Document Title	Specification of Generic Network Management Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	228
Document Classification	Standard

Document Version	1.4.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	1.4.0	AUTOSAR Release Management	• Correct chapter Expected Interfaces • Add missing callback APIs • Exclude DEM usage. • Solve duplicated requirement ID • Editorial changes
17.05.2012	1.3.0	AUTOSAR Administration	• Support of Shutdown of nested Subbusses added • Cleanup of API-Service Attribute Sync/Async
15.04.2011	1.2.0	AUTOSAR Administration	• Add interface support added (minor change) • Support of sending Nm statechanges in Nm user data. (minor change) • Introduction of CarWakeUp-functionality (minor change) • Backup Coordinator added (major change)
10.09.2010	1.1.0	AUTOSAR Administration	• Fix of description of Nm_State_Notification • Legal disclaimer revised
23.06.2008	1.0.1	AUTOSAR Administration	Legal disclaimer revised
03.12.2007	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

This document describes the concept, interfaces and configuration of the Generic Network Management Interface module.

The Generic Network Management Interface is an adaptation layer between the AUTOSAR Communication Manager and the AUTOSAR Bus specific network management modules (e.g. CAN Network Management or FlexRay Network Management)

Additionally, this document describes the interoperability between several networks connected to the same (coordinator) ECU that run AUTOSAR NM, where 'interoperability' means that these networks can be put to sleep synchronously. This functionality is also referred to as 'NM coordinator'. As an extension, some of the networks in question can be direct OSEK NM (ISO 17356-5) (see [9]).

Support of the NM coordinator functionality is optional. An AUTOSAR NM Interface implementation can either support:

- NM Interface functionality without any NM coordinator functionality
- NM Interface functionality with NM coordinator functionality limited to support synchronous shutdown of busses with AUTOSAR NM running on the busses
- NM Interface functionality with NM coordinator functionality to support synchronous shutdown of busses with AUTOSAR NM and direct OSEK NM running on the busses

2 Acronyms and abbreviations

Acronym:	Description:
CC	Communication Controller
CWU	Car Wake Up
NM	Network Management
NMS	Network Management State

Abbreviation:	Description:
CanIf	CAN Interface
CanNm	CAN NM
CDD	Complex Device Driver
ComM	Communication Manager
EcuM	ECU State Manager
DET	Development Error Tracer

Term:	Definition:
Bus-Sleep Mode	Network mode where all interconnected communication controllers are in the sleep mode.
NM-Channel	Logical channel associated with the NM-cluster
NM-Cluster	Set of NM nodes coordinated with use of the NM algorithm
NM-Coordinator	A functionality of the Generic NM Interface which allows coordination of network sleep for multiple NM Channels.
active NM-Coordinator	The NM-Coordinator which synchronizes the shutdown of all other NM-Coordinators connected to the Network
passive NM-Coordinator	An NM-Coordinator which is going to shutdown synchronized by an active NM-Coordinator
NM-Message	Packet of information exchanged for purposes of the NM algorithm.
NM-Timeout	Timeout in the NM algorithm that initiates transition into Bus-Sleep Mode.
NM User Data	Supplementary application specific piece of data that is attached to every NM message sent on the bus.
Node Identifier	Node address information exchanged for purposes of the NM algorithm.
Node Identifier List	List of Node Identifiers recognized by the NM algorithm.

3 Related documentation

3.1 Input documents

[1] AUTOSARLayered Software Architecture
AUTOSAR_SoftwareArchitecture.pdf

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

[3] AUTOSAR Requirements on Basic Software, Module NM
AUTOSAR_SRS_ NM.pdf

[4] AUTOSAR Software Specification of CAN Generic Network Management,
AUTOSAR_SWS_CAN _NM.pdf

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

[6] AUTOSAR Software Specification of Module Communication Manager
AUTOSAR_SWS_ComManager.pdf

[7] AUTOSAR Software Specification of Module ECU State Manager
AUTOSAR_SWS_EcuStateManager.pdf

[8] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

[9] OSEK/VDX NM Specification (ISO 17356-5), Version 2.5.3
http://ww.osek-vdx.org/

4 Constraints and assumptions

4.1 Limitations

a. The AUTOSAR NM Interface can only be applied to communication systems that support broadcast communication and Bus-sleep mode.
b. There will be only one instance of the NM Interface layer for all NM-Clusters, this instance manages all channels where AUTOSAR NM is run.
c. The NM Interface shall only include the common modes, definitions and return values of different Bus Specific Network management layers.

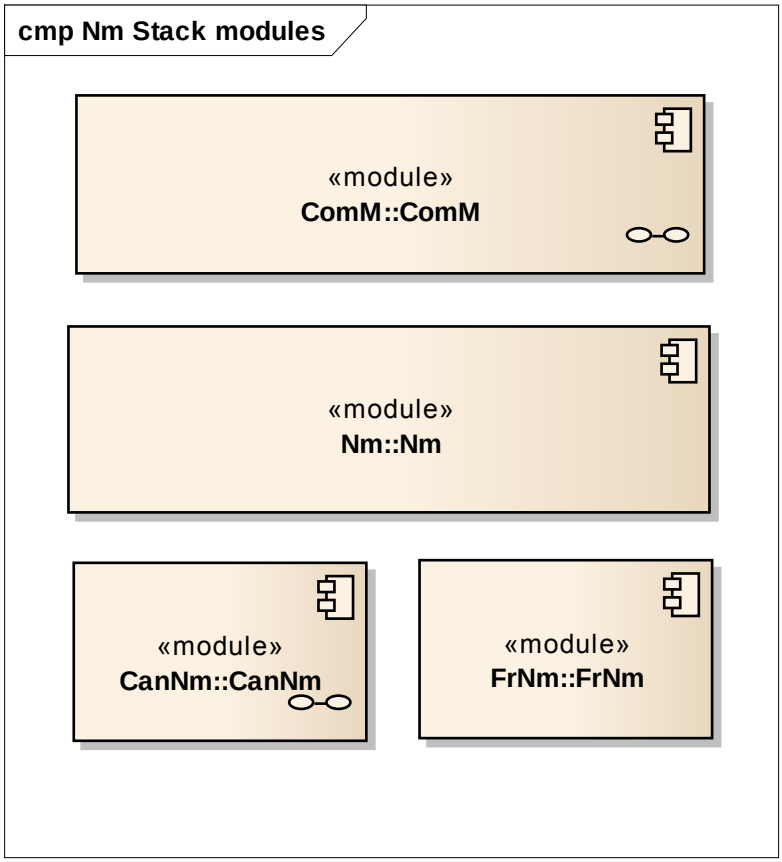

Figure 4.1 Nm Stack Modules

4.2 Specific limitations of the coordinator functionality for AUTOSAR release 3

For the inclusion of OSEK NM, the following restriction applies:

- No support to run AUTOSAR NM and direct OSEK NM on the same bus

In addition, for the inclusion of direct OSEK NM, the following assumption is made:

- **Nm049** If NM_COORDINATOR_SUPPORT_ENABLED and NM_OSEK_SUPPORT_ENABLED are set to TRUE, it has to be ensured that OSEK NM provides the RemoteSleepIndication functionality.

4.3 Applicability to car domains

The AUTOSAR NM Interface is generic and provides flexible configuration; it is independent of the underlying communication system and can be applied to any car domain under limitations provided above.

5 Dependencies to other modules

5.1 Interfaces to modules

5.1.1 ComM, CanNm, FrNm, LinNm, CDD

The Generic Network Management Interface Module provides services to the Communication Manager (ComM) and uses services of the respective bus specific Network management adaptation layers (CanNm, FrNm and LinNm[1]) as shown in Figure 5.1.
The CarWakeup-Functionality in Release 3.2 needs Complex Device Driver which Coordinates Basic Software Mode Management.

[1] LinNm is not within the scope of the current AUTOSAR NM Release

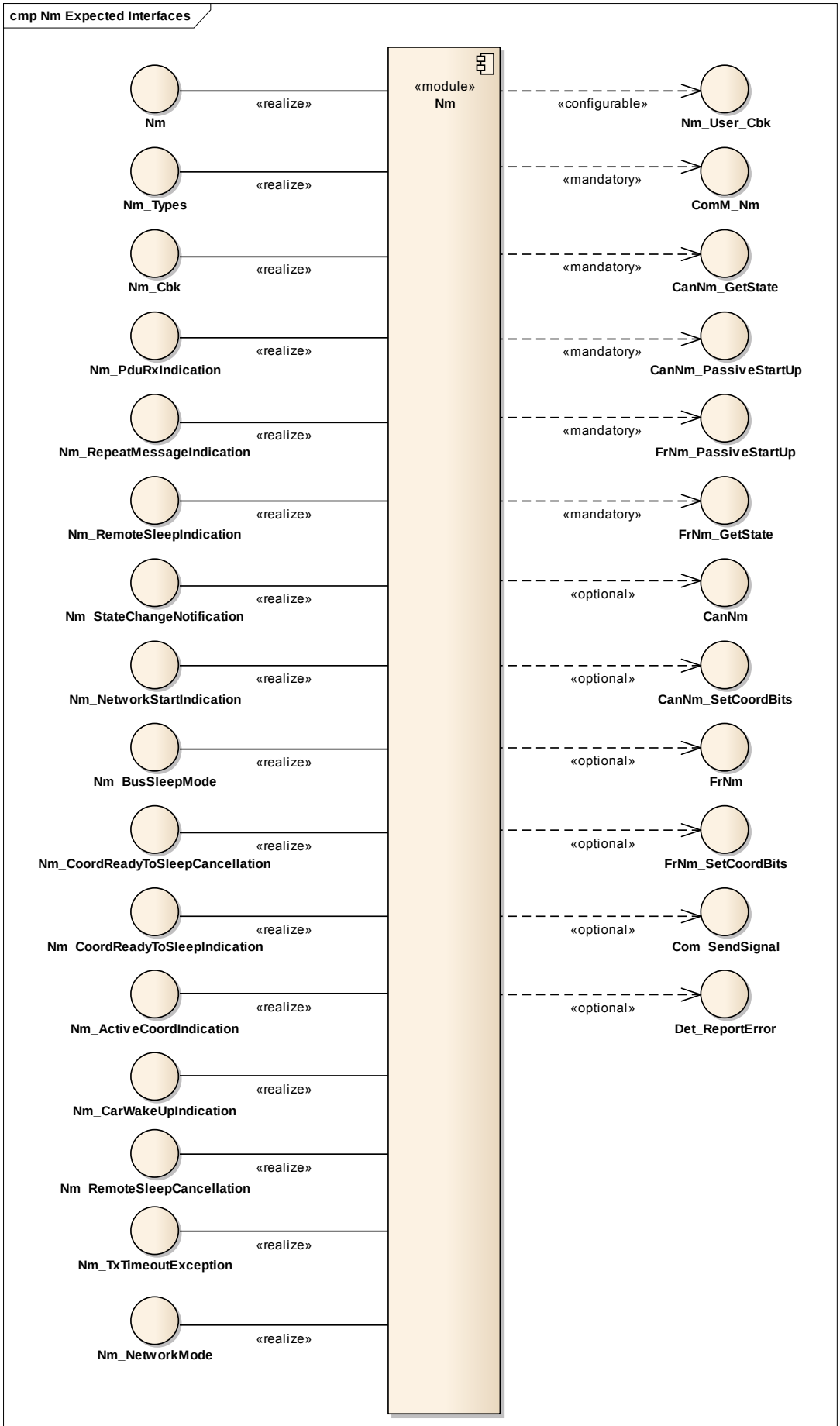
cmp Nm Expected Interfaces
«module»
Nm
Nm
«realize»
Nm_Types
«realize»
Nm_Cbk
«realize»
Nm_PduRxIndication
«realize»
Nm_RepeatMessageIndication
«realize»
Nm_RemoteSleepIndication
«realize»
Nm_StateChangeNotification
«realize»
Nm_NetworkStartIndication
«realize»
Nm_BusSleepMode
«realize»
Nm_CoordReadyToSleepCancellation
«realize»
Nm_CoordReadyToSleepIndication
«realize»
Nm_ActiveCoordIndication
«realize»
Nm_CarWakeUpIndication
«realize»
Nm_RemoteSleepCancellation
«realize»
Nm_TxTimeoutException
«realize»
Nm_NetworkMode
«configurable»
Nm_User_Cbk
«mandatory»
ComM_Nm
«mandatory»
CanNm_GetState
«mandatory»
CanNm_PassiveStartUp
«mandatory»
FrNm_PassiveStartUp
«mandatory»
FrNm_GetState
«optional»
CanNm
«optional»
CanNm_SetCoordBits
«optional»
FrNm
«optional»
FrNm_SetCoordBits
«optional»
Com_SendSignal
«optional»
Det_ReportError

