

|                                   |   |
|-----------------------------------|---|
| <b>Document Title</b>             | Specification of Network Management Interface |
| <b>Document Owner</b>             | AUTOSAR                                       |
| <b>Document Responsibility</b>    | AUTOSAR                                       |
| <b>Document Identification No</b> | 228   |

|                                 |                  |
|---------------------------------|------------------|
| <b>Document Status</b>          | published        |
| <b>Part of AUTOSAR Standard</b> | Classic Platform |
| <b>Part of Standard Release</b> | R20-11           |

| <b>Document Change History</b> |                |                            |  |
|--------------------------------|----------------|----------------------------|--|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>          | <b>Description</b>   |
| 2020-11-30                     | R20-11         | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Support for synchronized PNC shutdown functionality</li> <li>• Introduction of Dynamic PNC-to-channel-mapping and PNC Learning algorithm</li> <li>• Added limitation regarding Nm Coordinator functionality when using 10BASE-T1S in combination with PLCA media access</li> <li>• Support for passing NM state change to SwC</li> <li>• Editorial changes</li> </ul> |
| 2019-11-28                     | R19-11         | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Minor changes</li> <li>• Multicore Distribution support (draft) added</li> <li>• Changed Document Status from Final to published</li> </ul>   |
| 2018-10-31                     | 4.4.0          | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Removed LinNM from the architecture</li> <li>• Removed obsolete elements</li> <li>• Header File Cleanup</li> </ul>  |
| 2017-12-08                     | 4.3.1          | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>  |

|            |       |                            |  |
|------------|-------|----------------------------|--|
| 2016-11-30 | 4.3.0 | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Add functionality for synchronizing channel A and channel B</li> <li>• removed dependencies of ComMChannels to each other in respect to NMVariants</li> <li>• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>  |
| 2015-07-31 | 4.2.2 | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• "Coordination algorithm" and "Coordinated shutdown" redefined</li> <li>• Make the CarWakeup feature available</li> <li>• Debugging support marked as obsolete</li> <li>• Editorial changes</li> </ul>   |
| 2014-10-31 | 4.2.1 | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Corrections on the requirement tracing</li> <li>• Clarification at use of callback versus callout</li> <li>• Editorial changes</li> </ul>   |
| 2014-03-31 | 4.1.3 | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Rework of wakeup and abortion of coordinated shutdown</li> <li>• Rework of coordination of nested sub-busses</li> </ul>   |
| 2013-10-31 | 4.1.2 | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>• Remove DEM usage</li> <li>• Correct multiplicity and dependency of configuration parameter</li> <li>• Corrections on RemoteSleepIndication feature</li> <li>• Corrections on MainFunction and coordinated shutdown</li> <li>• Formal correction on REQ Tags</li> <li>• Editorial changes</li> <li>• Removed chapter(s) on change documentation</li> </ul> |

|            |       |                        |  |
|------------|-------|------------------------|--|
| 2013-03-15 | 4.1.1 | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Introduction J1939Nm</li> <li>• Merged and corrected calculation of delay timer for Coordination Algorithm</li> <li>• Correction of parametrization and Services for Coordinator Synchronization Algorithm</li> <li>• Moved Nm_Passive_Mode_Enabled Parameter back to global container</li> </ul>   |
| 2011-12-22 | 4.0.3 | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• NmMultipleChannelsEnabled removed</li> <li>• Added Mandatory Interfaces provided by ComM to Chapter 8.6.1</li> <li>• move NmPassiveMode</li> <li>• Enabled form global configuration to channel configuration</li> <li>• Removed Nm_ReturnType</li> <li>• Fixed some min and max values of FloatParamDef configuration parameters</li> <li>• Added support of NmCarWakup-Feature</li> <li>• Added support of coordinated shutdown of nested sub-busses</li> </ul> |
| 2009-12-18 | 4.0.1 | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Release check added</li> <li>• DET Error Code for false Pointer added</li> <li>• ChannelID harmonized in COM-Stack</li> <li>• Nm-State-changes in Userdata via NmIf</li> </ul>  |
| 2010-02-02 | 3.1.4 | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Remove explicit support for OSEK NM from specification</li> <li>• NM Coordinator functionality reworked (chapter 7.2 and 7.2.4)</li> <li>• Debugging functionality added</li> <li>• Link time configuration variant introduced</li> <li>• Legal disclaimer revised</li> </ul>   |
| 2008-08-13 | 3.1.1 | AUTOSAR Administration | Legal disclaimer revised   |
| 2007-12-21 | 3.0.1 | AUTOSAR Administration | Initial release  |

## Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

|         |   |    |
|---------|---|----|
| 1       | Introduction and functional overview                                      | 8  |
| 2       | Acronyms and Abbreviations  | 9  |
| 3       | Related documentation   | 11 |
| 3.1     | Input documents   | 11 |
| 3.2     | Related specification   | 11 |
| 4       | Constraints and assumptions   | 12 |
| 4.1     | Limitations   | 12 |
| 4.2     | Specific limitations of the current release                               | 12 |
| 4.3     | Applicability to automotive domains                                       | 13 |
| 5       | Dependencies to other modules   | 14 |
| 5.1     | Interfaces to modules   | 14 |
| 5.1.1   | ComM, CanNm, J1939Nm, FrNm, UdpNm, generic bus specific NM layers and CDD | 16 |
| 5.1.2   | Error handling modules  | 16 |
| 5.1.3   | BSW Scheduler   | 16 |
| 5.2     | File structure  | 17 |
| 5.2.1   | Code file structure   | 17 |
| 5.2.2   | Header file structure   | 17 |
| 6       | Requirements traceability   | 18 |
| 7       | Functional specification  | 38 |
| 7.1     | Base functionality  | 38 |
| 7.2     | NM Coordinator functionality  | 38 |
| 7.2.1   | Applicability of the NM Coordinator functionality                         | 39 |
| 7.2.2   | Keeping coordinated busses alive  | 40 |
| 7.2.3   | Shutdown of coordinated busses  | 41 |
| 7.2.4   | Coordination of nested sub-busses   | 43 |
| 7.2.5   | Calculation of shutdown timers  | 47 |
| 7.2.6   | Synchronization Use Case 1 - Synchronous command                          | 47 |
| 7.2.7   | Synchronization Use Case 2 - Synchronous initiation                       | 48 |
| 7.2.8   | Synchronization Use Case 3 - Synchronous network sleep                    | 48 |
| 7.2.8.1 | Examples  | 49 |
| 7.3     | Wakeup and abortion of the coordinated shutdown                           | 51 |
| 7.3.1   | External network wakeup   | 51 |
| 7.3.2   | Coordinated wakeup  | 51 |
| 7.3.3   | Abortion of the coordinated shutdown                                      | 51 |
| 7.4     | Synchronized PNC shutdown functionality                                   | 53 |
| 7.5     | Prerequisites of bus specific Network Management modules                  | 54 |
| 7.5.1   | Prerequisites for basic functionality                                     | 55 |
| 7.5.2   | Prerequisites for NM Coordinator functionality                            | 56 |

|          |  |    |
|----------|--|----|
| 7.5.3    | Prerequisites for synchronized PNC shutdown functionality    | 58 |
| 7.5.4    | Configuration of global parameters for bus specific networks | 58 |
| 7.6      | NM_BUSNM_LOCALNM   | 59 |
| 7.7      | Multicore Distribution                                       | 59 |
| 7.8      | Additional Functionality                                     | 59 |
| 7.8.1    | Nm_CarWakeUpIndication                                       | 59 |
| 7.8.2    | Nm_StateChangeNotification                                   | 60 |
| 7.9      | Error classification   | 61 |
| 7.9.1    | Development Errors   | 61 |
| 7.9.2    | Runtime Errors   | 61 |
| 7.9.3    | Transient Faults   | 61 |
| 7.9.4    | Production Errors  | 61 |
| 7.9.5    | Extended Production Errors                                   | 61 |
| 8        | API specification  | 62 |
| 8.1      | Imported types   | 62 |
| 8.2      | Type definitions   | 62 |
| 8.2.1    | Nm_ModeType  | 62 |
| 8.2.2    | Nm_StateType   | 62 |
| 8.2.3    | Nm_BusNmType   | 63 |
| 8.2.4    | Nm_ConfigType  | 63 |
| 8.3      | Function definitions   | 64 |
| 8.3.1    | Standard services provided by NM Interface                   | 64 |
| 8.3.1.1  | Nm_Init  | 64 |
| 8.3.1.2  | Nm_PassiveStartup  | 65 |
| 8.3.1.3  | Nm_NetworkRequest  | 65 |
| 8.3.1.4  | Nm_NetworkRelease  | 66 |
| 8.3.2    | Communication control services provided by NM Interface      | 67 |
| 8.3.2.1  | Nm_DisableCommunication                                      | 67 |
| 8.3.2.2  | Nm_EnableCommunication                                       | 68 |
| 8.3.3    | Partial Network services provided by NM Interface            | 69 |
| 8.3.3.1  | Nm_RequestSynchronizedPncShutdown                            | 69 |
| 8.3.4    | Extra services provided by NM Interface                      | 70 |
| 8.3.4.1  | Nm_SetUserData   | 70 |
| 8.3.4.2  | Nm_GetUserData   | 71 |
| 8.3.4.3  | Nm_GetPduData  | 72 |
| 8.3.4.4  | Nm_RepeatMessageRequest                                      | 73 |
| 8.3.4.5  | Nm_GetNodeIdentifier   | 74 |
| 8.3.4.6  | Nm_GetLocalNodeIdentifier                                    | 74 |
| 8.3.4.7  | Nm_CheckRemoteSleepIndication                                | 75 |
| 8.3.4.8  | Nm_GetState  | 76 |
| 8.3.4.9  | Nm_GetVersionInfo  | 77 |
| 8.3.4.10 | Nm_PnLearningRequest   | 77 |
| 8.4      | Call-back notifications                                      | 78 |
| 8.4.1    | Standard Call-back notifications                             | 79 |
| 8.4.1.1  | Nm_NetworkStartIndication                                    | 79 |

