Document Title	Specification of CAN Network Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	13

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	**Release**	**Changed by**	**Description**
2024-11-27	R24-11	AUTOSAR Release Management	• adapted interaction with lower layer to LSduR • Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes • Improvements and harmonization
2022-11-24	R22-11	AUTOSAR Release Management	• Fixes for Partial Networking and PNC Shutdown • Improved traceability
2021-11-25	R21-11	AUTOSAR Release Management	• Rework of Partial Networking • Fixes for Partial Networking extensions
2020-11-30	R20-11	AUTOSAR Release Management	• Harmonizing error sections • Partial Networking extensions introduced
2019-11-28	R19-11	AUTOSAR Release Management	• Clarification for CAN FD usage • Extended Wait Bus Sleep Handling • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Header File Cleanup • Removed obsolete elements • Fixed documentation structure

▽

△

2017-12-08	4.3.1	AUTOSAR Release Management	• Node Detection Configuration per channel • Runtime Errors introduced
2016-11-30	4.3.0	AUTOSAR Release Management	• API Harmonizations • Improved post-build parameter support and dependencies • Transmission of additional NM message on NM Coordinator Ready Sleep Bit change • Introduction of Reliable TX Confirmation
2015-07-31	4.2.2	AUTOSAR Release Management	• Clarification NM message transmission start • Clarification of configuration dependencies • Clarification NM timers while communication is disabled
2014-10-31	4.2.1	AUTOSAR Release Management	• Removed obsolete configuration parameters • Partial Network Handling Improvements • Const usage in APIs reworked
2014-03-31	4.1.3	AUTOSAR Release Management	• Rewording and improving Partial Networking Algorithm Requirements • Remote Sleep Indication Timeout handling corrected • Network Release handling during communication control clarified
2013-10-31	4.1.2	AUTOSAR Release Management	• Fixed Message Cycle Time Offset Handling • Corrected Active Wakeup Handling • Editorial changes • Removed chapter(s) on change documentation

▽

△

2013-03-15	4.1.1	AUTOSAR Administration	• Partial Network Handling corrected • Coordinator Support improved • Start-up Handling from Prepare-Bus Sleep clarified
2011-12-22	4.0.3	AUTOSAR Administration	• Support for Partial Networking • Support for Car Wakeup • Immediate Transmission of NM-PDUs • Support of a coordinated shutdown with multiple connected gateways
2009-12-18	4.0.1	AUTOSAR Administration	• Changed Signature of RxIndication and TriggerTransmit • Faster NM wakeup
2010-02-02	3.1.4	AUTOSAR Administration	• Nm User Data accessible through PduR • Changed PDU handle ID exchange with CanIf • No more instance specific CanNm MainFunction() APIs • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Administration	• Merge CAN NM and Generic NM • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	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
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Contents

1 Introduction and functional overview

This document describes the concept, core functionality, configurable 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 [1] (for limitations refer to Section 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 configurable 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.

For a general understanding of the AUTOSAR Network Management functionality please refer to [1, Specification of the AUTOSAR Network Management Protocol] and [2, SWS Network Management Interface].

[1] This includes all in AUTOSAR specified CAN protocols like CAN 2.0, CAN FD or TT CAN.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the CanNm module that are not included in the [3, AUTOSAR glossary].

Acronym/abbreviation:	Description:
CanIf	Abbreviation for the CAN Interface
CanNm	Abbreviation for CAN Network Management
CBV	Control Bit Vector
CWU	Car Wakeup
ERA	External Request Array
EIRA	External and Internal Request Array
LSduR	Linklayer SDU Router
NM	Network Management
PNC	Partial Network Cluster
PNI	Partial Network Information
PNL	Partial Network Learning
SNI	Source Node Identifier

Term	Description:
PDU transmission ability is disabled	This means that the Network Management PDU transmission has been disabled by the service CanNm_DisableCommunication.
Repeat Message Request Bit Indication	CanNm_RxIndication finds the RptMsgRequest set in the Control Bit Vector of a received Network Management PDU.
Top-level PNC coordinator	An ECU acts as top-level PNC coordinator for those PNCs which are actively coordinated on all assigned channels. This ECU has the PNC gateway functionality enabled. The top-level PNC coordinator triggers for those PNCs a synchronized PNC shutdown, if no other ECU in the network requests them and if the synchronized PNC shutdown is enabled.
Intermediate PNC coordinator	An ECU acts as intermediate PNC coordinator for those PNCs which are passively coordinated on at least one channel. This ECU has the PNC gateway functionality enabled. The intermediate PNC coordinator forwards a synchronized PNC shutdown to active coordinated channels for PNCs which are passively coordinated, if the synchronized PNC shutdown is enabled.
PNC leaf node	A PNC leaf node is an ECU that acts not as a PNC coordinator at all in the network. It processes PN shutdown message as usual NM messages.
PN shutdown message	A top-level PNC coordinator transmit PN shutdown messages to indicate a synchronized PNC shutdown across the PN topology. A PN shutdown message is as NM message which has PNSR bit in the control bit vector and all PNCs which are indicated for a synchronized shutdown set to ‘1’.
Immediate Transmission Confirmation	Every NM PDU transmission request is directly seen as confirmed by the bus and no timeout handling is required. This mechanism can be used on bus systems where the bus traffic is designed in a way that every transmit will always be sent on the bus.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Specification of the AUTOSAR Network Management Protocol
AUTOSAR_FO_PRS_NetworkManagementProtocol

[2] Specification of Network Management Interface
AUTOSAR_CP_SWS_NetworkManagementInterface

[3] Glossary
AUTOSAR_FO_TR_Glossary

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] Specification of Linklayer Sdu Routing Module
AUTOSAR_CP_SWS_LSduRouter

[6] Specification of CAN Interface
AUTOSAR_CP_SWS_CANInterface

[7] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[8] Requirements on AUTOSAR Network Management
AUTOSAR_FO_RS_NetworkManagement

[9] Specification of Communication Manager
AUTOSAR_CP_SWS_COMManager

[10] System Template
AUTOSAR_CP_TPS_SystemTemplate

[11] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[12] Guide to Mode Management
AUTOSAR_CP_EXP_ModeManagementGuide

[13] Specification of CAN State Manager
AUTOSAR_CP_SWS_CANStateManager

[14] Specification of Communication Stack Types
AUTOSAR_CP_SWS_CommunicationStackTypes

[15] Specification of Standard Types
AUTOSAR_CP_SWS_StandardTypes

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4, SWS BSW General], which is also valid for CAN Network Management.

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

4 Constraints and assumptions

4.1 Limitations

1. One channel of CanNm is associated with only one network management cluster in one network. One network management 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 [1] systems.

The Figure 4.1 presents an AUTOSAR Network Management stack within an example ECU that contains at least one CanNm cluster.

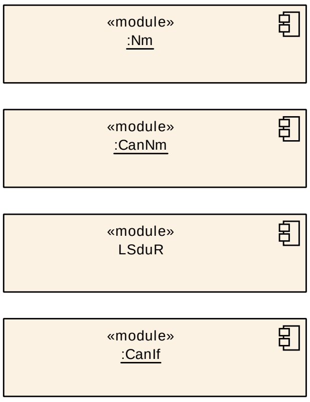


Figure 4.1: AUTOSAR NM Stack on CAN

4.2 Applicability to car domains

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

[1] This includes all in AUTOSAR specified CAN protocols like CAN 2.0, CAN FD or TT CAN.

5 Dependencies to other modules

CAN Network Management (CanNm) mainly uses services of Linklayer SDU Router (LSduR [5]) to address a lower layer (e.g CAN Interface: CanIf [6]) and provides services to the Network Management Interface (Nm [2]).

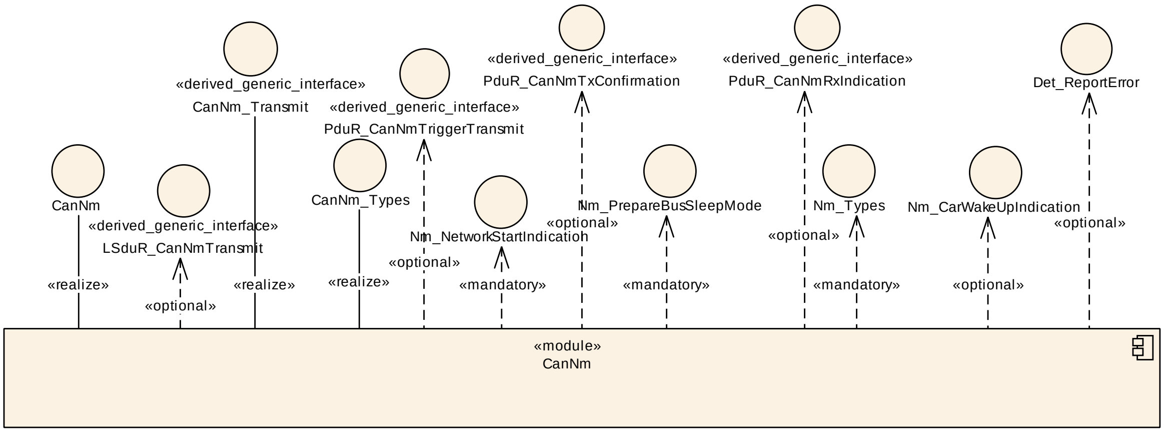


Figure 5.1: Dependencies to other modules

5.1 File Structure

5.1.1 Code File Structure

Please refer to the chapter 5.1.6 Code file structure in SWS_BSWGeneral [4].

5.1.2 Header File Structure

Please refer to the chapter 5.1.7 Header file structure in SWS_BSWGeneral [4].

[SWS_CanNm_00305]

Upstream requirements: SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00301

⌈ComStack_Types.h shall be included.

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)

⌋

[SWS_CanNm_00308]

Upstream requirements: SRS_BSW_00301

⌈Det.h shall be included for interfacing the Default Error Tracer.⌋

[SWS_CanNm_00309]

Upstream requirements: SRS_BSW_00301

⌈NmStack_Types.h shall be included for common network management types.⌋

[SWS_CanNm_00312]

Upstream requirements: SRS_BSW_00301

⌈LSduR_CanNm.h shall be included for interfacing the LSduR.⌋

[SWS_CanNm_00326]

Upstream requirements: SRS_BSW_00301

⌈PduR_CanNm.h shall be included if COM user data support is enabled.⌋

5.2 Protocol layer dependencies

The CAN Network Management is based on the protocol mentioned in [1, Specification of the AUTOSAR Network Management Protocol].

6 Requirements Tracing

The following tables reference the requirements specified in [7], and [8], and links to the fulfillment of these. Requirements that are not fulfilled by this document are linked to [SWS_CanNm_NA_00000] to [SWS_CanNm_NA_00008].

Requirement	Description	Satisfied by
[RS_Nm_00045]	Nm shall provide services to coordinate shutdown of Nm-clusters independently of each other	[SWS_CanNm_00104] [SWS_CanNm_00105]
[RS_Nm_00046]	It shall be possible to trigger the startup of all Nodes at any Point in Time	[SWS_CanNm_00129]
[RS_Nm_00047]	Nm shall provide a service to request to keep the bus awake and a service to cancel this request.	[SWS_CanNm_00103] [SWS_CanNm_00104] [SWS_CanNm_00105] [SWS_CanNm_00106] [SWS_CanNm_00110] [SWS_CanNm_00118]
[RS_Nm_00050]	The Nm shall provide the current state of Nm	[SWS_CanNm_00091]
[RS_Nm_00051]	Nm shall inform application when Nm state changes occur.	[SWS_CanNm_00097] [SWS_CanNm_00114] [SWS_CanNm_00126] [SWS_CanNm_00166]
[RS_Nm_00052]	The Nm interface shall signal to the application that all other ECUs are ready to sleep.	[SWS_CanNm_00150] [SWS_CanNm_00153]
[RS_Nm_00054]	There shall be a deterministic time from the point where all nodes agree to go to bus sleep to the point where bus is switched off.	[SWS_CanNm_00088]
[RS_Nm_00137]	Nm shall perform communication system error handling for errors that have impact on the Nm behavior.	[SWS_CanNm_00064] [SWS_CanNm_00065] [SWS_CanNm_00066] [SWS_CanNm_00193] [SWS_CanNm_00194] [SWS_CanNm_00446]
[RS_Nm_00142]	Nm shall provide a mechanism to limit its bus load.	[SWS_CanNm_00052] [SWS_CanNm_00069] [SWS_CanNm_00071] [SWS_CanNm_00156] [SWS_CanNm_00157]
[RS_Nm_00149]	The timing of Nm shall be configurable.	[SWS_CanNm_00088]
[RS_Nm_00151]	The Network Management algorithm shall allow any node to integrate into an already running Nm cluster	[SWS_CanNm_00099] [SWS_CanNm_00124] [SWS_CanNm_00127]
[RS_Nm_00153]	The Network Management shall optionally provide a possibility to detect present nodes	[SWS_CanNm_00014] [SWS_CanNm_00111] [SWS_CanNm_00112] [SWS_CanNm_00113] [SWS_CanNm_00119] [SWS_CanNm_00120] [SWS_CanNm_00121]
[RS_Nm_02503]	The Nm API shall optionally give the possibility to send user data	[SWS_CanNm_00013] [SWS_CanNm_00159] [SWS_CanNm_00328] [SWS_CanNm_00351] [SWS_CanNm_00510]
[RS_Nm_02504]	The Nm API shall optionally give the possibility to get user data	[SWS_CanNm_00160]
[RS_Nm_02505]	The Nm shall optionally set the local node identifier to the Nm-message	[SWS_CanNm_00074]
[RS_Nm_02506]	The Nm API shall give the possibility to read the source node identifier of the sender	[SWS_CanNm_00132]
[RS_Nm_02508]	Every node shall have a node identifier associated with it that is unique in the Nm-cluster.	[SWS_CanNm_00133]

▽

△

Requirement	Description	Satisfied by
[RS_Nm_02509]	The Nm interface shall signal to the application that at least one ECU is not ready to sleep anymore.	[SWS_CanNm_00151] [SWS_CanNm_00152] [SWS_CanNm_00153]
[RS_Nm_02511]	It shall be possible to configure the Network Management of a node so that it does not contribute to the cluster shutdown decision.	[SWS_CanNm_00161]
[RS_Nm_02512]	The Nm shall give the possibility to enable or disable the network management related communication configured for an active Nm node	[SWS_CanNm_00170] [SWS_CanNm_00173] [SWS_CanNm_00176] [SWS_CanNm_00178]
[RS_Nm_02513]	Nm shall provide functionality which enables upper layers to control the sleep mode.	[SWS_CanNm_00104] [SWS_CanNm_00105]
[RS_Nm_02516]	All AUTOSAR Nm instances shall support the Nm Coordinator functionality including Bus synchronization on demand	[SWS_CanNm_00130] [SWS_CanNm_00187] [SWS_CanNm_00226] [SWS_CanNm_00280]
[RS_Nm_02517]	CanNm shall support Partial Networking on CAN	[SWS_CanNm_00409] [SWS_CanNm_00410] [SWS_CanNm_00411] [SWS_CanNm_00413] [SWS_CanNm_00414] [SWS_CanNm_00444] [SWS_CanNm_00502] [SWS_CanNm_00503] [SWS_CanNm_00511] [SWS_CanNm_00518]
[RS_Nm_02519]	The Nm Control Bit Vector shall contain a PNI (Partial Network Information) bit.	[SWS_CanNm_00413] [SWS_CanNm_00414] [SWS_CanNm_00511] [SWS_CanNm_00518]
[RS_Nm_02527]	Nm shall implement a filter algorithm dropping all Nm messages that are not relevant for the ECU	[SWS_CanNm_00333] [SWS_CanNm_00410] [SWS_CanNm_00411] [SWS_CanNm_00502] [SWS_CanNm_00503]
[RS_Nm_02528]	Nm shall provide a service which allows for instantaneous sending of Nm messages.	[SWS_CanNm_00444]
[RS_Nm_02536]	Nm shall provide functionality to start-up without requesting the network.	[SWS_CanNm_00128]
[RS_Nm_02540]	The Nm Control Bit Vector shall contain a PN shutdown request bit.	[SWS_CanNm_00519]
[RS_Nm_02541]	Nm shall define a common layout of Nm messages.	[SWS_CanNm_00074] [SWS_CanNm_00075] [SWS_CanNm_00501]
[RS_Nm_02542]	The <Bus>Nm of the top-level PNC coordinator shall set the PN shutdown request bit if a least one PNC is released	[SWS_CanNm_00521]
[RS_Nm_02544]	Nm shall forward the indication of a PN shutdown message	[SWS_CanNm_00461]
[RS_Nm_02547]	<Bus>Nm shall be able to propagate and evaluate the need for Partial Networking Learning (optional)	[SWS_CanNm_00380] [SWS_CanNm_00381]
[RS_Nm_02548]	<Bus>Nm shall be able to propagate and evaluate the need for synchronized PNC shutdown in the role of a top-level PNC coordinator or intermediate PNC coordinator (optional)	[SWS_CanNm_00519]

▽

△

Requirement	Description	Satisfied by
[RS_Nm_02549]	Nm shall offer interfaces to Request and indicate Repeat Message Request (optional)	[SWS_CanNm_00111] [SWS_CanNm_00112] [SWS_CanNm_00113] [SWS_CanNm_00119] [SWS_CanNm_00120] [SWS_CanNm_00121]
[RS_Nm_02565]	<Bus>Nm shall communicate EIRA and ERA requests to the upper layers using dedicated APIs	[SWS_CanNm_00502]
[RS_Nm_02571]	Nm shall handle requests for synchronized PNC shutdown	[SWS_CanNm_00515] [SWS_CanNm_00516] [SWS_CanNm_00517]
[RS_Nm_02572]	<Bus>Nm shall transmit requests for synchronized PNC shutdown as NM-PDU	[SWS_CanNm_00513] [SWS_CanNm_00519] [SWS_CanNm_00520] [SWS_CanNm_00521] [SWS_CanNm_00522] [SWS_CanNm_00523] [SWS_CanNm_91005] [SWS_CanNm_91006]
[RS_Nm_02573]	<Bus>Nm shall handle retransmission of NM-PDUs	[SWS_CanNm_00514]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_CanNm_00305] [SWS_CanNm_00308] [SWS_CanNm_00309] [SWS_CanNm_00312] [SWS_CanNm_00326]
[SRS_BSW_00310]	API naming convention	[SWS_CanNm_00208] [SWS_CanNm_00211] [SWS_CanNm_00213] [SWS_CanNm_00214] [SWS_CanNm_00215] [SWS_CanNm_00216] [SWS_CanNm_00217] [SWS_CanNm_00218] [SWS_CanNm_00219] [SWS_CanNm_00220] [SWS_CanNm_00221] [SWS_CanNm_00222] [SWS_CanNm_00223] [SWS_CanNm_00224] [SWS_CanNm_00226] [SWS_CanNm_00227] [SWS_CanNm_00228] [SWS_CanNm_00231] [SWS_CanNm_00331] [SWS_CanNm_00338] [SWS_CanNm_91001] [SWS_CanNm_91002] [SWS_CanNm_91004] [SWS_CanNm_91005] [SWS_CanNm_91006]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_CanNm_00244]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_CanNm_91002]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_CanNm_00305]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_CanNm_00192] [SWS_CanNm_00195] [SWS_CanNm_00352] [SWS_CanNm_00507]
[SRS_BSW_00353]	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	[SWS_CanNm_00305]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_CanNm_00208]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_CanNm_00352]
[SRS_BSW_00385]	List possible error notifications	[SWS_CanNm_00316] [SWS_CanNm_00317]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_CanNm_00317]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_CanNm_00211] [SWS_CanNm_00213] [SWS_CanNm_00214] [SWS_CanNm_00215] [SWS_CanNm_00216] [SWS_CanNm_00217] [SWS_CanNm_00218] [SWS_CanNm_00219] [SWS_CanNm_00220] [SWS_CanNm_00221] [SWS_CanNm_00222] [SWS_CanNm_00223] [SWS_CanNm_00224] [SWS_CanNm_00227] [SWS_CanNm_00228] [SWS_CanNm_00231] [SWS_CanNm_00234] [SWS_CanNm_00331] [SWS_CanNm_00338] [SWS_CanNm_00344] [SWS_CanNm_91001] [SWS_CanNm_91004] [SWS_CanNm_91005] [SWS_CanNm_91006]
[SRS_BSW_00460]	Reentrancy Levels	[SWS_CanNm_00208] [SWS_CanNm_00211] [SWS_CanNm_00213] [SWS_CanNm_00214] [SWS_CanNm_00215] [SWS_CanNm_00216] [SWS_CanNm_00217] [SWS_CanNm_00218] [SWS_CanNm_00219] [SWS_CanNm_00220] [SWS_CanNm_00221] [SWS_CanNm_00222] [SWS_CanNm_00223] [SWS_CanNm_00224] [SWS_CanNm_00226] [SWS_CanNm_00227] [SWS_CanNm_00228] [SWS_CanNm_00231] [SWS_CanNm_00331] [SWS_CanNm_00338] [SWS_CanNm_91001] [SWS_CanNm_91002] [SWS_CanNm_91004] [SWS_CanNm_91005] [SWS_CanNm_91006]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_CanNm_00211] [SWS_CanNm_00213] [SWS_CanNm_00214] [SWS_CanNm_00215] [SWS_CanNm_00216] [SWS_CanNm_00217] [SWS_CanNm_00218] [SWS_CanNm_00219] [SWS_CanNm_00220] [SWS_CanNm_00221] [SWS_CanNm_00222] [SWS_CanNm_00223] [SWS_CanNm_00226] [SWS_CanNm_00227] [SWS_CanNm_00331] [SWS_CanNm_00338] [SWS_CanNm_91004] [SWS_CanNm_91005] [SWS_CanNm_91006]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_CanNm_00316]
[SRS_BSW_00481]	Invalid configuration set selection errors shall follow a naming rule	[SWS_CanNm_00316]
[SRS_BSW_00482]	Get version information function shall follow a naming rule	[SWS_CanNm_00224]
[SRS_BSW_00483]	BSW Modules shall handle buffer alignments internally	[SWS_CanNm_00035] [SWS_CanNm_00091] [SWS_CanNm_00132] [SWS_CanNm_00133] [SWS_CanNm_00138] [SWS_CanNm_00153] [SWS_CanNm_00159] [SWS_CanNm_00160] [SWS_CanNm_00333] [SWS_CanNm_00351] [SWS_CanNm_00510]
[SRS_BSW_00484]	Input parameters of scalar and enum types shall be passed as a value.	[SWS_CanNm_00211] [SWS_CanNm_00213] [SWS_CanNm_00214] [SWS_CanNm_00215] [SWS_CanNm_00216] [SWS_CanNm_00217] [SWS_CanNm_00218] [SWS_CanNm_00219] [SWS_CanNm_00220] [SWS_CanNm_00221] [SWS_CanNm_00222] [SWS_CanNm_00223] [SWS_CanNm_00226] [SWS_CanNm_00227] [SWS_CanNm_00228] [SWS_CanNm_00231] [SWS_CanNm_00331] [SWS_CanNm_00338] [SWS_CanNm_91001] [SWS_CanNm_91004] [SWS_CanNm_91005] [SWS_CanNm_91006]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00485]	Input parameters of structure type shall be passed as a reference to a constant structure	[SWS_CanNm_00208] [SWS_CanNm_00231] [SWS_CanNm_00331]
[SRS_BSW_00486]	Input parameters of array type shall be passed as a reference to the constant array base type	[SWS_CanNm_00217]
[SRS_BSW_00487]	Errors for module initialization shall follow a naming rule	[SWS_CanNm_00316]

Table 6.1: Requirements Tracing

Details about the SRS Requirements can be found in AUTOSAR General Requirements on Basic Software Modules [7] and Requirements on AUTOSAR Network Management [8].

