Document Title	Specification of Module Generic Network Management Interface
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	1.0.0
Document Status	Draft
Part of Release	2.1
Revision	0014

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.01.2007	1.0.0	AUTOSAR Administration	Initial release

Release Notes

Errata and known deficiencies

All modifications planned in the scope of Release 2.1 for the incorporation into this document are completed. The document, however, has not yet undergone the necessary finalization.

Disclaimer

Any use of these specifications requires membership within the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of these specifications.

Following the completion of the development of the AUTOSAR specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

Advice to users of AUTOSAR Specification Documents:

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 AUTOSAR Network Management Interface module.

The AUTOSAR 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)

2 Acronyms and abbreviations

Acronym:	*Description:*
CC	Communication Controller
NM	Network Management

Abbreviation:	*Description:*
CanIf	CAN Interface
CNm	CAN Generic NM
CanNm	CAN NM
ComM	Communication Manager
EcuM	ECU State Manager
DEM	Diagnostic Event 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-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.
Passive Node	A node where the Network Management is configured such that it cannot contribute directly to the cluster shutdown decision.

3 Related documentation

3.1 Input documents

[1] AUTOSARLayered Software Architecture
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SoftwareArchitecture.pdf

[2] AUTOSAR Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[3] AUTOSAR Requirements on Basic Software, Module NM
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_Generic_NM.pdf

[4] AUTOSAR Software Specification of CAN Generic Network Management,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_CAN_Generic_NM.pdf

[5] AUTOSAR Software Specification of FlexRay Network Management
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_FlexRay_NM.pdf

[6] AUTOSAR Software Specification of Module Communication Manager
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_ComManager.pdf

[7] AUTOSAR Software Specification of Module ECU State Manager
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_EcuStateManager.pdf

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.
c. The NM Interface shall only include the common modes, definitions and return values of different Bus Specific Network management layers.