Figure 5.1

5.1.2 DET

For development errors, please refer to Nm025

5.1.3 BSW Scheduler

In case of the NM coordinator functionality, depending on the implementation the NM Interface module may also need a main function, and thus interface to the BSW Scheduler.

5.2 File structure

5.2.1 Code file structure

The Nm module shall provide the following C-files:

- `Nm.c` (for implementation of provided functionality)

However, no C-files will be necessary if this module is implemented as a MACRO

5.2.2 Header file structure

The Nm Interface module shall provide the following header files:

- `Nm.h` (for declaration of provided interface functions)
- `Nm_Cbk.h` (for declaration of provided call-back functions)
- `Nm_Cfg.h` (for pre-compile time configurable parameters)
- `NmStack_Types.h`

The following header files will be included within the Nm Interface module:

- `Std_Types.h` (for AUTOSAR standard types - Note: `Platform_Types.h` (for platform specific types) and `Compiler.h` (for compiler specific language extensions) are indirectly included via AUTOSAR standard types)
- `MemMap.h` (for memory abstraction)
- `SchM_Nm.h` (for interfaces with the BSW Scheduler)
- `CanNm.h` (for interface of CAN Generic specific NM; included only if CAN Generic NM is available)
- `FrNm.h` (for interface of FlexRay specific NM; included only if FlexRay NM is available)
- `ComM_Nm.h` (for Communication Manager callback functions)

- <cdd>.h For CarWakeup-functionalitiy a CDD is needed. The name of the CDD is generic.

.

6 Requirements traceability

Document: AUTOSAR Requirements on Basic Software Modules [2]

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	NA (no link time parameters)
[BSW00404] Reference to post build time configuration	NA (no post built parameter)
[BSW00405] Reference to multiple configuration sets	Nm030
[BSW00345] Pre-compile-time configuration	Nm.h and Nm_Cfg.h defined
[BSW159] Tool-based configuration	Implementation provider requirement
[BSW167] Static configuration checking	NA
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	NA
[BSW171] Configurability of optional functionality	Several switches, e.g.: NmUserDataEnabled, NmNodeDetectionEnabled, Nm055
[BSW00380] Separate C-Files for configuration parameters	NA (no link time or post built parameters)
[BSW00419] Separate C-Files for pre-compile time configuration parameters	NA (no link time or post built parameters)
[BSW00381] Separate configuration header file for pre-compile time parameters	NA (no link time or post built parameters)
[BSW00412] Separate H-File for configuration parameters	NA (no link time or post built parameters)
[BSW00383] List dependencies of configuration files	Described in chapter 5
[BSW00384] List dependencies to other modules	Described in chapter 5
[BSW00387] Specify the configuration class of callback function	Nm091
[BSW00388] Introduce containers	Chapter 10.2
[BSW00389] Containers shall have names	NA
[BSW00390] Parameter content shall be unique within the module	ok
[BSW00391] Parameter shall have unique names	ok
[BSW00392] Parameters shall have a type	ok
[BSW00393] Parameters shall have a range	ok
[BSW00394] Specify the scope of the parameters	ok
[BSW00395] List the required parameters (per parameter)	ok
[BSW00396] Configuration classes	specified (pre-compile only)
[BSW00397] Pre-compile-time parameters	ok
[BSW00398] Link-time parameters	NA
[BSW00399] Loadable Post-build time parameters	NA
[BSW00400] Selectable Post-build time parameters	NA
[BSW00402] Published information	Included (Nm008)
[BSW00375] Notification of wake-up reason	NA (no wake-up interrupt handled)
[BSW101] Initialization interface	Nm030
[BSW00416] Sequence of Initialization	Nm121
[BSW00406] Check module initialization	Errors defined in interfaces
[BSW168] Diagnostic Interface of SW components	NA
[BSW00407] Function to read out published parameters	Function included
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	ok
[BSW00424] BSW main processing function task allocation	ok
[BSW00425] Trigger conditions for schedulable objects	Main function is periodic
[BSW00426] Exclusive areas in BSW modules	NA (no exclusive areas)
[BSW00427] ISR description for BSW modules	NA (no ISRs)

[BSW00428] Execution order dependencies of main processing functions	Nm014
[BSW00429] Restricted BSW OS functionality access	ok
[BSW00431] The BSW Scheduler module implements task bodies	ok
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	NA
[BSW00433] Calling of main processing functions	ok
[BSW00434] The Schedule Module shall provide an API for exclusive areas	NA
[BSW00336] Shutdown interface	NA
[BSW00337] Classification of errors	Ok: checking for NetworkHandle validity included in the interfaces, yields development errors
[BSW00338] Detection and Reporting of development errors	Nm022, Nm023, Nm025
[BSW00369] Do not return development error codes via API	ok
[BSW00339] Reporting of production relevant error status	Nm026
[BSW00417] Reporting of Error Events by Non-Basic Software	NA
[BSW00323] API parameter checking	Checking for NetworkHandle validity included in the interfaces
[BSW004] Version check	All necessary data available. Performance of preprocessor check is an implementation provider requirement
[BSW00409] Header files for production code error IDs	Implementation issue
[BSW00385] List possible error notifications	Chapter 8.2.1
[BSW00386] Configuration for detecting an error	NA
[BSW161] Microcontroller abstraction	Ok, NM Interface is HW independent
[BSW162] ECU layout abstraction	Ok, NM Interface is HW independent
[BSW005] No hard coded horizontal interfaces within MCAL	Ok
[BSW00415] User dependent include files	NA
[BSW164] Implementation of interrupt service routines	NA
[BSW00325] Runtime of interrupt service routines	NA
[BSW00326] Transition from ISRs to OS tasks	NA
[BSW00342] Usage of source code and object code	NA, NM Interface is pre-compile only
[BSW00343] Specification and configuration of time	Ok
[BSW160] Human-readable configuration data	Ok
[BSW007] HIS MISRA C	Ok in specification, rest is an implementation provider requirement
[BSW00300] Module naming convention	Ok (Nm)
[BSW00413] Accessing instances of BSW modules	NA (no instantiation of NM Interface)
[BSW00347] Naming separation of different instances of BSW drivers	NA
[BSW00305] Self-defined data types naming convention	Ok
[BSW00307] Global variables naming convention	Ok
[BSW00310] API naming convention	Ok
[BSW00373] Main processing function naming convention	Nm020
[BSW00327] Error values naming convention	Ok
[BSW00335] Status values naming convention	Ok
[BSW00350] Development error detection keyword	Chapter 10.2
[BSW00408] Configuration parameter naming convention	Ok
[BSW00410] Compiler switches shall have defined values	Chapter 10.2
[BSW00411] Get version info keyword	Chapter 10.2

[BSW00346] Basic set of module files	Chapter 5.2.2
[BSW158] Separation of configuration from implementation	Ok
[BSW00314] Separation of interrupt frames and service routines	NA
[BSW00370] Separation of callback interface from API	Chapter 5.2.2
[BSW00348] Standard type header	Chapter 5.2.2
[BSW00353] Platform specific type header	NA
[BSW00361] Compiler specific language extension header	NA
[BSW00301] Limit imported information	Nm117
[BSW00302] Limit exported information	Ok
[BSW00328] Avoid duplication of code	Ok
[BSW00312] Shared code shall be reentrant	Ok
[BSW006] Platform independency	Ok
[BSW00357] Standard API return type	Ok
[BSW00377] Module specific API return types	Ok
[BSW00304] AUTOSAR integer data types	Ok
[BSW00355] Do not redefine AUTOSAR integer data types	Ok
[BSW00378] AUTOSAR boolean type	Ok
[BSW00306] Avoid direct use of compiler and platform specific keywords	Ok
[BSW00308] Definition of global data	NA
[BSW00309] Global data with read-only constraint	Ok
[BSW00371] Do not pass function pointers via API	Ok
[BSW00358] Return type of init() functions	Ok
[BSW00414] Parameter of init function	Ok
[BSW00376] Return type and parameters of main processing functions	Ok
[BSW00359] Return type of callback functions	Ok
[BSW00360] Parameters of callback functions	Ok
[BSW00329] Avoidance of generic interfaces	Ok
[BSW00330] Usage of macros / inline functions instead of functions	The NM Interface module (except the coordinator part) may be delivered totally as macros (Nm091)
[BSW00331] Separation of error and status values	Ok
[BSW009] Module User Documentation	Implementation provider requirement
[BSW00401] Documentation of multiple instances of configuration parameters	NA
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (no internal scheduling policy)
[BSW010] Memory resource documentation	Implementation provider requirement
[BSW00333] Documentation of callback function context	Nm028
[BSW00374] Module vendor identification	Nm008
[BSW00379] Module identification	Nm008
[BSW003] Version identification	Nm008, Nm044
[BSW00318] Format of module version numbers	Nm008
[BSW00321] Enumeration of module version numbers	Implementation provider requirement
[BSW00341] Microcontroller compatibility documentation	NA
[BSW00334] Provision of XML file	Ok