7 Functional specification

7.1 Coordination algorithm

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

The AUTOSAR CanNm algorithm is based on periodic Network Management PDUs, which are received by all nodes in the cluster via broadcast transmission. Reception of Network Management PDUs indicates that sending nodes want to keep the network management cluster awake. If any node is ready to go to the Bus-Sleep Mode, it stops sending Network Management PDUs, but as long as Network Management PDUs from other nodes are received, it postpones transition to the Bus-Sleep Mode. Finally, if a dedicated timer elapses because no Network Management PDUs are received anymore, every node initiates transition to the Bus-Sleep Mode.

If any node in the network management cluster requires bus-communication, it can wake-up the network management cluster from the Bus-Sleep Mode by transmitting Network Management PDUs. For more details concerning wakeup procedure itself please refer to the AUTOSAR SWS ComM [9].

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

[SWS_CanNm_00089] ⌈The AUTOSAR CanNm state machine shall contain states, transitions and triggers required for the AUTOSAR CanNm algorithm seen from point of view of one single node in the network management cluster.⌋

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 CanNm algorithm are described in detail.

[SWS_CanNm_00092] ⌈The AUTOSAR CanNm shall contain three operational modes visible at the module's interface:

- Network Mode

- Prepare Bus-Sleep Mode
- Bus-Sleep Mode

⌋

[SWS_CanNm_00093] ⌈Changes of the AUTOSAR CanNm operational modes shall be notified to the upper layer by means of callback functions.⌋

[SWS_CanNm_00091]

Upstream requirements: RS_Nm_00050, SRS_BSW_00483

⌈When CanNm_GetState is called CanNm shall return the current NM state and mode.⌋

7.2.1 Network Mode

[SWS_CanNm_00094] ⌈The Network Mode shall consist of three internal states:

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

⌋

[SWS_CanNm_00314] ⌈When the Network Mode is entered from Bus-Sleep, by default, the CanNm module shall enter the Repeat Message State.⌋

[SWS_CanNm_00315] ⌈When the Network Mode is entered from Prepare Bus-Sleep Mode, by default, the CanNm module shall enter the Repeat Message State.⌋

[SWS_CanNm_00096] ⌈When the Network Mode is entered, the CanNm module shall start the NM-Timeout Timer.⌋

[SWS_CanNm_00097]

Upstream requirements: RS_Nm_00051

⌈When the Network Mode is entered, CanNm shall notify the upper layer of the new current operational mode by calling the callback function Nm_NetworkMode.⌋

[SWS_CanNm_00098] ⌈At successful reception of a Network Management PDU (call of CanNm_RxIndication) in the Network Mode, the CanNm module shall restart the NM-Timeout Timer if PDU transmission ability is enabled.⌋

[SWS_CanNm_00099]

Upstream requirements: RS_Nm_00151

⌈At successful transmission of a Network Management PDU (call of CanNm_TxConfirmation with E_OK) in the Network Mode, the CanNm module shall restart the NM-Timeout Timer.⌋

Note: If CanNmImmediateTxconfEnabled is enabled it is assumed that each Network Management PDU transmission request results in a successful Network Management PDU transmission.

[SWS_CanNm_00206] ⌈The CanNm module shall reset the NM-Timeout Timer every time it is started or restarted.⌋

[SWS_CanNm_00147] ⌈If CanNm_PassiveStartUp is called in the Network Mode, the CanNm module shall not execute this service and shall return E_NOT_OK.⌋

[SWS_CanNm_00380]

Upstream requirements: RS_Nm_02547

⌈If function CanNm_PnLearningRequest is called on a channel where CanNmDynamicPncToChannelMappingEnabled is set to TRUE and CanNm is in the Network Mode the CanNm module shall set the Repeat Message Bit and the Partial Network Learning Bit in the CBV to 1 on this channel and change to or restart the Repeat Message State.⌋

[SWS_CanNm_00381]

Upstream requirements: RS_Nm_02547

⌈If the bits Partial Network Learning and Repeat Message Request both are received with value 1 on a channel where CanNmDynamicPncToChannelMappingEnabled is set to TRUE and CanNm is in the Network Mode, then CanNm shall set the Partial Network Learning Bit in the CBV to 1 on this channel and change to or restart the Repeat Message State.⌋

Note: Restart in [SWS_CanNm_00380] or [SWS_CanNm_00381] means that CanNm is already in Repeat Message State and then a complete re-entry of the Repeat Message State has to be performed once.

7.2.1.1 Repeat Message State

For nodes that are not in passive mode (refer to Section 7.9.3) 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. It can be used for detection of present nodes.

[SWS_CanNm_00100] ⌈When the Repeat Message State is entered the CanNm module shall (re-)start transmission of Network Management PDUs unless passive mode is enabled and/or communication is disabled.⌋

[SWS_CanNm_00101] ⌈When the NM-Timeout Timer expires in the Repeat Message State, the CanNm module shall (re-)start the NM-Timeout Timer.⌋

[SWS_CanNm_00193]

Upstream requirements: RS_Nm_00137

⌈When the NM-Timeout Timer expires in the Repeat Message State the CanNm module shall report `CANNM_E_NETWORK_TIMEOUT` to the DET.⌋

[SWS_CanNm_00102] ⌈The network management state machine shall stay in the Repeat Message State for a configurable amount of time determined by the `CanNmRepeatMessageTime` (configuration parameter); after that time the CanNm module shall leave the Repeat Message State.⌋

[SWS_CanNm_00103]

Upstream requirements: RS_Nm_00047

⌈When Repeat Message State is left and if the network has been requested (see [SWS_CanNm_00104]), the CanNm module shall enter the Normal Operation State.⌋

[SWS_CanNm_00106]

Upstream requirements: RS_Nm_00047

⌈When Repeat Message State is left and if the network has been released (see [SWS_CanNm_00105]), the CanNm module shall enter the Ready Sleep State.⌋

[SWS_CanNm_00107] ⌈If `CanNmNodeDetectionEnabled` is set to TRUE CanNm shall clear the Repeat Message Bit when leaving the Repeat Message State.⌋

[SWS_CanNm_00137] ⌈If the service `CanNm_RepeatMessageRequest` is called in Repeat Message State, Prepare Bus-Sleep Mode or Bus-Sleep Mode, the CanNm module shall not execute the service and return `E_NOT_OK`.⌋

[SWS_CanNm_00382] ⌈If CanNmDynamicPncToChannelMappingEnabled is set to TRUE CanNm shall clear the Partial Network Learning Bit when leaving the Repeat Message State.⌋

7.2.1.2 Normal Operation State

The Normal Operation State ensures that any node can keep the network management cluster awake as long as the network is requested.

[SWS_CanNm_00116] ⌈When the Normal Operation State is entered from Ready Sleep State, the CanNm module shall start transmission of Network Management PDUs.⌋

Note: If passive mode is enabled or the Network Management PDU transmission ability has been disabled no NM PDUs are transmitted, therefore no action is required.

[SWS_CanNm_00117] ⌈When the NM-Timeout Timer expires in the Normal Operation State, the CanNm module shall (re-)start the NM-Timeout Timer.⌋

[SWS_CanNm_00194]

Upstream requirements: RS_Nm_00137

⌈When the NM-Timeout Timer expires in the Normal Operation State the CanNm module shall report CANNM_E_NETWORK_TIMEOUT to the DET.⌋

[SWS_CanNm_00118]

Upstream requirements: RS_Nm_00047

⌈When the network is released and the current state is Normal Operation State, the CanNm module shall enter the Ready Sleep state (refer to [SWS_CanNm_00105]).⌋

[SWS_CanNm_00119]

Upstream requirements: RS_Nm_00153, RS_Nm_02549

⌈If CanNmNodeDetectionEnabled is set to TRUE and Repeat Message Request Bit is received in the Normal Operation State, the CanNm module shall enter the Repeat Message State.⌋

[SWS_CanNm_00120]

Upstream requirements: RS_Nm_00153, RS_Nm_02549

⌈If CanNmNodeDetectionEnabled is set to TRUE and function CanNm_RepeatMessageRequest is called in the Normal Operation State, the CanNm module shall enter the Repeat Message State.⌋

[SWS_CanNm_00121]

Upstream requirements: RS_Nm_00153, RS_Nm_02549

⌈If CanNmNodeDetectionEnabled is set to TRUE and function CanNm_RepeatMessageRequest is called in the Normal Operation State the CanNm module shall set the Repeat Message Bit.⌋

7.2.1.3 Ready Sleep State

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

[SWS_CanNm_00108] ⌈When the Ready Sleep State is entered from Repeat Message State or Normal Operation State, the CanNm module shall stop transmission of Network Management PDUs.⌋

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

Note: If passive mode is disabled in some cases NM PDUs have to be transmitted in Ready Sleep State to grant a synchronized shutdown in the network, e.g. retransmission of PN shutdown messages.

[SWS_CanNm_00109] ⌈When the NM-Timeout Timer expires in the Ready Sleep State, the CanNm module shall enter the Prepare Bus-Sleep Mode.⌋

[SWS_CanNm_00110]

Upstream requirements: RS_Nm_00047

⌈When the network is requested and the current state is the Ready Sleep State, the CanNm module shall enter Normal Operation State (refer to [SWS_CanNm_00104]).⌋

[SWS_CanNm_00111]

Upstream requirements: RS_Nm_00153, RS_Nm_02549

⌈If CanNmNodeDetectionEnabled is set to TRUE and Repeat Message Request Bit is received in the Ready Sleep State, the CanNm module shall enter the Repeat Message State.⌋

[SWS_CanNm_00112]

Upstream requirements: RS_Nm_00153, RS_Nm_02549

⌈If CanNmNodeDetectionEnabled is set to TRUE and function CanNm_RepeatMessageRequest is called in the Ready Sleep State, the CanNm module shall enter the Repeat Message State.⌋

[SWS_CanNm_00113]

Upstream requirements: RS_Nm_00153, RS_Nm_02549

⌈If CanNmNodeDetectionEnabled is set to TRUE and function CanNm_RepeatMessageRequest is called in Ready Sleep State the CanNm module shall set the Repeat Message Bit.⌋

7.2.2 Prepare Bus-Sleep Mode

The purpose of the Prepare Bus-Sleep Mode is to ensure that all nodes have time to stop their network activity before the Bus-Sleep Mode is entered. In Prepare Bus-Sleep Mode the 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.

[SWS_CanNm_00114]

Upstream requirements: RS_Nm_00051

⌈When Prepare Bus-Sleep Mode is entered, the CanNm module shall notify the upper layer by calling Nm_PrepareBusSleepMode.⌋

[SWS_CanNm_00088]

Upstream requirements: RS_Nm_00054, RS_Nm_00149

⌈The parameter CanNmStayInPbsEnabled shall match parameter NmStayInPbsEnabled from the PRS_Nm_00506 specification ([1, Specification of the AUTOSAR Network Management Protocol]).⌋

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

[SWS_CanNm_00124]

Upstream requirements: RS_Nm_00151

⌈At successful reception of a Network Management PDU in the Prepare Bus-Sleep Mode, the CanNm Module shall enter the Network Mode; by default the CanNm Module shall enter the Repeat Message State (refer to [SWS_CanNm_00315]).⌋

[SWS_CanNm_00123] ⌈When the network is requested in the Prepare Bus-Sleep Mode, the CanNm module shall enter the Network Mode; by default the CanNm Module shall enter the Repeat Message State (refer to [SWS_CanNm_00315]).⌋

[SWS_CanNm_00122] ⌈When the network has been requested (see [SWS_CanNm_00104]) in the Prepare Bus-Sleep Mode and the CanNm module has entered Network Mode and if `CanNmImmediateRestartEnabled` (configuration parameter) is set to TRUE, the CanNm module shall transmit a Network Management PDU.⌋

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

Caused by the transmission offset for Network Management PDUs in CanNm, the transmission of the first Network Management PDU in Repeat Message State can be delayed significantly. In order to avoid a delayed re-start of the network the transmission of a Network Management PDU can be requested immediately.

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

7.2.3 Bus-Sleep Mode

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

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

Note: The parameters `CanNmTimeoutTime` and `CanNmWaitBusSleepTime` should have the same values within all network nodes of the network management 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),
- Network Management PDU waiting time in the Tx-queue (if transmission confirmation is made immediately after transmit request).

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

[SWS_CanNm_00126]

Upstream requirements: RS_Nm_00051

⌈When Bus-Sleep Mode is entered, except by default at initialization, the CanNm module shall notify the upper layer by calling the callback function Nm_BusSleepMode.⌋

[SWS_CanNm_00127]

Upstream requirements: RS_Nm_00151

⌈When the CanNm module successfully receives a Network Management PDU (call of CanNm_RxIndication) in the Bus-Sleep Mode, the CanNm module shall notify the upper layer by calling the callback function Nm_NetworkStartIndication.⌋

Rationale: To avoid race conditions and state inconsistencies between Network and Mode Management, 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.

[SWS_CanNm_00336] ⌈When the CanNm module successfully receives a Network Management PDU (call of CanNm_RxIndication) in the Bus-Sleep Mode, the CanNm module shall report the error CANNM_E_NET_START_IND to the DET.⌋

[SWS_CanNm_00128]

Upstream requirements: RS_Nm_02536

⌈If CanNm_PassiveStartUp is called in the Bus-Sleep Mode or Prepare Bus-Sleep Mode, the CanNm module shall enter the Network Mode; by default the CanNm module shall enter the Repeat Message State (refer to [SWS_CanNm_00314] and [SWS_CanNm_00315]).⌋

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

[SWS_CanNm_00129]

Upstream requirements: RS_Nm_00046

⌈When the network is requested in Bus-Sleep Mode, the CanNm module shall enter the Network Mode; by default the CanNm module shall enter the Repeat Message State (refer to [SWS_CanNm_00314] and [SWS_CanNm_00104]).⌋

7.3 Network states

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

[SWS_CanNm_00104]

Upstream requirements: RS_Nm_00045, RS_Nm_00047, RS_Nm_02513

⌈The function call `CanNm_NetworkRequest` shall request the network. I.e. the CanNm module shall change network state to 'requested'.⌋

[SWS_CanNm_00105]

Upstream requirements: RS_Nm_00045, RS_Nm_00047, RS_Nm_02513

⌈The function call `CanNm_NetworkRelease` shall release the network. I.e. the CanNm module shall change network state to 'released'.⌋

7.4 Initialization

[SWS_CanNm_00141] ⌈If the initialization of the CanNm module (`CanNm_Init`) is successful, the CanNm module shall set the Network Management State to Bus-Sleep Mode.⌋

Note: The CanNm module should be initialized after LSduR and the according lower layer (e.g. CanIf) are initialized and before any other network management service is called.

[SWS_CanNm_00143] ⌈When initialized, by default, the CanNm module shall set the network state to 'released'⌋

[SWS_CanNm_00144] ⌈When initialized, by default, the CanNm module shall enter the Bus-Sleep Mode.⌋

[SWS_CanNm_00060] ⌈The function `CanNm_Init` shall select the active configuration set by means of a configuration pointer parameter being passed (see `CanNm_Init`).⌋

[SWS_CanNm_00061] ⌈If `CanNmGlobalPnSupport` is set to TRUE and CanNm is initialized (call of `CanNm_Init`) then CanNm shall stop the NM Message Tx Timeout Timer.⌋

[SWS_CanNm_00023] ⌈During initialization the CanNm module shall deactivate the bus load reduction.⌋

[SWS_CanNm_00033] ⌈After initialization the CanNm module shall stop the transmission of Network Management PDUs by stopping the Message Cycle Timer.⌋

Note: If `CanNmPassiveModeEnabled` is set to TRUE the CanNm Message Cycle is not needed, because no Network Management PDUs are transmitted by such nodes.

[SWS_CanNm_00025] ⌈During initialization the CanNm module shall set each byte of the user data to 0xFF.⌋

[SWS_CanNm_00085] ⌈During initialization the CanNm module shall set the Control Bit Vector to 0x00.⌋

[SWS_CanNm_00500] ⌈During initialization and if `CanNmPnEnabled` is TRUE, the CanNm module shall set each byte of the PNC bit vector to 0x00.⌋

[SWS_CanNm_00511]

Upstream requirements: RS_Nm_02517, RS_Nm_02519

⌈If `CanNmSynchronizedPncShutdownEnabled` is set to TRUE, the CanNm module shall consider transmission of PN shutdown message as inactive after initialization .⌋

7.5 Execution

7.5.1 Processor architecture

[SWS_CanNm_00146] ⌈The AUTOSAR CanNm 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

[SWS_CanNm_00246] ⌈The configuration parameter `CanNmTimeoutTime` shall determine the AUTOSAR CanNm timing parameter NM-Timeout Time.⌋

[SWS_CanNm_00247] ⌈The configuration parameter `CanNmRepeatMessageTime` shall determine the AUTOSAR CanNm timing parameter Repeat Message Time.⌋

[SWS_CanNm_00248] ⌈The configuration parameter `CanNmWaitBusSleepTime` shall determine the AUTOSAR CanNm timing parameter Wait Bus-Sleep Time.⌋

[SWS_CanNm_00249] ⌈The configuration parameter `CanNmRemoteSleepIndTime` shall determine the AUTOSAR CanNm timing parameter Remote Sleep Indication Time.⌋

7.6 Network Management PDU Structure

The figure below shows the format of the Network Management PDU for an example with 8 bytes where Source Node Identifier (SNI) is located in the first byte and the Control Bit Vector (CBV) at the second byte, user data is used and partial network is enabled. User data range is located between the system bytes and the PNC bit vector:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 7	PNC bit vector – byte 3							
Byte 6	PNC bit vector – byte 2							
Byte 5	PNC bit vector – byte 1							
Byte 4	PNC bit vector – byte 0							
Byte 3	User data 1							
Byte 2	User data 0							

▽

△

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Control Bit Vector (CBV)							
Byte 0	Source Node Identifier (SNI)							

Table 7.1: Network Management PDU Format Example

[SWS_CanNm_00074]

Upstream requirements: RS_Nm_02505, RS_Nm_02541

⌈The location of the Source Node Identifier shall be configurable by means of CanNmPduNidPosition to Byte 0, Byte 1, or off.⌋

Note: Setting the CanNmPduNidPosition to off means that in the NM PDU no space is occupied by the Source Node Identifier. Hence one more byte is available for user data or PNC bit vector.

[SWS_CanNm_00075]

Upstream requirements: RS_Nm_02541

⌈The location of the Control Bit Vector shall be configurable by means of CanNmPduCbvPosition to Byte 0, Byte 1, or off.⌋

Note:

- Setting the CanNmPduCbvPosition to off means that in the NM PDU no space is occupied by the CBV. Hence one more byte is available for user data.
- The location of the PNC bit vector is configurable by means of NmPncBitVectorOffset and NmPncBitVectorLength of the corresponding NM-channel. The location of the PNC bit vector is placed after the system bytes (CBV and SNI) and within the PduLength of the NM-PDU.

[SWS_CanNm_00501]

Upstream requirements: RS_Nm_02541

⌈The remaining bytes not assigned to Nm System Bytes or PNC bit vector shall be available for User Data.⌋

Note: According to [10] ([TPS_SYST_03069], [TPS_SYST_03070], [TPS_SYST_03071], and [TPS_SYST_03072]) the use and location of user data is configurable. If user data is used, the user data is placed within the PduLength of the NM-PDU and does not overlap with the range of system bytes or PNC bit vector. If partial network functionaliy is enabled (CanNmPnEnabled is set to TRUE) and user data is used, the user data range is exclusively located either between the system bytes and the PNC bit vector or between the PNC bit vector and the end of the NM-PDU. The length of user data range has to be calculated according to the following restrictions:

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

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

Note: The length of the Network Management PDU is defined by the PduLength parameter in the “global” EcuC module ([ECUC_EcuC_00003], see Specification of ECU Configuration [11]).

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

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CBV	Reserved	Partial Network Information Bit	Partial Network Learning Bit	Active Wakeup Bit	NM Coordinator Sleep Ready Bit	Reserved	PN Shutdown Request Bit	Repeat Message Request

Table 7.2: Network Management PDU – Control Bit Vector (CBV)

Note: Bits 1 and 2 were used in R3.2 as NM Coordinator ID (Low Bit)

Note: The CBV is initialized with 0x00 during initialization (also refer to [SWS_CanNm_00085]).

[SWS_CanNm_00013]

Upstream requirements: RS_Nm_02503

⌈The CanNm module shall set the Source Node Identifier with the configuration parameter CanNmNodeId unless CanNmPduNidPosition is set to off.⌋

[SWS_CanNm_00401] ⌈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.⌋

[SWS_CanNm_00402] ⌈If the CanNm module leaves the Network Mode and CanNmActiveWakeupBitEnabled is TRUE, the CanNm module shall clear the Active WakeupBit in the CBV.⌋

7.7 Communication Scheduling

7.7.1 Transmission

The Network Management PDUs transmission ability is configurable by means of CanNmPassiveModeEnabled (refer also to Section 7.9.3). The transmission mechanisms described in this chapter are only relevant if CanNmPassiveModeEnabled is FALSE.