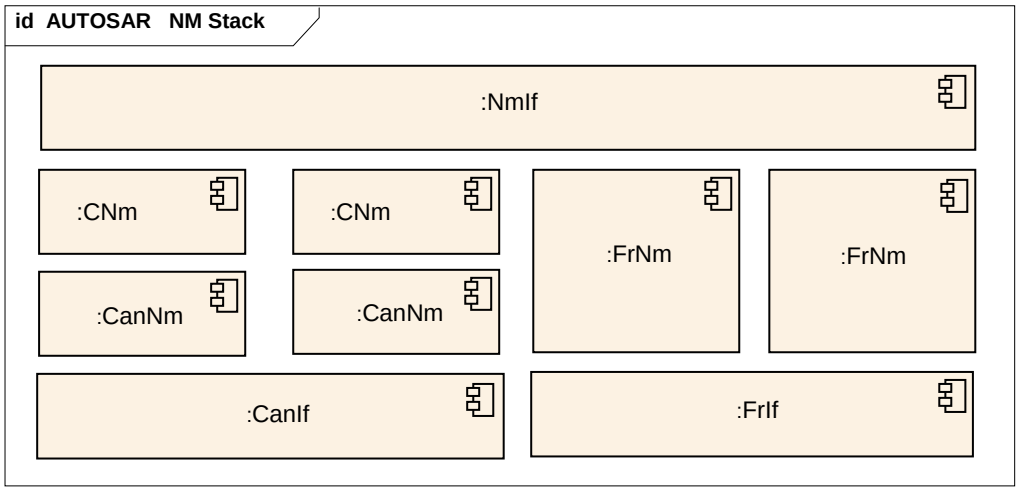


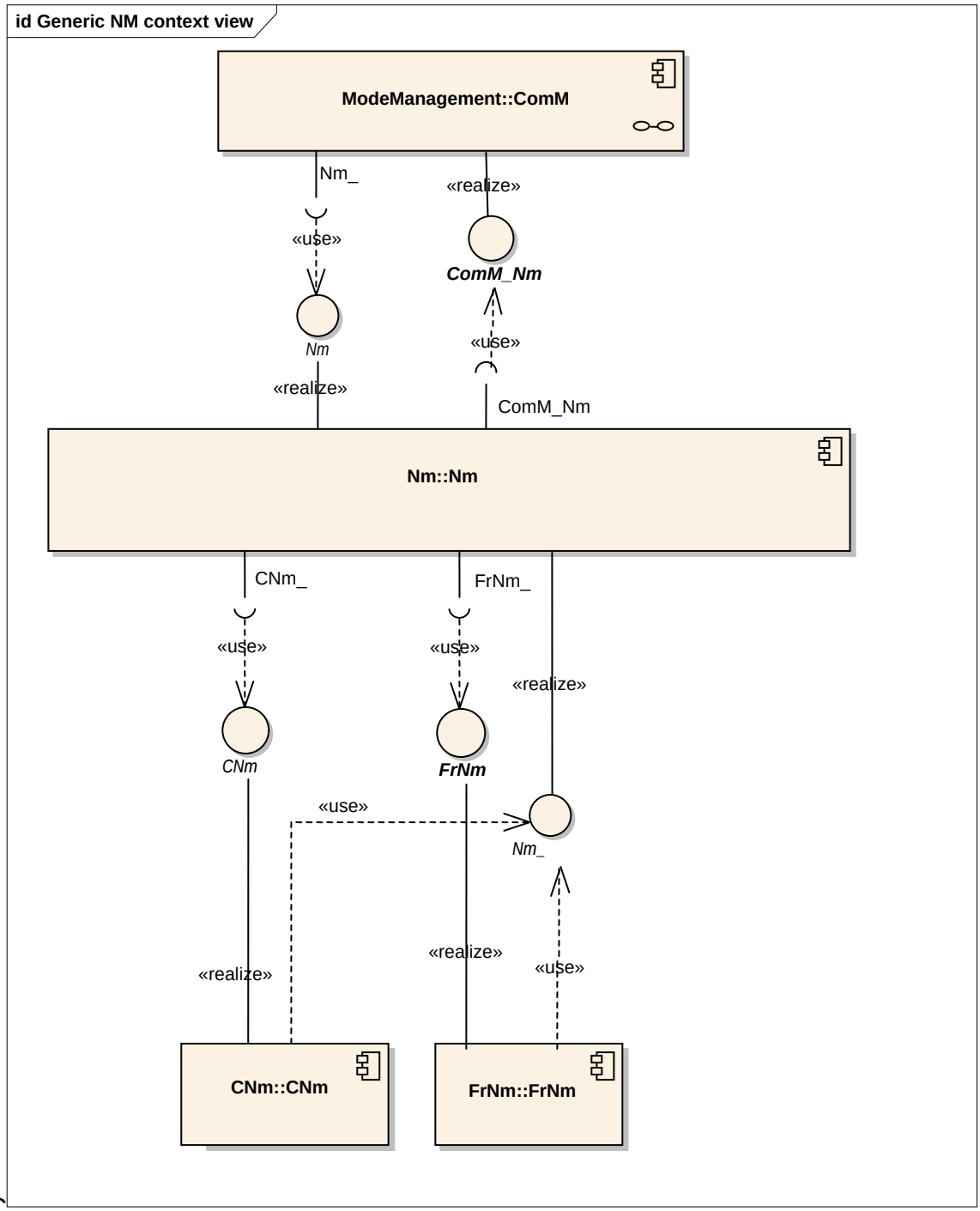
Figure 4.1

4.2 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

The Network management Interface module provides services to the Communication Manager (ComM) and uses services of the respective bus specific Network Management adaptation layers (CNm and FrNm) as shown in Figure 5.1.



C

Figure 5.1

5.1 File structure

5.1.1 Code file structure

The following C-files shall be provided by the Nm module

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

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

5.1.2 Header file structure

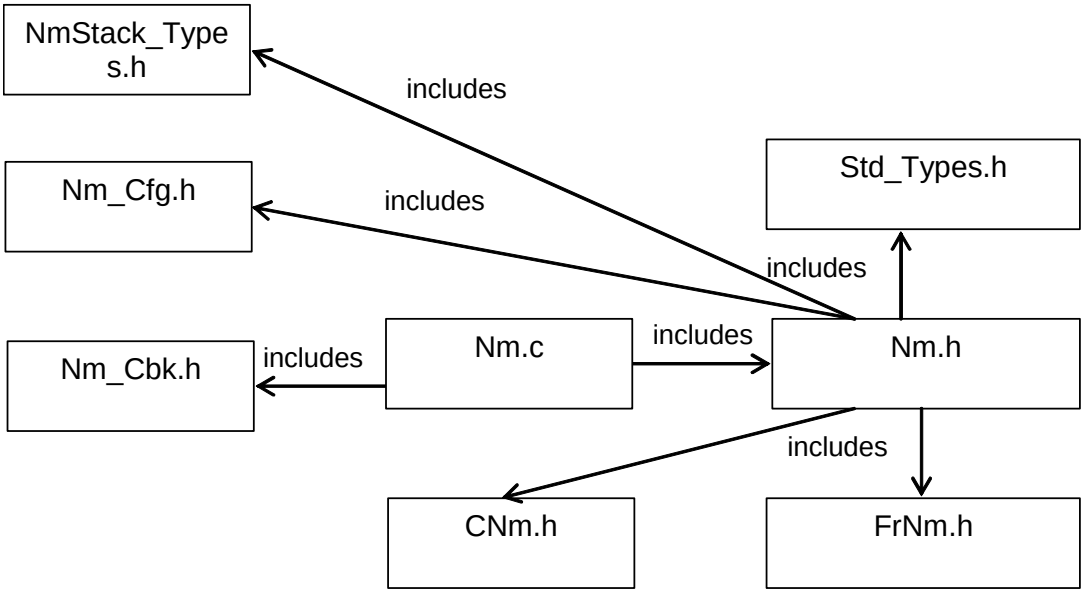


Figure 5.2

The following header files shall be provided by the Nm Interface module:

- `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)

- `CNm.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_Cbk.h` (for Communication Manager callback functions)

.