|          |   |     |
|----------|---|-----|
| 8.4.1.2  | Nm_NetworkMode . . . . .                      | 79  |
| 8.4.1.3  | Nm_BusSleepMode . . . . .                     | 80  |
| 8.4.1.4  | Nm_PrepareBusSleepMode . . . . .              | 80  |
| 8.4.1.5  | NM_SynchronizeMode . . . . .                  | 81  |
| 8.4.1.6  | Nm_RemoteSleepIndication . . . . .            | 81  |
| 8.4.1.7  | Nm_RemoteSleepCancellation . . . . .          | 82  |
| 8.4.1.8  | Nm_SynchronizationPoint . . . . .             | 83  |
| 8.4.1.9  | Nm_CoordReadyToSleepIndication . . . . .      | 83  |
| 8.4.1.10 | Nm_CoordReadyToSleepCancellation . . . . .    | 84  |
| 8.4.1.11 | Nm_ForwardSynchronizedPncShutdown . . . . .   | 85  |
| 8.4.2    | Extra Call-back notifications . . . . .       | 85  |
| 8.4.2.1  | Nm_PduRxIndication . . . . .                  | 85  |
| 8.4.2.2  | Nm_StateChangeNotification . . . . .          | 86  |
| 8.4.2.3  | Nm_RepeatMessageIndication . . . . .          | 87  |
| 8.4.2.4  | Nm_TxTimeoutException . . . . .               | 87  |
| 8.4.2.5  | Nm_CarWakeUpIndication . . . . .              | 88  |
| 8.5      | Scheduled functions . . . . .                 | 88  |
| 8.5.1    | Nm_MainFunction . . . . .                     | 89  |
| 8.6      | Expected interfaces . . . . .                 | 89  |
| 8.6.1    | Mandatory Interfaces . . . . .                | 89  |
| 8.6.2    | Optional Interfaces . . . . .                 | 90  |
| 8.6.3    | Configurable Interfaces . . . . .             | 92  |
| 8.6.3.1  | NmCarWakeUpCallout . . . . .                  | 92  |
| 8.7      | Version Check . . . . .                       | 92  |
| 9        | Sequence diagrams . . . . .                   | 93  |
| 9.1      | Basic functionality . . . . .                 | 93  |
| 9.2      | Seq of NM Coordinator functionality . . . . . | 93  |
| 10       | Configuration specification . . . . .         | 95  |
| 10.1     | How to read this chapter . . . . .            | 95  |
| 10.2     | Configuration parameters . . . . .            | 95  |
| 10.2.1   | Nm . . . . .                                  | 96  |
| 10.3     | Global configurable parameters . . . . .      | 96  |
| 10.3.1   | NmGlobalConfig . . . . .                      | 96  |
| 10.3.2   | NmGlobalConstants . . . . .                   | 98  |
| 10.3.3   | NmGlobalProperties . . . . .                  | 99  |
| 10.3.4   | NmGlobalFeatures . . . . .                    | 100 |
| 10.4     | Channel configurable parameters . . . . .     | 105 |
| 10.4.1   | NmChannelConfig . . . . .                     | 105 |
| 10.4.2   | NmBusType . . . . .                           | 111 |
| 10.4.3   | NmGenericBusNmConfig . . . . .                | 112 |
| 10.4.4   | NmStandardBusNmConfig . . . . .               | 113 |
| 10.5     | Published Information . . . . .               | 113 |
| A        | Not applicable requirements . . . . .         | 114 |

# 1 Introduction and functional overview

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

The **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 and FlexRay Network Management). This is also referred to as Basic functionality.

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 is also referred to as *NM Coordinator functionality*.

Support of the *NM Coordinator functionality* is optional. A **Network Management Interface** implementation can either support only Basic functionality or both Basic functionality and NM Coordinator functionality.

The **Network Management Interface** is constructed to support generic lower layer modules that follow a fixed set of requirement for bus specific NM modules. This will allow third parties to offer support for OEM specific or legacy NM protocols such as direct OSEK NM.



## 2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations and terms relevant to the Network Management Interface module that are not included in the [1, AUTOSAR glossary].

| Abbreviation / Acronym: | Description:   |
|-------------------------|--|
| CanIf                   | CAN Interface module   |
| CanNm                   | CAN Network Management module  |
| CC                      | Communication controller   |
| ComM                    | Communication Manager module   |
| EcuM                    | ECU State Manager module   |
| DEM                     | Diagnostic Event Manager module  |
| Nm                      | Generic Network Management Interface module, this is the abbreviation used for this module throughout this specification |
| NM                      | Network Management   |
| OEM                     | Original Equipment Manufacturer  |
| CBV                     | Control Bit Vector in NM-message   |
| PNC                     | Partial network cluster  |

| Terms:                 | 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 the use of the NM algorithm.  |
| NM-Coordinator         | A functionality of the Nm which allows coordination of network sleep for multiple NM Channels.         |
| 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.   |
| Bus                    | Physical communication medium to which a NM node/ecu is connected to.                                  |
| network                | Entity of all NM nodes/ecus which are connected to the same bus.                                       |
| channel                | Logical bus to which the NM node/ecu is connected to.  |
| Coordinated shutdown   | Shutdown of two or more busses in a way that their shutdown is finished coinciding.                    |
| Coordination algorithm | Initiation of coordinated shutdown in case all conditions are met.                                     |
| PN info                | Represent the Partial Network information in a NM frame  |
| PN info range          | Represent the length of a Partial Network information in bytes   |
| PN bit                 | One bit which represent a particular Partial Network in the Partial Network Info Range                 |

| Terms:                       | Definition:   |
|------------------------------|---|
| Top-level PNC coordinator    | The top-level PNC coordinator is an ECU that acts as PNC gateway in the network and that handles at least one PNC as actively coordinated on all assigned channels. If synchronized PNC shutdown is enabled, the top-level PNC coordinator triggers for these PNCs the shutdown, if no other ECU in the network request them.   |
| Intermediate PNC coordinator | An intermediate PNC coordinator is an ECU that acts as PNC gateway in the network and that handles at least one PNC as passively coordinated on at least one assigned channel. If synchronized PNC shutdown is enabled, it forwards received shutdown requests for these PNCs to the corresponding actively coordinated channels and starts their shutdown accordingly.   |
| Subordinated PNC node        | A subordinated PNC node is an ECU that act neither as top-level PNC coordinator nor as an intermediate PNC coordinator. It act as an ECU without a PNC gateway in the network and process PN shutdown message as usual NM messages.   |
| PN shutdown request message  | <p>A top-level PNC coordinator transmits PN shutdown messages to indicate a synchronized PNC shutdown across the PN topology. A PN shutdown message is a NM message which has PNSR bit in the control bit vector and all PNCs which are indicated for a synchronized shutdown set to '1'. An intermediate PNC coordinator which receive a PN shutdown message and forward the PN information as PN shutdown message on the affected channels.</p> <p>Note: An intermediate PNC coordinators has to forward the PN information of received PN shutdown message as fast as possible to ensure a nearly synchronized shutdown of the affected PNCs across the PN topology.</p> |

## 3 Related documentation

### 3.1 Input documents

- [1] Glossary  
AUTOSAR\_TR\_Glossary
- [2] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral
- [3] Specification of CAN Network Management  
AUTOSAR\_SWS\_CANNetworkManagement
- [4] Specification of FlexRay Network Management  
AUTOSAR\_SWS\_FlexRayNetworkManagement
- [5] Specification of UDP Network Management  
AUTOSAR\_SWS\_UDPNetworkManagement
- [6] Specification of Network Management for SAE J1939  
AUTOSAR\_SWS\_SAEJ1939NetworkManagement
- [7] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral
- [8] Requirements on Network Management  
AUTOSAR\_SRS\_NetworkManagement
- [9] Specification of Communication Manager  
AUTOSAR\_SWS\_COMManager
- [10] Guide to BSW Distribution  
AUTOSAR\_EXP\_BSWDistributionGuide

### 3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for the Generic Network Management Interface.

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

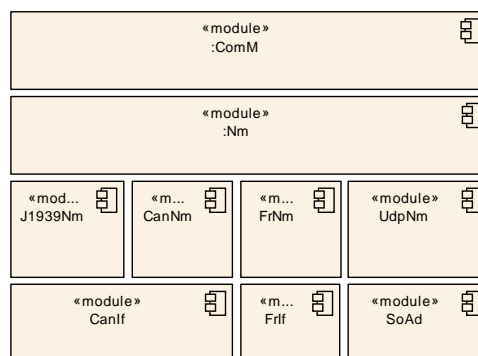
## 4 Constraints and assumptions

### 4.1 Limitations

1. The Generic Network Management Interface can only be applied to communication systems that support broadcast communication and 'bus-sleep mode'.
2. There is only one instance of the Generic Network Management Interface layer for all NM-Clusters. This instance manages all channels where a NM is used.
3. The Generic Network Management Interface shall only include the common modes, definitions and return values of different bus specific NM layers.
4. The Generic Network Management Interface shall only include the common modes, definitions and return values of different bus specific NM layers.
5. If 10BASE-T1S is used in combination with PLCA media access, then Nm Coordinator functionality are not supported.

**Note:** Consequently, the configuration parameter `NmCoordinatorSupportEnabled` shall be set to false.

Figure 4.1 shows a typical example of the AUTOSAR NM stack.



**Figure 4.1: Nm stack modules**

### 4.2 Specific limitations of the current release

The following limitations reflect desired functionality that has yet not been implemented or agreed upon, but might be added for future releases:

- No support of a back-up coordinator ECU (fault tolerance).

Also; explicit support for OSEK NM has been completely removed from this specification as of AUTOSAR Release 4.0. OSEK NM can still be supported by extending the CanNm or by introducing a Complex Driver (CDD) on BusNm level as a generic BusNm. Supporting the OSEK NM through a CDD is not specified by AUTOSAR.

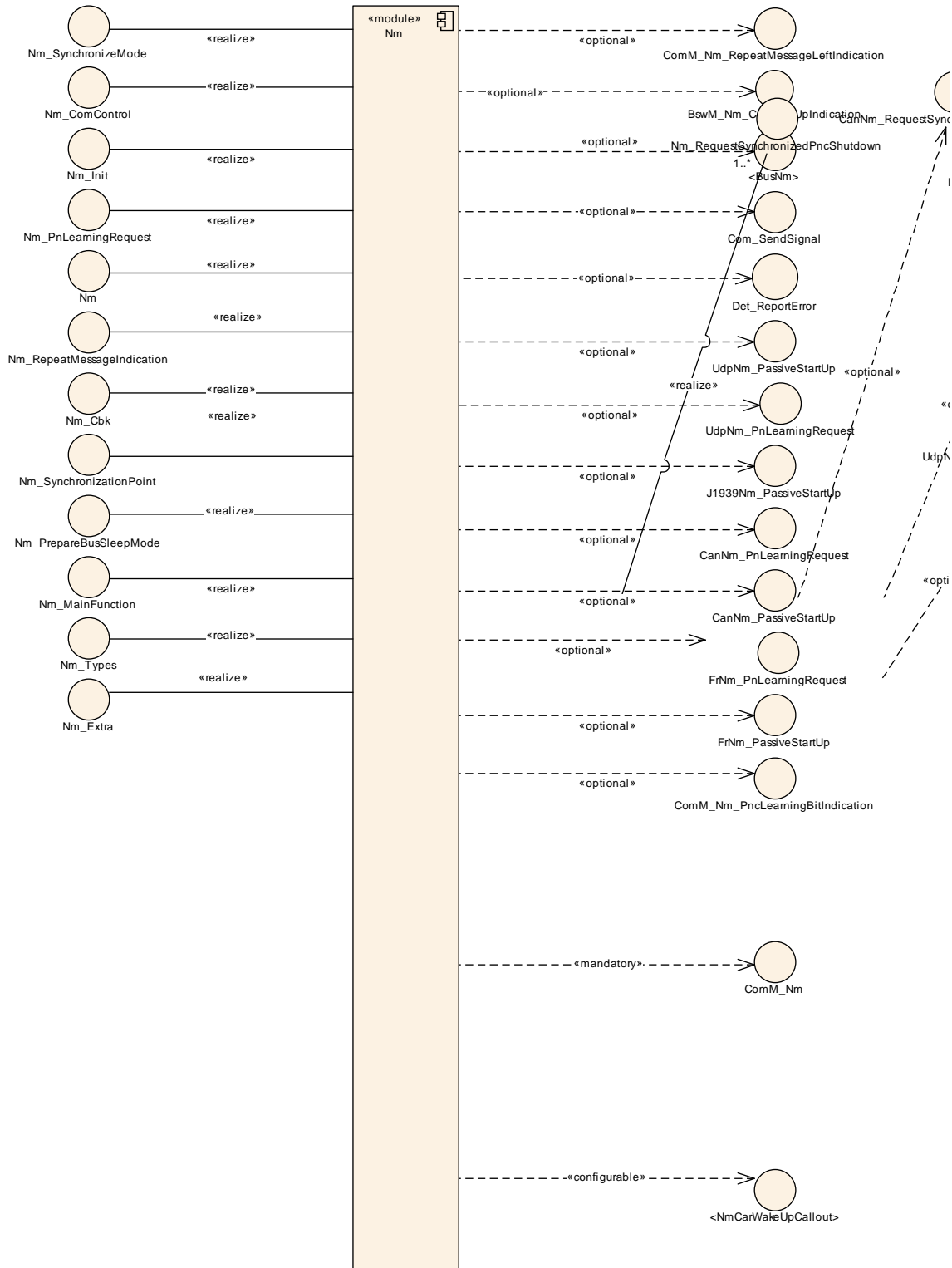
### **4.3 Applicability to automotive domains**

