmPduCbvPosition != CANNM_PDU_OFF			

SWS Item	:			
Name	CanNmPduLength {CANNM_PDU_LENGTH}			
Description	Defines the length of the NM PDU.			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 8			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: ECU			

SWS Item	:

Name	CanNmPduNidPosition {CANNM_PDU_NID_POSITION}	
Description	Defines the position of the source node identifier within the NM PDU. The value of the parameter represents the location of the source node identifier in the NM PDU (CanNMPduByte0 means byte 0, CanNmPduByte1 means byte 1, CanNmPduOff means source node identifier is not part of the NM PDU) ImplementationType: CanNm_PduPositionType	
Multiplicity	1	
Type	EnumerationParamDef	
Range	CANNM_PDU_BYTE_0	Byte 0 is used
	CANNM_PDU_BYTE_1	Byte 1 is used
	CANNM_PDU_OFF	Node Identification is not used
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: if(CANNM_PDU_NID_POSITION != CANNM_PDU_OFF && CANNM_PDU_CBV_POSITION != CANNM_PDU_OFF) then CANNM_PDU_NID_POSITION != CANNM_PDU_CBV_POSITION if(CANNM_PDU_NID_POSITION != CANNM_PDU_OFF && CANNM_PDU_CBV_POSITION == CANNM_PDU_OFF) then CANNM_PDU_NID_POSITION = CANNM_PDU_BYTE0 CanNmPduCbvPosition; If CanNmNodeIdEnabled == true then CanNmPduNidPosition != CANNM_PDU_OFF	

SWS Item	CANNM066_Conf :		
Name	CanNmPnEnabled		
Description	Enables or disables support of partial networking. false: Partial networking Range not supported true: Partial networking supported		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: If CanNmPnEnabled == TRUE then CanNmComUserDataSupport = TRUE		

SWS Item	CANNM067_Conf :		
Name	CanNmPnEraCalcEnabled		
Description	Specifies if CanNm calculates the PN request information for external requests. (ERA) false: PN request are not calculated true: PN request are calculated		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	

Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true

SWS Item	CANNM071_Conf :		
Name	CanNmPnHandleMultipleNetworkRequests		
Description	Specifies if CanNm performs an additional transition from Network Mode to Repeat Message State (true) or not (false).		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true		

SWS Item	:		
Name	CanNmRemoteSleepIndTime {CANNM_REMOTE_SLEEP_IND_TIME}		
Description	Timeout for Remote Sleep Indication. It defines the time in seconds how long it shall take to recognize that all other nodes are ready to sleep.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel dependency: CanNmRemoteSleepIndTime ≥ CanNmMsgCycleTime		

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

CanNmRepeatMessageTime = n * CanNmMsgCycleTime;
CanNmRepeatMessageTime > CanNmImmediateNmTransmissions *
CanNmImmediateNmCycleTime

SWS Item	ECUC_CanNm_00072 :		
Name	CanNmRetryFirstMessageRequest		
Description	Specifies if first message request in CanNm is repeated until accepted by CanIf.		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local dependency: CanNmRetryFirstMessageRequest = false if CanNmPassiveModeEnabled == true		

SWS Item	:		
Name	CanNmTimeoutTime {CANNM_TIMEOUT_TIME}		
Description	Network Timeout for NM PDUs. It denotes the time in seconds how long the NM shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.		
Multiplicity	1		
Type	FloatParamDef		
Range	0.002 .. 65.535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel dependency: It shall be equal for all nodes in the cluster. It shall be greater than CanNmMsgCycleTime. Typically it should be equal to: n * CanNmMsgCycleTime, where n denotes the number of NM PDU 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 PDUs that can be tolerated by the coordination algorithm. CanNmTimeoutTime = n * CanNmMsgCycleTime		

SWS Item	:		
Name	CanNmUserDataLength {CANNM_USER_DATA_LENGTH}		
Description	Defines the length of the user data contained in the NM PDU. Please note that this parameter is deprecated and will be removed in future.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 8		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: CanNmPduLength, CanNmPduNidPosition, CanNmPduCbvPosition		

SWS Item	:			
Name	CanNmWaitBusSleepTime {CANNM_WAIT_BUS_SLEEP_TIME}			
Description	Timeout for bus calm down phase. It denotes the time in seconds how long the NM shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.			
Multiplicity	1			
Type	FloatParamDef			
Range	0.001 .. 65.535			
Default value	--			
ConfigurationClass	**Pre-compile time**		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: Channel dependency: It shall be equal for all nodes in the cluster. It shall be long enough to make all Tx-buffer empty.			