6 Requirements traceability

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

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	
[BSW00404] Reference to post build time configuration	
[BSW00405] Reference to multiple configuration sets	
[BSW00345] Pre-compile-time configuration	
[BSW159] Tool-based configuration	
[BSW167] Static configuration checking	
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	
[BSW171] Configurability of optional functionality	
[BSW00380] Separate C-Files for configuration parameters	
[BSW00419] Separate C-Files for pre-compile time configuration parameters	
[BSW00381] Separate configuration header file for pre-compile time parameters	
[BSW00412] Separate H-File for configuration parameters	
[BSW00383] List dependencies of configuration files	
[BSW00384] List dependencies to other modules	
[BSW00387] Specify the configuration class of callback function	
[BSW00388] Introduce containers	
[BSW00389] Containers shall have names	
[BSW00390] Parameter content shall be unique within the module	
[BSW00391] Parameter shall have unique names	
[BSW00392] Parameters shall have a type	
[BSW00393] Parameters shall have a range	
[BSW00394] Specify the scope of the parameters	
[BSW00395] List the required parameters (per parameter)	
[BSW00396] Configuration classes	
[BSW00397] Pre-compile-time parameters	
[BSW00398] Link-time parameters	
[BSW00399] Loadable Post-build time parameters	
[BSW00400] Selectable Post-build time parameters	
[BSW00402] Published information	
[BSW00375] Notification of wake-up reason	
[BSW101] Initialization interface	
[BSW00416] Sequence of Initialization	
[BSW00406] Check module initialization	
[BSW168] Diagnostic Interface of SW components	
[BSW00407] Function to read out published parameters	
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	
[BSW00424] BSW main processing function task allocation	
[BSW00425] Trigger conditions for schedulable objects	
[BSW00426] Exclusive areas in BSW modules	
[BSW00427] ISR description for BSW modules	
[BSW00428] Execution order dependencies of main processing functions	
[BSW00429] Restricted BSW OS functionality access	
[BSW00431] The BSW Scheduler module implements task bodies	
[BSW00432] Modules should have separate main processing	

functions for read/receive and write/transmit data path	
[BSW00433] Calling of main processing functions	
[BSW00434] The Schedule Module shall provide an API for exclusive areas	
[BSW00336] Shutdown interface	
[BSW00337] Classification of errors	
[BSW00338] Detection and Reporting of development errors	
[BSW00369] Do not return development error codes via API	
[BSW00339] Reporting of production relevant error status	
[BSW00417] Reporting of Error Events by Non-Basic Software	
[BSW00323] API parameter checking	
[BSW004] Version check	
[BSW00409] Header files for production code error IDs	
[BSW00385] List possible error notifications	
[BSW00386] Configuration for detecting an error	
[BSW161] Microcontroller abstraction	
[BSW162] ECU layout abstraction	
[BSW005] No hard coded horizontal interfaces within MCAL	
[BSW00415] User dependent include files	
[BSW164] Implementation of interrupt service routines	
[BSW00325] Runtime of interrupt service routines	
[BSW00326] Transition from ISRs to OS tasks	
[BSW00342] Usage of source code and object code	
[BSW00343] Specification and configuration of time	
[BSW160] Human-readable configuration data	
[BSW007] HIS MISRA C	
[BSW00300] Module naming convention	
[BSW00413] Accessing instances of BSW modules	
[BSW00347] Naming separation of different instances of BSW drivers	
[BSW00305] Self-defined data types naming convention	
[BSW00307] Global variables naming convention	
[BSW00310] API naming convention	
[BSW00373] Main processing function naming convention	
[BSW00327] Error values naming convention	
[BSW00335] Status values naming convention	
[BSW00350] Development error detection keyword	
[BSW00408] Configuration parameter naming convention	
[BSW00410] Compiler switches shall have defined values	
[BSW00411] Get version info keyword	
[BSW00346] Basic set of module files	
[BSW158] Separation of configuration from implementation	
[BSW00314] Separation of interrupt frames and service routines	
[BSW00370] Separation of callback interface from API	
[BSW00348] Standard type header	
[BSW00353] Platform specific type header	
[BSW00361] Compiler specific language extension header	
[BSW00301] Limit imported information	
[BSW00302] Limit exported information	
[BSW00328] Avoid duplication of code	
[BSW00312] Shared code shall be reentrant	
[BSW006] Platform independency	
[BSW00357] Standard API return type	
[BSW00377] Module specific API return types	
[BSW00304] AUTOSAR integer data types	
[BSW00355] Do not redefine AUTOSAR integer data types	
[BSW00378] AUTOSAR boolean type	

[BSW00306] Avoid direct use of compiler and platform specific keywords	
[BSW00308] Definition of global data	
[BSW00309] Global data with read-only constraint	
[BSW00371] Do not pass function pointers via API	
[BSW00358] Return type of init() functions	
[BSW00414] Parameter of init function	
[BSW00376] Return type and parameters of main processing functions	
[BSW00359] Return type of callback functions	
[BSW00360] Parameters of callback functions	
[BSW00329] Avoidance of generic interfaces	
[BSW00330] Usage of macros / inline functions instead of functions	
[BSW00331] Separation of error and status values	
[BSW009] Module User Documentation	
[BSW00401] Documentation of multiple instances of configuration parameters	
[BSW172] Compatibility and documentation of scheduling strategy	
[BSW010] Memory resource documentation	
[BSW00333] Documentation of callback function context	
[BSW00374] Module vendor identification	
[BSW00379] Module identification	
[BSW003] Version identification	
[BSW00318] Format of module version numbers	
[BSW00321] Enumeration of module version numbers	
[BSW00341] Microcontroller compatibility documentation	
[BSW00334] Provision of XML file	

Document: AUTOSAR Requirements on Basic Software, Module NM [2].