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

Requirement	***Satisfied by***
[BSW150] Configuration of functionality	Ok, e.g. Nm055
[BSW151] Integration into running NM cluster	Nm031

[BSW043] Bus Traffic without NM Initialization	NA
[BSW044] Applicability to different types of communication systems	Currently limited to Can and Fr, but not by NM Interface
[BSW045] NM-cluster Independent Shutdown Coordination	via coordinator functionality (chapter 7) (Nm003, Nm007, Nm033)
[BSW046] Trigger of startup of all Nodes at any Point in Time	Ok: init function can be called any time, bus specific strategies not part of NM Interface
[BSW047] Bus Keep Awake Services	Nm032, Nm034
[BSW048] Bus Sleep Mode	Nm046
[BSW050] NM State Information	Nm043
[BSW051] NM State Change Indication	Ok, see ComM interfaces
[BSW052] Notification that all other ECUs are ready to sleep	NA (coordinator functionality merged with interface functionality, therefore handled internally)
[BSW02509] Notification that at least one other node is not ready to sleep anymore	Nm_NetworkStartIndication
[BSW02503] Sending user data	Nm035
[BSW02504] Receiving user data	Nm036
[BSW153] Detection of present nodes	Nm038
[BSW02508] Unambiguous node identification per bus	Nm040
[BSW02505] Sending node identifier	Nm039
[BSW02506] Receiving node identifier	Nm037
[BSW02511] Configurable Role in Cluster Shutdown	NA (bus specific part)
[BSW053] Deterministic Behavior in Case of Bus Unavailability	NA (bus specific part)
[BSW137] Communication system error handling	NA (bus specific part)
[BSW136] Coordination of coupled networks	Nm001, Nm002, Nm003, Nm004, Nm042; Nm112, Nm114, Nm115
[BSW140] Compliance with OSEK NM on a gateway	Nm002, Nm003, Nm004, Nm049, Nm053, Nm056, Nm115, Nm116
[BSW054] Deterministic Time for Bus Sleep	NA (bus specific part)
[BSW142] Limitation of NM bus load	NA (bus specific part)
[BSW143] Predictable NM bus load	NA (bus specific part)
[BSW144] ECU cluster size	NA (bus specific part)
[BSW145] Robustness against NM message losses	NA (bus specific part)
[BSW146] Robustness against NM message jitter	NA (bus specific part)
[BSW147] Processor independent algorithm	Nm095
[BSW149] Configurable Timing	Nm097
[BSW154] Bus independency of API	Chapter 8.3 (no bus specific parameter), Nm095, Nm006, Nm012
[BSW148] Separation of Communication system dependent parts	Separated into CanNm and FrNm
[BSW139] Compliance with OSEK NM on one cluster	Nm053, Nm056 (direct OSEK NM only)
[BSW02510] Immediate Transmission Confirmation	
[BSW02512] CommunicationControl (28 hex) service support	Nm028, Nm034

7 Functional specification

The NM Interface functionality consists of two parts:

- The base functionality necessary to run, together with the bus specific NM modules, AUTOSAR NM on an ECU
- The NM coordinator functionality used by gateway ECUs to synchronously shut down more than one bus

7.1 Base functionality

The AUTOSAR NM Interface module shall act as a bus-independent adaptation layer between the bus-specific Network Management modules (CanNm, and FrNm) and the AUTOSAR basic software module Communication Manager.

Note: No specific knowledge of the type of the underlying busses will be present in the NM Interface algorithm and module API.

Nm006 The NM Interface module shall convert generic function calls to bus specific functions of the bus specific NM layer.

Nm012 The NM Interface module shall convert callback functions of the bus specific NM layer to generic callbacks to the Communication Manager.

Nm091 It shall be allowed to implement the base functionality completely or partly using macros.

7.2 NM coordinator functionality

A NM coordinator is an ECU, which is connected to at least two busses, and where the requirement exists that shutdown of NM of at least two of these busses (also referred to as ‘coordinated busses’) has to be performed synchronously. These busses are referred to as ‘coordinated busses’, the algorithm to shutdown synchronously is referred to as ‘coordinator algorithm’.

If the NM Coordinator functionality is configured, the configuration parameter NmCycletimeMainFunction should be configured with the cycle time of the rate at which two successive calls to the Nm's main function (ref Nm118) are made. The NM Coordinator may use this to calculate the timeout status of internal timers.

Nm001 The coordinator algorithm shall be able to handle a topology where several coordinated busses are connected to a NM coordinator.
Nm051 On each of the coordinated busses it shall be possible to run AUTOSAR NM.
Nm055 It shall be configurable if the coordinator algorithm to support AUTOSAR NM is present.
Nm053 On each of the coordinated busses it shall be possible to run direct OSEK NM.

Nm056 It shall be configurable if the coordinator algorithm to support OSEK NM is present in addition to the coordinator algorithm for AUTOSAR NM (Nm055).

Nm002 As long as AUTOSAR NM or direct OSEK NM on a NM coordinator node is not ready to sleep on at least one of the coordinated busses (this means the node itself is not ready to sleep on all coordinated busses) the NM coordinator shall act on all coordinated busses as if it is not ready to sleep.

Because of this requirement, the normal AUTOSAR NM or direct OSEK NM algorithm is performed. As in both algorithms, a bus is kept awake if at least one node needs the bus, the NM coordinator will automatically keep all coordinated busses awake.

Nm152: The NM Coordinator shall only coordinate channels where NmSelectiveNmChannel is set to FALSE. If NmSelectiveNmChannel is set to TRUE the NM coordinator shall ignore the corresponding channel for the coordination algorithm

Nm003 If a NM coordinator is ready to sleep on all coordinated busses, but at least one of the coordinated busses is kept awake by a node that is not the coordinator node, the NM coordinator shall act on all coordinated busses as if it is not ready to sleep.
This means that in this case the coordinator node shall still act as if it is not ready to sleep on the coordinated busses.

Rational: The bus specific NMs will indicate to Nm if the bus is ready to go to sleep or not by calling the callbacks Nm_RemoteSleepIndication() and Nm_RemoteSleepCancellation(). The local ECU will indicate if it is ready to go to sleep or not on a network using the API functions Nm_NetworkRelease() and Nm_NetworkRequest().

The two requirements above can be summarized as follows: as long as at least one node keeps any of the coordinated busses awake, the NM coordinator shall keep all coordinated busses awake.

The algorithm to detect on AUTOSAR CAN NM if some other node on a bus is not ready to sleep depends on a specific property of the AUTOSAR NM algorithm. The timeout for nodes that have sent a NM message is defined such that a second node will send in-between before the first node will send another NM message.
If the NM coordinator keeps the bus awake, there are two possibilities:

- At least one other node on the same bus needs the bus. Because of the algorithm, one of these nodes will send in-between the NM messages of the NM coordinator
- No other node needs the bus. Because of the algorithm, the NM coordinator will be the only node to send NM messages on the bus

To detect the situation that no other node needs the bus – if the coordinator itself does not need the bus any more -, the RemoteSleepIndication functionality of the underlying layers is used.
For details of the algorithm, please refer to the Can NM SWS [4].

For AUTOSAR FlexRay NM, if no NM-messages with an indication to keep the bus awake are received in the Normal Operation State for a configurable amount of time AUTOSAR FlexRay NM notifies the NM Interface that all other nodes in the cluster are ready to sleep.

OSEK NM uses a token bus like algorithm: a specific message is passed from node to node. In this message, the sending node informs the other nodes if it is ready to sleep. Thus, if all nodes except the NM coordinator inform that they are ready to sleep (via RemoteSleepIndication, see Nm049), the NM coordinator can assume that this bus is ready to go to sleep. OSEK NM offers as extension a service to collect the information if all other nodes on a specific bus are ready to sleep.

Note: The Nm_RemoteSleepIndication() and Nm_RemoteSleepCancellation() are used by the coordination algorithm to determine when all conditions for initiating the coordinated shutdown are met. The indication will be called by the bus specific NM (CanNm or FrNm) when they detects that all other nodes on the network (except for itself) are ready to go to 'bus-sleep mode'. Some implementations may make use of the API call Nm_CheckRemoteSleepIndication().

If on all coordinated busses the only sending node is the NM coordinator, and the NM coordinator itself needs none of the coordinated busses any more, the NM coordinator can put all coordinated busses to sleep. To do this, it needs to send final NM messages on the busses. These final messages have to be timed according to the individual timing of the coordinated busses such that the busses are put to sleep synchronously.

Nm004 If the NM coordinator detects that for all coordinated busses all of the following conditions apply:

1. on all coordinated busses AUTOSAR NM is run, for a configurable number of NM message transmissions[2] of the NM coordinator (see Nm097), there has been no AUTOSAR NM message except the AUTOSAR NM messages sent by the NM coordinator
2. on all coordinated busses OSEK NM is run, for a configurable number of NM message transmissions of the NM coordinator (see Nm099), all other nodes have only sent NM messages which state that the node is ready to go to sleep (using RemoteSleepIndication, see Nm049)
3. the NM coordinator is ready to sleep

the NM coordinator shall request NM message transmission on all coordinated busses such that the a synchronous shutdown is achieved. (For limits, see comment further below.)

Nm115 For the algorithm described in Nm004, the coordinator shall ignore coordinated busses which have not successfully performed at all, or after the last

[2] The configurable number of rounds delays the detection that all busses are ready to go to sleep. This is useful if there is a high probability that within a certain amount of time one of the busses is needed again. In addition, if a NM message has been lost, this number must be set such that it is certain that the bus is not needed any more.

successful Nm_NetworkRelease (or the equivalent OSEK NM functionality), a successful Nm_NetworkRequest (or the equivalent OSEK NM functionality).

Please note that there are limits to how much synchrony can be achieved when shutting down busses. Whereas CAN NM can in its timing be influenced by requesting NM message transmissions, FlexRay has a fixes time value for rounds, and OSEK NM has indirectly a fixed time value for rounds via the number of nodes. Therefore, it is not possible to meet an arbitrary point in time, and synchrony is only possible within certain tolerances. This is also partly determined by configuration values (e.g. the time a FlexRay round needs).[3]

Synchronous wake up is not required and not included in the concept, because the COM Manager will wake up all coordinated busses when one if the coordinated busses informs the COM Manager about a wakeup event. However, there are races possible if the coordinated shutdown is performed and a bus comes up again before all busses have entered sleep mode.

Nm116: If NmIf has started to shut down the coordinated busses AND not all coordinated busses have signaled bus sleep state AND on at least on one of the coordinated busses NM is restarted, THEN the NM Interface shall call the callback function ComM_Nm_RestartIndication with the nmNetworkHandle of the waking channel as parameter AND the appropriate function in OSEK NM for all of the coordinated busses which run OSEK NM.