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

## 5 Dependencies to other modules

### 5.1 Interfaces to modules

Figure [5.1](#) shows the interfaces provided to and required from other modules in the AUTOSAR BSW.



**Figure 5.1: Nm's interfaces to other modules**

### 5.1.1 ComM, CanNm, J1939Nm, FrNm, UdpNm, generic bus specific NM layers and CDD

The Generic Network Management Interface module (**Nm**) provides services to the Communication Manager (**ComM**) and uses services of the bus specific Network Management modules:

- CAN Network Management ([3, **CanNm**])
- FlexRay Network Management ([4, **FrNm**])
- Ethernet Network Management ([5, **UdpNm**]).
- J1939 Network Management ([6, **J1939Nm**]).

For Buses which do not need to provide Network Management Information on the bus like for example a LIN-bus the Bus-Type can be configured as "local Nm". With respect to callbacks, the **Nm** provides notification callbacks to the bus specific Network Management modules and calls the notification callbacks provided by the **ComM**.

In addition to the official AUTOSAR NM-modules above, Nm also support generic bus specific NM layers (<**BusNm**>). Any component which implements the required provided interfaces and uses the provided callback functions of Nm can be used as a bus specific NM. See [section 7.5](#) for the prerequisites for a generic bus specific NM.

**Rationale:** Nm is specified to support generic bus specific NM layers by adding generic lower layer modules as Complex Drivers. As such, Nm does not explicitly use the services by the official AUTOSAR bus-NM modules (CanNm, FrNm and UdpNm), but rather the services of the generic <BusNm>. The AUTOSAR bus-NMs are then explicitly supported since they implement the interfaces of <BusNm>.

The optional CarWakeUp-Functionality needs a Complex Driver which Coordinates Basic Software Mode Management.

### 5.1.2 Error handling modules

Nm reports development errors to the Default Error Tracer according to [\[SWS\\_Nm\\_00232\]](#).

### 5.1.3 BSW Scheduler

In case of the NM Coordinator functionality and depending on the configuration, the Nm will need cyclic invocation of it's main scheduling function in order to evaluate and detect when timers have expired.



## 5.2 File structure

### 5.2.1 Code file structure

**[SWS\_Nm\_00247]** [The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

]([SRS\\_BSW\\_00159](#), [SRS\\_BSW\\_00345](#), [SRS\\_BSW\\_00419](#))

### 5.2.2 Header file structure

**[SWS\_Nm\_00124]** [The following header files shall be included by 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.
- ComM\_Nm.h (for Communication Manager callback functions)
- BswM\_Nm.h (If the BswM is used for CarWakeup-functionality)

]([SRS\\_BSW\\_00348](#), [SRS\\_BSW\\_00353](#), [SRS\\_BSW\\_00357](#), [SRS\\_BSW\\_00381](#), [SRS\\_BSW\\_00384](#), [SRS\\_BSW\\_00412](#))

**[SWS\_Nm\_00243]** [The Nm Interface shall optionally include the header file of Default Error Tracer (depending on the pre-processor switch NmDevErrorDetect, see ECUC\_Nm\_00203).

- Det.h for service of the Default Error Tracer.

]([SRS\\_BSW\\_00171](#), [SRS\\_BSW\\_00301](#), [SRS\\_BSW\\_00384](#))

## 6 Requirements traceability

The following tables references the requirements specified in [7] as well as [8] and links to the fulfillment of these.

| Requirement     | Description   | Satisfied by  |
|-----------------|---|---|
| [SRS_BSW_00003] | All software modules shall provide version and identification information   | [SWS_Nm_00044]  |
| [SRS_BSW_00004] | All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files                        | [SWS_Nm_00999]  |
| [SRS_BSW_00005] | Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces                                 | [SWS_Nm_00999]  |
| [SRS_BSW_00006] | The source code of software modules above the $\mu$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent. | [SWS_Nm_00999]  |
| [SRS_BSW_00007] | All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.  | [SWS_Nm_00999]  |
| [SRS_BSW_00009] | All Basic SW Modules shall be documented according to a common standard.  | [SWS_Nm_00999]  |
| [SRS_BSW_00010] | The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.   | [SWS_Nm_00999]  |
| [SRS_BSW_00101] | The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function            | [SWS_Nm_00030] [SWS_Nm_00127]<br>[SWS_Nm_00128] [SWS_Nm_00129]<br>[SWS_Nm_00131] [SWS_Nm_00133]<br>[SWS_Nm_00135] [SWS_Nm_00137]<br>[SWS_Nm_00139] [SWS_Nm_00141]<br>[SWS_Nm_00143] [SWS_Nm_00145]<br>[SWS_Nm_00147] [SWS_Nm_00149]<br>[SWS_Nm_00151] |
| [SRS_BSW_00158] | No description  | [SWS_Nm_00999]  |

| Requirement     | Description  | Satisfied by   |
|-----------------|--|----------------|
| [SRS_BSW_00159] | All modules of the AUTOSAR Basic Software shall support a tool based configuration   | [SWS_Nm_00247] |
| [SRS_BSW_00160] | Configuration files of AUTOSAR Basic SW module shall be readable for human beings  | [SWS_Nm_00999] |
| [SRS_BSW_00161] | The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers | [SWS_Nm_00999] |
| [SRS_BSW_00162] | The AUTOSAR Basic Software shall provide a hardware abstraction layer  | [SWS_Nm_00999] |
| [SRS_BSW_00164] | The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules                             | [SWS_Nm_00999] |
| [SRS_BSW_00167] | All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks                             | [SWS_Nm_00999] |
| [SRS_BSW_00168] | SW components shall be tested by a function defined in a common API in the Basis-SW  | [SWS_Nm_00999] |
| [SRS_BSW_00170] | The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands                       | [SWS_Nm_00999] |
| [SRS_BSW_00171] | Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time                       | [SWS_Nm_00243] |

| Requirement     | Description   | Satisfied by                  |
|-----------------|---|-------------------------------|
| [SRS_BSW_00172] | The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system    | [SWS_Nm_00999]                |
| [SRS_BSW_00300] | All AUTOSAR Basic Software Modules shall be identified by an unambiguous name   | [SWS_Nm_00999]                |
| [SRS_BSW_00301] | All AUTOSAR Basic Software Modules shall only import the necessary information  | [SWS_Nm_00117] [SWS_Nm_00243] |
| [SRS_BSW_00302] | All AUTOSAR Basic Software Modules shall only export information needed by other modules  | [SWS_Nm_00999]                |
| [SRS_BSW_00304] | All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types                                | [SWS_Nm_00999]                |
| [SRS_BSW_00305] | Data types naming convention  | [SWS_Nm_00999]                |
| [SRS_BSW_00306] | AUTOSAR Basic Software Modules shall be compiler and platform independent   | [SWS_Nm_00999]                |
| [SRS_BSW_00307] | Global variables naming convention  | [SWS_Nm_00999]                |
| [SRS_BSW_00308] | AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file                                | [SWS_Nm_00999]                |
| [SRS_BSW_00309] | All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword | [SWS_Nm_00999]                |
| [SRS_BSW_00310] | API naming convention   | [SWS_Nm_00999]                |
| [SRS_BSW_00312] | Shared code shall be reentrant  | [SWS_Nm_00999]                |

| Requirement     | Description   | Satisfied by  |
|-----------------|---|---|
| [SRS_BSW_00314] | All internal driver modules shall separate the interrupt frame definition from the service routine                | [SWS_Nm_00999]  |
| [SRS_BSW_00318] | Each AUTOSAR Basic Software Module file shall provide version numbers in the header file                          | [SWS_Nm_00999]  |
| [SRS_BSW_00321] | The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules                | [SWS_Nm_00999]  |
| [SRS_BSW_00323] | All AUTOSAR Basic Software Modules shall check passed API parameters for validity                                 | [SWS_Nm_00488] [SWS_Nm_00489]<br>[SWS_Nm_00490] [SWS_Nm_00491]<br>[SWS_Nm_00492] [SWS_Nm_00493]<br>[SWS_Nm_00494] [SWS_Nm_00495]<br>[SWS_Nm_00496] [SWS_Nm_00497]<br>[SWS_Nm_00498] [SWS_Nm_00499]<br>[SWS_Nm_00500] [SWS_Nm_00505]<br>[SWS_Nm_00508] |
| [SRS_BSW_00325] | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short | [SWS_Nm_00999]  |
| [SRS_BSW_00327] | Error values naming convention  | [SWS_Nm_00232]  |
| [SRS_BSW_00328] | All AUTOSAR Basic Software Modules shall avoid the duplication of code  | [SWS_Nm_00999]  |
| [SRS_BSW_00330] | It shall be allowed to use macros instead of functions where source code is used and runtime is critical          | [SWS_Nm_00091]  |
| [SRS_BSW_00331] | All Basic Software Modules shall strictly separate error and status information                                   | [SWS_Nm_00999]  |
| [SRS_BSW_00333] | For each callback function it shall be specified if it is called from interrupt context or not                    | [SWS_Nm_00028]  |