Requirement	***Satisfied by***
[BSW150] Configuration of functionality	
[BSW151] Integration into running NM cluster	
[BSW043] Bus Traffic without NM Initialization	
[BSW044] Applicability to different types of communication systems	
[BSW045] NM-cluster Independent Shutdown Coordination	
[BSW046] Trigger of startup of all Nodes at any Point in Time	
[BSW047] Bus Keep Awake Services	
[BSW048] Bus Sleep Mode	
[BSW050] NM State Information	
[BSW051] NM State Change Indication	
[BSW052] Notification that all other ECUs are ready to sleep	
[BSW02509] Notification that at least one other node is not ready to sleep anymore	
[BSW02503] Sending user data	
[BSW02504] Receiving user data	
[BSW153] Detection of present nodes	
[BSW02508] Unambiguous node identification per bus	
[BSW02505] Sending node identifier	
[BSW02506] Receiving node identifier	
[BSW02511] Configurable Role in Cluster Shutdown	
[BSW053] Deterministic Behavior in Case of Bus Unavailability	
[BSW137] Communication system error handling	
[BSW136] Coordination of coupled networks	

[BSW054] Deterministic Time for Bus Sleep	
[BSW142] Limitation of NM bus load	
[BSW143] Predictable NM bus load	
[BSW144] ECU cluster size	
[BSW145] Robustness against NM message losses	
[BSW146] Robustness against NM message jitter	
[BSW147] Processor independent algorithm	
[BSW149] Configurable Timing	
[BSW154] Bus independency of API	
[BSW148] Separation of Communication system dependent parts	
[BSW02510] Immediate Transmission Confirmation	
[BSW02512] CommunicationControl (28 hex) service support	

7 Functional specification

The AUTOSAR NM Interface module acts as a bus-independent adaptation layer between the bus-specific Network Management modules (CNm, FrNm and LinNm) and the AUTOSAR basic software module Communication Manager. The role NM Interface module is to convert Generic function calls to Bus specific functions of the Bus specific NM layer.

7.1 Error classification

This specification will not contain any error classification.

7.2 Error detection

This specification will not contain any error detection.

7.3 Error notification

This specification will not contain any error notification.

8 API specification

8.1 Imported types

8.1.1 Standard types

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

- Std_Types.h

Standard types included by NM

- Std_VersionInfoType

8.2 Type definitions

8.2.1 Nm_ReturnType

Type:	`enum Nm_ReturnType`	
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

Type:	`enum`	
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

Type:	`enum`	
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_ChannelHandleType

Type:	`uint8`	
Range:	0 .. 255	8-Bit unsigned integer (`unit8`)
Description:	Handle to identify a NM-Channel.	

8.2.5 Nm_BusNmType

Type:	`Enum`	
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	

8.2.6 Nm_BusNm_ConfigType

Type:	`structure`
Range:	N/A N/A
Description:	Data structure consisting of: • const CNm_ConfigType * const CanNmConfigPtr (Only available if `NM_BUSNM_CANNM_ENABLED` is set to ON) • const FrNm_ConfigType * const FrNmConfigPtr (Only available if `NM_BUSNM_FRNM_ENABLED` is set to ON) Data structure shall be pre-compile time configurable.

8.3 Function definitions

8.3.1 Services provided by NM Interface

8.3.1.1 Nm_Init

Service name:	Nm_Init	
Syntax:	For all configuration Variants `void Nm_Init ( Nm_BusNm_ConfigType * const nmConfigPtr );`	
Service ID:	`0x00`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmConfigPtr`	Pointer to the selected configuration set
	-	-
	-	-

Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	Initializes the NM Interface. This function shall in turn call the <BusNm>[1]_Init for all configured Bus specific NM's. In case the pointer to the corresponding configuration is a NULL pointer, the corresponding bus specific NM initialization function shall not be called.	
Caveats:	This service function has to be called after the initialization of the respective bus interface.	
Configuration:	Mandatory	

8.3.1.2 Nm_PassiveStartUp

Service name:	Nm_PassiveStartUp	
Syntax:	`Nm_ReturnType Nm_PassiveStartUp ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Passive start of network management has failed
	NM_E_NOT_EXECUTED	Passive start of network management has not executed
Description:	This function calls the <BusNm>_PassiveStartUp function (e.g. CNm_PassiveStartUp function is called if channel is configured as CAN).	
Caveats:	CanNm and FrNm are initialized correctly.	
Configuration:	Mandatory	

8.3.1.3 Nm_NetworkRequest

Service name:	Nm_NetworkRequest	
Syntax:	`Nm_ReturnType Nm_NetworkRequest ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel

[1] <BusNm> is a symbolic entry representing CNm, FrNm and LinNm for CAN, FlexRay and LIN NM modules respectively.

	-	-
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Requesting of bus communication has failed
Description:	This function calls the <BusNm>_NetworkRequest (e.g. CNm_NetworkRequest function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if NM_PASSIVE_NODE_ENABLED is set to OFF)	

8.3.1.4 Nm_NetworkRelease

Service name:	Nm_NetworkRelease	
Syntax:	`Nm_ReturnType Nm_NetworkRelease ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Releasing of bus communication has failed
	NM_E_NOT_EXECUTED	Function to release bus communication not executed
Description:	This function calls the <BusNm>_NetworkRelease bus specific function (e.g. CNm_NetworkRelease function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if NM_PASSIVE_NODE_ENABLED is set to OFF)	

8.3.1.5 Nm_DisableCommunication

Service name:	Nm_DisableCommunication	
Syntax:	`Nm_ReturnType Nm_DisableCommunication ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	0x0C	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Disabling of NM PDU transmission ability has failed.