Also the Network Management PDU transmission ability can be enabled and disabled by communication control service (refer to Section 7.9.6). The transmission mechanisms described in this chapter are only relevant if the Network Management PDU transmission ability is enabled.

[SWS_CanNm_00237] ⌈The CanNm module shall provide the periodic transmission mode. In this transmission mode the CanNm module shall send Network Management PDUs periodically.⌋

[SWS_CanNm_00238] ⌈The CanNm module shall optionally provide the periodic transmission mode with bus load reduction. In this transmission mode the CanNm module shall transmit Network Management PDUs due to a specific algorithm.⌋

The periodic transmission mode is used in “Repeat Message State” and “Normal Operation State”. Periodic transmission mode with bus load reduction is only available in “Normal Operation State”

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.

[SWS_CanNm_00071]

Upstream requirements: RS_Nm_00142

⌈The immediate transmission confirmation mechanism shall be configurable by means of the CanNmImmediateTxconfEnabled.⌋

Note: The immediate transmission confirmation mechanism is used for systems which don’t want to use the actual confirmation from the lower layer (e.g. CanIf).

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

Moreover, implementation of superfluous actual transmission confirmation in such a system only for one NM message would mean a significant performance loss regarding the execution time of the overall CAN Interface/Driver layer making the calculated time schedule inefficient.

[SWS_CanNm_00005] ⌈If the Repeat Message State is not entered via CanNm_NetworkRequest OR CanNmImmediateNmTransmissions is zero the transmission of NM PDU shall be delayed by CanNmMsgCycleOffset after entering the repeat message state.⌋

[SWS_CanNm_00334] ⌈When entering the Repeat Message State from Bus Sleep Mode or Prepare Bus Sleep Mode because of CanNm_NetworkRequest() (active wakeup) and if 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.⌋

[SWS_CanNm_00006] ⌈If Normal Operation State is entered from Ready Sleep State the transmission of NM PDUs shall be started immediately.⌋

[SWS_CanNm_00454] ⌈If CanNmPnHandleMultipleNetworkRequests is set to TRUE CanNm_NetworkRequest shall trigger a state transition from Network Mode to Repeat Message state. If PDU transmission ability is enabled the NM PDUs shall be transmitted using 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.⌋

Note: CanNmImmediateNmTransmissions has to be greater zero in this case due to [ECUC_CanNm_00056].

[SWS_CanNm_00335] ⌈If NM PDUs shall be transmitted with CanNmImmediateNmCycleTime (See [SWS_CanNm_00334] and [SWS_CanNm_00454]), CanNm shall ensure that CanNmImmediateNmTransmissions (including first immediate transmission) with this timing are requested succesfully. If a transmission request to the lower layer (e.g. CanIf) fails (E_NOT_OK is returned), CanNm shall retry the transmission request in the next main function. Afterwards CanNm shall continue transmitting NM PDUs 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).

[SWS_CanNm_00512] ⌈If transmission of Network Management PDUs has been started, the CanNm Message Cycle Timer expires and when CanNmSynchronizedPncShutdownEnabled is set to either FALSE or if set to TRUE and additionally the transmission of PN shutdown messages is inactive, then the CanNm module shall transmit a Network Management PDU by calling LSduR_CanNmTransmit.⌋

[SWS_CanNm_00513]

Upstream requirements: RS_Nm_02572

⌈If transmission of Network Management PDUs has been started, the CanNm Message Cycle Timer expires and when CanNmSynchronizedPncShutdownEnabled is set to TRUE and the transmission of PN shutdown messages is active, the transmission of this NM PDU shall be postponed to the next CanNm_MainFunction call.⌋

Note:

- NM-PDU transmitted as PN Shutdown message has to be sent immediately and therefore processing of cylic NM-PDUs transmitted with CanNmMsgCycleTime have to be delayed. In rare cases this could lead to a delay of more than one main function cycle time.
- The NM timing has to consider that an NM message transmitted with CanNmMsgCycleTime may be delayed for more than one main function cycle time. Therefore, the following condition has to be fulfilled to tolerate multiple delays of those NM Messages:

  (NmPnResetTime – CanNmMsgCycleTime) > n * CanNmMainFunctionPeriod, where n denotes the number of tolerated delays before the PnResetTime expires, if no NM message is received.

[SWS_CanNm_00514]

Upstream requirements: RS_Nm_02573

⌈If the CanNm module has requested a transmission of a NM-PDU, CanNmSynchronizedPncShutdownEnabled is set to TRUE, the transmission of PN shutdown messages is active, CanNm_TxConfirmation is called with result E_NOT_OK or the transmission request for this NM-PDU was not accepted (LSduR_CanNmTransmit returned E_NOT_OK), then the CanNm module shall perform a retransmission of a NM-PDU for this NM-Channel in the next main function call.⌋

Note:

- CanNm has to perform a retry transmission handling for a NM-PDU in the context of the main function calls, if the transmission of PN shutdown messages is active and if the transmission of this NM-PDU was not accepted or was not confirmed by the lower layer. The retry transmission requests should cover error cases, where the lower layer cannot transmit the Nm messages.

- The dependency to a pending transmission confirmation indicated by the lower layer, should support reliable communication, e.g. ensure PN shutdown message was transmitted on the bus or avoid transmissions of outdated PN shutdown messages, if for example queueing in the lower layer is configured.

[SWS_CanNm_00040] ⌈If the CanNm Message Cycle Timer expires the CanNm module shall restart with CanNmMsgCycleTime.⌋

[SWS_CanNm_00051] ⌈If transmission of Network Management PDUs has been stopped the CanNm module shall cancel the Message Cycle Timer.⌋

7.7.2 Reception

If a NM PDU has been successfully received, the lower layer (e.g. CanIf) will inform CanNm via the function call CanNm_RxIndication.

[SWS_CanNm_00035]

Upstream requirements: SRS_BSW_00483

⌈On the call of the callback function CanNm_RxIndication, the CanNm module shall copy the data of the Network Management PDU referenced in the function parameter to an internal buffer.⌋

7.8 Bus Load Reduction Mechanism

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

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

1. If a Network Management PDU is received the CanNm Message Cycle Timer is reloaded with the node specific timing parameter CanNmMsgReducedTime.

   The node specific time CanNmMsgReducedTime should be greater than 0.5 * CanNmMsgCycleTime and less than CanNmMsgCycleTime.

2. If a Network Management PDU is been transmitted the CanNm Message Cycle Timer is reloaded with the network management cluster specific timing parameter CanNmMsgCycleTime.

This leads to the following behavior:

Only the two nodes with the smallest CanNmMsgReducedTime time transmit alternating Network Management PDUs on the network. If one of the nodes stops transmission, the node with the next smallest CanNmMsgReducedTime time will start to transmit Network Management PDUs. If there is only one node on the network that requires bus communication, one Network Management PDU per CanNmMsgCycleTime is transmitted.

The algorithm ensures that the bus load is limited to a maximum two Network Management PDUs per CanNmMsgCycleTime.

An example can be found in Appendix A.

[SWS_CanNm_00052]

Upstream requirements: RS_Nm_00142

⌈The bus load reduction mechanism shall be statically configurable by means of the CanNmBusLoadReductionEnabled parameter.⌋

[SWS_CanNm_00156]

Upstream requirements: RS_Nm_00142

⌈When the Repeat Message State is entered from Bus-Sleep Mode, Prepare Bus-Sleep Mode, Normal Operation or Ready Sleep State the CanNm module shall deactivate the busload reduction.⌋

[SWS_CanNm_00157]

Upstream requirements: RS_Nm_00142

⌈When the Normal Operation State is entered from Repeat Message State or Ready Sleep State and CanNmBusLoadReductionEnabled is TRUE the CanNm module shall activate the busload reduction.⌋

[SWS_CanNm_00069]

Upstream requirements: RS_Nm_00142

⌈If the bus load reduction mechanism is globally enabled (CanNmBusLoadReductionEnabled is TRUE), for a particular network activated, PDU transmission ability is enabled and the function CanNm_RxIndication is called for this network, the CanNm module shall restart the CanNm Message Cycle Timer with the node specific time CanNmMsgReducedTime.⌋

7.9 Additional features

7.9.1 Detection of Remote Sleep Indication

The “Remote Sleep Indication” denotes a situation, where a node in Normal Operations States finds all other nodes in the cluster are ready to sleep (in Ready-Seep State). The node in Normal Operation State will still keep the bus awake.

[SWS_CanNm_00149] ⌈Detection of remote sleep indication shall be statically configurable with use of the CanNmRemoteSleepIndEnabled switch (configuration parameter).⌋

[SWS_CanNm_00150]

Upstream requirements: RS_Nm_00052

⌈If the CanNm module receives no Network Management PDUs in the Normal Operation State for a configurable amount of time determined by CanNmRemoteSleepIndTime (configuration parameter), the CanNm module shall call the callback function Nm_RemoteSleepIndication.⌋

With a call of Nm_RemoteSleepIndication CanNm notifies the module Nm that all nodes in the cluster are ready to sleep (the so-called ‘Remote Sleep Indication’).

[SWS_CanNm_00151]

Upstream requirements: RS_Nm_02509

⌈If Remote Sleep Indication has been previously detected and if a Network Management PDU is received in the Normal Operation State or Ready Sleep State again, the module CanNm shall call the callback function Nm_RemoteSleepCancellation.⌋

[SWS_CanNm_00152]

Upstream requirements: RS_Nm_02509

⌈If Remote Sleep Indication has been previously detected and if Repeat Message State is entered from Normal Operation State or Ready Sleep State, the module CanNm shall call the callback function Nm_RemoteSleepCancellation.⌋

With a call of Nm_RemoteSleepCancellation CanNm notifies the module Nm that some nodes in the cluster are not ready to sleep anymore (the so-called ‘Remote Sleep Cancellation’).

[SWS_CanNm_00154] ⌈When the service CanNm_CheckRemoteSleepIndication is called and the state is Bus-Sleep Mode, Prepare Bus-Sleep Mode or Repeat Message State the CanNm module shall not execute the service and shall return E_NOT_OK.⌋

7.9.2 User Data

NM user data in CanNm can be accessed for transmission in two ways: Either by writing data within the API CanNm_SetUserData (CanNmUserDataEnabled needs to be TRUE) or by writing according Com signals (CanNmComUserDataSupport needs to be TRUE). In second case the first option is not available.

Reading NM user data from received CanNm messages is only possible with the APIs CanNm_GetUserData or CanNm_GetPduData (CanNmUserDataEnabled needs to be TRUE).

[SWS_CanNm_00159]

Upstream requirements: RS_Nm_02503, SRS_BSW_00483

⌈When CanNm_SetUserData is called the CanNm module shall set the Network Management user data for the Network Management PDUs transmitted next on the bus.⌋

[SWS_CanNm_00160]

Upstream requirements: RS_Nm_02504, SRS_BSW_00483

⌈When CanNm_GetUserData is called CanNm module shall return the Network Management user data of the most recently received Network Management PDU.⌋

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

7.9.2.1 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.

[SWS_CanNm_00327] ⌈If CanNmComUserDataSupport is enabled the API CanNm_SetUserData shall not be available.⌋

[SWS_CanNm_00328]

Upstream requirements: RS_Nm_02503

⌈If CanNmComUserDataSupport is enabled and NM-PDU is not configured for triggered transmission in the lower layer (e.g. CanIf: CanIfTxPduTriggerTransmit set to FALSE) 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.⌋

Note: In case of triggered transmission no data is needed at the transmission request, just the length is needed. The data will be collected within CanNm_TriggerTransmit.

[SWS_CanNm_00450] ⌈If CanNmComUserDataSupport is enabled and PduR_CanNmTriggerTransmit returns E_NOT_OK, the NM shall use the last transmitted value for NmUserData.⌋

Note: The transmission of outdated NM data can be avoided by not stopping the IPdu in COM used for NmUserData transmission.

[SWS_CanNm_00329] ⌈If CanNmComUserDataSupport is enabled and CanNm_TxConfirmation is called CanNm shall forward the transmission confirmation result to PduR by calling PduR_CanNmTxConfirmation.⌋

[SWS_CanNm_00332] ⌈If CanNmComUserDataSupport is enabled and the number of available user data bytes does not match to the length of the referenced I-PDU an error shall be reported at generation time.⌋

7.9.3 Passive Mode

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

[SWS_CanNm_00161]

Upstream requirements: RS_Nm_02511

⌈Passive Mode shall be statically configurable with use of the CanNmPassiveModeEnabled switch (configuration parameter).⌋

Note: Passive Mode has to be either enabled or disabled for all NM networks within one ECU.

7.9.4 Network Management PDU Rx Indication

[SWS_CanNm_00037] ⌈On the call of the callback function CanNm_RxIndication, the CanNm module shall call the Nm callback function Nm_PduRxIndication, if and only if CanNmPduRxIndicationEnabled (configuration parameter) is set to TRUE.⌋

7.9.5 State change notification

[SWS_CanNm_00166]

Upstream requirements: RS_Nm_00051

⌈All changes of the AUTOSAR CanNm states shall be notified to the upper layer by calling Nm_StateChangeNotification if the callback Nm_StateChangeNotification is enabled (configuration parameter CanNmStateChangeIndEnabled is TRUE).⌋

7.9.6 Communication Control

Note: Communication Control is statically configurable by the configuration parameter CanNmComControlEnabled.

[SWS_CanNm_00170]

Upstream requirements: RS_Nm_02512

⌈If the service CanNm_DisableCommunication is called the CanNm module shall disable the Network Management PDU transmission ability.⌋

Note: This behavior shall also be applied in Repeat Message State. Communication Control feature does not influence the duration of the Repeat Message State.

[SWS_CanNm_00173]

Upstream requirements: RS_Nm_02512

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

[SWS_CanNm_00174] ⌈When the Network Management PDU transmission ability is disabled, the CanNm module shall stop the NM-Timeout Timer.⌋

[SWS_CanNm_00175] ⌈When the Network Management PDU transmission ability is disabled, the CanNm module shall stop the Remote Sleep Indication Detection.⌋

[SWS_CanNm_00178]

Upstream requirements: RS_Nm_02512

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

[SWS_CanNm_00179] ⌈When the Network Management PDU transmission ability is enabled, the CanNm module shall restart the NM-Timeout Timer.⌋

[SWS_CanNm_00180] ⌈If CanNmRemoteSleepIndEnabled is TRUE and the Network Management PDU transmission ability is enabled, the CanNm module shall restart the Remote Sleep Indication Detection.⌋

[SWS_CanNm_00181] ⌈The service CanNm_RequestBusSynchronization shall return E_NOT_OK if the Network Management PDU transmission ability is disabled.⌋

7.9.7 Coordinator Synchronization Support

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

[SWS_CanNm_00341] ⌈If CanNmCoordinatorSyncSupport is set to TRUE and 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.⌋

[SWS_CanNm_00348] ⌈If CanNmCoordinatorSyncSupport is set to TRUE and 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.⌋

[SWS_CanNm_00342] ⌈If CanNmCoordinatorSyncSupport is set to TRUE and the API CanNm_SetSleepReadyBit is called CanNm shall set the “NM Coordinator Sleep ready Bit” to the passed value and trigger a single Network Management PDU.⌋

7.10 Car Wakeup

[SWS_CanNm_00405] ⌈The position of the Car Wakeup bit in the NM-PDU is defined by the configuration parameters CanNmCarWakeUpBytePosition and CanNmCarWakeUpBitPosition.⌋

7.10.1 Rx Path

[SWS_CanNm_00406] ⌈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.⌋

[SWS_CanNm_00407] ⌈If CanNm_GetPduData is called in the context of Nm_CarWakeUpIndication and if CanNmNodeDetectionEnabled or CanNmUserDataEnabled or CanNmNodeIdEnabled is set to TRUE 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.

[SWS_CanNm_00408] ⌈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.

7.10.2 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.

7.11 Partial Networking

An overview regarding the partial network cluster functionality can be found in document Guide to Mode Management [12].

7.11.1 Rx Handling of NM PDUs

[SWS_CanNm_00409]

Upstream requirements: RS_Nm_02517

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

[SWS_CanNm_00410]

Upstream requirements: RS_Nm_02517, RS_Nm_02527

⌈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 Indication handling omitting the extensions for partial networking.⌋

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

[SWS_CanNm_00411]

Upstream requirements: RS_Nm_02517, RS_Nm_02527

⌈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.⌋

[SWS_CanNm_00502]

Upstream requirements: RS_Nm_02517, RS_Nm_02527, RS_Nm_02565

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

- CanNmSynchronizedPncShutdownEnabled is set to FALSE
- CanNmSynchronizedPncShutdownEnabled is set to TRUE and the PNSR bit is set to 0

then the CanNm module shall extract the PNC bit vector from the received NM-PDU according to the partial network configuration (NmPncBitVectorOffset and NmPncBitVectorLength of the corresponding NM-channel) and forward the PNC bit vector by calling Nm_PncBitVectorRxIndication.⌋

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

[SWS_CanNm_00503]

Upstream requirements: RS_Nm_02517, RS_Nm_02527

⌈If CanNmPnEnabled is set to TRUE and Nm_PncBitVectorRxIndication was called, a received NM PDU shall only be considered for further processing under the following conditions:

- CanNmAllNmMessagesKeepAwake is set to TRUE OR
- CanNm_ConfirmPnAvailability has not been called yet OR
- the output value of RelevantPncRequestDetectedPtr is set to TRUE

⌋

Note:

- CanNmAllNmMessagesKeepAwake is required to enable a gateway to stay awake on any kind of NM-PDU.
- If PN availability was not confirmed by CanSM, all PNC requests are considered as relevant and therefore the Nm restarts the NM-Timeout Timer when receiving a NM-PDU. This is required to allow a malfunctioning partial network depending hardware (e.g. PN capable CAN transceiver) to shut down synchronously with the remaining network.
- As consequence of [SWS_CanNm_00503], a NM PDU is not considered for further processing if not all messages shall keep the ECU awake and the PN availability was confirmed but no relevant PNC bit vector was detected.

Example:

- CanNmPduCbvPosition = 0
- CanNmPduNidPosition = 1
- NmPncBitVectorOffset = 4
- NmPncBitVectorLength = 4
- Calculated length of user data range = 2

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

Byte 4 to Byte 7 of the NM PDU contain the PNC bit vector:

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

Table 7.3: Example NM PDU containing relevant PNC bit vector

[SWS_CanNm_00461]

Upstream requirements: RS_Nm_02544

⌈If CanNmSynchronizedPncShutdownEnabled is set to TRUE, when a NM PDU is received where PNI bit and PNSR bit are 1 and the corresponding ComM-Channel configured via CanNmComMNetworkHandleRef is actively coordinated (ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE), CanNm module shall report the runtime error CANNM_E_INVALID_PN_SYNC_SHUTDOWN_REQUEST to DET, ignore the PNSR bit and handle the PDU as usual NM PDU.⌋

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

[SWS_CanNm_00504] ⌈If CanNmSynchronizedPncShutdownEnabled is TRUE, the PNI bit in the received NM-PDU is set to 1 and the PNSR bit is set to 1, CanNm module shall extract the PNC bit vector from the received NM-PDU according to the partial network configuration (NmPncBitVectorOffset and NmPncBitVectorLength of the corresponding NM-channel) and forward the PNC bit vector by calling Nm_ForwardSynchronizedPncShutdown.⌋

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

7.11.2 Tx Handling of NM PDUs

[SWS_CanNm_00413]

Upstream requirements: RS_Nm_02517, RS_Nm_02519

⌈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.

[SWS_CanNm_00414]

Upstream requirements: RS_Nm_02517, RS_Nm_02519

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

[SWS_CanNm_00515]

Upstream requirements: RS_Nm_02571

⌈If CanNmGlobalPnSupport is set to TRUE, the CanNm module shall store the latest PNC bit vector per NM-channel everytime the PNC bit vector has been fetched from the Nm modul via call of Nm_PncBitVectorTxIndication.⌋

[SWS_CanNm_00516]

Upstream requirements: RS_Nm_02571

⌈If CanNmGlobalPnSupport is set to TRUE, a NM-PDU has been transmitted on a NM-Channel and CanNm_TxConfirmation is called with result E_OK for this NM-PDU, then the CanNm shall forward the confirmation to Nm by calling Nm_PncBitVectorTxConfirmation with the stored PNC bit vector (see [SWS_CanNm_00515]) for this NM-channel with result set to E_OK.⌋

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

[SWS_CanNm_00517]

Upstream requirements: RS_Nm_02571

⌈If CanNmGlobalPnSupport is set to TRUE, a NM-PDU has been transmitted on a NM-Channel and CanNm_TxConfirmation is called with result E_NOT_OK or the transmission request for this NM-PDU was not accepted (LSduR_CanNmTransmit returned E_NOT_OK) for this NM-PDU, then the CanNm module shall forward the confirmation to Nm by calling Nm_PncBitVectorTxConfirmation with the stored PNC bit vector (see [SWS_CanNm_00515]) for this NM-Channel with result set to E_NOT_OK.⌋

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

[SWS_CanNm_00518]

Upstream requirements: RS_Nm_02517, RS_Nm_02519

⌈If CanNmPnEnabled is TRUE and a NM-PDU has to be transmitted (either as cylic NM-PDU transmitted with CanNmMsgCycleTime (see [SWS_CanNm_00512]) or as PN shutdown message), the CanNm module shall additionally fetch the PNC bit vector by calling Nm_PncBitVectorTxIndication and copy the PNC bit vector with respect to NmPncBitVectorOffset and NmPncBitVectorLength of the corresponding NM-channel to the NM-PDU before requesting the transmission of the NM-PDU.⌋

Note:

- The transmission of a NM-PDU has to consider user data if the usage of user data is configured. Please refer to Section 7.9.2 User Data.
- PNC bit vector is always fetched up front to a transmission request independent if NM-PDU is configured for triggered transmission or not in the lower layer (e.g. CanIf: CanIfTxPduTriggerTransmit set to TRUE or FALSE). This should ensure to re-start the PN reset timer of the affected PNC in the Nm on a transmission request.

[SWS_CanNm_00519]

Upstream requirements: RS_Nm_02540, RS_Nm_02548, RS_Nm_02572

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

Note: Transmission of PNC shutdown message is processed with higher priority, due to [SWS_CanNm_00513]. Cyclic NM messages are not transmitted in the same main function as the synchronized PNC shutdown message. They are delayed to the next mainfunction cycle as long as synchronized PNC shutdown requests are pending.