7.2.1 Coordinator Synchronization

To support the coordination of nested sub-busses, the Nm-Coordinators, which are linked together, have to be assigned different NM-Coordinator IDs to mark them as active or passive Nm-Coordinator on the respective sub-bus. Defining the active Nm-Coordinator by NMcoordinatorID allows a topology of 7 Nm-Coordinators, assuming that there is always one active Nm-Coordinator on one Network.

[3] A more precise algorithm is possible if information is available about when FlexRay rounds begin. As information about some state changes is communicated on FlexRay round boundaries between the Communication Manager and FlexRay NM, this information can be taken into account.

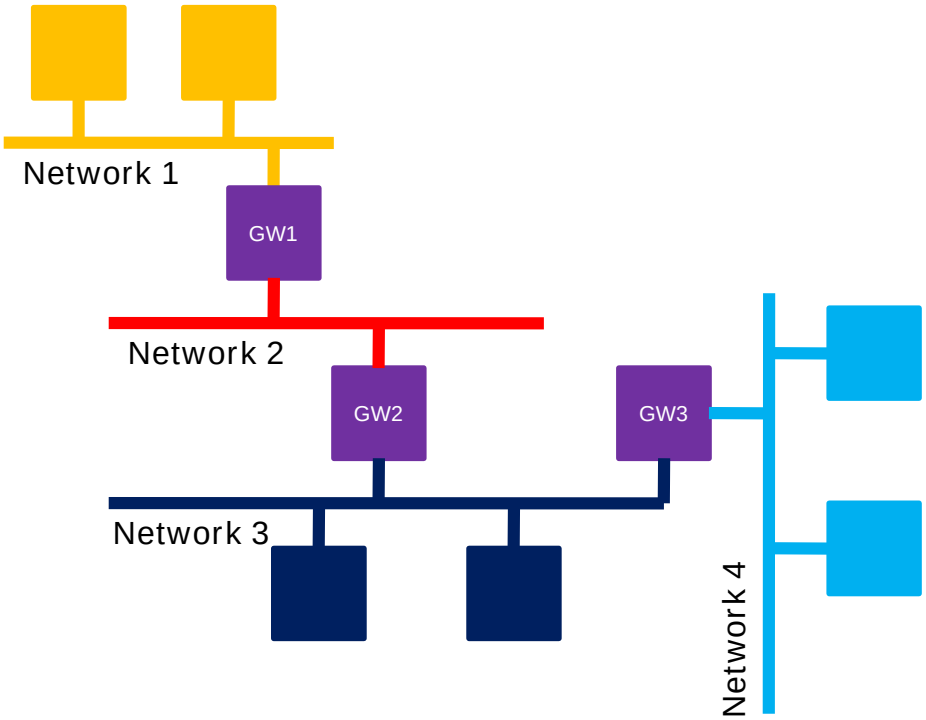

Figure 7.1**: Use Case Nested Gateways**

The topology shown in Figure 7.1 may have following coordination approach. GW 1 can be active Nm-Coordinator on Network 1 and 2 using NMcoordinatorID 3, where GW 2 is passive Nm-Coordinator on Network 2 but active Nm-Coordinator on Network 3 using the NMcoordinatorID 2. GW 3 then needs to be passive Nm-Coordinator on Network 3 but active Nm-Coordinator on Network43 using the NMcoordinatorID 1. This Topology is only an example of one possible topology shut down coordinated by **Nm** implemented after following requirements.

Nm129: The NM-Coordinator shall support two or more gateways connected to the same NM Cluster.

Nm140: There is always only one active NM-Coordinator per NM Channel.
Note:*This is realized by setting the priority of the NM coordinator to the according value. On a cascaded network the passive NM coordinator has a lower coordinator ID, which is again the highest on the nested sub-bus.*

Nm130: The NM Coordinator shall use <BusNm>_SetCoordBits to set 2 additional bits in the NM message (Bit 1 & Bit 2) stating that it is an actively coordinating gateway.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Res	Res	Res	Res	NM Coordinator Sleep Ready	NMcoodinator ID(High Bit)	NMcoodinator ID(Low Bit)	RptMsgRequest

Table7.1: Layout of Control Bit Vector (CBV)

Nm131: As long as the application(s) on the gateway are not ready to sleep, it shall act like a normal node.

Nm132: If the application(s) is(are) ready to sleep, but at least one of the coordinated busses is kept awake by a node that is not the coordinator node, the NM Coordinator shall still act as if it is not ready to sleep. In this case the NM Coordinator sets a specific gateway coordination ID.
Rationale: *To indicate that it does only coordinate the busses.*

Nm133: The NMcoordinatorID shall be configurable at pre-compile time and represent the priority of the NM Coordinator.

Nm134: An NM Coordinator with ID=0x03 shall have the highest priority. An NM Coordinator with ID=0x01 shall have the lowest priority.

Nm135: The NMcoordinationID shall be set to 0x00 if the NM Coordinator receives a NM message with a higher NMcoordinatorID than the own one via Nm_ActiveCoordIndication.
Rational*: This ensures the possibility to detect if a node is an actively coordinating gateway coordinator.*

Nm136:The passive coordinator(s) (NMcoordinationID=0x00) send Nm messages only if the node has a network management request pending or a connected network which is coordinated active by that coordinator is not ready to sleep.
Rationale*: This prevents that both, gateway and back-up gateway, send NM messages within CANNM_MSG_CYCLE_TIME when they are ready to sleep. Without this mechanism it would not be possible to detect if there is at least one other node active.*

Nm137: If the NM coordinator with the highest priority fails, the NM coordinator with the next highest priority takes over and sets its own NM Coordinator ID.

Nm141: The active NM Coordinator shall set the *NMcoordinatorSleepReady* bit in the NM message via `<BusNm>_SetSleepReadyBit(Nm_Channel, value)` to the value 1, if all nodes of the NM cluster are ready to sleep.
Note*: for Position of NMcoordinatorSleepReady bit in CBV see Table 7.1.*
Nm142: A passive NM Coordinator must not set the NMcoordinatorSleepReady bit ever on its passively coordinated NM channel but it has to forward a received bit on its actively coordinated channels.

Nm143: A passive NM Coordinator gets Indication of the Sleep Ready Bit set by `Nm_CoordReadyToSleepIndication` called by the <Bus>Nm.

Nm302: A passive NM Coordinator shall abort shutdown when `Nm_CoordReadyToSleepCancellation` is called on its passive channel.

Note: Nm_CoordReadyToSleepCancelation() provide the indication that the Sleep Ready Bit has been unset.

Nm144: The active and passive NM Coordinators shall start a coordination delay timer after the active NM Coordinator has set the NMcoordinatorSleepReady bit in the CBV.

Nm145: The *active and passive NM* Coordinators shall start the *shutdown network* sequence, once the coordination delay timer has expired.
***Note**: The coordination delay timer defines the time left before the NM cluster starts to shut down its network thus the shutdown network sequence of the nested bus shall be finished before the active NM Coordinator has shut down the Network to ensure a synchronous shutdown. See Figure 2: Shutdown with NmGlobalCoordinatorTime*

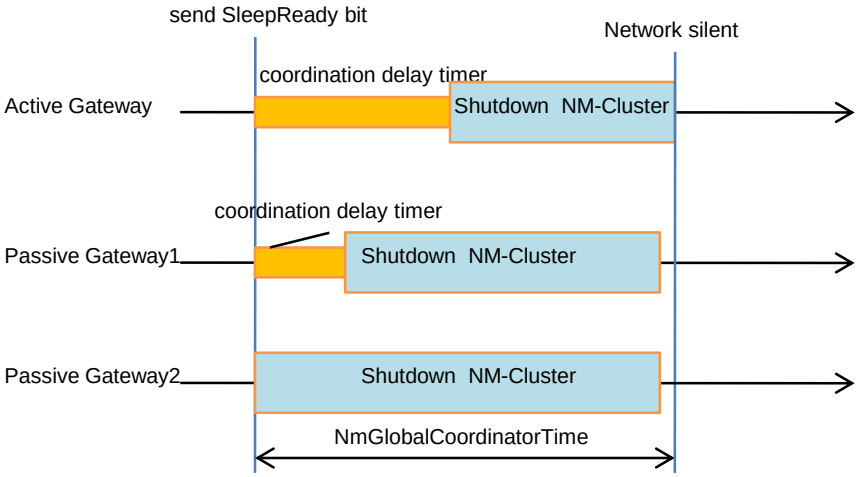

Figure 7.2: **Shutdown with NmGlobalCoordinatorTime**

Nm146: The value of the coordination delay timer is calculated from NmGlobalCoordinatorTime.
***Note:** The coordination delay timer shall be calculated following:*
NmGlobalCoordinatorTime - "channel-specific Nm shutdown time"
for CanNm:
Ready Sleep Time + Prepare Bus Sleep Time;
for FrNm:
*Ready Sleep Time, i.e. (FrNmReadySleepCnt+1) * FrNmRepetitionCycle * 'Duration of one FlexRay Cycle';*
***Note:** For OSEK_NM the coordinator delay timer needs to be calculated on runtime due to dynamic shutdown time of the nm_channel.*

Nm147: NmGlobalCoordinatorTime defines the maximum time needed to shut down all Networks coordinated.
***Note:**This includes all nested connections.(for example see Figure 7.2)*

Nm148: The NM Coordinator shall reset the *coordination delay timer*, if the network shutdown is aborted by any reason, e.g. an application or another network node requesting the network.

Nm149: If the network shutdown has been aborted for any reason, the NM Coordinator shall set the Sleep Ready bit back to the value 0 via `<BusNm>_SetSleepReadyBit(Nm_Channel, value)` on every actively coordinated channel.

7.3 Additional Functionality

7.3.1 Nm_CarWakeUpIndication

Nm126:
If the <bus>Nm calls Nm_CarWakeUpIndication, the NM Interface shall call the callback function defined by NmCarWakeUpCallback with nmNetworkHandle as parameter.

Note: The application, called by NmCarWakeUpCallback, is responsible to manage the Car Wake Up (CWU) request and distribute the Request to other Nm channels by setting the CWU bit in its own Nm message. This application has to drop the CWU request if the request is not repeated within a specific time.

7.3.2 Nm_StateChangeNotification

Nm124: When `NmStateReportEnabled` is set to `TRUE`, Nm_StateChangeNotification shall call Com_SendSignal(…) with `NmStateReportSignalRef` as `Com_signalIdType`. `NmStateReportSignalRef` points to a 6 bit signal, called Network Management State (NMS). The NMS needs to be configuered in Com. The NMS shall be set to the value according to the following table:

Bit	*Value*	*Name*	*Description*
0	1	NM_RM_BSM	NM in state RepeatMessage (transition from BusSleepMode)
1	2	NM_RM_PBSM	NM in state RepeatMessage (transition from PrepareBusSleepMode)
2	4	NM_NO_RM	NM in state NormalOperation (transition from RepeatMessage)
3	8	NM_NO_RS	NM in state NormalOperation (transition from ReadySleep)
4	16	NM_RM_RS	NM in state RepeatMessage (transition from ReadySleep)
5	32	NM_RM_NO	NM in state RepeatMessage (transition from NormalOperation)

7.4 Error classification

Nm301: The Nm shall be able to detect the following errors and exceptions depending on its configuration (development/production):

Type of error	*Relevance*	*Related error code*	*Value [hex]*

API service used without Nm interface initialization	Development	NM_E_UNINIT	0x00
API Service called with wrong parameter but not with NULL-pointer	Development	NM_E_HANDLE_UNDEF	0x01
API service called with a NULL pointer	Development	NM_E_PARAM_POINTER	0x02

7.5 Error detection

Nm022 The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time.
The switch `NmDevErrorDetect` shall activate or deactivate the detection of all development errors.