	NM_E_NOT_EXECUTED	Disabling of NM PDU transmission ability is not executed.
Description:	This function calls the CNm_DisableBusCommunication bus specific function. This disable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service	
Caveats:	CNm are initialized correctly	
Configuration:	Optional (Only available if NM_COM_CONTROL_ENABLED is defined)	

8.3.1.6 Nm_EnableCommunication

Service name:	Nm_EnableCommunication	
Syntax:	`Nm_ReturnType Nm_EnableCommunication ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	0x0D	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Enabling of NM PDU transmission ability has failed.
	NM_E_NOT_EXECUTED	Enabling of NM PDU transmission ability is not executed.
Description:	This function calls the CNm_NetworkRelease bus specific function.This Enables the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service	
Caveats:	CNm is initialized correctly.	
Configuration:	Optional (Only available if NM_COM_CONTROL_ENABLED is defined)	

8.3.1.7 Nm_SetUserData

Service name:	Nm_SetUserData	
Syntax:	`Nm_ReturnType Nm_SetUserData ( const Nm_ChannelHandleType nmChannelHandle, const uint8 * const nmUserDataPtr );`	
Service ID:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	nmUserData	User data for the next transmitted NM message
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Setting of user data has failed
Description:	Set user data for NM messages transmitted next on the bus. For that purpose <BusNm>_SetUserData shall be called (e.g.	

	CNm_SetUserData function is called if channel is configured as CAN).
Caveats:	The <BusNm> and the Nm itself are initialized correctly.
Configuration:	Optional (Only available if `NM_USER_DATA_ENABLED` and `NM_PASSIVE_MODE_ENABLED` is defined)

8.3.1.8 Nm_GetUserData

Service name:	Nm_GetUserData	
Syntax:	`Nm_ReturnType Nm_GetUserData ( const Nm_ChannelHandleType nmChannelHandle, uint8 * const nmUserDataPtr, );`	
Service ID:	`0x05`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
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_E_OK`	No error
	`NM_E_NOT_OK`	Getting of user data has failed
Description:	Get user data out of the last successfully received NM message. For that purpose `<BusNm>_GetUserData` shall be called (e.g. CNm_GetUserData function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if `NM_USER_DATA_ENABLED` is set to ON)	

8.3.1.9 Nm_GetPduData

Service name:	Nm_GetPduData	
Syntax:	`Nm_ReturnType Nm_GetPduData ( const Nm_ChannelHandleType nmChannelHandle, uint8 * const nmPduData );`	
Service ID:	`0x0A`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
Parameters (out):	`nmPduData`	Pointer where NM PDU shall be copied to.
	-	-
	-	-
Return value:	`NM_E_OK`	No error

	NM_E_NOT_OK	Getting of NM PDU data has failed
Description:	Get the whole PDU data out of the most recently received NM message. For that purpose <BusNm>_GetPduData shall be called (e.g. CNm_GetPduData function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and <Nm> are initialized correctly	
Configuration:	Optional (Only available if NM_NODE_ID_ENABLED or NM_NODE_DETECTION_ENABLED or NM_USER_DATA_ENABLED is defined).	

8.3.1.10 Nm_RepeatMessageRequest

Service name:	Nm_RepeatMessageRequest	
Syntax:	Nm_ReturnType **Nm_RepeatMessageRequest** (const Nm_ChannelHandleType nmChannelHandle);	
Service ID:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Setting of Repeat Message Request Bit has failed
	NM_E_NOT_EXECUTED	Repeat Message Request is currently not executed.
Description:	Set Repeat Message Request Bit for NM messages transmitted next on the bus. For that purpose <BusNm>_RepeatMessageRequest shall be called (e.g. CNm_RepeatMessageRequest function is called if channel is configured as CAN)	
Caveats:	FrNm and CNm itself are initialized correctly.	
Configuration:	Optional (Only available if NM_NODE_DETECTION_ENABLED is defined)	

8.3.1.11 Nm_GetNodeIdentifier

Service name:	Nm_GetNodeIdentifier	

Syntax:	Nm_ReturnType **Nm_GetNodeIdentifier**
	(
	  const Nm_ChannelHandleType nmChannelHandle,
	  uint8 * const nmNodeIdPtr
	);	

Service ID:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	nmNodeIdPtr	Pointer where node identifier out of the last successfully received NM-message shall be copied to
Return value:	NM_E_OK	No error

	NM_E_NOT_OK	Getting of the node identifier out of the last received NM-message has failed
Description:	Get node identifier out of the last successfully received NM-message. The function <BusNm>_GetNodeIdentifier shall be called.	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if NM_NODE_ID_ENABLED is defined)	

8.3.1.12 Nm_GetLocalNodeIdentifier

Service name:	Nm_GetLocalNodeIdentifier	
Syntax:	Nm_ReturnType **Nm_GetLocalNodeIdentifier** (const Nm_ChannelHandleType nmChannelHandle, uint8 * const nmNodeIdPtr);	
Service ID:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	nmNodeIdPtr	Pointer where node identifier of the local node shall be copied to
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Getting of the node identifier of the local node has failed
Description:	Get node identifier configured for the local node. For that purpose <BusNm>_GetLocalNodeIdentifier shall be called (e.g. CNm_GetLocalNodeIdentifier function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if NM_NODE_ID_ENABLED is defined)	

8.3.1.13 Nm_RequestBusSynchronization

Service name:	Nm_RequestBusSynchronization	
Syntax:	Nm_ReturnType **Nm_RequestBusSynchronization** (const Nm_ChannelHandleType mmChannelHandle);	
Service ID:	0xC0	
Sync/Async:	Synchronous	
Reentrancy:	Non-Reentrant	
Parameters (in):	mmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Requesting of bus synchronization has failed
	NM_E_NOT_EXECUTED	Bus synchronization is not currently not executed.