SWS Item	:			
Name	CanNmChannelIdRef {CANNM_CHANNEL_ID}			
Description	Channel identifier configured for the respective AUTOSAR NM cluster. It is used by referring to the respective NM channel handle. It must be unique for each AUTOSAR NM cluster within one ECU.			
Multiplicity	1			
Type	Reference to [NmChannelConfig]			
ConfigurationClass	**Pre-compile time**		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: Channel			

SWS Item	:			
Name	CanNmNetworkHandle {CANNM_CHANNEL_HANDLE}			
Description	Channel identifier configured for the respective CanNm channel. The CanNmChannelHandle shall be encoded in the canNmRxPduId parameter which is passed to CanNm_RxIndication() function called by the CanIf.			
Multiplicity	1			
Type	Reference to [CanStateManagerNetworks]			
ConfigurationClass	**Pre-compile time**		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME
	Post-build time		X	VARIANT-POST-BUILD
Scope / Dependency	scope: Channel			

SWS Item	**CANNM065_Conf :**			
Name	CanNmPnEraRxNSduRef			
Description	Reference to a Pdu in the COM-Stack. The SduRef is required for every CanNm Channel, because ERA is reported per channel.			
Multiplicity	0..1			
Type	Reference to [Pdu]			
ConfigurationClass	**Pre-compile time**		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD

	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEraCalcEnabled == true		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanNmRxPdu	1	This container is used to configure the Rx PDU Properties that are used for the CanNm Channel.
CanNmTxPdu	0..1	This container is used to configure the Tx PDU Properties that are used for the CanNm Channel.
CanNmUserDataTxPdu	0..1	This optional container is used to configure the UserNm PDU. This container is only available if CanNmComUserDataSupport is enabled.

10.3.6 CanNmRxPdu

SWS Item	:
Container Name	CanNmRxPdu
Description	This container is used to configure the Rx PDU Properties that are used for the CanNm Channel.
Configuration Parameters	

SWS Item	:		
Name	CanNmRxPduId {CANNM_RX_PDU_ID}		
Description	This parameter defines the Rx PDU ID of the CanIf L-PDU range that is associated with this CanNmChannel. ImplementationType: PduIdType		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Channel		

SWS Item	:		
Name	CanNmRxPduRef		
Description	Reference to the global PDU that is used by this CanNmChannel.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3.7 CanNmTxPdu

SWS Item	:
Container Name	CanNmTxPdu

Description	This container is used to configure the Tx PDU Properties that are used for the CanNm Channel.
Configuration Parameters	

SWS Item	:		
Name	CanNmTxPduRef		
Description	The reference to the common PDU structure.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3.8 CanNmUserDataTxPdu

SWS Item	:
Container Name	CanNmUserDataTxPdu
Description	This optional container is used to configure the UserNm PDU. This container is only available if CanNmComUserDataSupport is enabled.
Configuration Parameters	

SWS Item	:		
Name	CanNmTxUserDataPduId {CANNM_TX_USER_DATA_PDU_ID}		
Description	This parameter defines the Handle ID of the NM User Data I-PDU.		
Multiplicity	1		
Type	IntegerParamDef (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, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	:		
Name	CanNmTxUserDataPduRef		
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, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3.9 CanNmPnInfo

SWS Item	CANNM069_Conf :	
Container Name	CanNmPnInfo	
Description	PN information configuration	
Configuration Parameters		

SWS Item	CANNM061_Conf :			
Name	CanNmPnInfoLength			
Description	Specifies the length of the PN request information in the NM PDU.			
Multiplicity	1			
Type	IntegerParamDef			
Range	1 .. 7			
Default value	1			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel.			

SWS Item	CANNM060_Conf :			
Name	CanNmPnInfoOffset			
Description	Specifies the offset of the PN request information in the NM PDU.			
Multiplicity	1			
Type	IntegerParamDef			
Range	1 .. 7			
Default value	1			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE
	Link time		X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time		--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel.			

Included Containers		
Container Name	Multiplicity	Scope / Dependency
CanNmPnFilterMaskByte	0..7	Filter mask byte configuration

10.3.10 CanNmPnFilterMaskByte

SWS Item	CANNM062_Conf :	
Container Name	CanNmPnFilterMaskByte	
Description	Filter mask byte configuration	
Configuration Parameters		

SWS Item	CANNM063_Conf :			
Name	CanNmPnFilterMaskByteIndex			
Description	Index of the filter mask byte. Specifies the position within the filter mask byte array.			
Multiplicity	1			
Type	IntegerParamDef			
Range	0 .. 6			
Default value	--			
ConfigurationClass	Pre-compile time		X	VARIANT-PRE-COMPILE

Document ID 013:AUTOSAR_SWS_CAN_NM

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel. CanNmPnFilterMaskByteIndex < CanNmPnFilterMaskLength		

SWS Item	CANNM064_Conf :		
Name	CanNmPnFilterMaskByteValue		
Description	Parameter to configure the filter mask byte.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	0		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if CanNmPnEnabled == true for at least one CanNm Channel.		

No Included Containers

10.4 Published parameters

CANNM021: The following table specifies configuration parameters that shall be published in the module's header file.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [18] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.