7.11.3 Handling of Internal Requested Partial Network Clusters

All internal PNC requests are maintained by ComM. ComM forwards the aggregated internal PNC requests per channel as PNC bit vector to Nm. This PNC bit vector carries the so-called “Internal Request Array”. The CanNm has to retrieve the latest IRA from Nm every time an NM PDU is transmitted. Nm provides the IRA information to CanNm and updates the PNC reset timer (each time a relevant PNC is transmitted, the PNC reset timer is re-started).

Note: For all configured NM-channel where CanNmPnEnabled is set TRUE, the CanNm will call Nm_PncBitVectorTxIndication(<NM-channel>, < buffer to store the unfiltered PNC bit vector of aggregated internal PNC requests>) (see [SWS_CanNm_00518], [SWS_CanNm_00521] and [SWS_CanNm_00523]) to indicate the transmission and to retrieve the current internal PNC requests as PNC bit vector with respect to the configured NmPncBitVectorLength. The CanNm will copy received internal PNC requests to the PNC bit vector bytes of the NM-PDU.

7.11.4 Spontaneous Transmission of NM PDUs via CanNm_NetworkRequest

[SWS_CanNm_00444]

Upstream requirements: RS_Nm_02517, RS_Nm_02528

⌈If CanNm_NetworkRequest is called, CanNmPnHandleMultipleNetworkRequests is 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.⌋

Note: If CanNmPnHandleMultipleNetworkRequests is set to TRUE the CanNm feature 'Immediate Transmission' is mandatory.

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

7.12 Transmission Error Handling

Depending on configuration the CanNm will evaluate the confirmation function that a Network Management PDU has been successfully transmitted or not. CanNm will monitor these confirmations and alarm the upper layers if a transmission confirmation is received with result E_NOT_OK or not received within a specific amount of time. Timeout Monitoring is required for Partial Networking to ensure that the first message gets acknowledged when all ECUs on the network use Partial Network transceivers. Otherwise CanSM is informed and will restart CAN Driver (see also SWS CanSM [13]).

[SWS_CanNm_00073] ⌈If CanNmPassiveModeEnabled is set to TRUE or CanNmImmediateTxconfEnabled is set to TRUE CanNm shall not perform transmission error handling and omit the requirements [SWS_CanNm_00061], [SWS_CanNm_00064], [SWS_CanNm_00065], [SWS_CanNm_00066] and [SWS_CanNm_00446].⌋

Rationale: Transmission error handling makes only sense if a node is allowed to transmit Network Management PDUs and the real confirmation from the lower layer (e.g. CanIf) is evaluated.

[SWS_CanNm_00064]

Upstream requirements: RS_Nm_00137

⌈If CanNmGlobalPnSupport is set to TRUE and CanNmMsgTimeoutTime is defined and CanNm requests the transmission of a NM PDU (call of LSduR_CanNmTransmit) then CanNm shall start the NM Message Tx Timeout Timer with CanNmMsgTimeoutTime.⌋

[SWS_CanNm_00065]

Upstream requirements: RS_Nm_00137

⌈If CanNmGlobalPnSupport is set to TRUE and CanNmMsgTimeoutTime is defined and CanNm_TxConfirmation is called then CanNm shall stop the NM Message Tx Timeout Timer.⌋

[SWS_CanNm_00066]

Upstream requirements: RS_Nm_00137

⌈If CanNm_TxConfirmation is called with result E_NOT_OK or if CanNmGlobalPnSupport is set to TRUE and NM Message Tx Timeout Timer has expired then CanNm shall call the function Nm_TxTimeoutException.⌋

[SWS_CanNm_00446]

Upstream requirements: RS_Nm_00137

⌈If CanNmGlobalPnSupport is set to TRUE and NM Message Tx Timeout Timer has expired then CanNm shall call the function CanSM_TxTimeoutException.⌋

7.13 Functional requirements on CanNm API

[SWS_CanNm_00014]

Upstream requirements: RS_Nm_00153

⌈If CanNmRepeatMsgIndEnabled is set to TRUE and the Repeat Message Request bit is received CanNm module shall call the callout function Nm_RepeatMessageIndication only the first time until Repeat Message State has been left again. In case the Partial Network Learning Bit is also received with value 1 and CanNmDynamicPncToChannelMappingEnabled is set to TRUE the parameter pnLearningBit Set shall be set to TRUE in this function call, otherwise to FALSE.⌋

Note: When Repeat Message Bit is received NM will enter or restart Repeat Message State, but the bits will still be received as requestor will send until he leaves Repeat Message State to be fault-tolerant regarding possible loss of messages. State Change and callout are only needed once the first time the node received it.

[SWS_CanNm_00086] ⌈If CanNmUserDataEnabled is enabled but no user data bytes are available, the CanNm module shall raise an error during configuration or compilation time.⌋

7.14 Error Classification

Section “Error Handling” of the document [4] “General Specification of Basic Software Modules” describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.14.1 Development Errors

[SWS_CanNm_00316] Definiton of development errors in module CanNm

Upstream requirements: SRS_BSW_00385, SRS_BSW_00480, SRS_BSW_00481, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Error value*
API service used without module initialization	CANNM_E_UNINIT	0x01
API service called with wrong channel handle	CANNM_E_INVALID_CHANNEL	0x02
API service called with wrong PDU-ID	CANNM_E_INVALID_PDUID	0x03
CanNm initialization has failed, e.g. selected configuration set doesn’t exist.	CANNM_E_INIT_FAILED	0x05
Null pointer has been passed as an argument	CANNM_E_PARAM_POINTER	0x12
DeInit API service called when not all CAN networks are in Bus Sleep mode	CANNM_E_NOT_IN_BUS_SLEEP	0x13

⌋

7.14.2 Runtime Errors

[SWS_CanNm_00317] Definiton of runtime errors in module CanNm

Upstream requirements: SRS_BSW_00385, SRS_BSW_00452

⌈

Type of error	*Related error code*	*Error value*
Reception of NM PDUs in Bus-Sleep Mode.	CANNM_E_NET_START_IND	0x04
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 Network Management PDU can not be transmitted in time; this race condition applies to event-triggered systems	CANNM_E_NETWORK_TIMEOUT	0x11
A NM message with PN Shutdown Request Bit was received on a channel that is actively coordinated by the ComM PNC Gateway.	CANNM_E_INVALID_PN_SYNC_SHUTDOWN_ REQUEST	0x20

⌋

7.14.3 Transient Faults

There are no transient faults.

7.14.4 Production Errors

There are no production errors.

7.14.5 Extended Production Errors

There are no extended production errors.

7.15 Scheduling of the main function

For details refer to the chapter 8.5 “Scheduled functions” in SWS_BSWGeneral [4].

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

[SWS_CanNm_00185] ⌈Support of bus synchronization on demand shall be statically configurable with use of the `CanNmBusSynchronizationEnabled` switch (configuration parameter).⌋

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

7.17 Summary of CanNm Timing Requirements

This section gives a summary of the CanNm timing requirements. Please note that this chapter is a summary only and does not replace or act as requirement. Moreover this section does not require any specific way of implementation

Type of timing	Requirements
Nm timeout related	[SWS_CanNm_00061] [SWS_CanNm_00096] [SWS_CanNm_00098] [SWS_CanNm_00099] [SWS_CanNm_00101] [SWS_CanNm_00109] [SWS_CanNm_00117] [SWS_CanNm_00174] [SWS_CanNm_00179] [SWS_CanNm_00193] [SWS_CanNm_00194] [SWS_CanNm_00206]
Tx confirmation timeout related	[SWS_CanNm_00064] [SWS_CanNm_00065] [SWS_CanNm_00066]
NmPdu transmission related	[SWS_CanNm_00005] [SWS_CanNm_00040] [SWS_CanNm_00051] [SWS_CanNm_00061] [SWS_CanNm_00069] [SWS_CanNm_00173] [SWS_CanNm_00178] [SWS_CanNm_00512]
Remote sleep indication related	[SWS_CanNm_00175] [SWS_CanNm_00180]

7.18 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 / Dynamic PNC-to-channel-mapping related transitions in blue.

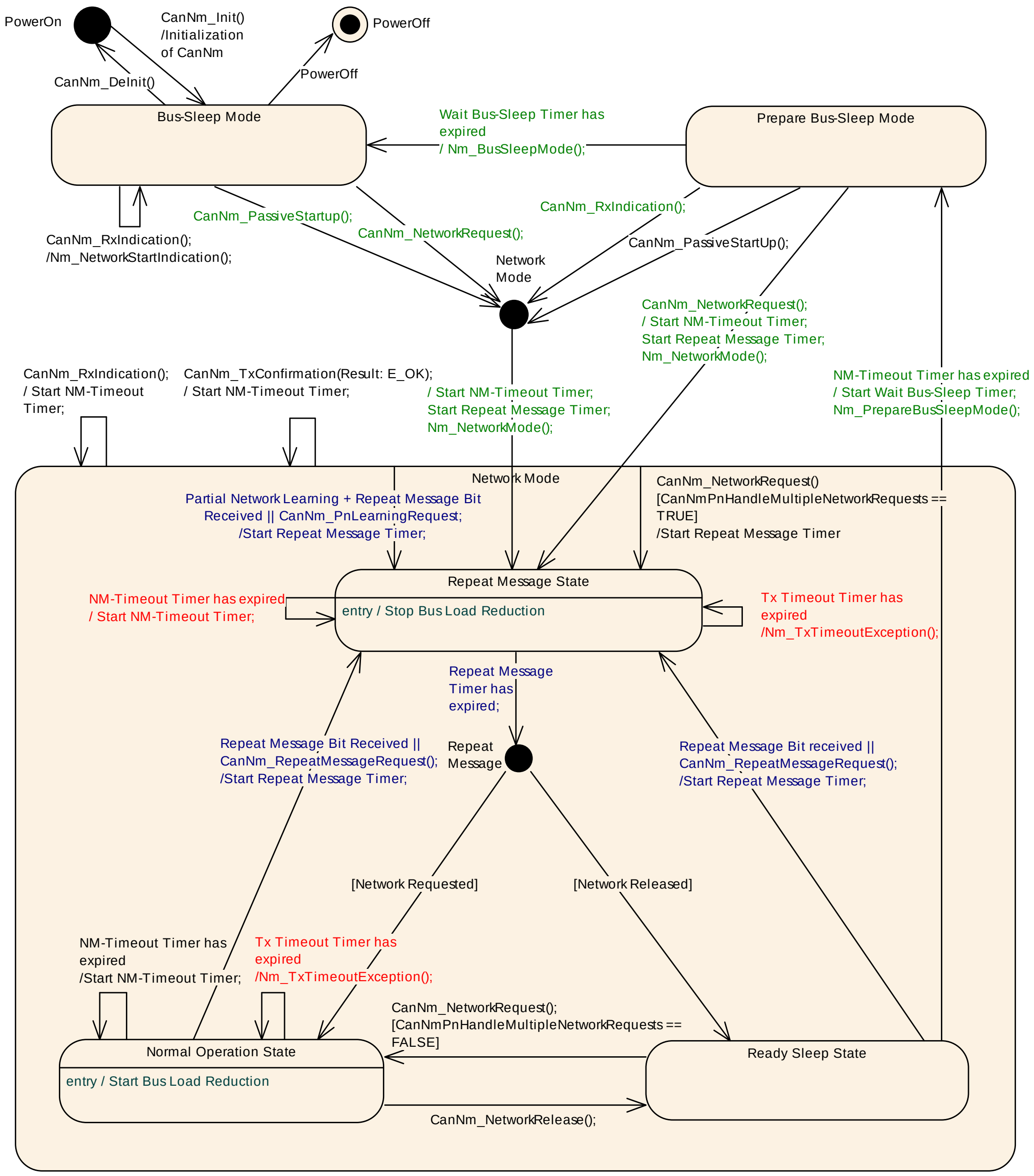

PowerOn
CanNm_Init()
/Initialization
of CanNm
PowerOff
CanNm_DeInit()
PowerOff
Bus-Sleep Mode
Wait Bus-Sleep Timer has
expired
/ Nm_BusSleepMode();
Prepare Bus-Sleep Mode
CanNm_RxIndication();
/Nm_NetworkStartIndication();
CanNm_PassiveStartup();
CanNm_NetworkRequest();
CanNm_RxIndication();
CanNm_PassiveStartUp();
Network
Mode
CanNm_NetworkRequest();
/ Start NM-Timeout Timer;
Start Repeat Message Timer;
Nm_NetworkMode();
NM-Timeout Timer has expired
/ Start Wait Bus-Sleep Timer;
Nm_PrepareBusSleepMode();
CanNm_RxIndication();
/ Start NM-Timeout
Timer;
CanNm_TxConfirmation(Result: E_OK);
/ Start NM-Timeout Timer;
/ Start NM-Timeout Timer;
Start Repeat Message Timer;
Nm_NetworkMode();
Network Mode
Partial Network Learning + Repeat Message Bit
Received || CanNm_PnLearningRequest;
/Start Repeat Message Timer;
CanNm_NetworkRequest()
[CanNmPnHandleMultipleNetworkRequests ==
TRUE]
/Start Repeat Message Timer
Repeat Message State
entry / Stop Bus Load Reduction
NM-Timeout Timer has expired
/ Start NM-Timeout Timer;
Tx Timeout Timer has
expired
/Nm_TxTimeoutException();
Repeat Message
Timer has
expired;
Repeat Message Bit Received ||
CanNm_RepeatMessageRequest();
/Start Repeat Message Timer;
Repeat
Message
Repeat Message Bit received ||
CanNm_RepeatMessageRequest();
/Start Repeat Message Timer;
[Network Requested]
[Network Released]
NM-Timeout Timer has
expired
/Start NM-Timeout Timer;
Tx Timeout Timer has
expired
/Nm_TxTimeoutException();
CanNm_NetworkRequest();
[CanNmPnHandleMultipleNetworkRequests ==
FALSE]
Normal Operation State
entry / Start Bus Load Reduction
Ready Sleep State
CanNm_NetworkRelease();

Figure 7.1: CanNm Algorithm

8 API specification

[SWS_CanNm_00189] ⌈The CanNm module shall not return development errors by API functions; in case of a development error, the execution of the respective API function shall be aborted and E_NOT_OK shall be returned, if applicable.⌋

[SWS_CanNm_00190] ⌈The CanNm module shall not return production errors by API functions; in case of a production error, the execution of the respective API function shall be aborted and E_NOT_OK shall be returned, if applicable.⌋

[SWS_CanNm_00192]

Upstream requirements: SRS_BSW_00350

⌈When a CanNm service with an invalid network handle is called, the called function shall not be executed and it shall return E_NOT_OK to the calling function if applicable. If development error detection is enabled (CanNmDevErrorDetect is set to TRUE) the corresponding error CANNM_E_INVALID_CHANNEL shall be reported to DET.⌋

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

[SWS_CanNm_00507]

Upstream requirements: SRS_BSW_00350

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

[SWS_CanNm_00195]

Upstream requirements: SRS_BSW_00350

⌈When a CanNm service with an invalid PDU ID is called, the called function shall not be executed and it shall return E_NOT_OK to the calling function if applicable. If development error detection is enabled (CanNmDevErrorDetect is set to TRUE) the corresponding error CANNM_E_INVALID_PDUID shall be reported to DET.⌋

[SWS_CanNm_00244]

Upstream requirements: SRS_BSW_00323

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

AUTOSAR CanNm 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

In this chapter all types included from the following modules are listed:

[SWS_CanNm_00245] Definition of imported datatypes of module CanNm ⌈

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

⌋

For further details of these types refer to the according specifications [14], [2] and [15].

8.2 Type definitions

8.2.1 CanNm_ConfigType

[SWS_CanNm_00447] Definition of datatype CanNm_ConfigType ⌈

Name	CanNm_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	This type shall contain at least all parameters that are post-build able according to chapter 10.	

▽

△

Available via	CanNm.h

⌋

8.3 Function definitions

8.3.1 CanNm_Init

[SWS_CanNm_00208] Definition of API function CanNm_Init

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00358, SRS_BSW_00485

⌈

Service Name	CanNm_Init	

Syntax	`void CanNm_Init (`
	`  const CanNm_ConfigType* 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 CanNm module.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00253] ⌈Caveats of `CanNm_Init`: The function `CanNm_Init` has to be called after initialization of the the LSduR and the according lower layer module (e.g. CanIf).⌋

8.3.2 CanNm_DeInit

[SWS_CanNm_91002] Definition of API function CanNm_DeInit

Upstream requirements: SRS_BSW_00336, SRS_BSW_00310, SRS_BSW_00460

⌈

Service Name	CanNm_DeInit

Syntax	`void CanNm_DeInit (`
	`  void`
	`)`

Service ID [hex]	0x10
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	De-initializes the CanNm module.
Available via	CanNm.h

⌋

Note: General behavior and constraints on de-initialization functions are specified by [SWS_BSW_00152], [SWS_BSW_00072], [SWS_BSW_00232], and [SWS_BSW_00233].

Caveat: Caller of the CanNm_DeInit function has to ensure all CAN networks are in the Bus Sleep mode.

[SWS_CanNm_00352]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00350

⌈If development error detection for the CanNm module is enabled: The function CanNm_DeInit shall raise the error CANNM_E_NOT_IN_BUS_SLEEP if not all CAN networks are in Bus Sleep mode.⌋

8.3.3 CanNm_PassiveStartUp

[SWS_CanNm_00211] Definition of API function CanNm_PassiveStartUp

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

⌈

Service Name	CanNm_PassiveStartUp	

Syntax	`Std_ReturnType CanNm_PassiveStartUp (`
	`  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	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Passive startup of network management has failed
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. Caveats: CanNm is initialized correctly.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00254] ⌈Caveats of `CanNm_PassiveStartUp`: The CanNm module is initialized correctly.⌋

8.3.4 CanNm_NetworkRequest

[SWS_CanNm_00213] Definition of API function CanNm_NetworkRequest

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

⌈

Service Name	CanNm_NetworkRequest

Syntax	`Std_ReturnType CanNm_NetworkRequest (`
	`  NetworkHandleType nmChannelHandle`
	`)`

Service ID [hex]	0x02
Sync/Async	Asynchronous
Reentrancy	Reentrant (but not for the same NM-channel)

▽

△

Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Requesting of network has failed
Description	Request the network, since ECU needs to communicate on the bus.	
Available via	CanNm.h	

⌋

()

[SWS_CanNm_00255] ⌈The function `CanNm_NetworkRequest` shall change the Network state to 'requested'.⌋

[SWS_CanNm_00256] ⌈Caveats of `CanNm_NetworkRequest`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00257] ⌈Configuration of `CanNm_NetworkRequest`: Optional (Only available if `CanNmPassiveModeEnabled` is not defined).⌋

8.3.5 `CanNm_NetworkRelease`

[SWS_CanNm_00214] Definition of API function CanNm_NetworkRelease

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

⌈

Service Name	CanNm_NetworkRelease	

Syntax	`Std_ReturnType CanNm_NetworkRelease (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Releasing of network has failed
Description	Release the network, since ECU doesn't have to communicate on the bus.	

▽

△

Available via	CanNm.h

⌋

[SWS_CanNm_00259] ⌈Caveats of `CanNm_NetworkRelease`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00260] ⌈Configuration of `CanNm_NetworkRelease`: Optional (Only available if `CanNmPassiveModeEnabled` is not defined)⌋

8.3.6 CanNm_DisableCommunication

[SWS_CanNm_00215] Definition of API function CanNm_DisableCommunication

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

⌈

Service Name	CanNm_DisableCommunication	

Syntax	`Std_ReturnType CanNm_DisableCommunication (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

Service ID [hex]	0x0c	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Disabling of NM PDU transmission ability has failed
Description	Disable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service	
Available via	CanNm.h	

⌋

[SWS_CanNm_00261] ⌈Caveats of `CanNm_DisableCommunication`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00262] ⌈Configuration of `CanNm_DisableCommunication`: Optional (Only available if `CanNmComControlEnabled` is set to TRUE)⌋

[SWS_CanNm_00172] ⌈The service `CanNm_DisableCommunication` shall return `E_NOT_OK`, if the current mode is not Network Mode.⌋

8.3.7 CanNm_EnableCommunication

[SWS_CanNm_00216] Definition of API function CanNm_EnableCommunication

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

⌈

Service Name	CanNm_EnableCommunication	

Syntax	`Std_ReturnType CanNm_EnableCommunication (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

Service ID [hex]	0x0d	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Enabling of NM PDU transmission ability has failed
Description	Enable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service	
Available via	CanNm.h	

⌋

[SWS_CanNm_00176]

Upstream requirements: RS_Nm_02512

⌈The service `CanNm_EnableCommunication` shall enable the Network Management PDU transmission ability if the Network Management PDU transmission ability is disabled.⌋

[SWS_CanNm_00177] ⌈The service `CanNm_EnableCommunication` shall return `E_NOT_OK` if the Network Management PDU transmission ability is enabled.⌋

[SWS_CanNm_00295] ⌈The service `CanNm_EnableCommunication` shall return `E_NOT_OK`, if the current mode is not Network Mode.⌋

[SWS_CanNm_00263] ⌈Caveats of `CanNm_EnableCommunication`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00264] ⌈Configuration of `CanNm_EnableCommunication`: Optional (Only available if `CanNmComControlEnabled` is set to TRUE).⌋

8.3.8 CanNm_SetUserData

[SWS_CanNm_00217] Definition of API function CanNm_SetUserData

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00484, SRS_BSW_00486, SRS_BSW_00459

⌈

Service Name	CanNm_SetUserData	

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

Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same NM-channel)	
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	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Setting of user data has failed
Description	Set user data for NM PDUs transmitted next on the bus.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00265] ⌈Caveats of `CanNm_SetUserData`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00266] ⌈Configuration of `CanNm_SetUserData`: Optional (Only available if `CanNmUserDataEnabled` is set to TRUE and `CanNmPassiveModeEnabled` is not defined)⌋

8.3.9 CanNm_GetUserData

[SWS_CanNm_00218] Definition of API function CanNm_GetUserData

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

⌈

Service Name	CanNm_GetUserData	

Syntax	`Std_ReturnType CanNm_GetUserData (`
	`  NetworkHandleType nmChannelHandle,`
	`  uint8* 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	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Getting of user data has failed
Description	Get user data out of the most recently received NM PDU.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00267] ⌈Caveats of `CanNm_GetUserData`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00268] ⌈Configuration of `CanNm_GetUserData`: Optional (Only available if `CanNmUserDataEnabled` is set to TRUE).⌋

8.3.10 CanNm_Transmit

[SWS_CanNm_00331] Definition of API function CanNm_Transmit

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00484, SRS_BSW_00485, SRS_BSW_00459

⌈

Service Name	CanNm_Transmit	

Syntax	`Std_ReturnType CanNm_Transmit (`
	`  PduIdType TxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

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

⌋

[SWS_CanNm_00330] ⌈If CanNmComUserDataSupport or CanNmGlobalPnSupport is enabled the CanNm implementation shall provide an API CanNm_Transmit.⌋

[SWS_CanNm_00333]

Upstream requirements: RS_Nm_02527, SRS_BSW_00483

⌈If CanNmComUserDataSupport is enabled and if CanNm is in RepeatMessage state or NormalOperation state and if CanNm_Transmit() is called CanNm shall request an additional transmission of the NM PDU with the current user data.⌋

Note: The call of CanNm_Transmit request to transmit a NM PDU between the periodic transmissions with the current data (e.g., system bytes, user data and PNC bit vector).