Nm023 If the `NmDevErrorDetect` switch is enabled, API parameter checking shall be enabled.

7.6 Error notification

Nm025 Detected development errors shall be reported to the `Det_ReportError` service of the Development Error Tracer (DET) if the pre-processor switch `NmDevErrorDetect` is set, except if the respective service is implemented as macro (Nm091).

8 API specification

8.1 Imported types

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

Nm117:

Module	*Imported Type*
Com	Com_SignalIdType
ComStack_Types	NetworkHandleType
Std_Types	Std_VersionInfoType

8.2 Type definitions

8.2.1 Nm_ReturnType

Name:	Nm_ReturnType	
Type:	Enumeration	
Range:	NM_E_OK	Function call has been successfully accomplished and returned.
	NM_E_NOT_OK	Function call has been unsuccessfully accomplished and returned because of an internal execution error.
	NM_E_NOT_EXECUTED	Function call is not executed.
Description:	Return type for NM functions. Derived from Std_ReturnType.	

8.2.2 Nm_ModeType

Name:	Nm_ModeType	
Type:	Enumeration	
Range:	NM_MODE_BUS_SLEEP	Bus-Sleep Mode
	NM_MODE_PREPARE_BUS_SLEEP	Prepare-Bus Sleep Mode
	NM_MODE_SYNCHRONIZE	Synchronize Mode
	NM_MODE_NETWORK	Network Mode
Description:	Operational modes of the network management.	

8.2.3 Nm_StateType

Name:	Nm_StateType	
Type:	Enumeration	
Range:	NM_STATE_UNINIT	Uninitialized State (0)
	NM_STATE_BUS_SLEEP	Bus-Sleep State (1)
	NM_STATE_PREPARE_BUS_SLEEP	Prepare-Bus State (2)
	NM_STATE_READY_SLEEP	Ready Sleep State (3)
	NM_STATE_NORMAL_OPERATION	Normal Operation State (4)
	NM_STATE_REPEAT_MESSAGE	Repeat Message State (5)
	NM_STATE_SYNCHRONIZE	Synchronize State (6)
Description:	States of the network management state machine.	

8.2.4 Nm_BusNmType

Name:	`Nm_BusNmType`	
Type:	`Enumeration`	
Range:	`NM_BUSNM_CANNM`	CAN NM type
	`NM_BUSNM_FRNM`	FR NM type
	`NM_BUSNM_LINNM`	LIN NM type
	`NM_BUSNM_UNDEF`	NM type undefined; it shall be defined as FFh
Description:	BusNm Type	

8.2.5 Nm_ConfigType

Name:	`Nm_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
Description:	This type contains the initialization data for the NmIf module.	

8.3 Function definitions

8.3.1 Services provided by NM Interface

8.3.1.1 Nm_Init

Nm030:

Service name:	Nm_Init	
Syntax:	`void Nm_Init( Nm_ConfigType* const nmConfigPtr )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	nmConfigPtr	Pointer to the selected configuration set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the NM Interface.	

Caveats of Nm_Init: This service function has to be called after the initialization of the respective bus interface.

Configuration of Nm_Init: Mandatory

8.3.1.2 Nm_PassiveStartUp

Nm031:

Service name:	Nm_PassiveStartUp	
Syntax:	`Nm_ReturnType Nm_PassiveStartUp( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x01	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Passive start of network management has failed NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Passive start of network management has not executed
Description:	This function calls the <BusNm>_PassiveStartUp function (e.g. CanNm_PassiveStartUp function is called if channel is configured as CAN).	

Caveats of Nm_PassiveStartUp: CanNm and FrNm are initialized correctly.

Configuration of Nm_PassiveStartUp: Mandatory

8.3.1.3 Nm_NetworkRequest

Nm032:

Service name:	Nm_NetworkRequest	
Syntax:	`Nm_ReturnType Nm_NetworkRequest( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x02	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Requesting of bus communication has failed NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Function to request bus communication not executed
Description:	This function calls the <BusNm>_NetworkRequest (e.g. CanNm_NetworkRequest function is called if channel is configured as CAN).	

Caveats of Nm_NetworkRequest: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_NetworkRequest: Optional (Only available if `NmPassiveModeEnabled` is set to OFF)

8.3.1.4 Nm_NetworkRelease

Nm046:

Service name:	Nm_NetworkRelease	
Syntax:	`Nm_ReturnType Nm_NetworkRelease( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Releasing of bus communication has failed NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Function to release bus communication not executed
Description:	This function calls the <BusNm>_NetworkRelease bus specific function (e.g. CanNm_NetworkRelease function is called if channel is configured as CAN).	

Caveats of Nm_NetworkRelease: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_NetworkRelease: Optional (Only available if `NmPassiveModeEnabled` is set to OFF)

8.3.1.5 Nm_DisableCommunication

Nm033:

Service name:	Nm_DisableCommunication	
Syntax:	`Nm_ReturnType Nm_DisableCommunication( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x04	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Disabling of NM PDU transmission ability has failed.

		NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Disabling of NM PDU transmission ability is not executed.
Description:	This function calls the CanNm_DisableCommunication function, to disable the NM PDU transmission.	

Caveats of Nm_DisableCommunication: CanNm are initialized correctly

Configuration of Nm_DisableCommunication: Optional (Only available if `NmComControlEnabled` is defined)

8.3.1.6 Nm_EnableCommunication

Nm034:

Service name:	Nm_EnableCommunication	

Syntax:	`Nm_ReturnType Nm_EnableCommunication(`
	`  const NetworkHandleType NetworkHandle`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Enabling of NM PDU transmission ability has failed. NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Enabling of NM PDU transmission ability is not executed.
Description:	This function calls the CanNm_EnableCommunication function, to enable the NM PDU transmission.	

Caveats of Nm_EnableCommunication: CanNm is initialized correctly.

Configuration of Nm_EnableCommunication: Optional (Only available if `NmComControlEnabled` is defined)

8.3.1.7 Nm_SetUserData

Nm035:

Service name:	Nm_SetUserData

Syntax:	`Nm_ReturnType Nm_SetUserData(`
	`  const NetworkHandleType NetworkHandle,`
	`  const uint8 * const nmUserDataPtr`
	`)`

Service ID[hex]:	0x06

Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
	nmUserDataPtr	User data for the next transmitted NM message
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Setting of user data has failed NetworkHandle does not exist (development only) module not yet initialized (development only)
Description:	Set user data for NM messages transmitted next on the bus. For that purpose <BusNm>_SetUserData shall be called (e.g. CanNm_SetUserData function is called if channel is configured as CAN).	

Caveats of Nm_SetUserData: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_SetUserData: Optional (Only available if `NmUserDataEnabled` and `NmPassiveModeEnabled` is defined)

Configuration of `Nm_SetUserData`: If `NmComUserDataSupport` is `True` the API `Nm_SetUserData` shall not be available.

8.3.1.8 Nm_GetUserData

Nm036:

Service name:	Nm_GetUserData	

Syntax:	`Nm_ReturnType Nm_GetUserData(`
	`  const NetworkHandleType NetworkHandle,`
	`  uint8 * const nmUserDataPtr,`
	`  uint8 * const nmNodeIdPtr`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmUserDataPtr	Pointer where user data out of the last successfully received NM message shall be copied to
	nmNodeIdPtr	Pointer where node identifier out of the last successfully received NM message shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of user data has failed NetworkHandle does not exist (development only) module not yet initialized (development only)
Description:	Get user data out of the last successfully received NM message. For that purpose <BusNm>_GetUserData shall be called (e.g. CanNm_GetUserData function is called if channel is configured as CAN).	

Caveats of Nm_GetUserData: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_GetUserData: Optional (Only available if `NmUserDataEnabled` is set to ON)

8.3.1.9 Nm_GetPduData

Nm037:

Service name:	Nm_GetPduData	
Syntax:	`Nm_ReturnType Nm_GetPduData( const NetworkHandleType NetworkHandle, uint8 * const nmPduData )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmPduData	Pointer where NM PDU shall be copied to.
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of NM PDU data has failed NetworkHandle does not exist (development only) module not yet initialized (development only)
Description:	Get the whole PDU data out of the most recently received NM message. For that purpose CanNm_GetPduData shall be called.	

Caveats of Nm_GetPduData: The <BusNm> and <Nm> are initialized correctly

Configuration of Nm_GetPduData: Optional (Only available if `NmNodeIdEnabled or NmNodeDetectionEnabled` or `NmUserDataEnabled` is set to

8.3.1.10 Nm_RepeatMessageRequest

Nm038:

Service name:	Nm_RepeatMessageRequest	
Syntax:	`Nm_ReturnType Nm_RepeatMessageRequest( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x09	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Setting of Repeat Message Request Bit has failed NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Repeat Message Request is

		currently not executed.
Description:	Set Repeat Message Request Bit for NM messages transmitted next on the bus. For that purpose <BusNm>_RepeatMessageRequest shall be called (e.g. CanNm_RepeatMessageRequest function is called if channel is configured as CAN)	

Caveats of Nm_RepeatMessageRequest: FrNm and CanNm itself are initialized correctly.

Configuration of Nm_RepeatMessageRequest: Optional (Only available if `NmNodeDetectionEnabled` is defined)

8.3.1.11 Nm_GetNodeIdentifier

Nm039:

Service name:	Nm_GetNodeIdentifier	
Syntax:	`Nm_ReturnType Nm_GetNodeIdentifier( const NetworkHandleType NetworkHandle, uint8 * const nmNodeIdPtr )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer where node identifier out of the last successfully received NM-message shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of the node identifier out of the last received NM-message has failed NetworkHandle does not exist (development only) module not yet initialized (development only)
Description:	Get node identifier out of the last successfully received NM-message. The function <BusNm>_GetNodeIdentifier shall be called.	