| Requirement     | Description  | Satisfied by   |
|-----------------|--|----------------|
| [SRS_BSW_00334] | All Basic Software Modules shall provide an XML file that contains the meta data   | [SWS_Nm_00999] |
| [SRS_BSW_00335] | Status values naming convention  | [SWS_Nm_00999] |
| [SRS_BSW_00336] | Basic SW module shall be able to shutdown  | [SWS_Nm_00999] |
| [SRS_BSW_00337] | Classification of development errors   | [SWS_Nm_00232] |
| [SRS_BSW_00339] | Reporting of production relevant error status  | [SWS_Nm_00999] |
| [SRS_BSW_00341] | Module documentation shall contains all needed informations  | [SWS_Nm_00999] |
| [SRS_BSW_00342] | It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed | [SWS_Nm_00999] |
| [SRS_BSW_00343] | The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit                   | [SWS_Nm_00999] |
| [SRS_BSW_00344] | BSW Modules shall support link-time configuration  | [SWS_Nm_00030] |
| [SRS_BSW_00345] | BSW Modules shall support pre-compile configuration  | [SWS_Nm_00247] |
| [SRS_BSW_00346] | All AUTOSAR Basic Software Modules shall provide at least a basic set of module files  | [SWS_Nm_00999] |
| [SRS_BSW_00347] | A Naming separation of different instances of BSW drivers shall be in place  | [SWS_Nm_00999] |
| [SRS_BSW_00348] | All AUTOSAR standard types and constants shall be placed and organized in a standard type header file                                | [SWS_Nm_00124] |

| Requirement     | Description   | Satisfied by  |
|-----------------|---|---|
| [SRS_BSW_00350] | All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.                           | [SWS_Nm_00999]  |
| [SRS_BSW_00351] | Encapsulation of compiler specific methods to map objects   | [SWS_Nm_00999]  |
| [SRS_BSW_00353] | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header                          | [SWS_Nm_00124]  |
| [SRS_BSW_00357] | For success/failure of an API call a standard return type shall be defined  | [SWS_Nm_00124]  |
| [SRS_BSW_00358] | The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void   | [SWS_Nm_00030]  |
| [SRS_BSW_00359] | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible  | [SWS_Nm_00112] [SWS_Nm_00114]<br>[SWS_Nm_00154] [SWS_Nm_00156]<br>[SWS_Nm_00159] [SWS_Nm_00162]<br>[SWS_Nm_00192] [SWS_Nm_00193]<br>[SWS_Nm_00194] [SWS_Nm_00230]<br>[SWS_Nm_00234] [SWS_Nm_00250]<br>[SWS_Nm_00254] [SWS_Nm_00272] |
| [SRS_BSW_00360] | AUTOSAR Basic Software Modules callback functions are allowed to have parameters  | [SWS_Nm_00999]  |
| [SRS_BSW_00361] | All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header | [SWS_Nm_00999]  |

| Requirement     | Description  | Satisfied by  |
|-----------------|--|---|
| [SRS_BSW_00369] | All AUTOSAR Basic Software Modules shall not return specific development error codes via the API   | [SWS_Nm_00488] [SWS_Nm_00489]<br>[SWS_Nm_00490] [SWS_Nm_00491]<br>[SWS_Nm_00492] [SWS_Nm_00493]<br>[SWS_Nm_00494] [SWS_Nm_00495]<br>[SWS_Nm_00496] [SWS_Nm_00497]<br>[SWS_Nm_00498] [SWS_Nm_00499]<br>[SWS_Nm_00500] [SWS_Nm_00505]<br>[SWS_Nm_00508] |
| [SRS_BSW_00371] | The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules  | [SWS_Nm_00999]  |
| [SRS_BSW_00373] | The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention                             | [SWS_Nm_00020]  |
| [SRS_BSW_00374] | All Basic Software Modules shall provide a readable module vendor identification   | [SWS_Nm_00999]  |
| [SRS_BSW_00375] | Basic Software Modules shall report wake-up reasons  | [SWS_Nm_00999]  |
| [SRS_BSW_00377] | A Basic Software Module can return a module specific types   | [SWS_Nm_00999]  |
| [SRS_BSW_00378] | AUTOSAR shall provide a boolean type   | [SWS_Nm_00999]  |
| [SRS_BSW_00379] | All software modules shall provide a module identifier in the header file and in the module XML description file.                              | [SWS_Nm_00999]  |
| [SRS_BSW_00380] | Configuration parameters being stored in memory shall be placed into separate c-files  | [SWS_Nm_00999]  |
| [SRS_BSW_00381] | No description   | [SWS_Nm_00124]  |
| [SRS_BSW_00383] | The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description | [SWS_Nm_00999]  |



| Requirement     | Description  | Satisfied by   |
|-----------------|--|--|
| [SRS_BSW_00384] | The Basic Software Module specifications shall specify at least in the description which other modules they require  | [SWS_Nm_00124] [SWS_Nm_00243]  |
| [SRS_BSW_00385] | List possible error notifications  | [SWS_Nm_00232]   |
| [SRS_BSW_00386] | The BSW shall specify the configuration for detecting an error   | [SWS_Nm_00232] [SWS_Nm_00488]<br>[SWS_Nm_00489] [SWS_Nm_00490]<br>[SWS_Nm_00491] [SWS_Nm_00492]<br>[SWS_Nm_00493] [SWS_Nm_00494]<br>[SWS_Nm_00495] [SWS_Nm_00496]<br>[SWS_Nm_00497] [SWS_Nm_00498]<br>[SWS_Nm_00499] [SWS_Nm_00500]<br>[SWS_Nm_00505] [SWS_Nm_00508] |
| [SRS_BSW_00388] | Containers shall be used to group configuration parameters that are defined for the same object  | [SWS_Nm_00999]   |
| [SRS_BSW_00389] | Containers shall have names  | [SWS_Nm_00999]   |
| [SRS_BSW_00390] | Parameter content shall be unique within the module  | [SWS_Nm_00999]   |
| [SRS_BSW_00392] | Parameters shall have a type   | [SWS_Nm_00999]   |
| [SRS_BSW_00393] | Parameters shall have a range  | [SWS_Nm_00999]   |
| [SRS_BSW_00394] | The Basic Software Module specifications shall specify the scope of the configuration parameters   | [SWS_Nm_00999]   |
| [SRS_BSW_00395] | The Basic Software Module specifications shall list all configuration parameter dependencies   | [SWS_Nm_00999]   |
| [SRS_BSW_00396] | The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/container | [SWS_Nm_00999]   |

| Requirement     | Description  | Satisfied by   |
|-----------------|--|----------------|
| [SRS_BSW_00397] | The configuration parameters in pre-compile time are fixed before compilation starts   | [SWS_Nm_00999] |
| [SRS_BSW_00398] | The link-time configuration is achieved on object code basis in the stage after compiling and before linking   | [SWS_Nm_00999] |
| [SRS_BSW_00399] | Parameter-sets shall be located in a separate segment and shall be loaded after the code   | [SWS_Nm_00999] |
| [SRS_BSW_00400] | Parameter shall be selected from multiple sets of parameters after code has been loaded and started  | [SWS_Nm_00999] |
| [SRS_BSW_00401] | Documentation of multiple instances of configuration parameters shall be available   | [SWS_Nm_00999] |
| [SRS_BSW_00402] | Each module shall provide version information  | [SWS_Nm_00999] |
| [SRS_BSW_00403] | The Basic Software Module specifications shall specify for each parameter/container whether it supports different values or multiplicity in different configuration sets | [SWS_Nm_00999] |
| [SRS_BSW_00404] | BSW Modules shall support post-build configuration   | [SWS_Nm_00999] |
| [SRS_BSW_00405] | BSW Modules shall support multiple configuration sets  | [SWS_Nm_00030] |
| [SRS_BSW_00406] | A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called                           | [SWS_Nm_00999] |

| Requirement     | Description   | Satisfied by  |
|-----------------|---|---|
| [SRS_BSW_00407] | Each BSW module shall provide a function to read out the version information of a dedicated module implementation                         | [SWS_Nm_00044]  |
| [SRS_BSW_00408] | All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule                            | [SWS_Nm_00999]  |
| [SRS_BSW_00409] | All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration | [SWS_Nm_00999]  |
| [SRS_BSW_00410] | Compiler switches shall have defined values   | [SWS_Nm_00999]  |
| [SRS_BSW_00411] | All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API                              | [SWS_Nm_00999]  |
| [SRS_BSW_00412] | No description  | [SWS_Nm_00124]  |
| [SRS_BSW_00413] | An index-based accessing of the instances of BSW modules shall be done  | [SWS_Nm_00999]  |
| [SRS_BSW_00414] | Init functions shall have a pointer to a configuration structure as single parameter  | [SWS_Nm_00030] [SWS_Nm_00282]<br>[SWS_Nm_00283]   |
| [SRS_BSW_00415] | Interfaces which are provided exclusively for one module shall be separated into a dedicated header file                                  | [SWS_Nm_00999]  |
| [SRS_BSW_00416] | The sequence of modules to be initialized shall be configurable   | [SWS_Nm_00127] [SWS_Nm_00128]<br>[SWS_Nm_00129] [SWS_Nm_00131]<br>[SWS_Nm_00133] [SWS_Nm_00135]<br>[SWS_Nm_00137] [SWS_Nm_00139]<br>[SWS_Nm_00141] [SWS_Nm_00143]<br>[SWS_Nm_00145] [SWS_Nm_00147]<br>[SWS_Nm_00149] [SWS_Nm_00151]<br>[SWS_Nm_00999] |

| Requirement     | Description  | Satisfied by                  |
|-----------------|--|-------------------------------|
| [SRS_BSW_00417] | Software which is not part of the SW-C shall report error events only after the DEM is fully operational.              | [SWS_Nm_00999]                |
| [SRS_BSW_00419] | If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file     | [SWS_Nm_00247]                |
| [SRS_BSW_00422] | Pre-de-bouncing of error status information is done within the DEM   | [SWS_Nm_00999]                |
| [SRS_BSW_00423] | BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template                           | [SWS_Nm_00999]                |
| [SRS_BSW_00424] | BSW module main processing functions shall not be allowed to enter a wait state  | [SWS_Nm_00118] [SWS_Nm_00999] |
| [SRS_BSW_00425] | The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects | [SWS_Nm_00118]                |
| [SRS_BSW_00426] | BSW Modules shall ensure data consistency of data which is shared between BSW modules                                  | [SWS_Nm_00999]                |
| [SRS_BSW_00427] | ISR functions shall be defined and documented in the BSW module description template                                   | [SWS_Nm_00999]                |
| [SRS_BSW_00428] | A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence         | [SWS_Nm_00999]                |
| [SRS_BSW_00429] | Access to OS is restricted   | [SWS_Nm_00999]                |