Description:	Request bus synchronization. This function shall in turn calls the Bus specific function <BusNm>_RequestBusSynchronization (e.g. CNm_RequestBusSynchronization function is called if channel is configured as CAN).
Caveats:	The <BusNm> and the Nm itself are initialized correctly.
Configuration:	Optional (Only available if `NM_BUS_SYNCHRONIZATION_ENABLED` is set to ON)

8.3.1.14 Nm_CheckRemoteSleepIndication

Service name:	Nm_CheckRemoteSleepIndication	
Syntax:	`Nm_ReturnType Nm_CheckRemoteSleepIndication ( const Nm_ChannelHandleType nmChannelHandle, Boolean * const nmRemoteSleepIndPtr );`	
Service ID:	`0xD0`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
Parameters (out):	`NmRemoteSleepIndPtr`	Pointer where check result of remote sleep indication shall be copied to
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Checking of remote sleep indication bits has failed
	`NM_E_NOT_EXECUTED`	Checking of remote sleep indication 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. CNm_CheckRemoteSleepIndication function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if `NM_REMOTE_SLEEP_INDICATION_ENABLED` is set to ON)	

8.3.1.15 Nm_GetState

Service name:	Nm_GetState	
Syntax:	`Nm_ReturnType Nm_GetState ( const Nm_ChannelHandleType nmChannelHandle, Nm_StateType * const nmStatePtr Nm_ModeType * const nmModePtr );`	
Service ID:	`0x0B`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
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_E_OK	No error
	NM_E_NOT_OK	Getting of NM state has failed
Description:	Returns the state of the network management. This function in turn calls the <BusNm>_GetState function (e.g. CNm_GetState function is called if channel is configured as CAN).	
Caveats:	The <BusNm> and the Nm itself are initialized correctly.	
Configuration:	Mandatory	

8.3.1.16 Nm_GetVersionInfo

Service name:	Nm_ GetVersionInfo	
Syntax:	`void Nm_GetVersionInfo { Std_VersionInfoType *NmVerInfoPtr }`	
Service ID:	0xF1	
Sync/Async:	Synchronous	
Reentrancy:	Yes	
Parameters (in):	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.	
Caveats:	--	
Configuration:	Optional (only available if NM_VERSION_INFO_API is set to ON)	

8.4 Call-back notifications

CBK function	*Module*	*Description*
`void Nm_NetworkStartIndication ( const Nm_ChannelHandleType );`	NM Interface	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.
`void Nm_NetworkMode ( const Nm_ChannelHandleType );`	NM Interface	Notification that the network management has entered Network Mode. The callback function shall enable transmission of application messages.
`void Nm_PrepareBusSleepMode (`	NM Interface	Notification that the network management has entered Prepare Bus-Sleep Mode.

`const Nm_ChannelHandleType );`		The callback function shall disable transmission of application messages.
`void` **`Nm_BusSleepMode`** `( const Nm_ChannelHandleType );`	NM Interface	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.

CBK function	*Module*	*Description*
`void` **`Nm_RemoteSleepIndication`** `( const Nm_ChannelHandleType );`	NM Interface	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. *Configuration parameter:* **`NM_REMOTE_SLEEP_IND_ENABLED`**
`void` **`Nm_RemoteSleepCancelation`** `( const Nm_ChannelHandleType );`	NM Interface	Notification that the network management has detected that no more all other nodes are ready to sleep. The NM gateway shall check if the Bus is again required. *Configuration parameter:* **`NM_REMOTE_SLEEP_IND_ENABLED`**
`void` **`Nm_PduRxIndication`** `( const Nm_ChannelHandleType );`	NM Interface	Notification that a NM message has been received. *Configuration parameter:* **`NM_PDU_RX_INDICATION_ENABLED`**
`Nm_StateChangeNotification` `( const Nm_ChannelHandleType, const Nm_StateType nmPreviousState, const Nm_StateType nmCurrentState )`	NM Interface	Notification that the CAN Generic NM state has changed. *Configuration parameter:* **`NM_STATE_CHANGE_IND_ENABLED`**
`Nm_RepeatMessageIndication` `( const Nm_ChannelHandleType, )`	NM Interface	Service to indicate that a NM message with set Repeat Message Request Bit has been received.

8.5 Scheduled functions

There is no scheduled function defined for this release.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

API function	*Module*	*Description*
ComM_Nm_NetworkStartIndication	ComM	Indication 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.
ComM_Nm_NetworkMode	ComM	Notification that the network management has entered Network Mode.
ComM_Nm_PrepareBusSleepModeNotification	ComM	Notification that the network management has entered Prepare Bus-Sleep Mode.
ComM_Nm_BusSleepModeNotification	ComM	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.

9 Sequence diagrams

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

Use Case 01 – NM Initialization

Sequence in [4] shows how to initialize the Network Management modules. The ComManager calls Nm_Init only once (and not each channel individually), and the latter calls all configured BusNm_Init functions.