8.3.11 CanNm_GetNodeIdentifier

[SWS_CanNm_00219] Definition of API function CanNm_GetNodeIdentifier

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

⌈

Service Name	CanNm_GetNodeIdentifier	

Syntax	`Std_ReturnType CanNm_GetNodeIdentifier (`
	`  NetworkHandleType nmChannelHandle,`
	`  uint8* nmNodeIdPtr`
	`)`	

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

⌋

[SWS_CanNm_00132]

Upstream requirements: RS_Nm_02506, SRS_BSW_00483

⌈The service call `CanNm_GetNodeIdentifier` shall provide the node identifier out of the most recently received Network Management PDU if `CanNmNodeIdEnabled` is set to TRUE.⌋

[SWS_CanNm_00269] ⌈Caveats of `CanNm_GetNodeIdentifier`: The CanNm module is initialized correctly.⌋

8.3.12 CanNm_GetLocalNodeIdentifier

[SWS_CanNm_00220] Definition of API function CanNm_GetLocalNodeIdentifier

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

⌈

Service Name	CanNm_GetLocalNodeIdentifier	

Syntax	`Std_ReturnType CanNm_GetLocalNodeIdentifier (`
	`  NetworkHandleType nmChannelHandle,`
	`  uint8* nmNodeIdPtr`
	`)`	

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

⌋

[SWS_CanNm_00133]

Upstream requirements: RS_Nm_02508, SRS_BSW_00483

⌈The service call `CanNm_GetLocalNodeIdentifier` shall provide the node identifier configured for the local host node if `CanNmNodeIdEnabled` is set to TRUE.⌋

[SWS_CanNm_00271] ⌈Caveats of `CanNm_GetLocalNodeIdentifier`: The CanNm module is initialized correctly.⌋

8.3.13 CanNm_RepeatMessageRequest

[SWS_CanNm_00221] Definition of API function CanNm_RepeatMessageRequest

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

⌈

Service Name	CanNm_RepeatMessageRequest	

Syntax	`Std_ReturnType CanNm_RepeatMessageRequest (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

Service ID [hex]	0x08	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Setting of Repeat Message Request Bit has failed or is not configured for this network handle.
Description	Set Repeat Message Request Bit for NM PDUs transmitted next on the bus.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00273] ⌈Caveats of `CanNm_RepeatMessageRequest`: The CanNm module is initialized correctly.⌋

8.3.14 CanNm_GetPduData

[SWS_CanNm_00222] Definition of API function CanNm_GetPduData

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

⌈

Service Name	CanNm_GetPduData

Syntax	`Std_ReturnType CanNm_GetPduData (`
	`  NetworkHandleType nmChannelHandle,`
	`  uint8* nmPduDataPtr`
	`)`

Service ID [hex]	0x0a
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmPduDataPtr	Pointer where NM PDU shall be copied to
Return value	Std_ReturnType	`E_OK`: No error `E_NOT_OK`: Getting of NM PDU Data has failed or is not configured for this network handle.
Description	Get the whole PDU data out of the most recently received NM PDU.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00275] ⌈Caveats of `CanNm_GetPduData`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00138]

Upstream requirements: SRS_BSW_00483

⌈The service call `CanNm_GetPduData` shall provide whole PDU data (Node ID, Control Bit Vector and User Data) of the most recently received Network Management PDU if `CanNmNodeDetectionEnabled` or `CanNmUserDataEnabled` or `CanNmNodeIdEnabled` is set to TRUE.⌋

8.3.15 `CanNm_GetState`

[SWS_CanNm_00223] Definition of API function CanNm_GetState

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

⌈

Service Name	CanNm_GetState	

Syntax	`Std_ReturnType CanNm_GetState (`
	`  NetworkHandleType nmChannelHandle,`
	`  Nm_StateType* nmStatePtr,`
	`  Nm_ModeType* nmModePtr`
	`)`	

Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmStatePtr	Pointer where state of the network management shall be copied to

▽

△

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

⌋

[SWS_CanNm_00277] ⌈Caveats of CanNm_GetState: The CanNm module is initialized correctly.⌋

8.3.16 CanNm_GetVersionInfo

[SWS_CanNm_00224] Definition of API function CanNm_GetVersionInfo

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00482, SRS_BSW_00459

⌈

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.	
Available via	CanNm.h	

⌋

8.3.17 CanNm_RequestBusSynchronization

[SWS_CanNm_00226] Definition of API function CanNm_RequestBusSynchronization

Upstream requirements: RS_Nm_02516, SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00484

⌈

Service Name	CanNm_RequestBusSynchronization	

Syntax	`Std_ReturnType CanNm_RequestBusSynchronization (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

Service ID [hex]	0xc0	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: Requesting of bus synchronization has failed
Description	Request bus synchronization.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00279] ⌈Caveats of CanNm_RequestBusSynchronization: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00280]

Upstream requirements: RS_Nm_02516

⌈Configuration of CanNm_RequestBusSynchronization: Optional (Only available if CanNmBusSynchronizationEnabled is set to TRUE) and CanNmPassiveModeEnabled is not defined.⌋

[SWS_CanNm_00130]

Upstream requirements: RS_Nm_02516

⌈The service call CanNm_RequestBusSynchronization shall trigger transmission of a single Network Management PDU if CanNmPassiveModeEnabled (configuration parameter) is not defined.⌋

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

[SWS_CanNm_00187]

Upstream requirements: RS_Nm_02516

⌈If CanNm_RequestBusSynchronization is called in Bus-Sleep Mode and Prepare Bus-Sleep Mode the CanNm module shall not execute the service and shall return E_NOT_OK.⌋

8.3.18 CanNm_CheckRemoteSleepIndication

[SWS_CanNm_00227] Definition of API function CanNm_CheckRemoteSleepIndication

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

⌈

Service Name	CanNm_CheckRemoteSleepIndication	

Syntax	`Std_ReturnType CanNm_CheckRemoteSleepIndication (`
	`  NetworkHandleType nmChannelHandle,`
	`  boolean* nmRemoteSleepIndPtr`
	`)`	

Service ID [hex]	0xd0	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	nmRemoteSleepIndPtr	Pointer where check result of remote sleep indication shall be copied to
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: Checking of remote sleep indication bits has failed
Description	Check if remote sleep indication takes place or not.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00153]

Upstream requirements: RS_Nm_00052, RS_Nm_02509, SRS_BSW_00483

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

[SWS_CanNm_00281] ⌈Caveats of CanNm_CheckRemoteSleepIndication: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00282] ⌈Configuration of CanNm_CheckRemoteSleepIndication: Optional (Only available if CanNmRemoteSleepIndEnabled is set to TRUE).⌋

8.3.19 CanNm_SetSleepReadyBit

[SWS_CanNm_00338] Definition of API function CanNm_SetSleepReadyBit

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

⌈

Service Name	CanNm_SetSleepReadyBit	

Syntax	`Std_ReturnType CanNm_SetSleepReadyBit (`
	`  NetworkHandleType nmChannelHandle,`
	`  boolean nmSleepReadyBit`
	`)`	

Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same NM-channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
	nmSleepReadyBit	Value written to ReadySleep Bit in CBV
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: Writing of remote sleep indication bit has failed
Description	Set the NM Coordinator Sleep Ready bit in the Control Bit Vector	
Available via	CanNm.h	

⌋

[SWS_CanNm_00339] ⌈Caveats of `CanNm_SetSleepReadyBit`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00340] ⌈Configuration of `CanNm_SetSleepReadyBit`: Optional (Only available if `CanNmCoordinatorSyncSupport` is set to TRUE).⌋

8.3.20 CanNm_PnLearningRequest

[SWS_CanNm_91004] Definition of API function CanNm_PnLearningRequest

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

⌈

Service Name	CanNm_PnLearningRequest	

Syntax	`Std_ReturnType CanNm_PnLearningRequest (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

Service ID [hex]	0xf2	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM-channel)	
Parameters (in)	nmChannelHandle	Identification of the NM-channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: No error E_NOT_OK: PN Learning Requesthas failed or is not configured for this network handle.
Description	Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. This will force all nodes to enter the PNC Learning Phase. This is needed for the optional Dynamic PNC-to-channel-mapping feature.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00384] ⌈If the function CanNm_PnLearningRequest is called in Prepare Bus-Sleep Mode or Bus Sleep Mode no functionality shall be executed and E_NOT_OK shall be returned.⌋

[SWS_CanNm_00385] ⌈The function CanNm_PnLearningRequest shall only be available if CanNmDynamicPncToChannelMappingSupport is set to TRUE.⌋

8.3.21 CanNm_ActivateTxPnShutdownMsg

[SWS_CanNm_91005] Definition of API function CanNm_ActivateTxPnShutdownMsg

Upstream requirements: RS_Nm_02572, SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00484, SRS_BSW_00459

⌈

Service Name	CanNm_ActivateTxPnShutdownMsg	

Syntax	`Std_ReturnType CanNm_ActivateTxPnShutdownMsg (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

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

⌋

[SWS_CanNm_00520]

Upstream requirements: RS_Nm_02572

⌈If `CanNmSynchronizedPncShutdownEnabled` is set to TRUE the CanNm implementation shall provide the API `CanNm_ActivateTxPnShutdownMsg`.⌋

[SWS_CanNm_00521]

Upstream requirements: RS_Nm_02542, RS_Nm_02572

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

8.3.22 CanNm_DeactivateTxPnShutdownMsg

[SWS_CanNm_91006] Definition of API function CanNm_DeactivateTxPnShutdownMsg

Upstream requirements: RS_Nm_02572, SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00461, SRS_BSW_00484, SRS_BSW_00459

⌈

Service Name	CanNm_DeactivateTxPnShutdownMsg	

Syntax	`Std_ReturnType CanNm_DeactivateTxPnShutdownMsg (`
	`  NetworkHandleType nmChannelHandle`
	`)`	

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

⌋

[SWS_CanNm_00522]

Upstream requirements: RS_Nm_02572

⌈If CanNmSynchronizedPncShutdownEnabled is set to TRUE the CanNm implementation shall provide the API CanNm_DeactivateTxPnShutdownMsg.⌋

[SWS_CanNm_00523]

Upstream requirements: RS_Nm_02572

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

8.4 Callback notifications

This is a list of functions provided for other modules.

8.4.1 CanNm_TxConfirmation

[SWS_CanNm_00228] Definition of callback function CanNm_TxConfirmation

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00484, SRS_BSW_00459

⌈

Service Name	CanNm_TxConfirmation	

Syntax	`void CanNm_TxConfirmation (`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`	

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

⌋

[SWS_CanNm_00283] ⌈Caveats of `CanNm_TxConfirmation`:

- 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.

⌋

[SWS_CanNm_00284] ⌈Configuration of `CanNm_TxConfirmation`: Optional (Only available if `CanNmPassiveModeEnabled` and `CanNmImmediateTxconfEnabled` are set to FALSE).⌋

8.4.2 CanNm_RxIndication

[SWS_CanNm_00231] Definition of callback function CanNm_RxIndication

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00484, SRS_BSW_00485, SRS_BSW_00459

⌈

Service Name	CanNm_RxIndication	

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

Service ID [hex]	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication of a received PDU from a lower layer communication interface module.	
Available via	CanNm.h	

⌋

Note: The callback function CanNm_RxIndication called by the CAN Interface and implemented by the CanNm module. It is called in case of a receive indication event of the CAN Driver.

[SWS_CanNm_00285] ⌈Caveats of CanNm_RxIndication:

- Until this service returns the CAN Interface will not access canSduPtr. The can SduPtr 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 CanNm-RxPduId 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.

⌋

8.4.3 CanNm_ConfirmPnAvailability

[SWS_CanNm_00344] Definition of API function CanNm_ConfirmPnAvailability

Upstream requirements: SRS_BSW_00459

⌈

Service Name	CanNm_ConfirmPnAvailability	

Syntax	`void CanNm_ConfirmPnAvailability (`
	`  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 CanNmGlobalPnSupport is TRUE.	
Available via	CanNm.h	

⌋

[SWS_CanNm_00345] ⌈Caveats of `CanNm_ConfirmPnAvailability`: The CanNm module is initialized correctly.⌋

[SWS_CanNm_00346] ⌈Configuration of `CanNm_ConfirmPnAvailability`: Optional (Only available if `CanNmGlobalPnSupport` is set to TRUE).⌋

8.4.4 CanNm_TriggerTransmit

[SWS_CanNm_91001] Definition of callback function CanNm_TriggerTransmit

Upstream requirements: SRS_BSW_00310, SRS_BSW_00460, SRS_BSW_00484, SRS_BSW_00459

⌈

Service Name	CanNm_TriggerTransmit

Syntax	`Std_ReturnType CanNm_TriggerTransmit (`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`

Service ID [hex]	0x41
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Available via	CanNm.h	

⌋

Note: The PNC bit vector is not updated within the call of CanNm_TriggerTransmit but upfront of each NM message transmission request (see [SWS_CanNm_00518]). This ensures a common handling independent if the NM-PDU is configured for triggered transmission or not in the lower layer (e.g. CanIf: CanIfTxPduTriggerTransmit set to TRUE or FALSE).

[SWS_CanNm_00510]

Upstream requirements: RS_Nm_02503, SRS_BSW_00483

⌈If CanNm_TriggerTransmit is called and CanNmComUserDataSupport is enabled, CanNm shall collect the NM User Data from the referenced NM I-PDU by calling PduR_CanNmTriggerTransmit and copy the data to the user data range of the NM-PDU.⌋

[SWS_CanNm_00351]

Upstream requirements: RS_Nm_02503, SRS_BSW_00483

⌈The function CanNm_TriggerTransmit shall copy the NM PDU data of the according NM PDU requested by TxPduId⌋

Note: The function CanNm_TriggerTransmit might be called by the lower layer (e.g. CanIf) in an interrupt context.

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.5.1 CanNm_MainFunction

[SWS_CanNm_00234] Definition of scheduled function CanNm_MainFunction

Upstream requirements: SRS_BSW_00459

⌈

Service Name	CanNm_MainFunction

Syntax	`void CanNm_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x13
Description	Main function of the CanNm which processes the algorithm describes in that document.
Available via	SchM_CanNm.h

⌋

Note that as requirement [SWS_BSW_00037] specifies, CanNm_MainFunction will return without executing any functionality if the module is not initialized.

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

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

[SWS_CanNm_00324] Definition of mandatory interfaces required by module CanNm ⌈

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

▽

△

API Function	*Header File*	*Description*
Nm_PrepareBusSleepMode	Nm.h	Notification that the network management has entered Prepare Bus-Sleep Mode.

⌋

8.6.2 Optional interfaces

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

[SWS_CanNm_00325] Definition of optional interfaces requested by module CanNm ⌈

API Function	*Header File*	*Description*
CanSM_TxTimeoutException	CanSM_CanIf.h	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 within the respective network state machine of the CanSM module.
Det_ReportError	Det.h	Service to report development errors.
LSduR_CanNmTransmit (draft)	LSduR_CanNm.h	Requests transmission of a PDU.
Nm_CarWakeUpIndication	Nm.h	This function is called by a <Bus>Nm to indicate reception of a CWU request.
Nm_CoordReadyToSleepCancellation	Nm.h	Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0.
Nm_CoordReadyToSleepIndication	Nm.h	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set
Nm_ForwardSynchronizedPncShutdown	Nm.h	Notification that the network management has received a PN shutdown message on a particular NM-channel. This is used to grant a nearly synchronized PNC shutdown across the entire PN topology.
Nm_PduRxIndication	Nm.h	Notification that a NM message has been received.
Nm_PncBitVectorRxIndication	Nm.h	Indication that a bus specific network management has received a NM message on a particular NM-channel that contain a PNC bit vector. This is used to aggregate the external PNC requests. The function evaluate if a relevant PNC request (PNC bit set to '1') is available in the given PNC bit vector. If a relevant PNC request is available (PNC bit passes the PNC bit vector filter), then the RelevantPncRequestDetectedPtr refers to a boolean with value set to TRUE. Otherwise refer to booelan with value set to FALSE. RelevantPncRequestDetectedPtr is evaluated by the callee <Bus>Nm module to qualify the further processing of the received NM-PDU.

▽

△

API Function	Header File	Description
Nm_PncBitVectorTxConfirmation	Nm.h	Function called by <Bus>Nms to confirm the state of the transmission for the given PNC bit vector on the given NM-Channel.
Nm_PncBitVectorTxIndication	Nm.h	Function called by <Bus>Nms to request the aggregated internal PNC requests for transmission within the Nm message.
Nm_RemoteSleepCancellation	Nm.h	Notification that the network management has detected that not all other nodes on the network are longer ready to enter Bus-Sleep Mode.
Nm_RemoteSleepIndication	Nm.h	Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode.
Nm_RepeatMessageIndication	Nm.h	Service to indicate that an NM message with set Repeat Message Re- quest Bit has been received. This is needed for node detection and the Dynamic PNC-to-channel-mapping feature.
Nm_StateChangeNotification	Nm.h	Notification that the state of the lower layer <Bus>Nm has changed.
Nm_TxTimeoutException	Nm.h	Service to indicate that an attempt to send an NM message failed.
PduR_CanNmRxIndication	PduR_CanNm.h	Indication of a received PDU from a lower layer communication interface module.
PduR_CanNmTriggerTransmit	PduR_CanNm.h	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->Sdu Length. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->Sdu Length. If not, it returns E_NOT_OK without changing PduInfoPtr.
PduR_CanNmTxConfirmation	PduR_CanNm.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.

⌋

8.6.3 Configurable interfaces

CanNm does not provide any configurable interfaces.

8.6.4 Job End Notification

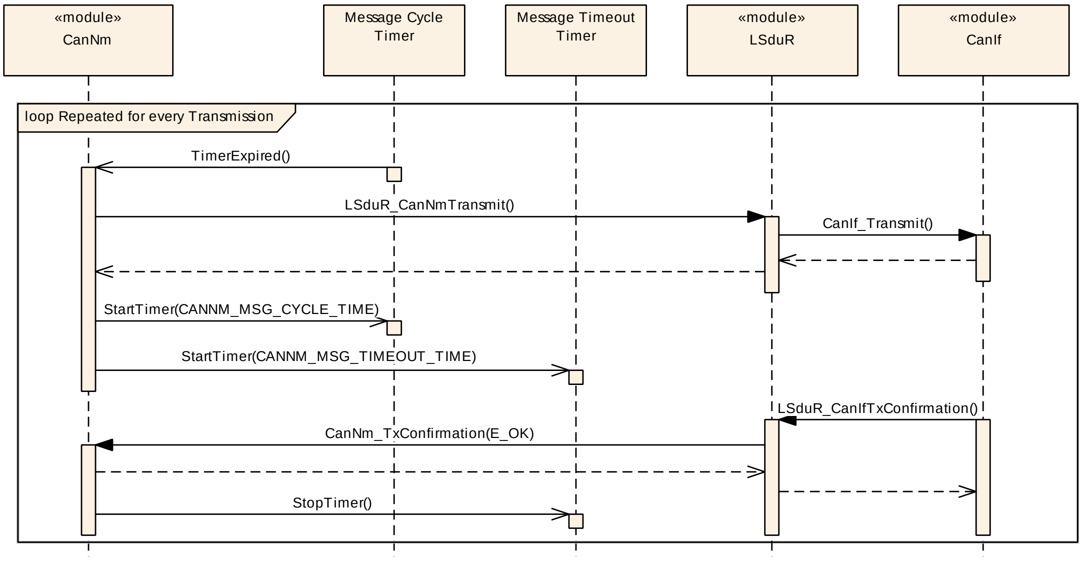
CanNm does not provide any job end notifications.

8.7 Service Interfaces

CanNm does not provide any service interfaces.