| Requirement     | Description  | Satisfied by   |
|-----------------|--|----------------|
| [SRS_BSW_00432] | Modules should have separate main processing functions for read/receive and write/transmit data path                             | [SWS_Nm_00999] |
| [SRS_BSW_00433] | Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler                           | [SWS_Nm_00999] |
| [SRS_BSW_00437] | Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup               | [SWS_Nm_00999] |
| [SRS_BSW_00438] | Configuration data shall be defined in a structure   | [SWS_Nm_00999] |
| [SRS_BSW_00439] | Enable BSW modules to handle interrupts  | [SWS_Nm_00999] |
| [SRS_BSW_00440] | The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API | [SWS_Nm_00999] |
| [SRS_BSW_00441] | Naming convention for type, macro and function   | [SWS_Nm_00999] |
| [SRS_BSW_00447] | Standardizing Include file structure of BSW Modules<br>Implementing Autosar Service  | [SWS_Nm_00999] |
| [SRS_BSW_00448] | Module SWS shall not contain requirements from Other Modules   | [SWS_Nm_00999] |
| [SRS_BSW_00449] | BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType  | [SWS_Nm_00999] |
| [SRS_BSW_00450] | A Main function of a un-initialized module shall return immediately  | [SWS_Nm_00121] |

| Requirement     | Description  | Satisfied by   |
|-----------------|--|--|
| [SRS_BSW_00451] | Hardware registers shall be protected if concurrent access to these registers occur                    | [SWS_Nm_00999]   |
| [SRS_BSW_00452] | Classification of runtime errors   | [SWS_Nm_00999]   |
| [SRS_BSW_00453] | BSW Modules shall be harmonized  | [SWS_Nm_00999]   |
| [SRS_BSW_00454] | An alternative interface without a parameter of category DATA_REFERENCE shall be available.            | [SWS_Nm_00999]   |
| [SRS_BSW_00456] | A Header file shall be defined in order to harmonize BSW Modules                                       | [SWS_Nm_00999]   |
| [SRS_BSW_00457] | Callback functions of Application software components shall be invoked by the Basis SW                 | [SWS_Nm_00999]   |
| [SRS_BSW_00458] | Classification of production errors  | [SWS_Nm_00999]   |
| [SRS_BSW_00459] | It shall be possible to concurrently execute a service offered by a BSW module in different partitions | [SWS_Nm_00484] [SWS_Nm_00485]<br>[SWS_Nm_00486] [SWS_Nm_00999] |
| [SRS_BSW_00460] | Reentrancy Levels  | [SWS_Nm_00999]   |
| [SRS_BSW_00461] | Modules called by generic modules shall satisfy all interfaces requested by the generic module         | [SWS_Nm_00999]   |
| [SRS_BSW_00462] | All Standardized Autosar Interfaces shall have unique requirement Id / number                          | [SWS_Nm_00999]   |
| [SRS_BSW_00463] | Naming convention of callout prototypes  | [SWS_Nm_00999]   |
| [SRS_BSW_00464] | File names shall be considered case sensitive regardless of the filesystem in which they are used      | [SWS_Nm_00999]   |
| [SRS_BSW_00465] | It shall not be allowed to name any two files so that they only differ by the cases of their letters   | [SWS_Nm_00999]   |

| Requirement     | Description  | Satisfied by   |
|-----------------|--|----------------|
| [SRS_BSW_00466] | Classification of extended production errors   | [SWS_Nm_00999] |
| [SRS_BSW_00467] | The init / deinit services shall only be called by BswM or EcuM  | [SWS_Nm_00999] |
| [SRS_BSW_00469] | Fault detection and healing of production errors and extended production errors  | [SWS_Nm_00999] |
| [SRS_BSW_00470] | Execution frequency of production error detection  | [SWS_Nm_00999] |
| [SRS_BSW_00471] | Do not cause dead-locks on detection of production errors - the ability to heal from previously detected production errors | [SWS_Nm_00999] |
| [SRS_BSW_00472] | Avoid detection of two production errors with the same root cause.   | [SWS_Nm_00999] |
| [SRS_BSW_00473] | Classification of transient faults   | [SWS_Nm_00999] |
| [SRS_BSW_00477] | The functional interfaces of AUTOSAR BSW modules shall be specified in C90   | [SWS_Nm_00999] |
| [SRS_BSW_00478] | Timing limits of main functions  | [SWS_Nm_00292] |
| [SRS_BSW_00479] | Interfaces for handling request from external devices  | [SWS_Nm_00999] |
| [SRS_BSW_00480] | NullPointer Errors shall follow a naming rule  | [SWS_Nm_00999] |
| [SRS_BSW_00481] | Invalid configuration set selection errors shall follow a naming rule  | [SWS_Nm_00999] |
| [SRS_BSW_00482] | Get Version Informationfunction shall follow a naming rule   | [SWS_Nm_00044] |
| [SRS_Nm_00043]  | NM shall not prohibit bus traffic with NM not being initialized  | [SWS_Nm_00999] |

| Requirement    | Description  | Satisfied by  |
|----------------|--|---|
| [SRS_Nm_00044] | The NM shall be applicable to different types of communication systems which are in the scope of Autosar and support a bus sleep mode. | [SWS_Nm_00051] [SWS_Nm_00172]<br>[SWS_Nm_00274] [SWS_Nm_00276]  |
| [SRS_Nm_00045] | NM has to provide services to coordinate shutdown of NM-clusters independently of each other   | [SWS_Nm_00167] [SWS_Nm_00168]   |
| [SRS_Nm_00046] | It shall be possible to trigger the startup of all Nodes at any Point in Time.   | [SWS_Nm_00031] [SWS_Nm_00032]   |
| [SRS_Nm_00047] | NM shall provide a service to request to keep the bus awake and a service to cancel this request.                                      | [SWS_Nm_00032] [SWS_Nm_00034]<br>[SWS_Nm_00171]   |
| [SRS_Nm_00048] | NM shall put the communication controller into sleep mode if there is no bus communication   | [SWS_Nm_00046]  |
| [SRS_Nm_00050] | The NM shall provide the current state of NM   | [SWS_Nm_00043] [SWS_Nm_00114]<br>[SWS_Nm_00275]   |
| [SRS_Nm_00051] | NM shall inform application when NM state changes occur.   | [SWS_Nm_00031] [SWS_Nm_00032]<br>[SWS_Nm_00046] [SWS_Nm_00156]<br>[SWS_Nm_00158] [SWS_Nm_00159]<br>[SWS_Nm_00161] [SWS_Nm_00162]<br>[SWS_Nm_00163] [SWS_Nm_00249]<br>[SWS_Nm_00487] |
| [SRS_Nm_00052] | The NM interface shall signal to the application that all other ECUs are ready to sleep.   | [SWS_Nm_00192] [SWS_Nm_00999]   |
| [SRS_Nm_00053] | NM on a node which is or become bus unavailable shall have a deterministic Behavior  | [SWS_Nm_00999]  |



| Requirement    | Description   | Satisfied by                                    |
|----------------|---|---|
| [SRS_Nm_00054] | There shall be a deterministic time from the point where all nodes agree to go to bus sleep to the point where bus is switched off.     | [SWS_Nm_00999]                                  |
| [SRS_Nm_00137] | NM shall perform communication system error handling for errors that have impact on the NM behavior.                                    | [SWS_Nm_00999]                                  |
| [SRS_Nm_00142] | NM shall guarantee an upper limit for the bus load generated by NM itself.  | [SWS_Nm_00999]                                  |
| [SRS_Nm_00143] | The bus load caused by NM shall be predictable.   | [SWS_Nm_00999]                                  |
| [SRS_Nm_00144] | NM shall support communication clusters of up to 64 ECUs  | [SWS_Nm_00999]                                  |
| [SRS_Nm_00145] | On a properly configured node, NM shall tolerate a loss of a predefined number of NM messages   | [SWS_Nm_00999]                                  |
| [SRS_Nm_00146] | The NM shall tolerate a time jitter of NM messages in one or more ECUs  | [SWS_Nm_00999]                                  |
| [SRS_Nm_00147] | The NM algorithm shall be processor independent.  | [SWS_Nm_00999]                                  |
| [SRS_Nm_00148] | The specification and implementation shall be split-up into a communication system independent and communication system dependent parts | [SWS_Nm_00999]                                  |
| [SRS_Nm_00149] | The timing of NM shall be configurable.   | [SWS_Nm_00175] [SWS_Nm_00281]<br>[SWS_Nm_00284] |

| Requirement    | Description   | Satisfied by  |
|----------------|---|---|
| [SRS_Nm_00150] | Specific functions of the Network Management shall be statically configurable at pre-compile time     | [SWS_Nm_00055] [SWS_Nm_00130]<br>[SWS_Nm_00132] [SWS_Nm_00134]<br>[SWS_Nm_00136] [SWS_Nm_00138]<br>[SWS_Nm_00140] [SWS_Nm_00150]<br>[SWS_Nm_00164] [SWS_Nm_00165]<br>[SWS_Nm_00166] [SWS_Nm_00241]<br>[SWS_Nm_00251] [SWS_Nm_00255]<br>[SWS_Nm_00273] [SWS_Nm_00277]<br>[SWS_Nm_00278] [SWS_Nm_00279]<br>[SWS_Nm_00286] [SWS_Nm_00287]<br>[SWS_Nm_00288] [SWS_Nm_00289]<br>[SWS_Nm_00290] |
| [SRS_Nm_00151] | The Network Management algorithm shall allow any node to integrate into an already running NM cluster | [SWS_Nm_00031]  |
| [SRS_Nm_00153] | The Network Management shall optionally provide a possibility to detect present nodes                 | [SWS_Nm_00038] [SWS_Nm_00230]   |
| [SRS_Nm_00154] | The Network Management API shall be independent from the communication bus                            | [SWS_Nm_00006] [SWS_Nm_00012]<br>[SWS_Nm_00276]   |
| [SRS_Nm_02503] | The NM API shall optionally give the possibility to send user data                                    | [SWS_Nm_00035] [SWS_Nm_00250]<br>[SWS_Nm_00252] [SWS_Nm_00285]  |
| [SRS_Nm_02504] | The NM API shall optionally give the possibility to get user data                                     | [SWS_Nm_00036] [SWS_Nm_00291]   |
| [SRS_Nm_02505] | The NM shall optionally set the local node identifier to the NM-message                               | [SWS_Nm_00039]  |
| [SRS_Nm_02506] | The NM API shall give the possibility to read the source node identifier of the sender                | [SWS_Nm_00037]  |
| [SRS_Nm_02508] | Every node shall have associated with it a node identifier that is unique in the NM-cluster           | [SWS_Nm_00040]  |