Caveats of Nm_GetNodeIdentifier: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_GetNodeIdentifier: Optional (Only available if `NmNodeIdEnabled` is set to `TRUE`)

8.3.1.12 Nm_GetLocalNodeIdentifier

Nm040:

Service name:	Nm_GetLocalNodeIdentifier
Syntax:	`Nm_ReturnType Nm_GetLocalNodeIdentifier( const NetworkHandleType NetworkHandle, uint8 * const nmNodeIdPtr )`

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer where node identifier of the local node shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of the node identifier of the local node has failed NetworkHandle does not exist (development only) module not yet initialized (development only)
Description:	Get node identifier configured for the local node. For that purpose <BusNm>_GetLocalNodeIdentifier shall be called (e.g. CanNm_GetLocalNodeIdentifier function is called if channel is configured as CAN).	

Caveats of Nm_GetLocalNodeIdentifier: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_GetLocalNodeIdentifier: Optional (Only available if `NmNodeIdEnabled` is set to `TRUE`)

8.3.1.13 Nm_CheckRemoteSleepIndication

Nm042:

Service name:	Nm_CheckRemoteSleepIndication	
Syntax:	`Nm_ReturnType Nm_CheckRemoteSleepIndication( const NetworkHandleType nmNetworkHandle, boolean * const nmRemoteSleepIndPtr )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmRemoteSleepIndPtr	Pointer where check result of remote sleep indication shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Checking of remote sleep indication bits has failed NetworkHandle does not exist (development only) module not yet initialized (development only) NM_E_NOT_EXECUTED: Checking of remote sleep indication bits has not executed
Description:	Check if remote sleep indication takes place or not. This in turn calls the <BusNm>_CheckRemoteSleepIndication for the bus specific NM layer (e.g. CanNm_CheckRemoteSleepIndication function is called if channel is configured as CAN).	

Caveats of Nm_CheckRemoteSleepIndication: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_CheckRemoteSleepIndication: Optional (Only available if `NM_REMOTE_SLEEP_INDICATION_ENABLED` is set to `ON`)

8.3.1.14 Nm_GetState

Nm043:

Service name:	Nm_GetState	
Syntax:	`Nm_ReturnType Nm_GetState( const NetworkHandleType nmNetworkHandle, Nm_StateType* const nmStatePtr, Nm_ModeType* const nmModePtr )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmStatePtr	Pointer where state of the network management shall be copied to
	nmModePtr	Pointer to the location where the mode of the network management shall be copied to
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of NM state has failed NetworkHandle does not exist (development only) module not yet initialized (development only)
Description:	Returns the state of the network management. This function in turn calls the <BusNm>_GetState function (e.g. CanNm_GetState function is called if channel is configured as CAN).	

Caveats of Nm_GetState: The <BusNm> and the Nm itself are initialized correctly.

Configuration of Nm_GetState: Mandatory

8.3.1.15 Nm_GetVersionInfo

Nm044:

Service name:	Nm_GetVersionInfo	
Syntax:	`void Nm_GetVersionInfo( Std_VersionInfoType* nmVerInfoPtr )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	nmVerInfoPtr	Pointer to where to store the version information of this

		module.
Return value:	None	
Description:	This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	

Configuration of Nm_ GetVersionInfo: Optional (only available if `NmVersionInfoApi` is set to `ON`)

8.4 Call-back notifications

Nm028: All callbacks shall assume that they can run either in task or in interrupt context.

8.4.1 Nm_NetworkStartIndication

Nm104:

Service name:	Nm_NetworkStartIndication	

Syntax:	`void Nm_NetworkStartIndication(`
	`  const NetworkHandleType nmNetworkHandle`
	`)`	

Service ID[hex]:	0x11	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that a NM-message has been received in the Bus-Sleep Mode, what indicates that some nodes in the network have already entered the Network Mode. The callback function shall start the network management state machine.	

Configuration of Nm_NetworkStartIndication: Mandatory

8.4.2 Nm_NetworkMode

Nm105:

Service name:	Nm_NetworkMode

Syntax:	`void Nm_NetworkMode(`
	`  const NetworkHandleType nmNetworkHandle`
	`)`

Service ID[hex]:	0x12	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has entered Network Mode. The callback function shall enable transmission of application messages.	

Configuration of Nm_NetworkMode: Mandatory

8.4.3 Nm_PrepareBusSleepMode

Nm106:

Service name:	Nm_PrepareBusSleepMode	
Syntax:	`void Nm_PrepareBusSleepMode( const NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x13	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has entered Prepare Bus-Sleep Mode. The callback function shall disable transmission of application messages.	

Configuration of Nm_PrepareBusSleepMode: Mandatory

8.4.4 Nm_BusSleepMode

Nm107:

Service name:	Nm_BusSleepMode	
Syntax:	`void Nm_BusSleepMode( const NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x14	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has entered Bus-Sleep Mode.	

Configuration of Nm_BusSleepMode: Mandatory

8.4.5 Nm_PduRxIndication

Nm112:

Service name:	Nm_PduRxIndication	
Syntax:	`void Nm_PduRxIndication( const NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x15	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that a NM message has been received.	

Configuration of Nm_PduRxIndication: Optional (only available if **`NmPduRxIndicationEnabled`** is set to `TRUE`).

The notification that an NM message has been received is only needed for OEM specific extensions of the *Nm*.

8.4.6 Nm_StateChangeNotification

Nm114:

Service name:	Nm_StateChangeNotification	
Syntax:	`void Nm_StateChangeNotification( const NetworkHandleType nmNetworkHandle, const Nm_StateType nmPreviousState, const Nm_StateType nmCurrentState )`	
Service ID[hex]:	0x16	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
	nmPreviousState	Previous state of the NM-channel
	nmCurrentState	Current (new) state of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the state of the lower layer <BusNm> has changed.	

Nm123:
Configuration of Nm_StateChangeNotification: Optional (only available if `NmStateChangeIndEnabled` is set to `TRUE`)

8.4.7 Nm_CarWakeUpIndication

Nm125:

Service name:	Nm_CarWakeUpIndication	

Syntax:	`void Nm_CarWakeUpIndication(`
	`  const NetworkHandleType nmNetworkHandle`
	`)`	

Service ID[hex]:	0x1d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by a <Bus>Nm to indicate reception of a CWU request.	

Nm128:
Configuration of Nm_CarWakeUpIndication: Optional
If `NmCarWakeUpRxEnabled` is `TRUE`, The Nm shall provide the API Nm_CarWakeUpIndication ().

8.4.8 Nm_ActiveCoordIndication

Nm138:

Service name:	Nm_ActiveCoordIndication	

Syntax:	`void Nm_ActiveCoordIndication(`
	`  const NetworkHandleType nmNetworkHandle,`
	`  const uint8 nmCoordPrio`
	`)`	

Service ID[hex]:	0x1c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
	nmCoordPrio	Priority of the NM coordinator
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Set the two bits in the Control Bit Vector which represent the NM coordinator ID.	

Nm139::
Configuration of Nm_ActiveCoordIndication : Optional (only available if `NmCoordinatorSyncSupport` is set to `TRUE`)

8.4.9 Nm_CoordReadyToSleepIndication

Nm150:

Service name:	Nm_CoordReadyToSleepIndication
Syntax:	`void Nm_CoordReadyToSleepIndication(`

	`const NetworkHandleType nmChannelHandle` `)`
Service ID[hex]:	0x1e
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	nmChannelHandle — Identification of the NM-channel
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set

Nm151: Configuration of Nm_CoordReadyToSleepIndication: Optional (only available if `NmCoordinatorSyncSupport` is set to `TRUE`)

8.4.10 Nm_CoordReadyToSleepCancelation

Nm303:

Service name:	Nm_CoordReadyToSleepCancellation

Syntax:	`void Nm_CoordReadyToSleepCancellation(`
	`const NetworkHandleType nmChannelHandle`
	`)`

Service ID[hex]:	0x1f
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	nmChannelHandle — Identification of the NM-channel
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0.

Nm304: Configuration of Nm_CoordReadyToSleepCancelation: Optional (only available if `NmCoordinatorSyncSupport` is set to `TRUE`).

8.4.11 Nm_RemoteSleepIndication

Nm305:

Service name:	Nm_RemoteSleepIndication

Syntax:	`void Nm_RemoteSleepIndication(`
	`const NetworkHandleType channel`
	`)`

Service ID[hex]:	0x17

Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	channel	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has detected that all other nodes are ready to sleep. The NM gateway shall check if the Bus is still required.	

8.4.12 Nm_RemoteSleepCancellation

Nm306:

Service name:	Nm_RemoteSleepCancellation	
Syntax:	`void Nm_RemoteSleepCancellation( const NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has detected that not all other nodes on the network are any longer ready to enter Bus-Sleep Mode.	

8.4.13 Nm_TxTimeoutException

Nm307:

Service name:	Nm_TxTimeoutException	
Syntax:	`void Nm_TxTimeoutException( const NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x1b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service to indicate that an attempt to send an NM message failed.	

The notification that an attempt to send an NM message failed is only needed for OEM specific extensions of the *Nm*.

8.5 Scheduled functions

[**Nm121**: In case a main function exists and it is called before the NM Interface has been initialized, the main function shall immediately return without yielding an error.

8.5.1 Nm_MainFunction

Nm118:

Service name:	Nm_MainFunction

Syntax:	`void Nm_MainFunction(`
	
	`)`

Service ID[hex]:	0x10
Timing:	FIXED_CYCLIC
Description:	This function implements the processes of the NM Interface, which need a fix cyclic scheduling.

Pre condition of Nm_MainFunction: NM Interface must be initialized before.

Nm020: Configuration of Nm_MainFunction: Optional (only available if `NmCoordinatorSupportEnabled` {NM_COORDINATOR_SUPPORT_ENABLED} is set to `TRUE`).

8.6 Expected Interfaces

This chapter lists all interfaces required from other modules.

8.6.1 Mandatory Interfaces

This chapter lists all interfaces required from other modules.

Nm119:

API function	***Description***
CanNm_GetState	Returns the state and the mode of the network management. Caveats: CanNm is initialized correctly. Configuration: Mandatory
CanNm_PassiveStartUp	Passive startup of the AUTOSAR CAN NM. It triggers the transition from Bus-Sleep Mode or Prepare Bus Sleep Mode to the Network Mode in Repeat Message State. This service has no effect if the current state is not equal to Bus-Sleep Mode or Prepare Bus Sleep Mode. In that case NM_E_NOT_EXECUTED is returned. Caveats: CanNm is initialized correctly. Configuration: Mandatory

ComM_Nm_BusSleepMode	Notification that the network management has entered Bus-Sleep Mode. This callback function should perform a transition of the hardware and transceiver to bus-sleep mode.
ComM_Nm_NetworkMode	Notification that the network management has entered Network Mode.
ComM_Nm_NetworkStartIndication	Indication that a NM-message has been received in the Bus Sleep Mode, which indicates that some nodes in the network have already entered the Network Mode.
ComM_Nm_PrepareBusSleepMode	Notification that the network management has entered Prepare Bus-Sleep Mode. Reentrancy: Reentrant (but not for the same NM-Channel)
ComM_Nm_RestartIndication	If NmIf has started to shut down the coordinated busses, AND not all coordinated busses have indicated bus sleep state, AND on at least on one of the coordinated busses NM is restarted, THEN the NM Interface shall call the callback function ComM_Nm_RestartIndication with the nmNetworkHandle of the channels which have already indicated bus sleep state.
FrNm_GetState	This function returns the state and the mode of the network management.
FrNm_PassiveStartUp	Initiates the Passive Startup of the FlexRay NM.