9 Sequence diagrams

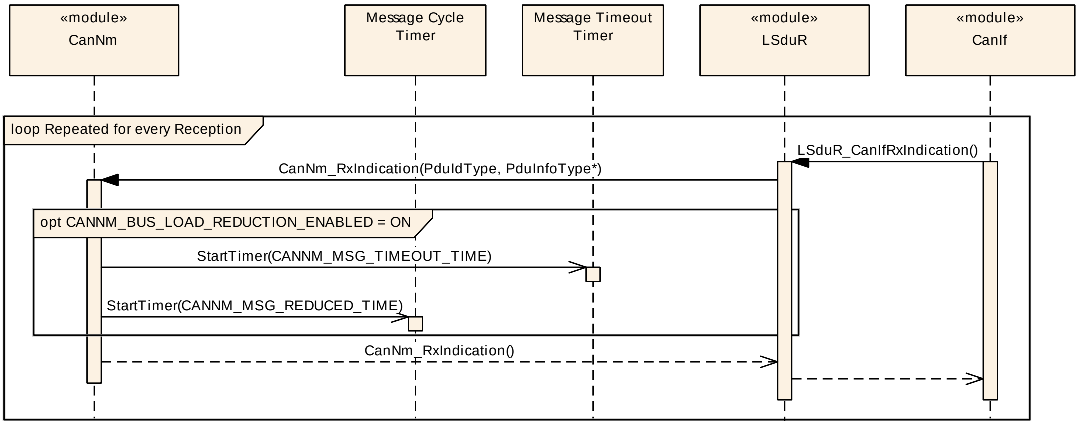
9.1 CanNm Transmission


«module»
CanNm
Message Cycle Timer
Message Timeout Timer
«module»
LSduR
«module»
CanIf
loop Repeated for every Transmission
TimerExpired()
LSduR_CanNmTransmit()
CanIf_Transmit()
StartTimer(CANNM_MSG_CYCLE_TIME)
StartTimer(CANNM_MSG_TIMEOUT_TIME)
LSduR_CanIfTxConfirmation()
CanNm_TxConfirmation(E_OK)
StopTimer()

Status: Proposed (CanNm 1.8.0), changes due to CAN NM merge by UH9
Comments:

Figure 9.1: CanNm Transmission

9.2 CanNm Reception


«module»
CanNm
Message Cycle Timer
Message Timeout Timer
«module»
LSduR
«module»
CanIf
loop Repeated for every Reception
LSduR_CanIfRxIndication()
CanNm_RxIndication(PduIdType, PduInfoType*)
opt CANNM_BUS_LOAD_REDUCTION_ENABLED = ON
StartTimer(CANNM_MSG_TIMEOUT_TIME)
StartTimer(CANNM_MSG_REDUCED_TIME)
CanNm_RxIndication()

Status: Proposed (CanNm 1.8.0), changes due to CAN NM merge by UH9
Comments:

Figure 9.2: CanNm Reception

9.3 Nm Coordination

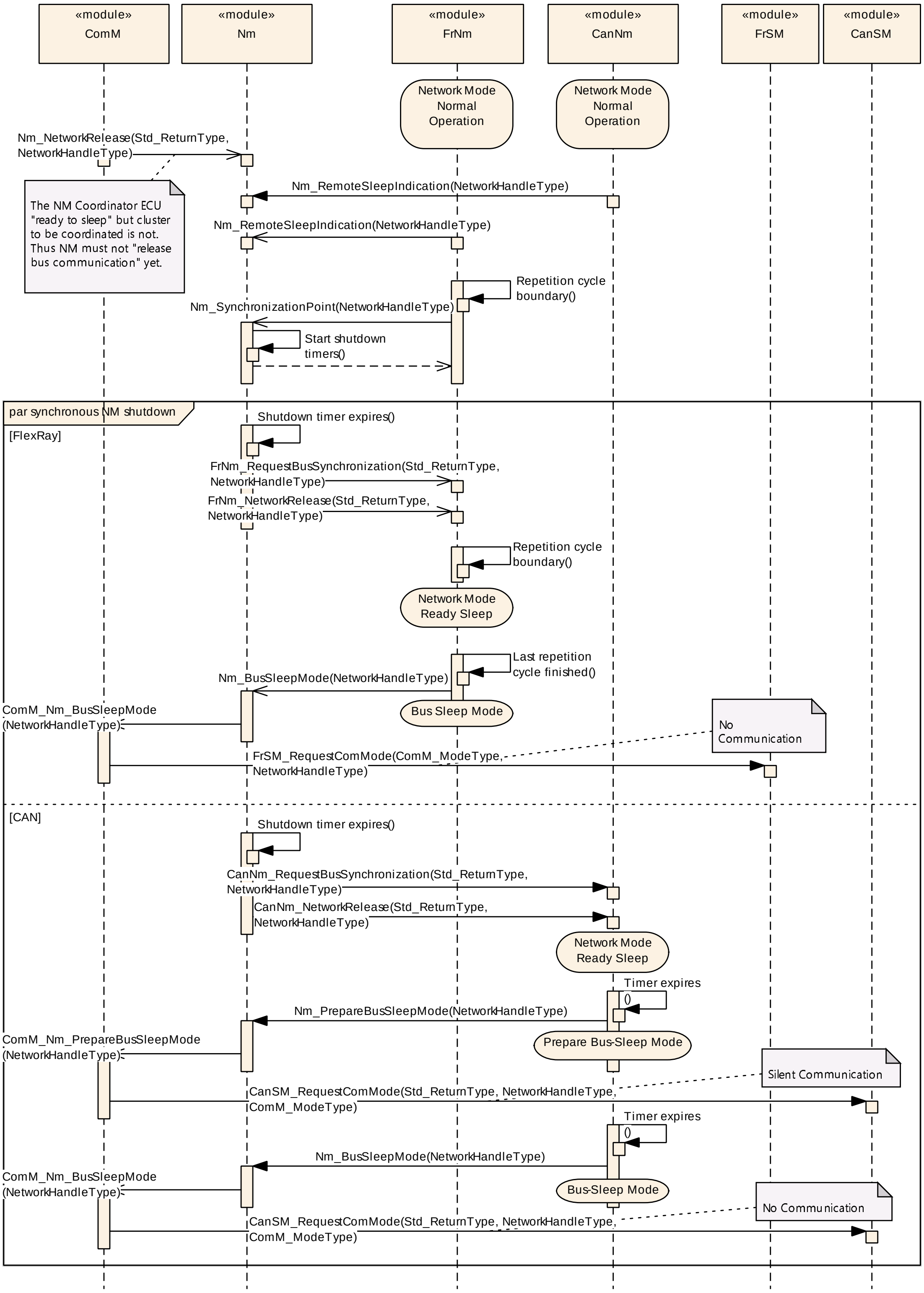

«module» ComM
«module» Nm
«module» FrNm
«module» CanNm
«module» FrSM
«module» CanSM
Network Mode Normal Operation
Network Mode Normal Operation
Nm_NetworkRelease(Std_ReturnType, NetworkHandleType)
The NM Coordinator ECU "ready to sleep" but cluster to be coordinated is not. Thus NM must not "release bus communication" yet.
Nm_RemoteSleepIndication(NetworkHandleType)
Nm_RemoteSleepIndication(NetworkHandleType)
Repetition cycle boundary()
Nm_SynchronizationPoint(NetworkHandleType)
Start shutdown timers()
par synchronous NM shutdown
[FlexRay]
Shutdown timer expires()
FrNm_RequestBusSynchronization(Std_ReturnType, NetworkHandleType)
FrNm_NetworkRelease(Std_ReturnType, NetworkHandleType)
Repetition cycle boundary()
Network Mode Ready Sleep
Last repetition cycle finished()
Nm_BusSleepMode(NetworkHandleType)
ComM_Nm_BusSleepMode (NetworkHandleType)
Bus Sleep Mode
No Communication
FrSM_RequestComMode(ComM_ModeType, NetworkHandleType)
[CAN]
Shutdown timer expires()
CanNm_RequestBusSynchronization(Std_ReturnType, NetworkHandleType)
CanNm_NetworkRelease(Std_ReturnType, NetworkHandleType)
Network Mode Ready Sleep
Timer expires ()
Nm_PrepareBusSleepMode(NetworkHandleType)
Prepare Bus-Sleep Mode
ComM_Nm_PrepareBusSleepMode (NetworkHandleType)
Silent Communication
CanSM_RequestComMode(Std_ReturnType, NetworkHandleType, ComM_ModeType)
Timer expires ()
Nm_BusSleepMode(NetworkHandleType)
ComM_Nm_BusSleepMode (NetworkHandleType)
Bus-Sleep Mode
No Communication
CanSM_RequestComMode(Std_ReturnType, NetworkHandleType, ComM_ModeType)

Figure 9.3: Nm Coordination

10 Configuration specification

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

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

Section 10.3 specifies published information of the module CanNm.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral [4].

Additionally it is highly recommended to read the document Specification of ECU Configuration [11]. This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration meta model in detail.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

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.

10.2.1 CanNm

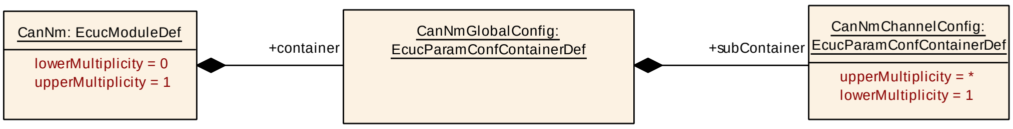


Figure 10.1: CanNm top level configuration overview

[ECUC_CanNm_00087] Definition of EcucModuleDef CanNm ⌈

Module Name	CanNm
Description	Configuration Parameters for the Can Nm module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanNmGlobalConfig	1	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_Config Type (for parameters where it is possible) which is passed to the CanNm_Init function.

⌋

10.2.2 CanNmGlobalConfig

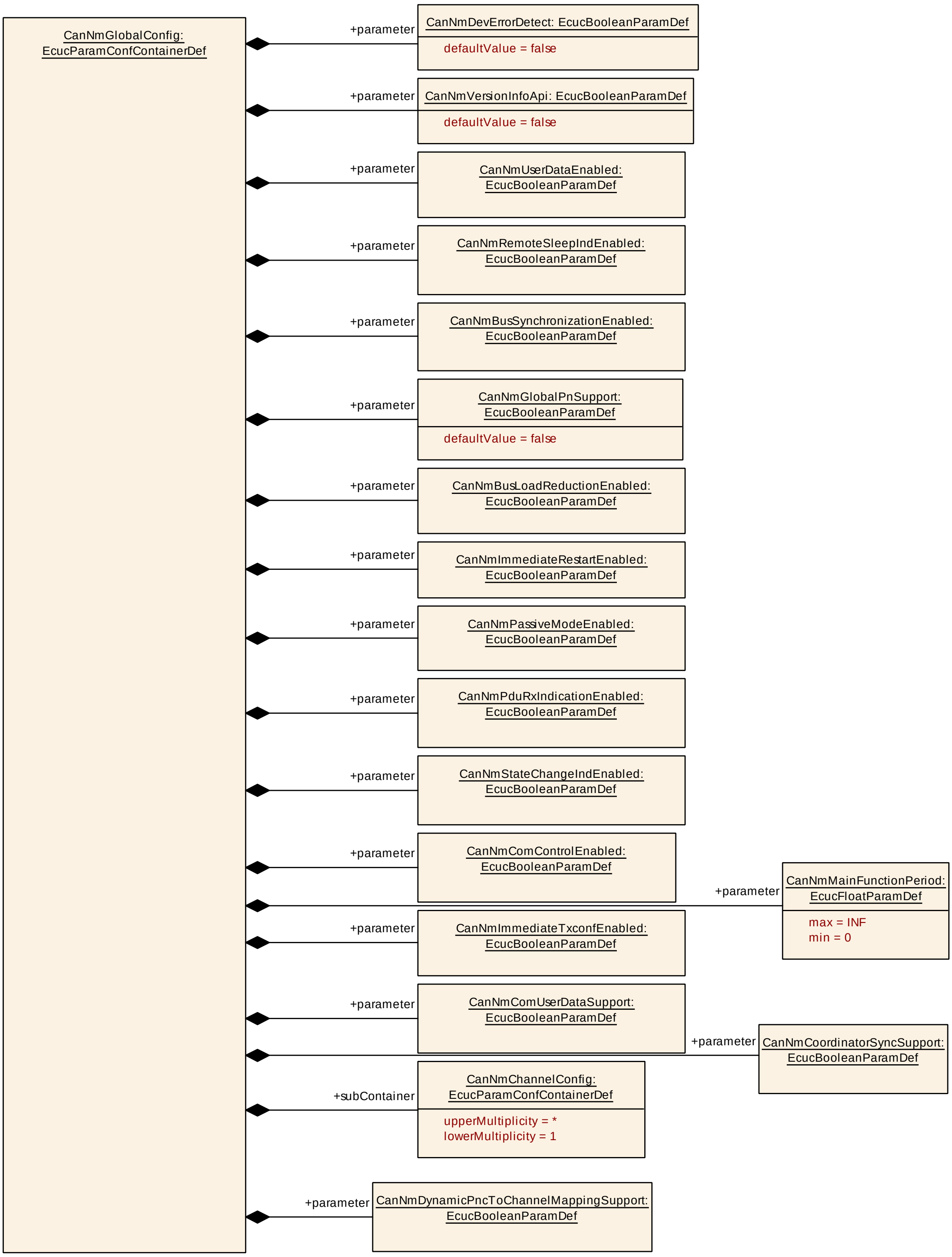

CanNmGlobalConfig: EcucParamConfContainerDef
+parameter
CanNmDevErrorDetect: EcucBooleanParamDef
defaultValue = false
+parameter
CanNmVersionInfoApi: EcucBooleanParamDef
defaultValue = false
+parameter
CanNmUserDataEnabled: EcucBooleanParamDef
+parameter
CanNmRemoteSleepIndEnabled: EcucBooleanParamDef
+parameter
CanNmBusSynchronizationEnabled: EcucBooleanParamDef
+parameter
CanNmGlobalPnSupport: EcucBooleanParamDef
defaultValue = false
+parameter
CanNmBusLoadReductionEnabled: EcucBooleanParamDef
+parameter
CanNmImmediateRestartEnabled: EcucBooleanParamDef
+parameter
CanNmPassiveModeEnabled: EcucBooleanParamDef
+parameter
CanNmPduRxIndicationEnabled: EcucBooleanParamDef
+parameter
CanNmStateChangeIndEnabled: EcucBooleanParamDef
+parameter
CanNmComControlEnabled: EcucBooleanParamDef
+parameter
CanNmMainFunctionPeriod: EcucFloatParamDef
max = INF
min = 0
+parameter
CanNmImmediateTxconfEnabled: EcucBooleanParamDef
+parameter
CanNmComUserDataSupport: EcucBooleanParamDef
+parameter
CanNmCoordinatorSyncSupport: EcucBooleanParamDef
+subContainer
CanNmChannelConfig: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
CanNmDynamicPncToChannelMappingSupport: EcucBooleanParamDef

Figure 10.2: Parameters of CanNm global configuration CanNmGlobalConfig

[ECUC_CanNm_00001] Definition of EcucParamConfContainerDef CanNmGlobalConfig ⌈

Container Name	CanNmGlobalConfig
Parent Container	CanNm
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.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanNmBusLoadReductionEnabled	1	[ECUC_CanNm_00040]
CanNmBusSynchronizationEnabled	1	[ECUC_CanNm_00006]
CanNmComControlEnabled	1	[ECUC_CanNm_00013]
CanNmComUserDataSupport	1	[ECUC_CanNm_00044]
CanNmCoordinatorSyncSupport	1	[ECUC_CanNm_00080]
CanNmDevErrorDetect	1	[ECUC_CanNm_00002]
CanNmDynamicPncToChannelMappingSupport	1	[ECUC_CanNm_00094]
CanNmGlobalPnSupport	1	[ECUC_CanNm_00086]
CanNmImmediateRestartEnabled	1	[ECUC_CanNm_00009]
CanNmImmediateTxconfEnabled	1	[ECUC_CanNm_00041]
CanNmMainFunctionPeriod	1	[ECUC_CanNm_00032]
CanNmPassiveModeEnabled	1	[ECUC_CanNm_00010]
CanNmPduRxIndicationEnabled	1	[ECUC_CanNm_00011]
CanNmRemoteSleepIndEnabled	1	[ECUC_CanNm_00055]
CanNmStateChangeIndEnabled	1	[ECUC_CanNm_00012]
CanNmUserDataEnabled	1	[ECUC_CanNm_00004]
CanNmVersionInfoApi	1	[ECUC_CanNm_00003]

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

⌋

[ECUC_CanNm_00040] Definition of EcucBooleanParamDef CanNmBusLoadReductionEnabled ⌈

Parameter Name	CanNmBusLoadReductionEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling busload reduction support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

▽

△

	Post-build time	–	
Scope / Dependency	scope: local dependency: CanNmBusLoadReductionEnabled = false if CanNmPassiveModeEnabled == true or CanNmGlobalPnSupport == true		

⌋

[ECUC_CanNm_00006] Definition of EcucBooleanParamDef CanNmBusSynchronizationEnabled ⌈

Parameter Name	CanNmBusSynchronizationEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling bus synchronization support. This feature is required for gateway nodes only.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: calculationFormula = If (CanNmPassiveModeEnabled == False) then Equal(NmBusSynchronizationEnabled) else Equal(False)		

⌋

[ECUC_CanNm_00013] Definition of EcucBooleanParamDef CanNmComControlEnabled ⌈

Parameter Name	CanNmComControlEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling the Communication Control support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: If (CanNmPassiveModeEnabled == False) then Equal(NmComControlEnabled) else Equal(False)		

⌋

[ECUC_CanNm_00044] Definition of EcucBooleanParamDef CanNmComUserDataSupport ⌈

Parameter Name	CanNmComUserDataSupport		
Parent Container	CanNmGlobalConfig		
Description	Preprocessor switch for enabling the Tx path of Com User Data. Use case: Setting of NMUserData via SWC.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: If CanNmPassiveModeEnabled == True OR if all bytes of the NM PDU are used for NM System Bytes and for the PNC bit vector and no space is left for user data, then CanNmComUserDataSupport shall be set to False.		

⌋

[ECUC_CanNm_00080] Definition of EcucBooleanParamDef CanNmCoordinatorSyncSupport ⌈

Parameter Name	CanNmCoordinatorSyncSupport		
Parent Container	CanNmGlobalConfig		
Description	Enables/disables the coordinator synchronization support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmCoordinatorSyncSupport has to be set to FALSE if CanNmPassiveModeEnabled is set to TRUE.		

⌋

[ECUC_CanNm_00002] Definition of EcucBooleanParamDef CanNmDevErrorDetect ⌈

Parameter Name	CanNmDevErrorDetect		
Parent Container	CanNmGlobalConfig		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CanNm_00094] Definition of EcucBooleanParamDef CanNmDynamicPncToChannelMappingSupport ⌈

Parameter Name	CanNmDynamicPncToChannelMappingSupport		
Parent Container	CanNmGlobalConfig		
Description	Precompile time switch to enable the dynamic PNC-to-channel-mapping handling. False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmDynamicPncToChannelMappingSupport == TRUE only allowed if CanNmGlobalPnSupport == TRUE and CanNmPassiveModeEnabled == FALSE		

⌋

[ECUC_CanNm_00086] Definition of EcucBooleanParamDef CanNmGlobalPnSupport ⌈

Parameter Name	CanNmGlobalPnSupport
Parent Container	CanNmGlobalConfig
Description	Pre-processor switch for enabling partial networking support globally.
Multiplicity	1

▽

△

Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CanNm_00009] Definition of EcucBooleanParamDef CanNmImmediateRestartEnabled ⌈

Parameter Name	CanNmImmediateRestartEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling the immediate transmission of a NM PDU upon bus-communication request in Prepare-Bus-Sleep mode.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: Must not be defined if CanNmPassiveModeEnabled==true		

⌋

[ECUC_CanNm_00041] Definition of EcucBooleanParamDef CanNmImmediateTxconfEnabled ⌈

Parameter Name	CanNmImmediateTxconfEnabled		
Parent Container	CanNmGlobalConfig		
Description	Enable/disable the immediate tx confirmation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmImmediateTxconfEnabled shall not be enabled if CanNmPasiveModeEnabled is enabled.		

⌋

[ECUC_CanNm_00032] Definition of EcucFloatParamDef CanNmMainFunctionPeriod ⌈

Parameter Name	CanNmMainFunctionPeriod		
Parent Container	CanNmGlobalConfig		
Description	Call cycle in seconds of CanNm_MainFunction.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CanNm_00010] Definition of EcucBooleanParamDef CanNmPassiveModeEnabled ⌈

Parameter Name	CanNmPassiveModeEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling support of the Passive Mode.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_CanNm_00011] Definition of EcucBooleanParamDef CanNmPduRxIndicationEnabled ⌈

Parameter Name	CanNmPduRxIndicationEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling the PDU Rx Indication.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: calculationFormula = Equal(NmPduRxIndicationEnabled)		

⌋

[ECUC_CanNm_00055] Definition of EcucBooleanParamDef CanNmRemoteSleepIndEnabled ⌈

Parameter Name	CanNmRemoteSleepIndEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling remote sleep indication support. This feature is required for gateway nodes only.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: calculationFormula = If (CanNmPassiveModeEnabled == False) then Equal(NmRemoteSleepIndEnabled) else Equal(False)		

⌋

[ECUC_CanNm_00012] Definition of EcucBooleanParamDef CanNmStateChangeIndEnabled ⌈

Parameter Name	CanNmStateChangeIndEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling the CAN NM state change notification.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: calculationFormula = Equal(NmStateChangeIdEnabled)		

⌋

[ECUC_CanNm_00004] Definition of EcucBooleanParamDef CanNmUserDataEnabled ⌈

Parameter Name	CanNmUserDataEnabled		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling user data support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmUserDataEnabled shall be set to FALSE, if all bytes of the NM PDU are used for NM System Bytes and for the PNC bit vector and no space is left for user data. Otherwise the parameter shall be set according the following formular: calculationFormula =Equal(NmUserDataEnabled).		