11 Examples

11.1 Example of periodic transmission mode with bus load reduction

Three nodes are connected to the bus and are in "normal operation" state. The nodes (Node 1 and Node 2) with the smallest CANNM_MSG_REDUCED_TIME are sending alternating their NM PDUs. After a while node 1 goes into "ready sleep" state. Now node 2 and node 3 are sending alternating NM PDU. After a while also node 2 goes into "ready sleep" state. Since node 3 is the last node on the bus only node 3 is sending PDUs with CANNM_MSG_CYCLE_TIME.
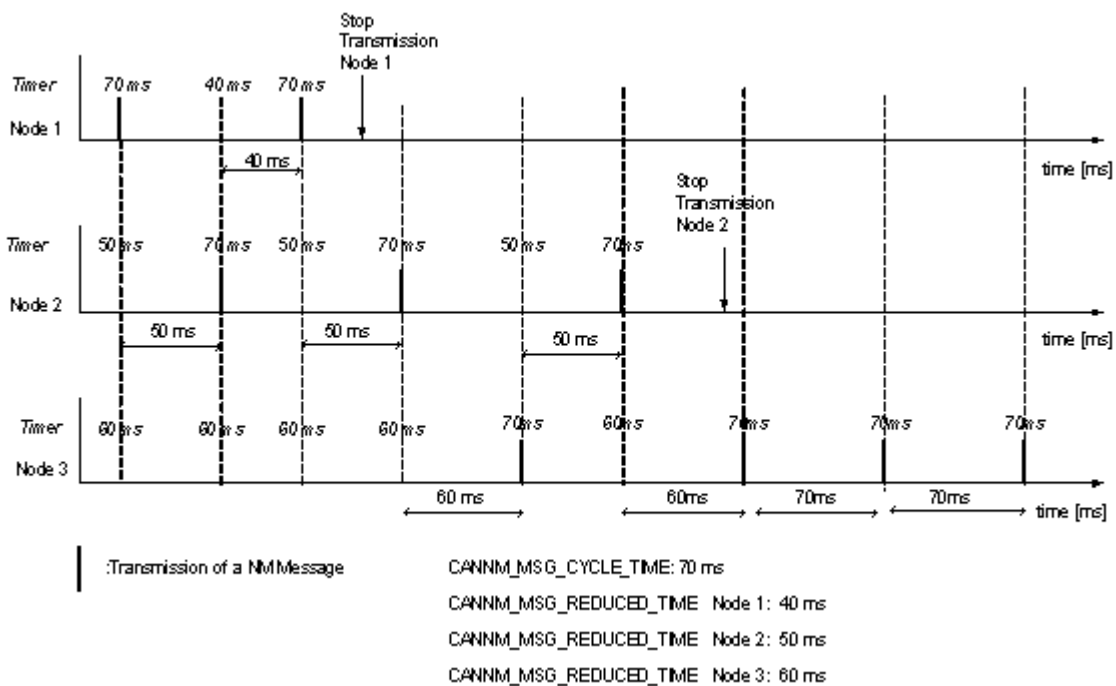
11.1.1 Example timing behaviour for NM PDUs

Assume an example network of three nodes 1, 2, 3 (**Figure 11-1**). Nodes specific cycle offsets are equal respectively to t1 < t2 < t3 < T. NM cycle time is equal to T (**Figure 11-2**).
NM PDUs sent on the bus within the Repeat Message State are presented in the Figure 11-3, and within the Normal Operation / Ready Sleep State in Figure 11-4. Each dot in Figure 11-4denotes restart of the NM-Timeout Timer.
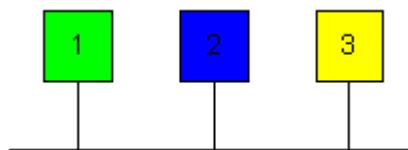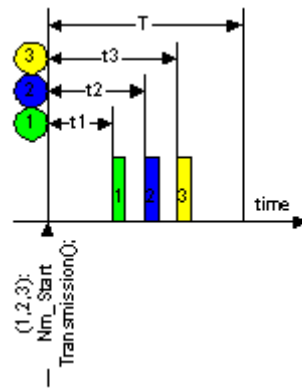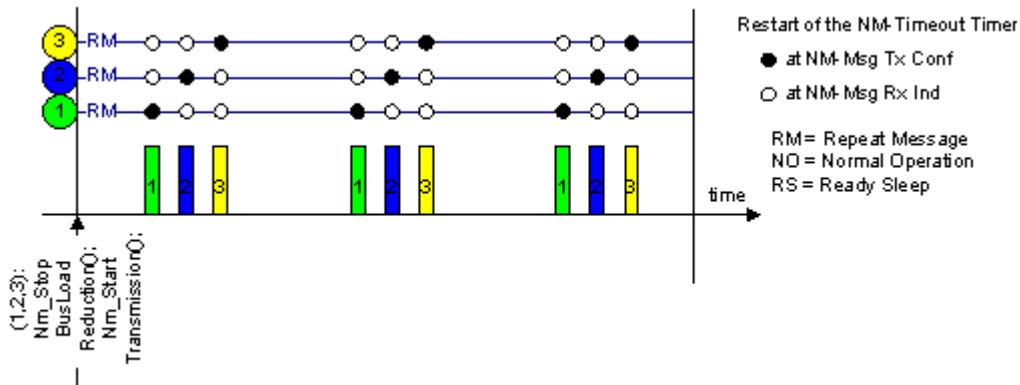

Figure 11-1

Figure 11-2

Figure 11-3

Figure 11-4