8.6.2 Optional Interfaces

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

Nm122:

API function	*Description*
CanNm_CheckRemoteSleepIndication	Check if remote sleep indication takes place or not. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_REMOTE_SLEEP_INDICATION_ENABLED is defined).
CanNm_DisableCommunication	Disable the NM PDU transmission ability. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_COM_CONTROL_ENABLED is defined).
CanNm_EnableCommunication	Enable the NM PDU transmission ability. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_COM_CONTROL_ENABLED is defined).
CanNm_GetLocalNodeIdentifier	Get node identifier configured for the local node. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_ID_ENABLED is defined).
CanNm_GetNodeIdentifier	Get node identifier out of the most recently received NM PDU.

	Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_ID_ENABLED is defined).
CanNm_GetPduData	Get the whole PDU data out of the most recently received NM PDU. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_ID_ENABLED or CANNM_NODE_DETECTION_ENABLED or CANNM_USER_DATA_ENABLED is defined).
CanNm_GetUserData	Get user data out of the most recently received NM PDU. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_USER_DATA_ENABLED is defined).
CanNm_NetworkRelease	Release the network, since ECU doesn't have to communicate on the bus. Network state shall be changed to 'released'. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_PASSIVE_MODE_ENABLED is not defined).
CanNm_NetworkRequest	Request the network, since ECU needs to communicate on the bus. Network state shall be changed to 'requested'. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_PASSIVE_MODE_ENABLED is not defined).
CanNm_RepeatMessageRequest	Set Repeat Message Request Bit for NM PDUs transmitted next on the bus. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_NODE_DETECTION_ENABLED is defined).
CanNm_RequestBusSynchronization	Request bus synchronization. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_BUS_SYNCHRONIZATION_ENABLED is defined) and CANNM_PASSIVE_MODE_ENABLED is not defined.
CanNm_SetCoordBits	Set the two bits in the Control Bit Vector which represent the NM coordinator ID.
CanNm_SetSleepReadyBit	Set the NM Coordinator Sleep Ready bit in the Control Bit Vector
CanNm_SetUserData	Set user data for NM PDUs transmitted next on the bus. Caveats: CanNm is initialized correctly. Configuration: Optional (Only available if CANNM_USER_DATA_ENABLED is defined and

	CANNM_PASSIVE_MODE_ENABLED is not defined).
Com_SendSignal	The service Com_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter. If the signal has the Triggered transfer property, the update is followed by immediate transmission (within the next main function at the latest) of the I-PDU associated with the signal except when the signal is packed into an I-PDU with Periodic transmission mode; in this case, no transmission is initiated by the call to this service. If the signal has the Pending transfer property, no transmission is caused by the update.
Det_ReportError	Service to report development errors.
FrNm_CheckRemoteSleepIndication	This function checks if remote sleep indication has taken place or not.
FrNm_GetLocalNodeIdentifier	This function gets the node identifier configured for the local node.
FrNm_GetNodeIdentifier	This function gets the node identifier from the last successfully received NM-message.
FrNm_GetPduData	Gets PDU data.
FrNm_GetUserData	This function gets user data from the last successfully received NM message.
FrNm_NetworkRelease	This function releases the network because the ECU doesn't have to communicate on the bus. Network state shall be changed to 'released'.
FrNm_NetworkRequest	This function requests the network because the ECU needs to communicate on the bus. Network state shall be changed to 'requested'.
FrNm_RepeatMessageRequest	This function causes a Repeat Message Request to be transmitted next on the bus.
FrNm_RequestBusSynchronization	This function has no functionality - the service is provided only to be compatible to future extensions and to be compatible to the CAN-NM interface.
FrNm_SetCoordBits	Set the two bits in the Control Bit Vector which represent the NM coordinator ID.
FrNm_SetSleepReadyBit	Set the NM Coordinator Sleep Ready bit in the Control Bit Vector
FrNm_SetUserData	This function sets user data for NM-Data transmitted next on the bus.

9 Sequence diagrams

The role of the NM Interface module is to act as a dispatcher of functions.

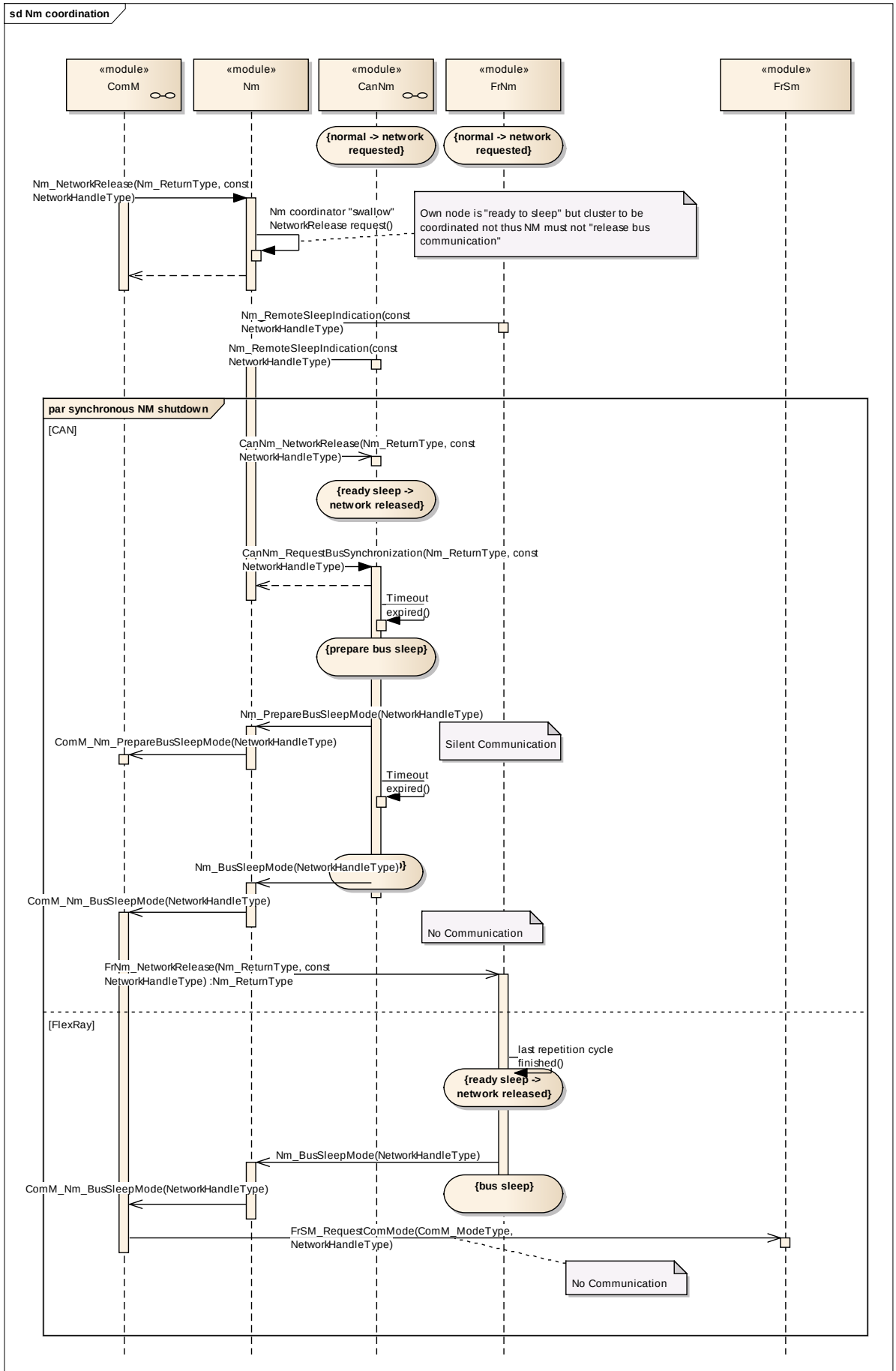
sd Nm coordination
«module» ComM
«module» Nm
«module» CanNm
«module» FrNm
«module» FrSm
{normal -> network requested}
{normal -> network requested}
Nm_NetworkRelease(Nm_ReturnType, const NetworkHandleType)
Nm coordinator "swallow" NetworkRelease request()
Own node is "ready to sleep" but cluster to be coordinated not thus NM must not "release bus communication"
Nm_RemoteSleepIndication(const NetworkHandleType)
Nm_RemoteSleepIndication(const NetworkHandleType)
par synchronous NM shutdown
[CAN]
CanNm_NetworkRelease(Nm_ReturnType, const NetworkHandleType)
{ready sleep -> network released}
CanNm_RequestBusSynchronization(Nm_ReturnType, const NetworkHandleType)
Timeout expired()
{prepare bus sleep}
Nm_PrepareBusSleepMode(NetworkHandleType)
ComM_Nm_PrepareBusSleepMode(NetworkHandleType)
Silent Communication
Timeout expired()
Nm_BusSleepMode(NetworkHandleType)
ComM_Nm_BusSleepMode(NetworkHandleType)
No Communication
FrNm_NetworkRelease(Nm_ReturnType, const NetworkHandleType) :Nm_ReturnType
[FlexRay]
last repetition cycle finished()
{ready sleep -> network released}
Nm_BusSleepMode(NetworkHandleType)
ComM_Nm_BusSleepMode(NetworkHandleType)
{bus sleep}
FrSM_RequestComMode(ComM_ModeType, NetworkHandleType)
No Communication

Figure 9.1

10 Configuration specification

The following chapter contains tables of all configuration parameters and switches used to determine the functional units of the Generic Network Management Interface. The default values of configuration parameters are denoted as bold.

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. Chapter 10.2 specifies the structure (containers) and the parameters of the module <Module Name>. Chapter 10.3 specifies published information of the module <Module Name>.