| Requirement    | Description  | Satisfied by   |
|----------------|--|--|
| [SRS_Nm_02509] | The NM interface shall signal to the application that at least one other ECUs is not ready to sleep anymore.                         | [SWS_Nm_00193] [SWS_Nm_00999]  |
| [SRS_Nm_02510] | For CAN NM it shall be optionally possible to immediately transmit the confirmation  | [SWS_Nm_00999]   |
| [SRS_Nm_02511] | It shall be possible to configure the Network Management of a node in Cluster Shutdown   | [SWS_Nm_00168] [SWS_Nm_00228]  |
| [SRS_Nm_02512] | The NM shall give the possibility to enable or disable the network management related communication configured for an active NM node | [SWS_Nm_00033] [SWS_Nm_00034]  |
| [SRS_Nm_02513] | NM shall provide functionality which enables upper layers to control the sleep mode.   | [SWS_Nm_00006] [SWS_Nm_00012]<br>[SWS_Nm_00031] [SWS_Nm_00032]<br>[SWS_Nm_00033] [SWS_Nm_00042]<br>[SWS_Nm_00154] [SWS_Nm_00155] |
| [SRS_Nm_02514] | It shall be possible to group networks into NM Coordination Clusters   | [SWS_Nm_00001] [SWS_Nm_00002]<br>[SWS_Nm_00003] [SWS_Nm_00168]<br>[SWS_Nm_00173]   |
| [SRS_Nm_02515] | NM shall offer a generic possibility to run other NMs than the AUTOSAR-NMs   | [SWS_Nm_00051] [SWS_Nm_00119]<br>[SWS_Nm_00166] [SWS_Nm_00276]   |
| [SRS_Nm_02516] | All AUTOSAR NM instances shall support the NM Coordinator functionality including Bus synchronization on demand                      | [SWS_Nm_00169] [SWS_Nm_00171]<br>[SWS_Nm_00174] [SWS_Nm_00175]<br>[SWS_Nm_00176] [SWS_Nm_00177]<br>[SWS_Nm_00194] [SWS_Nm_00284] |
| [SRS_Nm_02517] | <Bus>Nm shall support Partial Networking on CAN, FlexRay and Ethernet  | [SWS_Nm_00999]   |
| [SRS_Nm_02518] | <Bus>Nm shall be able to distinguish between NM Messages   | [SWS_Nm_00999]   |

| Requirement    | Description  | Satisfied by   |
|----------------|--|----------------|
| [SRS_Nm_02519] | The NM Control Bit Vector shall contain a PNI (Partial Network Information) bit.                           | [SWS_Nm_00999] |
| [SRS_Nm_02520] | <Bus>Nm shall evaluate the PNI bit in the NM message   | [SWS_Nm_00999] |
| [SRS_Nm_02521] | <Bus>Nm shall set the PNI bit for requesting Partial Network functionality                                 | [SWS_Nm_00999] |
| [SRS_Nm_02522] | <Bus>Nm shall calculate the combined partial network request status EIRA                                   | [SWS_Nm_00999] |
| [SRS_Nm_02523] | <Bus>Nm shall calculate the status of the external partial network requests ERA                            | [SWS_Nm_00999] |
| [SRS_Nm_02524] | <Bus>Nm shall communicate EIRA and ERA requests to the upper layers using virtual PDUs                     | [SWS_Nm_00999] |
| [SRS_Nm_02525] | <Bus>Nm shall support channel-specific configuration for ERA   | [SWS_Nm_00999] |
| [SRS_Nm_02526] | <Bus>Nm shall support a global configuration for EIRA over all channels                                    | [SWS_Nm_00999] |
| [SRS_Nm_02527] | CanNm shall implement a filter algorithm dropping all NM messages that are not relevant for the ECU        | [SWS_Nm_00999] |
| [SRS_Nm_02528] | CanNm shall provide a service which allows for spontaneous sending of NM messages.                         | [SWS_Nm_00999] |
| [SRS_Nm_02529] | If partial networking is used, the ECU shall secure that the first message on the bus is the wakeup frame. | [SWS_Nm_00999] |
| [SRS_Nm_02530] | CanIf shall provide an optional channel-specific TX filter   | [SWS_Nm_00999] |

| Requirement    | Description   | Satisfied by  |
|----------------|---|---|
| [SRS_Nm_02531] | CanIf shall provide the possibility to initiate clear and check wake-up flags in the transceiver                  | [SWS_Nm_00999]  |
| [SRS_Nm_02532] | When full communication is requested, CanSm shall enable pass mode on the CanIf TX filter                         | [SWS_Nm_00999]  |
| [SRS_Nm_02533] | CanSm shall provide the possibility to initiate clear and check wake-up flags in the transceiver                  | [SWS_Nm_00999]  |
| [SRS_Nm_02534] | CanSm shall support a validPN shutdown sequence   | [SWS_Nm_00999]  |
| [SRS_Nm_02535] | NM coordination on Nested Sub-Buses   | [SWS_Nm_00254] [SWS_Nm_00256]<br>[SWS_Nm_00257] [SWS_Nm_00259]<br>[SWS_Nm_00261] [SWS_Nm_00262]<br>[SWS_Nm_00267] [SWS_Nm_00271]<br>[SWS_Nm_00272] [SWS_Nm_00280] |
| [SRS_Nm_02536] | NM shall provide an interface which triggers the transition to the Network Mode without keeping the network awake | [SWS_Nm_00031] [SWS_Nm_00119]<br>[SWS_Nm_00245]   |
| [SRS_Nm_02537] | The NM Coordinator shall be able to abort the coordinated shutdown  | [SWS_Nm_00181] [SWS_Nm_00182]<br>[SWS_Nm_00183] [SWS_Nm_00185]<br>[SWS_Nm_00235] [SWS_Nm_00236]<br>[SWS_Nm_00267]   |
| [SRS_Nm_02543] | NM shall forward requests for synchronized PNC shutdown   | [SWS_Nm_00506] [SWS_Nm_91005]   |
| [SRS_Nm_02544] | NM shall forward the indication of a PN shutdown message  | [SWS_Nm_00507] [SWS_Nm_91004]   |
| [SRS_Nm_02545] | <Bus>NM shall handle requests for synchronized PNC shutdown   | [SWS_Nm_91005]  |

## 7 Functional specification

The NM Interface functionality consists of three 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 one or more busses.
- The *synchronized PNC shutdown functionality* used by PNC gateway ECUs in the role of a top-level PNC coordinator or intermediate PNC coordinator to synchronously shutdown particular PNCs across the PNC network topology.

### 7.1 Base functionality

The Generic Network Management Interface module (Nm) shall act as a bus-independent adaptation layer between the bus-specific Network Management modules (such as CanNm, J1939Nm, FrNm and UdpNm) and the Communication Manager module (ComM).

**Note:** The Nm does not provide interface functions beyond those specified in this document. The Nm will provide an interface to the ComM, that does not contain specific knowledge about the type of the underlying busses, and that nevertheless is sufficient to accomplish the necessary network management functions. The algorithm handled by the Nm is bus independent.

**Note:** It is also required that other service layer modules access network management functions exclusively via Nm and that no bypasses to bus specific NM functions exist

**[SWS\_Nm\_00006]** [The Nm shall convert generic function calls from the ComM to bus specific functions of the bus specific NM layer.] ([SRS\\_Nm\\_00154](#), [SRS\\_Nm\\_02513](#))

**[SWS\_Nm\_00012]** [The Nm shall convert callback functions called by the bus specific NM layers to generic callbacks to the ComM.] ([SRS\\_Nm\\_00154](#), [SRS\\_Nm\\_02513](#))

**[SWS\_Nm\_00091]** [The Base functionality of Nm may be implemented completely or partly using macros.] ([SRS\\_BSW\\_00330](#))

### 7.2 NM Coordinator functionality

*NM Coordinator functionality* is a functionality of **Nm** that uses a [coordination algorithm](#) to coordinate the shutdown of **NM** on all, or one or more independent subsets of the busses that the ECU is connected to.

Dependent on configuration, the [coordination algorithm](#) can be configured to achieve different levels of synchronization of the shutdown.

An ECU using an [NM](#) that actively performs the *NM Coordinator functionality* is commonly referred to as an [NM Coordinator](#). However, in this specification this term is synonymous with the *NM Coordinator functionality* when used in requirements.

**Note:** Consider that certain bus types have different nomenclature on the terms [Network](#), [Channel](#), [Cluster](#).

**[SWS\_Nm\_00292]** [If the *NM Coordinator functionality* is configured, the configuration parameter [NmCycletimeMainFunction](#) shall be configured (see [\[SWS\\_Nm\\_00118\]](#)).] ([SRS\\_BSW\\_00478](#))

**Note:** The [NM Coordinator](#) may use this to calculate the timeout status of internal timers.

### 7.2.1 Applicability of the NM Coordinator functionality

**[SWS\_Nm\_00001]** [The [coordination algorithm](#) shall be able to handle a topology where several coordinated busses are connected to one [NM Coordinator](#).] ([SRS\\_Nm\\_02514](#))

**[SWS\_Nm\_00256]** [The [NM-Coordinator](#) shall support two or more [NM-Coordinator](#)s connected to the same [NM Cluster](#).] ([SRS\\_Nm\\_02535](#))

**[SWS\_Nm\_00051]** [The [NM Coordinator](#) shall be able to coordinate busses running the official AUTOSAR bus specific NMs as well as all other generic bus NMs implementing the required functionality, callbacks and interfaces.] ([SRS\\_Nm\\_00044](#), [SRS\\_Nm\\_02515](#))

**Note:** The required functionality, callbacks and interfaces are specified in [subsection 7.5.2](#). Coordinator Support for **J1939Nm** is not needed as the **J1939Nm** does not support shutdown handling.

**[SWS\_Nm\_00055]** [The [NM Interface](#) configuration shall provide the parameter [Nm-CoordinatorSupportEnabled](#) to define if the support of the *NM Coordinator functionality* is present or not.] ([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00167]** [It shall be possible to configure multiple [NM coordination clusters](#) that shall be coordinated independently.] ([SRS\\_Nm\\_00045](#))

**[SWS\_Nm\_00168]** [Each bus shall belong to zero or one [NM coordination cluster](#).] ([SRS\\_Nm\\_00045](#), [SRS\\_Nm\\_02511](#), [SRS\\_Nm\\_02514](#))

**Rationale:** The configuration parameter [NmCoordClusterIndex](#) is used for specifying to which coordination cluster a bus belongs. If this parameter is undefined for a channel, the corresponding bus does not belong to an [NM coordination cluster](#).

**[SWS\_Nm\_00169]** [Shutdown shall only be coordinated on the presently awake networks of a coordination cluster. Networks that are already in "bus-sleep mode" shall still be monitored but not coordinated.] ([SRS\\_Nm\\_02516](#))

**Rationale:** The **NM Coordinator** does not require all busses in a coordination cluster to be awake, working with subsets of the coordination cluster resp. partial networks, to perform **coordinated shutdown**. It always monitors the shutdown initiation conditions and when these are met, it performs a coordinated shutdown of all the presently awake busses in the coordination cluster.

**Note:** It is outside the scope of the **Nm** to provide synchronized wakeup for coordinated busses. It is up to the application (-> vehicle mode management) to wake up the required resp. all channels if one channel wake up occurs.