⌋

[ECUC_CanNm_00003] Definition of EcucBooleanParamDef CanNmVersionInfo Api ⌈

Parameter Name	CanNmVersionInfoApi		
Parent Container	CanNmGlobalConfig		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

10.2.3 CanNmChannelConfig

[SWS_CanNm_00202] ⌈The container CanNmChannelConfig specifies configuration parameter that shall be located in a data structure of type CanNm_ConfigType.⌋

[SWS_CanNm_00203] ⌈Runtime configurable parameters listed below shall be configurable for each network management cluster separately.⌋

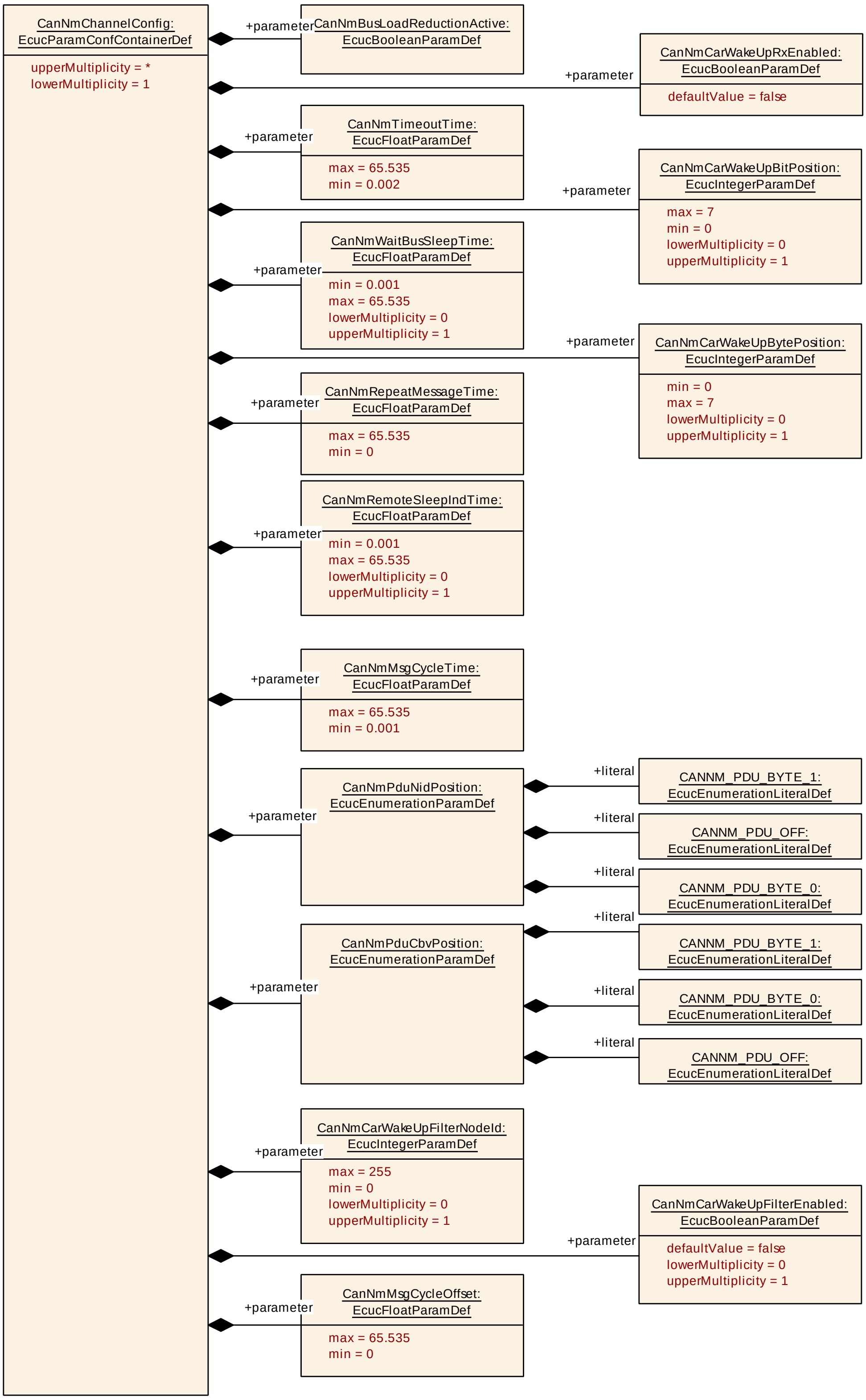

CanNmChannelConfig: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
CanNmBusLoadReductionActive: EcucBooleanParamDef
+parameter
CanNmCarWakeUpRxEnabled: EcucBooleanParamDef
defaultValue = false
+parameter
CanNmTimeoutTime: EcucFloatParamDef
max = 65.535
min = 0.002
+parameter
CanNmCarWakeUpBitPosition: EcucIntegerParamDef
max = 7
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanNmWaitBusSleepTime: EcucFloatParamDef
min = 0.001
max = 65.535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanNmCarWakeUpBytePosition: EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanNmRepeatMessageTime: EcucFloatParamDef
max = 65.535
min = 0
+parameter
CanNmRemoteSleepIndTime: EcucFloatParamDef
min = 0.001
max = 65.535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanNmMsgCycleTime: EcucFloatParamDef
max = 65.535
min = 0.001
+parameter
CanNmPduNidPosition: EcucEnumerationParamDef
+literal
CANNM_PDU_BYTE_1: EcucEnumerationLiteralDef
+literal
CANNM_PDU_OFF: EcucEnumerationLiteralDef
+literal
CANNM_PDU_BYTE_0: EcucEnumerationLiteralDef
+parameter
CanNmPduCbvPosition: EcucEnumerationParamDef
+literal
CANNM_PDU_BYTE_1: EcucEnumerationLiteralDef
+literal
CANNM_PDU_BYTE_0: EcucEnumerationLiteralDef
+literal
CANNM_PDU_OFF: EcucEnumerationLiteralDef
+parameter
CanNmCarWakeUpFilterNodeId: EcucIntegerParamDef
max = 255
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanNmCarWakeUpFilterEnabled: EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanNmMsgCycleOffset: EcucFloatParamDef
max = 65.535
min = 0

Figure 10.3: CanNm Channel Configuration `CanNmChannelConfig` Overview (1)

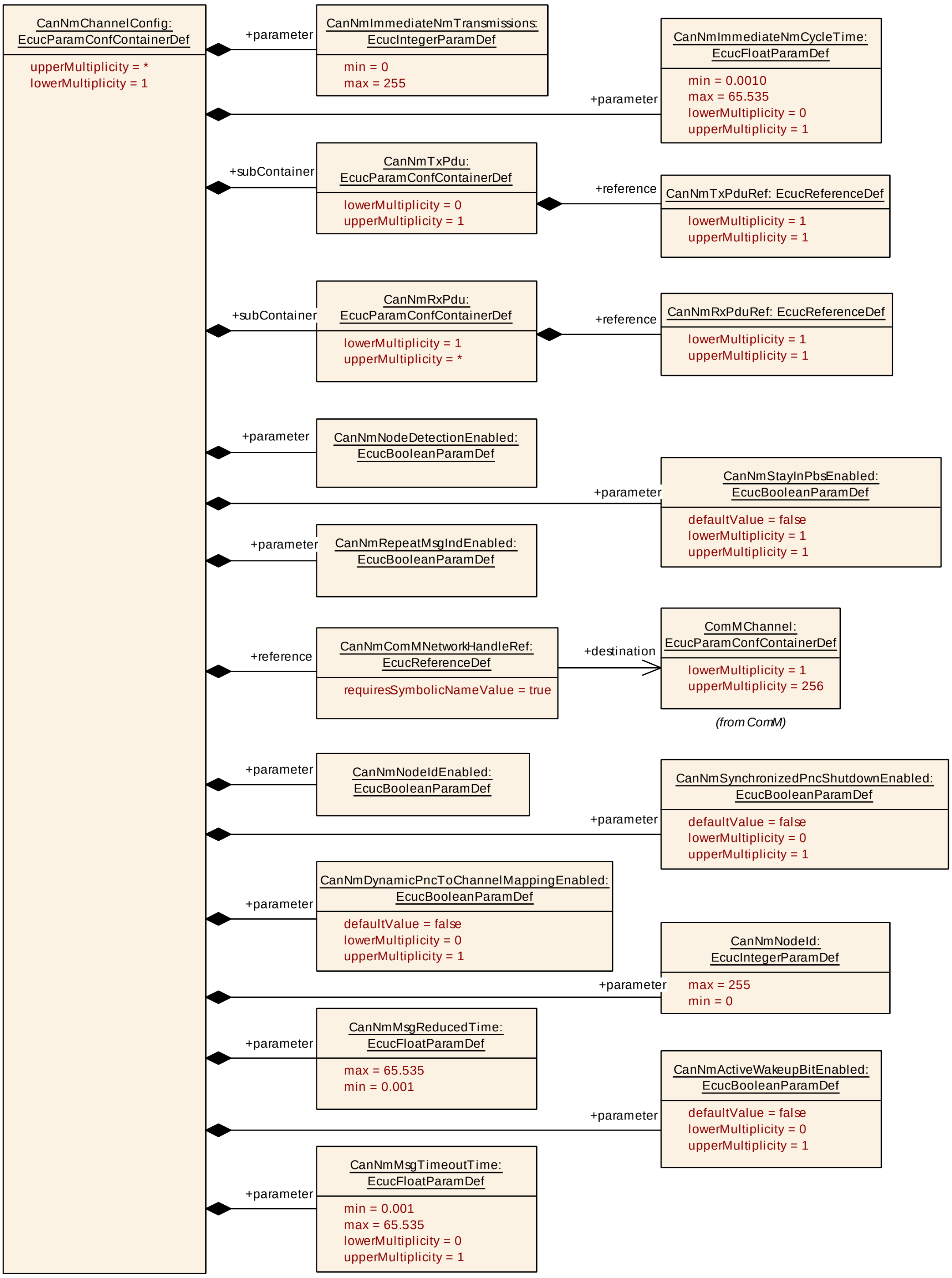


Figure 10.4: CanNm Channel Configuration CanNmChannelConfig Overview (2)

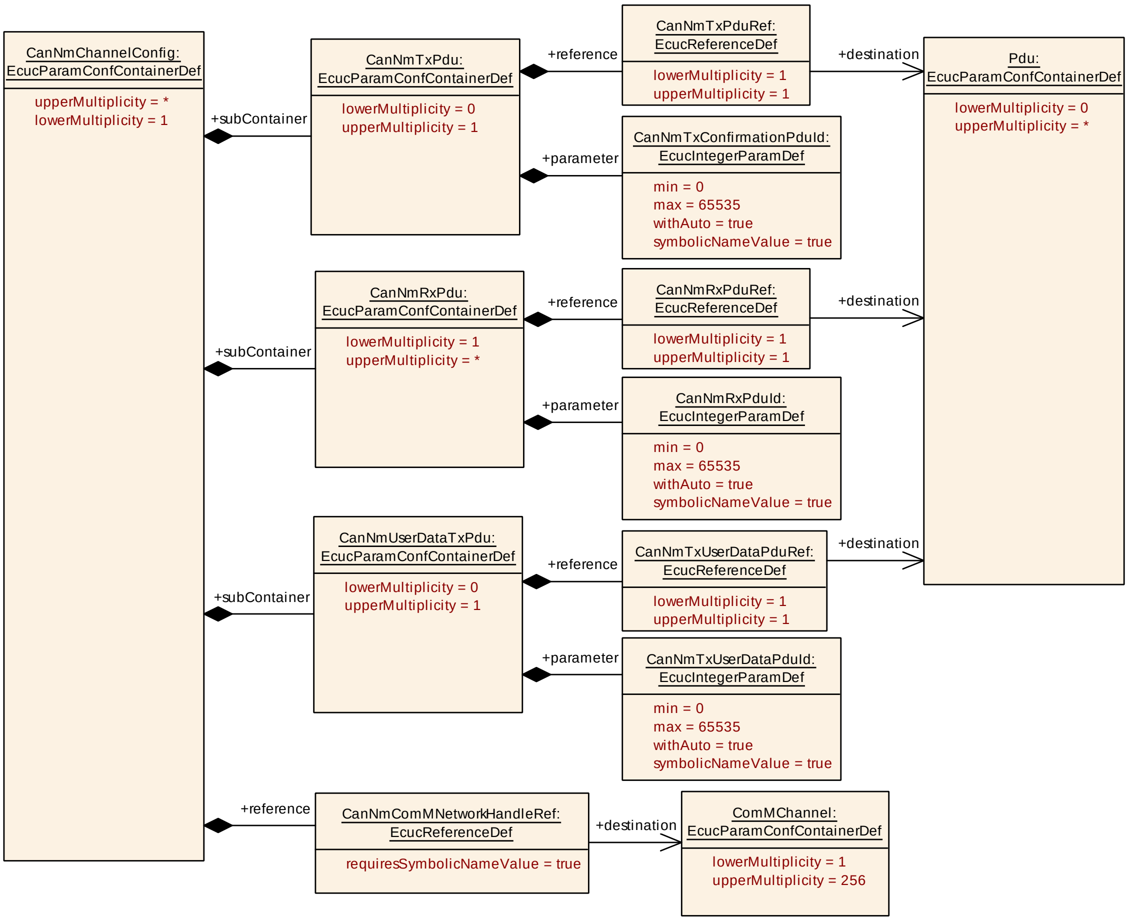


Figure 10.5: CanNm Channel Configuration `CanNmChannelConfig` Overview (3) –

[ECUC_CanNm_00017] Definition of EcucParamConfContainerDef CanNmChannelConfig ⌈

Container Name	CanNmChannelConfig		
Parent Container	CanNmGlobalConfig		
Description	This container contains the channel specific configuration parameter of the CanNm.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanNmActiveWakeupBitEnabled	0..1	[ECUC_CanNm_00084]
CanNmAllNmMessagesKeepAwake	0..1	[ECUC_CanNm_00068]
CanNmBusLoadReductionActive	1	[ECUC_CanNm_00042]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanNmCarWakeUpBitPosition	0..1	[ECUC_CanNm_00075]
CanNmCarWakeUpBytePosition	0..1	[ECUC_CanNm_00076]
CanNmCarWakeUpFilterEnabled	0..1	[ECUC_CanNm_00077]
CanNmCarWakeUpFilterNodeId	0..1	[ECUC_CanNm_00078]
CanNmCarWakeUpRxEnabled	1	[ECUC_CanNm_00074]
CanNmDynamicPncToChannelMappingEnabled	0..1	[ECUC_CanNm_00093]
CanNmImmediateNmCycleTime	0..1	[ECUC_CanNm_00057]
CanNmImmediateNmTransmissions	1	[ECUC_CanNm_00056]
CanNmMsgCycleOffset	1	[ECUC_CanNm_00029]
CanNmMsgCycleTime	1	[ECUC_CanNm_00028]
CanNmMsgReducedTime	1	[ECUC_CanNm_00043]
CanNmMsgTimeoutTime	0..1	[ECUC_CanNm_00030]
CanNmNodeDetectionEnabled	1	[ECUC_CanNm_00088]
CanNmNodeId	1	[ECUC_CanNm_00031]
CanNmNodeIdEnabled	1	[ECUC_CanNm_00090]
CanNmPduCbvPosition	1	[ECUC_CanNm_00026]
CanNmPduNidPosition	1	[ECUC_CanNm_00025]
CanNmPnEnabled	0..1	[ECUC_CanNm_00066]
CanNmPnHandleMultipleNetworkRequests	0..1	[ECUC_CanNm_00073]
CanNmRemoteSleepIndTime	0..1	[ECUC_CanNm_00023]
CanNmRepeatMessageTime	1	[ECUC_CanNm_00022]
CanNmRepeatMsgIndEnabled	1	[ECUC_CanNm_00089]
CanNmStayInPbsEnabled	1	[ECUC_CanNm_00092]
CanNmSynchronizedPncShutdownEnabled	0..1	[ECUC_CanNm_00097]
CanNmTimeoutTime	1	[ECUC_CanNm_00020]
CanNmWaitBusSleepTime	0..1	[ECUC_CanNm_00021]
CanNmComMNetworkHandleRef	1	[ECUC_CanNm_00018]

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 contains the CanNmTxConfirmationPduId and the CanNmTxPduRef.
CanNmUserDataTxPdu	0..1	This optional container is used to configure the UserNm PDU. This container is only available if CanNmComUserDataSupport is enabled.

⌋

[ECUC_CanNm_00084] Definition of EcucBooleanParamDef CanNmActiveWakeupBitEnabled ⌈

Parameter Name	CanNmActiveWakeupBitEnabled		
Parent Container	CanNmChannelConfig		
Description	Enables/Disables the handling of the Active Wakeup Bit in the CanNm module.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**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.		

⌋

[ECUC_CanNm_00068] Definition of EcucBooleanParamDef CanNmAllNmMessagesKeepAwake ⌈

Parameter Name	CanNmAllNmMessagesKeepAwake		
Parent Container	CanNmChannelConfig		
Description	Specifies if CanNm drops irrelevant NM PDUs. false: Only NM PDUs with a PNI bit = true and containing a PN request for this ECU triggers the standard RX indication handling true: Every NM PDU triggers the standard RX indication handling		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

▽

△

Scope / Dependency	scope: local dependency: only valid if NmPnEiraCalcEnabled == true or NmPnEraCalcEnabled == true		

⌋

[ECUC_CanNm_00042] Definition of EcucBooleanParamDef CanNmBusLoadReductionActive ⌈

Parameter Name	CanNmBusLoadReductionActive		
Parent Container	CanNmChannelConfig		
Description	This parameter defines if bus load reduction for the respective NM channel is active or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: CanNmBusLoadReductionActive = false if CanNmBusLoadReduction Enabled == false		

⌋

[ECUC_CanNm_00075] Definition of EcucIntegerParamDef CanNmCarWakeUp BitPosition ⌈

Parameter Name	CanNmCarWakeUpBitPosition		
Parent Container	CanNmChannelConfig		
Description	Specifies the Bit position of the CWU within the NM PDU.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

▽

△

Scope / Dependency	scope: local dependency: only available if CanNmCarWakeUpRxEnabled == TRUE

⌋

[ECUC_CanNm_00076] Definition of EcucIntegerParamDef CanNmCarWakeUpBytePosition ⌈

Parameter Name	CanNmCarWakeUpBytePosition		
Parent Container	CanNmChannelConfig		
Description	Specifies the Byte position of the CWU within the NM PDU.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: only available if CanNmCarWakeUpRxEnabled == TRUE CanNmCarWakeupBytePosition >= number of enabled system bytes (CBV, NID)		

⌋

[ECUC_CanNm_00077] Definition of EcucBooleanParamDef CanNmCarWakeUpFilterEnabled ⌈

Parameter Name	CanNmCarWakeUpFilterEnabled		
Parent Container	CanNmChannelConfig		
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	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD

▽

△

	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: only available if CanNmCarWakeUpRxEnabled == TRUE		

⌋

[ECUC_CanNm_00078] Definition of EcucIntegerParamDef CanNmCarWakeUpFilterNodeId ⌈

Parameter Name	CanNmCarWakeUpFilterNodeId		
Parent Container	CanNmChannelConfig		
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	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: only available if CanNmCarWakeUpFilterEnabled == TRUE		

⌋

[ECUC_CanNm_00074] Definition of EcucBooleanParamDef CanNmCarWakeUpRxEnabled ⌈

Parameter Name	CanNmCarWakeUpRxEnabled
Parent Container	CanNmChannelConfig
Description	Enables or disables support of CarWakeUp bit evaluation in received NM PDUs. FALSE - CarWakeUp not supported TRUE - CarWakeUp supported
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false

▽

△

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_CanNm_00093] Definition of EcucBooleanParamDef CanNmDynamicPncToChannelMappingEnabled ⌈

Parameter Name	CanNmDynamicPncToChannelMappingEnabled		
Parent Container	CanNmChannelConfig		
Description	Channel-specific parameter to enable the dynamic PNC-to-channel-mapping feature. False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: Shall only be TRUE if CanNmDynamicPncToChannelMappingSupport is TRUE		

⌋

[ECUC_CanNm_00057] Definition of EcucFloatParamDef CanNmImmediateNmCycleTime ⌈

Parameter Name	CanNmImmediateNmCycleTime		
Parent Container	CanNmChannelConfig		
Description	Defines the immediate NM PDU cycle time in seconds which is used for CanNmImmediateNmTransmissions NM PDU transmissions.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD

▽

△

	Post-build time	–	
Value Configuration Class	**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 CanNmImmediateNmTransmissions is greater one.		

⌋

[ECUC_CanNm_00056] Definition of EcucIntegerParamDef CanNmImmediateNmTransmissions ⌈

Parameter Name	CanNmImmediateNmTransmissions		
Parent Container	CanNmChannelConfig		
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	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: If CanNmImmediateRestartEnabled = true then CanNmImmediateNmTransmissions = 0 If CanNmPnHandleMultipleNetworkRequests == True" then "CanNmImmediateNmTransmissions > 0		

⌋

[ECUC_CanNm_00029] Definition of EcucFloatParamDef CanNmMsgCycleOffset ⌈

Parameter Name	CanNmMsgCycleOffset		
Parent Container	CanNmChannelConfig		
Description	Time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: Parameter value < CanNmMsgCycleTime This parameter is only valid if CanNmPassiveModeEnabled is False.		

⌋

[ECUC_CanNm_00028] Definition of EcucFloatParamDef CanNmMsgCycleTime ⌈

Parameter Name	CanNmMsgCycleTime		
Parent Container	CanNmChannelConfig		
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	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: This parameter is only valid if CanNmPassiveModeEnabled is False.		

⌋

[ECUC_CanNm_00043] Definition of EcucFloatParamDef CanNmMsgReducedTime ⌈

Parameter Name	CanNmMsgReducedTime		
Parent Container	CanNmChannelConfig		
Description	Node specific bus cycle time in the periodic transmission mode with bus load reduction. Specified in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

▽

△

Scope / Dependency	scope: local dependency: 0,5 * CanNmMsgCycleTime <= CanNmMsgReducedTime < CanNmMsgCycleTime This parameter is only valid if CanNmBusLoadReductionEnabled == True and CanNmBusLoadReductionActive == True and CanNmPassiveModeEnabled == False Otherwise this parameter is notused.		

⌋

[ECUC_CanNm_00030] Definition of EcucFloatParamDef CanNmMsgTimeoutTime ⌈

Parameter Name	CanNmMsgTimeoutTime		
Parent Container	CanNmChannelConfig		
Description	When using Partial Network and this timeout is defined then CanNm monitors that a NM-PDU is transmitted successfully within this Transmission Timeout Time and provides an error notification otherwise.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: CanNmMsgTimeoutTime < CanNmMsgCycleTime This parameter is only valid if CanNmPassiveModeEnabled and CanNmImmediateTxConfEnabled are set to FALSE and CanNmPnEnabled is set to TRUE.		