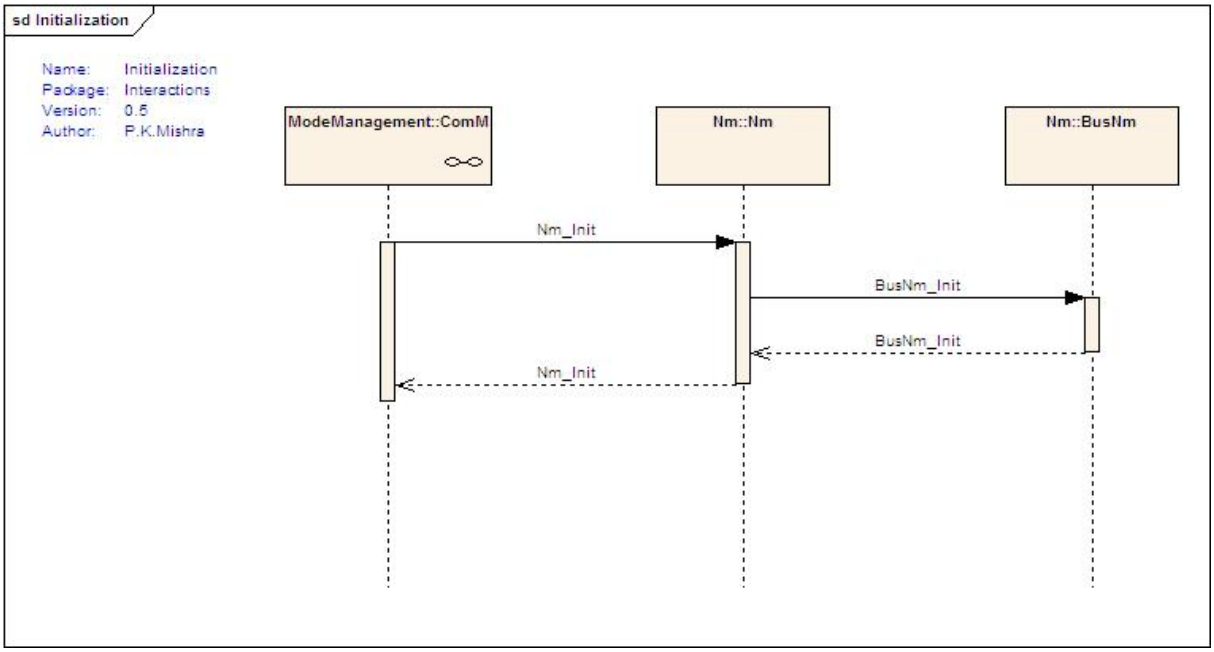


Figure 9.1

9.1 Use Case 02 – NM Network Request

Sequence in **Figure 9.2**. shows how to request activation of the network. This is an illustrative representation of how the NM Interface acts as a dispatcher of functions.

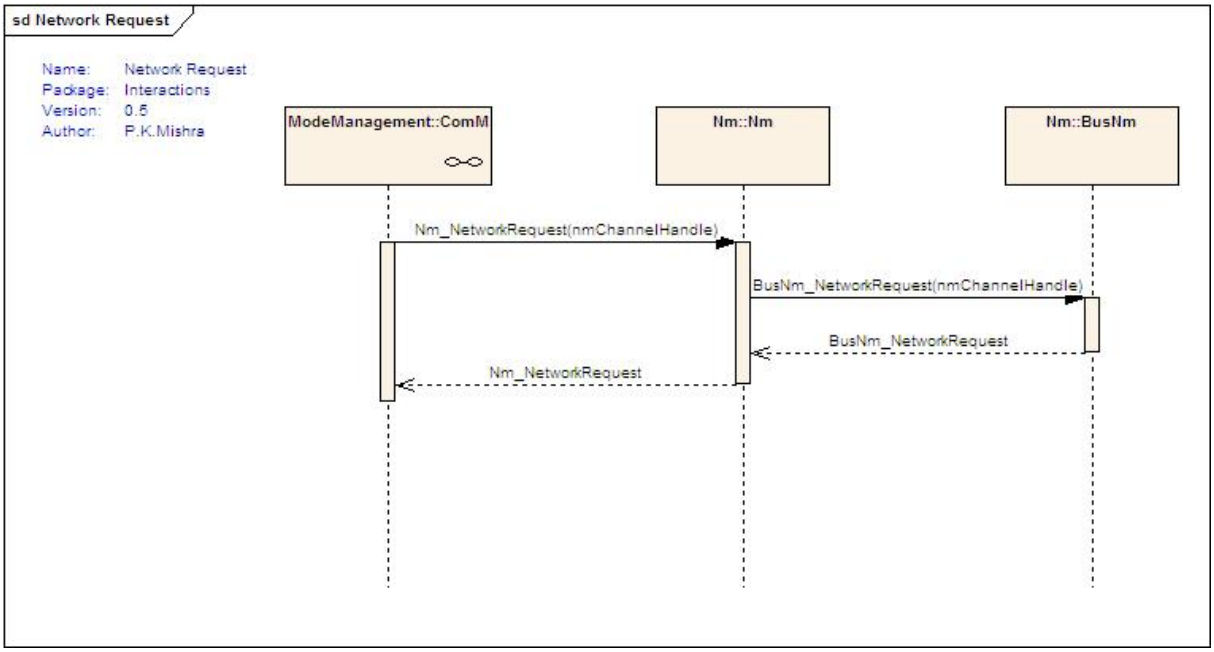


Figure 9.2

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.