10.1 How to read this chapter

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

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

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

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

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

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

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

10.2 Containers and configuration parameters

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

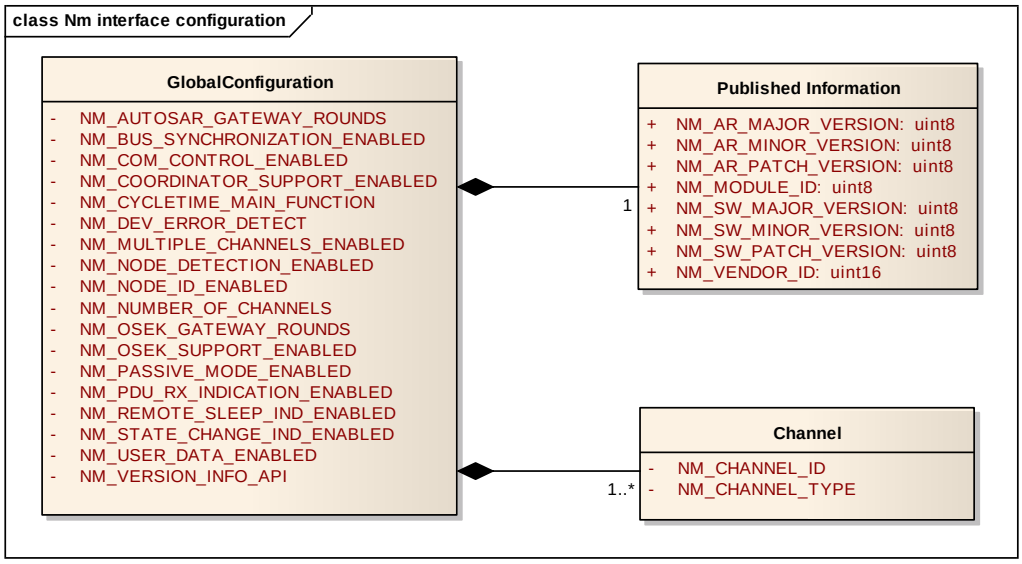

Figure 10.1

Figure 10.1 specifies all configurable parameters in this module.

10.2.1 Variants

Nm120: Variant 1: All configuration parameters are configurable at pre-compile time.
Use case: Source code optimizations

Variant 2: not supported

Variant 3: not supported

10.2.2 Nm

SWS Item	**Nm233_Conf :**
Module Name	*Nm*
Module Description	The Generic Network Management Interface module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
NmGlobalConfig	1	This container contains all global configuration parameters of the Nm Interface.

10.2.3 NmGlobalConfig

SWS Item	**Nm196_Conf :**
Container Name	NmGlobalConfig{Nm_GlobalConfig}
Description	This container contains all global configuration parameters of the Nm Interface.
Configuration Parameters	

SWS Item	**Nm240_Conf :**		
Name	NmAutosarGatewayRounds {NM_AUTOSAR_GATEWAY_ROUNDS}		
Description	Number of rounds the coordinator shall keep a bus which runs AUTOSAR NM awake after all nodes including itself are ready to sleep.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: The number is the same for all coordinated AUTOSAR busses. Only valid if NM_COORDINATOR_SUPPORT_ENABLED is set to TRUE.		

SWS Item	**Nm208_Conf :**		
Name	NmBusSynchronizationEnabled {NM_BUS_SYNCHRONIZATION_ENABLED}		
Description	Pre-processor switch for enabling bus synchronization support. This feature is required for gateway nodes only. true: Enabled false:Disabled		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: This parameter shall be enabled if NmCoordinatorSupportEnabled is enabled.		

SWS Item	**Nm234_Conf :**
Name	NmCarWakeUpCallback {NM_CAR_WAKE_UP_CALLBACK}
Description	Name of the callback function to be called if Nm_CarWakeUpIndication() is

	called.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only needed if NmCarWakeUpRxEnabled == TRUE		

SWS Item	**Nm235_Conf :**		
Name	NmCarWakeUpRxEnabled {NM_CAR_WAKE_UP_RX_ENABLED}		
Description	Enables or disables CWU detection. FALSE - CarWakeUp not supported TRUE - CarWakeUp supported		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm210_Conf :**		
Name	NmComControlEnabled {NM_COM_CONTROL_ENABLED}		
Description	Pre-processor switch for enabling the Communication Control support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm230_Conf :**		
Name	NmComUserDataSupport {NM_COM_USER_DATA_SUPPORT}		
Description	Enable/Disable setting of NMUserData via SW-C. If NmComUserDataSupport is enabled the API Nm_SetUserData shall not be available.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm206_Conf :**

Name	NmCoordinatorSupportEnabled {NM_COORDINATOR_SUPPORT_ENABLED}		
Description	Switch to inform if NM coordinator needs to be supported.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Only valid if NmRemoteSleepIndEnabled AND NmNumberOfChannels > 1		

SWS Item	**Nm236_Conf :**		
Name	NmCoordinatorSyncPrio {NM_COORDINATOR_SYNC_PRIO}		
Description	Priority of the coordinator		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	1 .. 3		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Only valid if NM_COORDINATOR_SYNC_SUPPORT is set to TRUE.		

SWS Item	**Nm242_Conf :**		
Name	NmCoordinatorSyncSupport {NM_COORDINATOR_SYNC_SUPPORT}		
Description	Pre-processor switch for enabling the synchronisation of multiple coordinators.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Only valid if NM_COORDINATOR_SUPPORT_ENABLED is set to TRUE.		

SWS Item	**Nm205_Conf :**		
Name	NmCycletimeMainFunction {NM_CYCLETIME_MAIN_FUNCTION}		
Description	The period between successive calls to the Main Function of the NM Interface in seconds.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

	dependency: If NmCoordinatorSupportEnabled is set to TRUE, then the NmCycletimeMainFunction parameter shall be configured.

SWS Item	**Nm203_Conf :**		
Name	NmDevErrorDetect {NM_DEV_ERROR_DETECT}		
Description	Pre-processor switch for enabling development error detection and notification.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm237_Conf :**		
Name	NmGlobalCoordinatorTime {NM_GLOBAL_COORDINATOR_TIME}		
Description	This parameter defines the maximum shutdown time of a connected and coordinated NM-Cluster. Note:This includes nested connections.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Shall be available if NmCoordinatorSupportEnabled is enabled.		

SWS Item	**Nm212_Conf :**		
Name	NmNodeDetectionEnabled {NM_NODE_DETECTION_ENABLED}		
Description	Pre-processor switch for enabling the Request Repeat Message Request support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Only valid if NM_NODE_ID_ENABLED is set to TRUE		

SWS Item	**Nm213_Conf :**		
Name	NmNodeIdEnabled {NM_NODE_ID_ENABLED}		
Description	Pre-processor switch for enabling the source node identifier.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build	--	

	time		
Scope / Dependency	scope: Module		

SWS Item	**Nm201_Conf :**		
Name	NmNumberOfChannels {NM_NUMBER_OF_CHANNELS}		
Description	Number of NM channels allowed within one ECU.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm238_Conf :**		
Name	NmOsekGatewayRounds {NM_OSEK_GATEWAY_ROUNDS}		
Description	Number of rounds the coordinator shall keep a bus which runs OSEK NM awake after all nodes including itself are ready to sleep.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: The number is the same for all coordinated OSEK busses. Only valid if NM_OSEK_SUPPORT_ENABLEDis setto TRUE.		

SWS Item	**Nm239_Conf :**		
Name	NmOsekSupportEnabled {NM_OSEK_SUPPORT_ENABLED}		
Description	Switch to inform if NM coordinator needs to support direct OSEK NM.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Only valid if NM_COORDINATOR_SUPPORT_ENABLED		

SWS Item	**Nm209_Conf :**		
Name	NmPassiveModeEnabled {NM_PASSIVE_MODE_ENABLED}		
Description	Pre-processor switch for enabling support of the Passive Mode.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: Module

SWS Item	**Nm214_Conf :**		
Name	NmPduRxIndicationEnabled {NM_PDU_RX_INDICATION_ENABLED}		
Description	Pre-processor switch for enabling the PDU Rx Indication.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm207_Conf :**		
Name	NmRemoteSleepIndEnabled {NM_REMOTE_SLEEP_IND_ENABLED}		
Description	Pre-processor switch for enabling remote sleep indication support. This feature is required for gateway nodes only.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: It must not be enabled if NmPassiveModeEnabled is enabled.		

SWS Item	**Nm215_Conf :**		
Name	NmStateChangeIndEnabled {NM_STATE_CHANGE_IND_ENABLED}		
Description	Pre-processor switch for enabling the CAN Network Management state change notification.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**Nm211_Conf :**		
Name	NmUserDataEnabled {NM_USER_DATA_ENABLED}		
Description	Pre-processor switch for enabling user data support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: Module

SWS Item	**Nm204_Conf :**		
Name	NmVersionInfoApi {NM_VERSION_INFO_API}		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
NmChannelConfig	1..*	This container contains the configuration (parameters) of the bus channel(s). The channel parameter shall be harmonized within the whole communication stack.

Nm083 The Global Scope specifies configuration parameters that shall be defined in the module's configuration header file **`Nm_Cfg.h`**.

10.2.4 NmChannelConfig

SWS Item	**Nm197_Conf :**
Container Name	NmChannelConfig{Channel}
Description	This container contains the configuration (parameters) of the bus channel(s). The channel parameter shall be harmonized within the whole communication stack.
Configuration Parameters	

SWS Item	**Nm218_Conf :**		
Name	NmBusType {NM_BUS_TYPE}		
Description	Identifies the bus type of the channel. LIN is not yet supported.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	NM_BUS_TYPE_CAN	CAN bus	
	NM_BUS_TYPE_FR	FlexRay bus	
	NM_BUS_TYPE_LIN	LIN bus. Not yet supported!	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**Nm216_Conf :**	
Name	NmChannelId {NM_CHANNEL_ID}	
Description	Channel identification number of the corresponding channel.	
Multiplicity	1	
Type	IntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	--	

ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Shall be harmonized with channel IDs of the whole communication stack.		

SWS Item	**Nm241_Conf :**		
Name	NmSelectiveNmChannel {NM_SELECTIVE_NM_CHANNEL}		
Description	If this value is set to FALSE the corresponding channel will be coordinated. If this value is set to TRUE the channel will be excluded from coordination.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Network dependency: Only valid if NM_COORDINATOR_SUPPORT_ENABLED is set to TRUE.		

SWS Item	**Nm231_Conf :**		
Name	NmStateReportEnabled {NM_STATE_REPORT_ENABLED}		
Description	Specifies if the NMS shall be set for the corresponding network. false: No NMS shall be set true: The NMS shall be set		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: only available if NM_STATE_CHANGE_IND_ENABLED is TRUE and <Bus>NmComUserDataSupport is configured		

SWS Item	**Nm217_Conf :**		
Name	NmComMChannelRef		
Description	Reference to the corresponding ComM Channel.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**Nm232_Conf :**
Name	NmStateReportSignalRef {NM_STATE_REPORT _SIGNAL_REF}
Description	Reference to the signal for setting the NMS by calling Com_SendSignal for the respective channel.

Multiplicity	0..1		
Type	Reference to [ComSignal]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Signal must be configured in COM. Only available if NmStateReportEnabled == true		

No Included Containers

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

Nm008 The following table specifies configuration parameters that shall be published in the module's header file **`Nm.h`** and also in the module's description file.

The standard common published information like

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

is provided in the BSW Module Description Template (see [8] Figure 4.1 and Figure 7.1).

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