## 7.2.2 Keeping coordinated busses alive

**[SWS\_Nm\_00002]** [As long as the node implementing the **NM Coordinator** is not ready to go to sleep on at least one of the busses in a coordination cluster (i.e. that it has actively requested the network), the **NM Coordinator** shall ensure that the network is requested on all currently active busses in that coordination cluster.] ([SRS\\_Nm\\_02514](#))

**[SWS\_Nm\_00003]** [As long as at least one bus in the coordination cluster is not ready to sleep (i.e. because another node than the **NM Coordinator** is requesting that bus), the **NM Coordinator** shall still ensure that the network is requested on all currently active busses in that coordination cluster even if the local ECU itself is ready to go to sleep on all busses of that coordination cluster.] ([SRS\\_Nm\\_02514](#))

**Rationale:** 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**.

**Rationale:** The **Nm** requests the network on a bus by calling the bus specific NM function `<BusNm>_NetworkRequest`.

Since all AUTOSAR bus specific **NMs** are built on the principle that one AUTOSAR node can keep the bus alive as long as it keeps the network requested, the **NM Coordinator** will keep all busses of the coordination cluster awake by requesting the network for the **bus specific NMs**.

The two requirements **[SWS\_Nm\_00002]** and **[SWS\_Nm\_00003]** above can be summarized as follows: as long as at least one node (including the node implementing the **NM Coordinator**) keeps any of the busses in the coordination cluster awake, the **NM Coordinator** shall keep all busses of that coordination cluster awake.

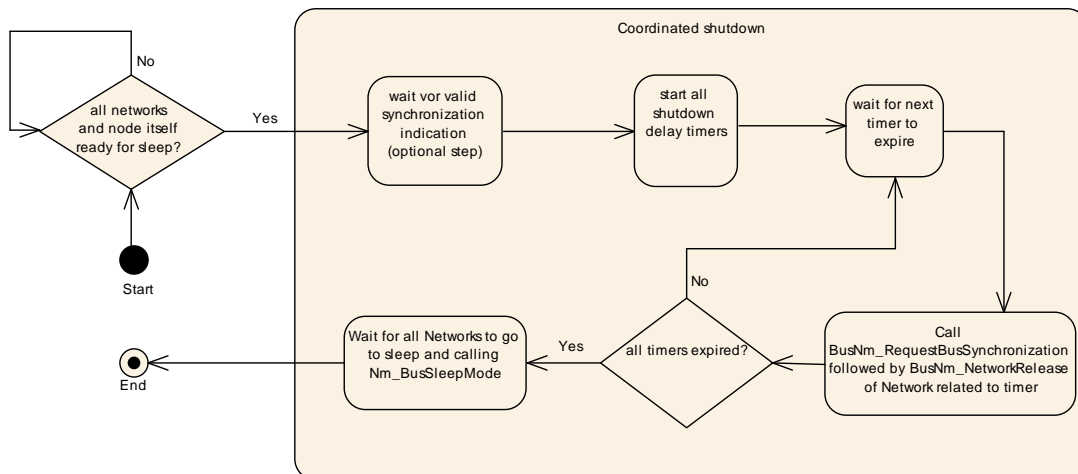
**[SWS\_Nm\_00228]** [If a bus of a coordination cluster has the parameter **NmChannelSleepMaster** set to **TRUE**, the **NM Coordinator** shall consider that bus ready to sleep at all times and shall not await an invocation of **Nm\_RemoteSleepIndication** from that bus before starting shutdown of that network.] ([SRS\\_Nm\\_02511](#))



**Rationale:** This property shall be set for all **bus specific NMs** where the sleep of the bus can be absolutely decided by the local node only and that no other nodes of that bus can oppose that decision. An example of such a network is LIN where the local AUTOSAR ECU will always be the LIN bus master and can always solely decide when the network shall go to sleep.

### 7.2.3 Shutdown of coordinated busses

The level of synchronization achievable is dependent on the configuration. See [subsection 7.2.5](#), [Figure 7.1](#) shows an overview of the [coordination algorithm](#). As described in Section 7.2.1, the [coordination algorithm](#) and [coordinated shutdown](#) shall be applied independently per NM coordination cluster.



**Figure 7.1: Overview of the coordination algorithm with the coordinated shutdown as part of it**

**Note:** There is no limitation where the actions performed by the [coordination algorithm](#) shall take place.

This can be done either by the Nm main function ( `NmMainFunction` ) or module indication / callbacks.

**[SWS\_Nm\_00171]** [When all networks of a coordination cluster are either ready to go to sleep or already in "bus-sleep mode" the [NM Coordinator](#) shall start the [coordinated shutdown](#) on all awake networks. The [NM Coordinator](#) shall evaluate continuously if the [coordinated shutdown](#) can be started.]([SRS\\_Nm\\_00047](#), [SRS\\_Nm\\_02516](#))

**Rationale:** Evaluation of shutdown conditions can be also done in other API calls than the main function. The evaluation can be segmented then to check only the specific conditions affected by the API calls there, hence it is not necessary to re-evaluate all conditions in every main processing period and every API call.

**[SWS\_Nm\_00172]** [If the configuration parameter `NmSynchronizingNetwork` is `TRUE` for any of the busses in a coordination cluster, the coordination shutdown shall be

delayed until a network that is configured as synchronizing network for this coordination cluster invoked `Nm_SynchronizationPoint`.] ([SRS\\_Nm\\_00044](#))

**[SWS\_Nm\_00293]** [If on a coordinated network the coordinator detects a mode change to `NM_MODE_SYNCHRONIZE`, `NM_MODE_PREPARE_BUS_SLEEP` or `NM_BUS_SLEEP` AND the coordinated cluster this network belongs to has not started the shutdown process AND if there is no internal network mode request for that channel by ComM, the coordinator shall treat this network as remote sleep and shall call `<BusNm>_NetworkRelease` for this network. If additionally for this network the configuration parameter `NmSynchronizingNetwork` is `TRUE` then the coordinator shall not wait for `Nm_SynchronizationPoint` on this network.] ()

**Rationale:** If one or more of the networks in the NM coordination clusters is cyclic (such as FlexRay), a higher level of synchronized shutdown will be achieved if the algorithm is synchronized with one of the included cyclic networks. If configured so, the shutdown timers for all coordinated networks will not be started until the synchronizing network has called the `Nm_SynchronizationPoint`.

**Rationale:** Although only one network per NM coordination cluster should be configured to indicate synchronization points, this will allow the *NM Coordinator functionality* to filter out all synchronization indications except those that originate from the network that is configured to be the synchronizing network of each coordination cluster.

**[SWS\_Nm\_00173]** [If not all conditions to start the `coordinated shutdown` have been met, or if the `coordinated shutdown` has already been started (but not aborted), calls to `Nm_SynchronizationPoint` shall be ignored.] ([SRS\\_Nm\\_02514](#))

**Rationale:** In some cases, non-synchronizing networks can take longer time to go to sleep. If this happens, the `coordinated shutdown` will be started based on one synchronization indication, but as the synchronizing network will not be released directly it will continue to invoke (several) more synchronization indications which can safely be ignored.

**[SWS\_Nm\_00174]** [If the configuration parameter `NmSynchronizingNetwork` is `FALSE` for all of the presently awake busses in a coordination cluster, the timers shall be started after all shutdown conditions have been met, without waiting for a call to `Nm_SynchronizationPoint`(). (see also [\[SWS\\_Nm\\_00172\]](#)).] ([SRS\\_Nm\\_02516](#))

**[SWS\_Nm\_00175]** [When the `coordinated shutdown` is started, a shutdown delay timer shall be activated for each currently awake channel in the coordination cluster. Each timer shall be set to `NmGlobalCoordinatorTime`. In case `NmBusType` is not set to `NM_BUSNM_LOCALNM` additionally the shutdown time of the specific channel `TSHUTDOWN_CHANNEL` shall be subtracted.] ([SRS\\_Nm\\_00149](#), [SRS\\_Nm\\_02516](#))

**[SWS\_Nm\_00284]** [If the `NmGlobalCoordinatorTime` is zero the shutdown delay timer of all channels shall also be zero.] ([SRS\\_Nm\\_00149](#), [SRS\\_Nm\\_02516](#))

**Note:** The `TSHUTDOWN_CHANNEL` can be calculated as described in [subsection 7.2.5](#) or with following formulas:

CanNm: Ready Sleep Time + Prepare BusSleep Time

FrNm: Ready Sleep Time, e.g.:  $(FrNmReadySleepCnt+1) * FrNmRepetitionCycle *$   
"Duration of one Flexray Cycle"

GenericNm: NmGenericBusNmShutdownTime

**[SWS\_Nm\_00176]** [When a shutdown timer expires for a network, **Nm** shall in case `BusNmType` is not set to `NM_BUSNM_LOCALNM` release the network by calling the `<BusNm>_RequestBusSynchronization` followed by `<BusNm>_NetworkRelease`. In case `BusNmType` is set to `NM_BUSNM_LOCALNM` **Nm** shall inform `ComM` about shutdown by calling `ComM_Nm_BusSleepMode()`.] ([SRS\\_Nm\\_02516](#))

**Note:** In the AUTOSAR Classic Platform, `CanNm_PassiveStartUp`, `J1939Nm_PassiveStartUp`, `FrNm_PassiveStartUp` and `UdpNm_PassiveStartUp` have been specified as the predefined interfaces corresponding to `<BusNm>_PassiveStartUp`.

**[SWS\_Nm\_00177]** [**Nm** shall keep track of all networks that have been released but have not yet reported "bus-sleep mode". If the shutdown is aborted, these networks shall still be considered active networks.] ([SRS\\_Nm\\_02516](#))

**Note:** See Section [subsection 7.3.3](#).

Definition: When all networks have been released and all networks are in "bus-sleep mode", the [coordinated shutdown](#) is completed.

## 7.2.4 Coordination of nested sub-busses

To support the coordination of nested sub-busses the Nm-Coordinators need be configured to build up a coordination hierarchy. The top most `NM Coordinator` has only actively coordinated channels (`NmActiveCoordinator == TRUE`) per coordination cluster. This `NM Coordinator` has to initiate the [coordinated shutdown](#) for all other coordinators. An nested `NM Coordinator` receive his shutdown indication information from his passively configured channel (`NmActiveCoordinator == FALSE`) and provides this information to following `NM Coordinators` via his actively coordinated channels (`NmActiveCoordinator == TRUE`).

The [Figure 7.2](#) will explain this as an example.