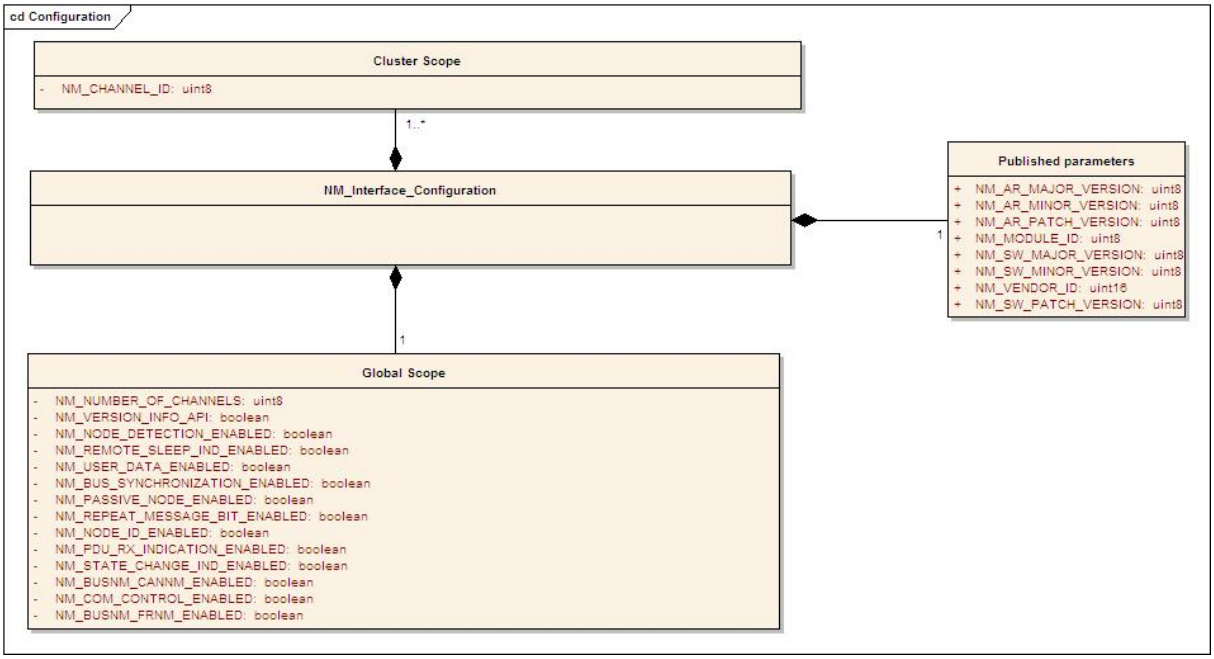


Figure 10.1

Figure 10.1 specifies all configurable parameters in this module.

10.2.1 Variants

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

10.2.2 Nm_GlobalConfig

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

SWS Item	--
Container Name	Nm_GlobalConfig
Description	This container contains all global configuration parameters of CAN Generic NM configured from the NM Module perspective.
Configuration Parameters	

Name	**NM_VERSION_INFO_API**		
Description	Pre-processor switch for enabling version info API support.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_NUMBER_OF_CHANNELS**		
Description	Number of NM channels allowed within one ECU.		
Type	Integer		
Unit	--		
Range	1..255		
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_USER_DATA_ENABLED**		
Description	Pre-processor switch for enabling user data support.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_NODE_DETECTION_ENABLED**		
Description	Pre-processor switch for enabling the Request Repeat Message Request support.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_REMOTE_SLEEP_IND_ENABLED**	
Description	Pre-processor switch for enabling remote sleep indication support.	
Type	Boolean	
Unit	--	
Range	TRUE	Option is enabled
	FALSE	Option is disabled

Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This feature is required for gateway nodes only. It must not be defined if **NM_PASSIVE_MODE_ENABLED** is defined.		

Name	**NM_BUS_SYNCHRONIZATION_ENABLED**		
Description	Pre-processor switch for enabling bus synchronization support.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This feature is required for gateway nodes only. It must not be defined if **NM_PASSIVE_MODE_ENABLED** is defined.		

Name	**NM_PASSIVE_MODE_ENABLED**		
Description	Pre-processor switch for enabling support of the Passive Mode.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_REPEAT_MESSAGE_BIT_ENABLED**		
Description	Pre-processor switch for enabling support for repeat message		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_NODE_ID_ENABLED**		
Description	Pre-processor switch for enabling the source node identifier.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**NM_PDU_RX_INDICATION_ENABLED**		
Description	Pre-processor switch for enabling the PDU Rx Indication.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	The `CNM_PDU_RX_INDICATION_ENABLED` shall be derived from this parameter.		

Name	**NM_STATE_CHANGE_IND_ENABLED**		
Description	Pre-processor switch for enabling the CAN Generic NM state change notification.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	The parameter `CNM_STATE_CHANGE_IND_ENABLED` and `FRNM_STATE_CHANGE_IND_ENABLED` shall be derived from this parameter.		

Name	**NM_COM_CONTROL_ENABLED**		
Description	Pre-processor switch for enabling the Communication Control support.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This parameter `CNM_COM_CONTROL_ENABLED` shall be derived from this.		

Name	**NM_BUSNM_CANNM_ENABLED**		
Description	Pre-processor switch for enabling the CAN NM support.		
Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency			

Name	**NM_BUSNM_FRNM_ENABLED**
Description	Pre-processor switch for enabling the FlexRay NM support.

Type	Boolean		
Unit	--		
Range	TRUE	Option is enabled	
	FALSE	Option is disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency			

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.

NM080: 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.

SWS Item	NM080	
Information elements		
Information element name	***Type / Range***	***Information element description***
`NM_VENDOR_ID`	`#define, / uint16`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`NM_MODULE_ID`	`#define, / 0x1D`	Module ID of this module from Module List
`NM_AR_MAJOR_VERSION`	`#define, / uint8`	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`NM_AR_MINOR_VERSION`	`#define, / uint8`	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`NM_AR_PATCH_VERSION`	`#define, / uint8`	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`NM_SW_MAJOR_VERSION`	`#define, / uint8`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`NM_SW_MINOR_VERSION`	`#define, / uint8`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`NM_SW_PATCH_VERSION`	`#define, / uint8`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.