⌋

[ECUC_CanNm_00088] Definition of EcucBooleanParamDef CanNmNodeDetectionEnabled ⌈

Parameter Name	CanNmNodeDetectionEnabled		
Parent Container	CanNmChannelConfig		
Description	Precompile time switch to enable the node detection feature.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: Only valid if CanNmNodeIdEnabled is set to TRUE If CanNmPassiveModeEnabled == True then CanNmNodeDetection = False		

⌋

[ECUC_CanNm_00031] Definition of EcucIntegerParamDef CanNmNodeId ⌈

Parameter Name	CanNmNodeId		
Parent Container	CanNmChannelConfig		
Description	Node identifier of local node.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: This parameter is only valid if CanNmNodeIdEnabled == True		

⌋

[ECUC_CanNm_00090] Definition of EcucBooleanParamDef CanNmNodeIdEnabled ⌈

Parameter Name	CanNmNodeIdEnabled		
Parent Container	CanNmChannelConfig		
Description	Pre-processor switch for enabling the source node identifier.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: calculationFormula = Equal(NmNodeIdEnabled)		

⌋

[ECUC_CanNm_00026] Definition of EcucEnumerationParamDef CanNmPduCbvPosition ⌈

Parameter Name	CanNmPduCbvPosition		
Parent Container	CanNmChannelConfig		
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	EcucEnumerationParamDef		
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	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmPduNidPosition; If CanNmNodeDetectionEnabled == true then CanNmPduCbvPosition != CANNM_PDU_OFF if(CanNmPduCbvPosition != CANNM_PDU_OFF && CanNmPduNidPosition != CANNM_PDU_OFF) then CanNmPduCbvPosition != CanNmPduNidPosition if(CanNmPduCbvPosition != CANNM_PDU_OFF && CanNmPduNidPosition == CANNM_PDU_OFF) then CanNmPduCbvPosition = CANNM_PDU_BYTE0		

⌋

[ECUC_CanNm_00025] Definition of EcucEnumerationParamDef CanNmPduNidPosition ⌈

Parameter Name	CanNmPduNidPosition		
Parent Container	CanNmChannelConfig		
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 (CANNM_PDU_BYTE_0 means byte 0, CANNM_PDU_BYTE_1 means byte 1, CANNM_PDU_OFF means source node identifier is not part of the NM PDU) ImplementationType: CanNm_PduPositionType		
Multiplicity	1		
Type	EcucEnumerationParamDef		
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	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

▽

△

Scope / Dependency	scope: ECU dependency: CanNmPduCbvPosition; If CanNmNodeIdEnabled == true then CanNmPduNidPosition != CANNM_PDU_OFF if(CanNmPduNidPosition != CANNM_PDU_OFF && CanNmPduCbvPosition != CANNM_PDU_OFF) then CanNmPduNidPosition != CanNmPduCbvPosition if(CanNmPduNidPosition != CANNM_PDU_OFF && CanNmPduCbvPosition == CANNM_PDU_OFF) then CanNmPduNidPosition = CANNM_PDU_BYTE0

⌋

[ECUC_CanNm_00066] Definition of EcucBooleanParamDef CanNmPnEnabled ⌈

Parameter Name	CanNmPnEnabled		
Parent Container	CanNmChannelConfig		
Description	Enables or disables support of partial networking. false: Partial networking Range not supported true: Partial networking supported		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: only valid if CanNmGlobalPnSupport == true		

⌋

[ECUC_CanNm_00073] Definition of EcucBooleanParamDef CanNmPnHandleMultipleNetworkRequests ⌈

Parameter Name	CanNmPnHandleMultipleNetworkRequests		
Parent Container	CanNmChannelConfig		
Description	Specifies if CanNm performs an additional transition from Network Mode to Repeat Message State (true) or not (false).		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

▽

△

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: only valid if CanNmGlobalPnSupport == true		

⌋

[ECUC_CanNm_00023] Definition of EcucFloatParamDef CanNmRemoteSleepIndTime ⌈

Parameter Name	CanNmRemoteSleepIndTime		
Parent Container	CanNmChannelConfig		
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	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: CanNmRemoteSleepIndTime >= CanNmMsgCycleTime CanNmRemoteSleepIndTime is only required if CanNmRemoteSleepIndEnabled = true		

⌋

[ECUC_CanNm_00022] Definition of EcucFloatParamDef CanNmRepeatMessageTime ⌈

Parameter Name	CanNmRepeatMessageTime		
Parent Container	CanNmChannelConfig		
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	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

▽

△

Scope / Dependency	scope: local dependency: CanNmRepeatMessageTime = n * CanNmMsgCycleTime; CanNmRepeatMessageTime > CanNmImmediateNmTransmissions * CanNmImmediateNmCycleTime 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.

⌋

[ECUC_CanNm_00089] Definition of EcucBooleanParamDef CanNmRepeatMsgIndEnabled ⌈

Parameter Name	CanNmRepeatMsgIndEnabled		
Parent Container	CanNmChannelConfig		
Description	Enable/disable the notification that a RepeatMessageRequest bit has been received.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmRepeatMsgIndEnabled = FALSE if CanNmPassiveModeEnabled == TRUE or (CanNmNodeDetectionEnabled == FALSE && CanNmDynamicPncToChannelMappingEnabled == FALSE). CanNmRepeatMsgIndEnabled = TRUE if CanNmDynamicPncToChannelMappingEnabled == TRUE.		

⌋

[ECUC_CanNm_00092] Definition of EcucBooleanParamDef CanNmStayInPbsEnabled ⌈

Parameter Name	CanNmStayInPbsEnabled		
Parent Container	CanNmChannelConfig		
Description	If this parameter is disabled Prepare Bus-Sleep Mode is left after CanNmWaitBusSleepTime. If this parameter is enabled Prepare Bus-Sleep Mode can only be left if ECU is powered off or any restart reason applies.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD

▽

△

	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CanNm_00097] Definition of EcucBooleanParamDef CanNmSynchronizedPncShutdownEnabled ⌈

Parameter Name	CanNmSynchronizedPncShutdownEnabled		
Parent Container	CanNmChannelConfig		
Description	Specifies if CanNm handle PN shutdown messages to support a synchronized PNC shutdown across a PN topology. This is only used for ECUs in the role of a top-level PNC coordinator or intermediate PNC coordinator. Thus, the PNC gateway functionality is enabled and therefore ERA calculation is used. FALSE: synchronized PNC shutdown is disabled TRUE: synchronized PNC shutdown is enabled		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: Only available if CanNmPnEnabled == TRUE and NmPnEraCalcEnabled == TRUE.		

⌋

[ECUC_CanNm_00020] Definition of EcucFloatParamDef CanNmTimeoutTime ⌈

Parameter Name	CanNmTimeoutTime		
Parent Container	CanNmChannelConfig		
Description	If NM is in Ready Sleep State it denotes the time in seconds how long after the last NM PDU transmission or reception state transition into the Prepare Bus-Sleep Mode is initiated. If NM is in Repeat Message or Normal Operation state and no NM PDU can be transmitted or received within this time, a run-time error is raised.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0.002 .. 65.535]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CanNmTimeoutTime > CanNmMsgCycleTime It shall be equal for all nodes in the cluster. It shall be greater than CanNmMsgCycleTime.		

⌋

[ECUC_CanNm_00021] Definition of EcucFloatParamDef CanNmWaitBusSleepTime ⌈

Parameter Name	CanNmWaitBusSleepTime		
Parent Container	CanNmChannelConfig		
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	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: It shall be equal for all nodes in the cluster. It shall be long enough to make all Tx-buffer empty. In case CanNmStayInPbsEnabled is disabled this parameter shall be mandatory.		

⌋

[ECUC_CanNm_00018] Definition of EcucReferenceDef CanNmComMNetworkHandleRef ⌈

Parameter Name	CanNmComMNetworkHandleRef		
Parent Container	CanNmChannelConfig		
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId.		
Multiplicity	1		
Type	Symbolic name reference to ComMChannel		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.4 CanNmRxPdu

[ECUC_CanNm_00038] Definition of EcucParamConfContainerDef CanNmRxPdu ⌈

Container Name	CanNmRxPdu
Parent Container	CanNmChannelConfig
Description	This container is used to configure the Rx PDU properties that are used for the CanNm Channel.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanNmRxPduId	1	[ECUC_CanNm_00054]
CanNmRxPduRef	1	[ECUC_CanNm_00039]

No Included Containers

⌋

[ECUC_CanNm_00054] Definition of EcucIntegerParamDef CanNmRxPduId ⌈

Parameter Name	CanNmRxPduId		
Parent Container	CanNmRxPdu		
Description	This parameter defines the Rx PDU ID of the CanIf L-PDU range that is associated with this CanNm channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU withAuto = true		

⌋

[ECUC_CanNm_00039] Definition of EcucReferenceDef CanNmRxPduRef ⌈

Parameter Name	CanNmRxPduRef
Parent Container	CanNmRxPdu
Description	Reference to the global PDU that is used by this CanNm channel.
Multiplicity	1
Type	Reference to Pdu

▽

△

Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.5 CanNmTxPdu

[ECUC_CanNm_00036] Definition of EcucParamConfContainerDef CanNmTxPdu ⌈

Container Name	CanNmTxPdu
Parent Container	CanNmChannelConfig
Description	This container contains the CanNmTxConfirmationPduId and the CanNmTxPduRef.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanNmTxConfirmationPduId	1	[ECUC_CanNm_00048]
CanNmTxPduRef	1	[ECUC_CanNm_00037]

No Included Containers

⌋

[ECUC_CanNm_00048] Definition of EcucIntegerParamDef CanNmTxConfirmationPduId ⌈

Parameter Name	CanNmTxConfirmationPduId		
Parent Container	CanNmTxPdu		
Description	Handle Id to be used by the Lower Layer to confirm the transmission of the CanNmTx Pdu to the LowerLayer.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: ECU withAuto = true

⌋

[ECUC_CanNm_00037] Definition of EcucReferenceDef CanNmTxPduRef ⌈

Parameter Name	CanNmTxPduRef		
Parent Container	CanNmTxPdu		
Description	The reference to the common PDU structure.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.2.6 CanNmUserDataTxPdu

[ECUC_CanNm_00045] Definition of EcucParamConfContainerDef CanNmUserDataTxPdu ⌈

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

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanNmTxUserDataPduId	1	[ECUC_CanNm_00047]
CanNmTxUserDataPduRef	1	[ECUC_CanNm_00046]

No Included Containers

⌋

[ECUC_CanNm_00047] Definition of EcucIntegerParamDef CanNmTxUserDataPduId ⌈

Parameter Name	CanNmTxUserDataPduId		
Parent Container	CanNmUserDataTxPdu		
Description	This parameter defines the Handle ID of the NM User Data I-PDU.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU withAuto = true		

⌋

[ECUC_CanNm_00046] Definition of EcucReferenceDef CanNmTxUserDataPduRef ⌈

Parameter Name	CanNmTxUserDataPduRef		
Parent Container	CanNmUserDataTxPdu		
Description	Reference to the NM User Data I-PDU in the global PDU collection.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral [4].

A Examples

A.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 CanNmMsgReducedTime are sending alternating their Network Management PDUs. After a while node 1 goes into “ready sleep” state. Now node 2 and node 3 are sending alternating Network Management 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 messages with CanNmMsgCycleTime.

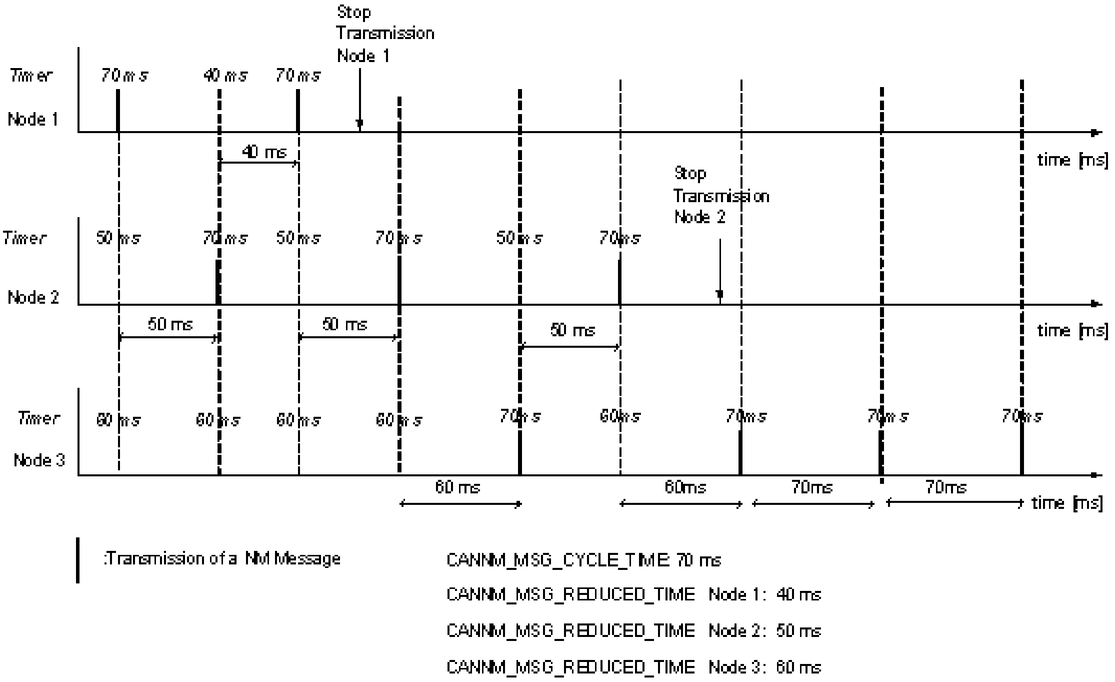


Figure A.1: Example for Bus Load Reduction

A.2 Example timing behavior for Network Management PDUs

Assume an example network of three nodes 1, 2, 3 (see also Figure A.2). Nodes specific cycle offsets are equal respectively to t1 < t2 < t3 < T. NM cycle time is equal to T (see Figure A.3).

Network Management PDUs sent on the bus within the Repeat Message State are presented in the Figure A.4, and within the Normal Operation / Ready Sleep State in Figure A.5. Each dot in Figure A.5 denotes restart of the NM-Timeout Timer.

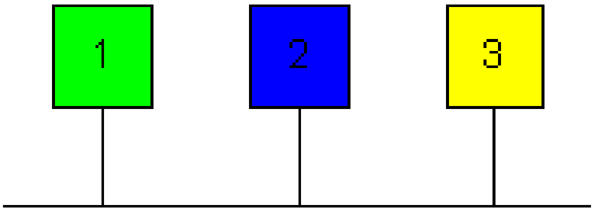


Figure A.2: Example for 3 ECUs connected to a network

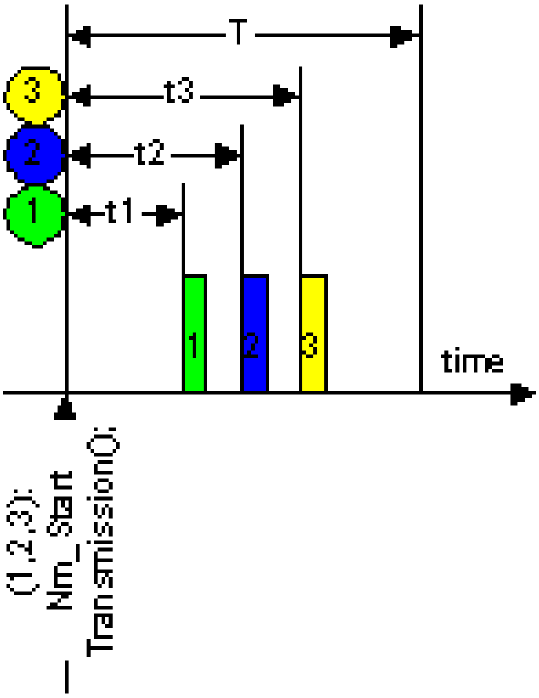


Figure A.3: Example for NM Transmission Start of different ECUs

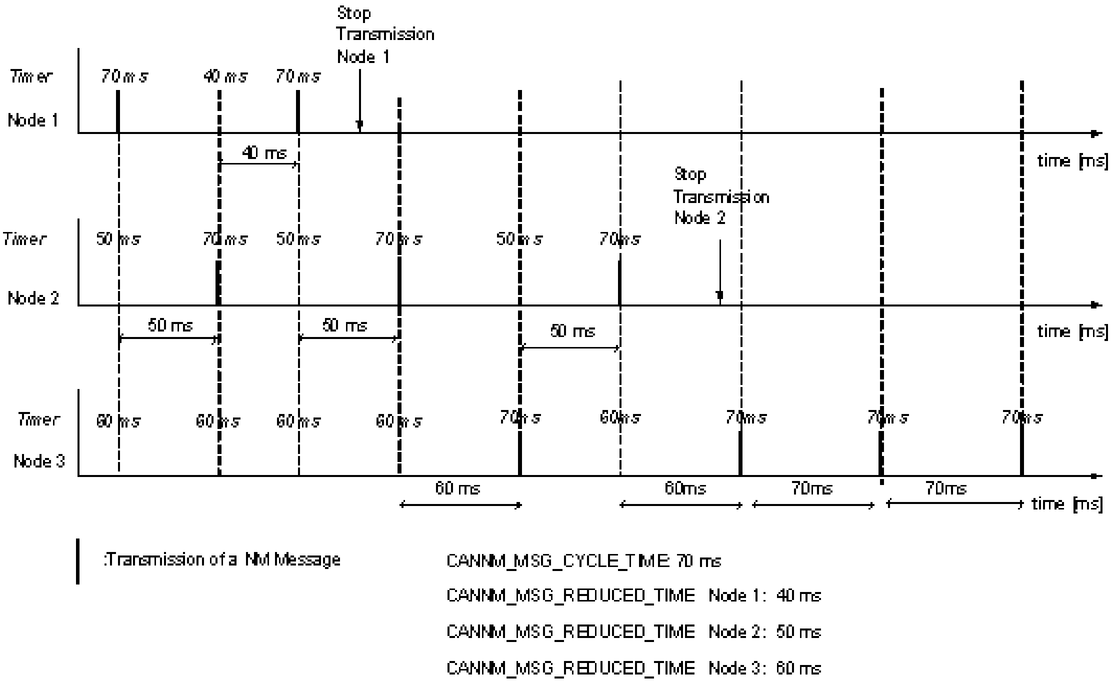


Figure A.4: Example for NM Transmission Handling of multiple ECUs

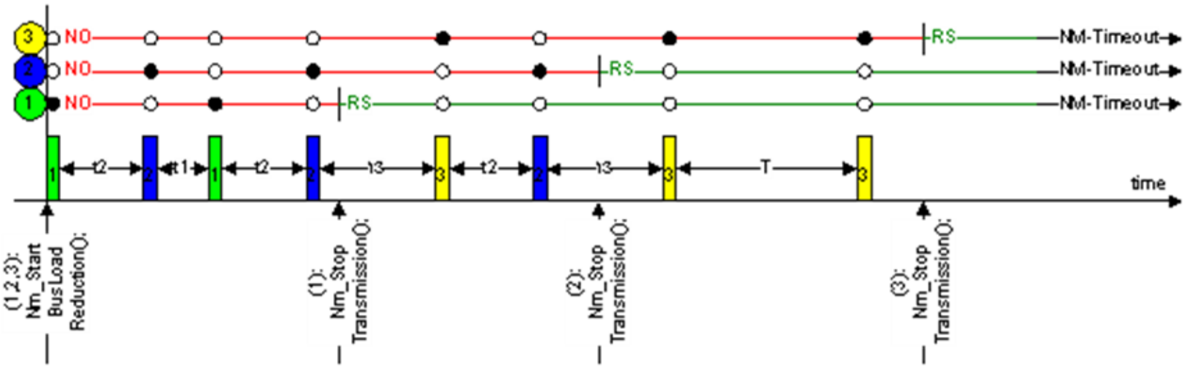


Figure A.5: Example for NM Timeout Handling

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

none

B.1.2 Changed Specification Items in R24-11

Number	Heading
[SWS_CanNm_00064]	
[SWS_CanNm_00253]	
[SWS_CanNm_00312]	
[SWS_CanNm_00325]	Definition of optional interfaces requested by module CanNm
[SWS_CanNm_00328]	
[SWS_CanNm_00335]	
[SWS_CanNm_00512]	
[SWS_CanNm_00514]	
[SWS_CanNm_00517]	
[SWS_CanNm_00519]	

Table B.1: Changed Specification Items in R24-11

B.1.3 Deleted Specification Items in R24-11

none

C Not applicable requirements

[SWS_CanNm_NA_00000]

Upstream requirements: SRS_BSW_00170, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00449, SRS_BSW_00454, SRS_BSW_00457

⌈This specification item references requirements that are not applicable, because CanNm has no interdepencies to SW Components.⌋

[SWS_CanNm_NA_00001]

Upstream requirements: SRS_BSW_00375, SRS_BSW_00424, SRS_BSW_00429, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00314, SRS_BSW_00479

⌈This specification item references requirements that are not applicable, because CanNm does not implement any interrupts, is not a driver or MCAL abstraction layer or has any direct access to OS.⌋

[SWS_CanNm_NA_00002]

Upstream requirements: SRS_BSW_00488, SRS_BSW_00489, SRS_BSW_00490, SRS_BSW_00491, SRS_BSW_00492, SRS_BSW_00493

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

[SWS_CanNm_NA_00003]

Upstream requirements: SRS_BSW_00425, SRS_BSW_00427

⌈This specification item references requirements that are not applicable, because BSW module description template is not part of the CanNm SWS.⌋

[SWS_CanNm_NA_00004]

Upstream requirements: SRS_BSW_00426

⌈This specification item references requirements that are not applicable, because CanNm does not share any data with other BSW.⌋

[SWS_CanNm_NA_00005]

Upstream requirements: SRS_BSW_00432

⌈This specification item references requirements that are not applicable, because CanNm does not propagate data through different layers.⌋

[SWS_CanNm_NA_00007]

Upstream requirements: SRS_BSW_00417, SRS_BSW_00386, SRS_BSW_00458, SRS_BSW_00466, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472

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

[SWS_CanNm_NA_00008]

Upstream requirements: SRS_BSW_00160, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00341, SRS_BSW_00416, SRS_BSW_00312, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00343, SRS_BSW_00345, SRS_BSW_00351, SRS_BSW_00357, SRS_BSW_00377, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00399, SRS_BSW_00401, SRS_BSW_00402, SRS_BSW_00406, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00448, SRS_BSW_00453, SRS_BSW_00456, SRS_BSW_00462, SRS_BSW_00478, RS_Nm_00043, RS_Nm_00044, RS_Nm_00048, RS_Nm_00145, RS_Nm_00146, RS_Nm_00150, RS_Nm_00154, RS_Nm_02514, RS_Nm_02515, RS_Nm_02535, RS_Nm_02537, RS_Nm_02574, RS_Nm_00144, RS_Nm_00152, RS_Nm_02546, RS_Nm_02550, RS_Nm_02561, RS_Nm_02562, RS_Nm_02563, RS_Nm_02564, RS_Nm_02566, RS_Nm_02567

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

[SWS_CanNm_NA_00009]

Upstream requirements: SRS_BSW_00459, SRS_BSW_00494

⌈This specification item references requirements that are not applicable, because CanNm does not have any service functionality.⌋