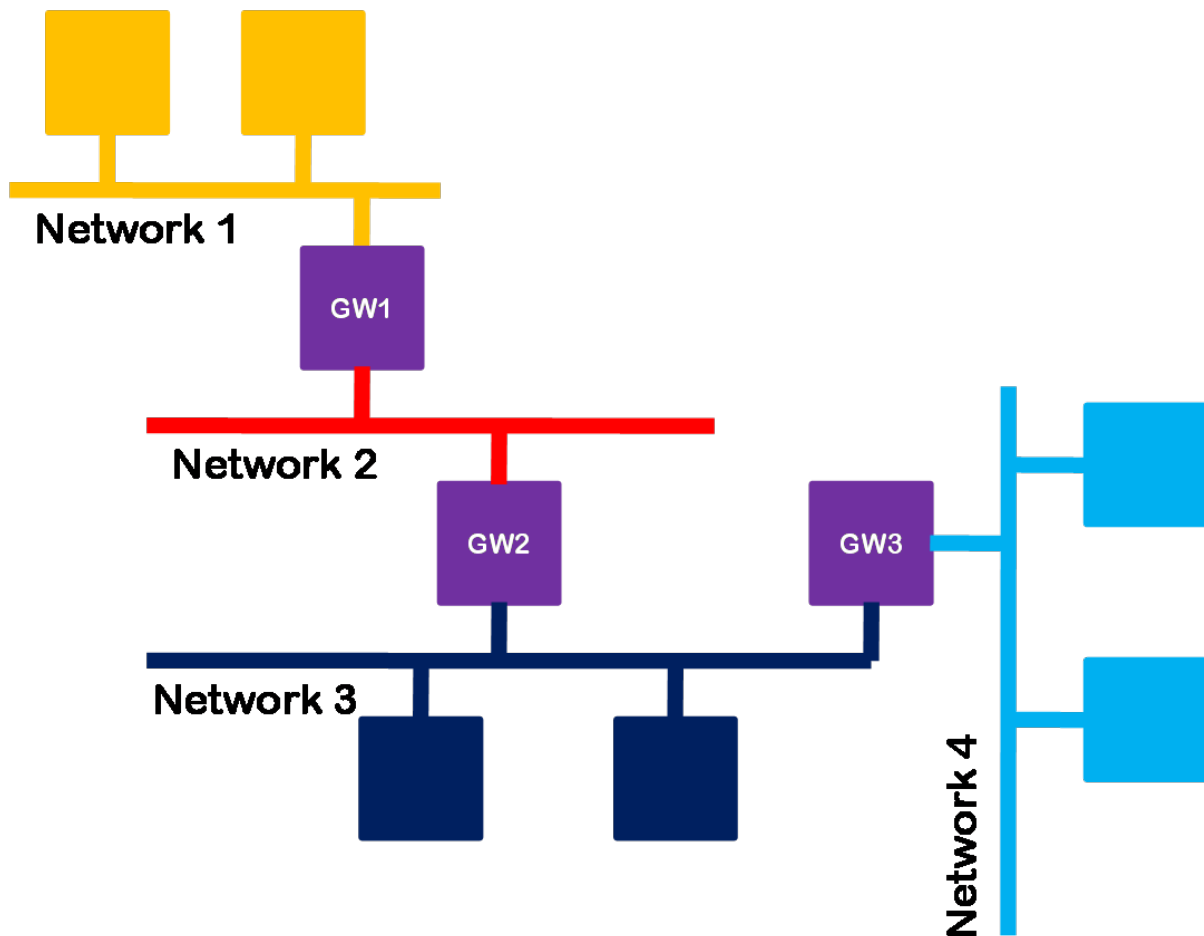


Figure 7.2: Use Case Nested Gateways

The exemplary topology shown in [Figure 7.2](#) has the following coordination approach. GW 1 have configured the channel onto Network 1 and Network 2 as actively coordinating channels. Where GW 2 is configured with Network 2 connection as passively coordinated channel, but with actively coordinated channel on Network 3. GW 3 then needs to be configured on Network 3 as passively coordinated channel but as actively coordinated channel for his connection to the Network 4.

**[SWS\_Nm\_00280]** [The functionality of coordinating nested sub busses shall be available if the `NmCoordinatorSyncSupport` parameter is set to `TRUE`.] ([SRS\\_Nm\\_02535](#))

**Note:** All requirements within this chapter are valid “per Nm Coordination Cluster” (see [\[SWS\\_Nm\\_00167\]](#)).

The `NmActiveCoordinator` parameter indicates, if an `NM Coordinator` behaves on this channel in actively manner

( Actively coordinated channel ) [`NmActiveCoordinator` = `TRUE`]

or behave in a passively manner

( Passively coordinated channel ) [`NmActiveCoordinator` = `FALSE`].

**[SWS\_Nm\_00257]** [On its passively coordinated channels a *NM-Coordinator* shall send Nm messages only if the node has a network management request pending or a connected network which is coordinated actively by that *NM Coordinator* is not ready to sleep.](*SRS\_Nm\_02535*)

**Rationale:** This prevents that 2 *NM Coordinators* at the same channel, send NM messages when they are ready to sleep and therefore keep the bus awake. Without this mechanism it would not be possible to detect if there is at least one other node active.

**[SWS\_Nm\_00259]** [The *NM Coordinator* shall set the *NMCoordinatorSleepReady* bit in the Nm message via `<BusNm>_SetSleepReadyBit` to the value 1 at his actively coordinated channels,

IF

all nodes of the *NM Coordination cluster* are ready to sleep (*RemoteSleepIndication*)

AND

IF *NmSynchronizingNetwork* is enabled a *NmSynchronizationPoint()* call has been received on the corresponding channel

AND

If all channels of this *NM Coordination cluster* are configured as *NmActiveCoordinator* == TRUE. AND

If the *NmBusType* is not configured to *NM\_BUSNM\_LOCALNM*.](*SRS\_Nm\_02535*)

**Note:** for Position of Coordinator Bits in CBV see according `<BusNm>` specifications.

**Note:** This applies to the top most coordinator (no passively coordinated channel).

**Rationale:** Nodes which contain passively coordinated channels do not need a synchronization point as they are synchronized by the sleep ready bit of their active coordinator already.

**Note:** Nodes which contain a passively coordinated channel will set the bit according to the requirement in [*SWS\_Nm\_00261*].

**[SWS\_Nm\_00261]** [If *NmCoordReadyToSleepIndication* is received on a passively coordinated channel the *NmCoordinator* shall set the *NMCoordinatorSleepReady* bit to SET (1) via API call to `<BusNm>_SetSleepReadyBit` on all actively coordinated channels.](*SRS\_Nm\_02535*)

**[SWS\_Nm\_00271]** [If *NmCoordReadyToSleepCancellation* is received on a passively coordinated channel the *NmCoordinator* shall set the *NMCoordinatorSleepReady* bit to UNSET (0) via API call to `<BusNm>_SetSleepReadyBit` on all actively coordinated channels.](*SRS\_Nm\_02535*)

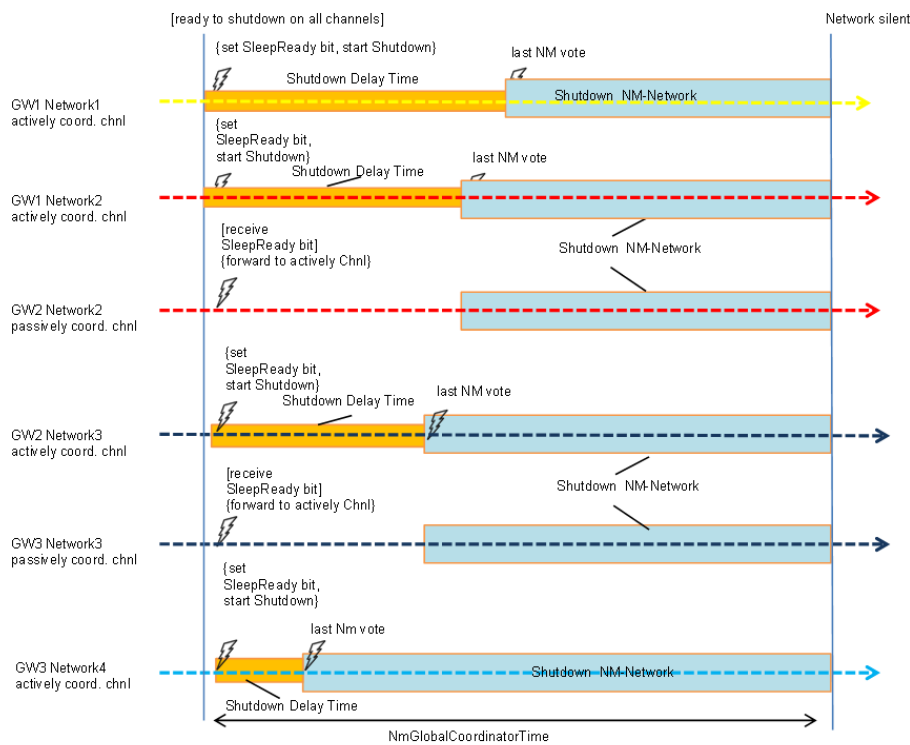
**Note:** On its passively coordinated channel a *NM Coordinator* would not set the *Sleep Ready* bit ever (via `<busNm>` function call) but forward a received status change of *Sleep ready* bit onto its actively coordinated channels.

**Note:** On its actively coordinated channel(s) a *NM Coordinator* a call of *Nm\_CoordReadyToSleepIndication* and *Nm\_CoordReadyToSleepCancellation* is not expected.

**[SWS\_Nm\_00262]** [The *NM Coordinator* shall start *coordinated shutdown* after the *Sleep Ready* Bit with *SET* status has been requested.] (*SRS\_Nm\_02535*)

**[SWS\_Nm\_00281]** [*NmGlobalCoordinatorTime* shall be set at least to the maximum time needed to shut down all Networks coordinated.] (*SRS\_Nm\_00149*)

**Note:** This includes all nested connections. (for example see [Figure 7.3](#))



**Figure 7.3: Shutdown with *Nm\_GlobalCoordinatorTime***

**[SWS\_Nm\_00267]** [*NM Coordinator* shall set the *NMCoordinatorSleepReady* bit to UNSET (0) via API call to *<BusNm>\_SetSleepReadyBit* on all actively coordinated channels if the *coordinated shutdown* has been aborted for any reason.] (*SRS\_Nm\_02535*, *SRS\_Nm\_02537*)

**Note:** Details about aborted shutdown can be found in [subsection 7.3.3](#).

## 7.2.5 Calculation of shutdown timers

The `coordination algorithm` is quite flexible since the level of synchronization achievable depends on the configuration of switches and timers. Depending on which event or point in time that is the goal to synchronize on, the configuration shall be done differently. This Chapter contains guide on how to achieve three different levels of synchronization. It is up to the configuration to follow these guidelines or to achieve a separate order of synchronization by choosing his/her own particular configuration. Therefore, this Section will not contain any requirement, only recommendations.

Note that absolute synchronization will never be possible to achieve. The jitter factors that determine the preciseness of the synchronization involve the processing period of the **Nm**, the exactness of the timers and the busload for non-deterministic busses. Correctly configured, the Use Cases described below will give the best possible synchronization that is achievable considering these circumstances.

Previous version of the `NM Coordinator` included the possibility for the coordinator algorithm to delay the start of the `coordinated shutdown` "a number of rounds". This specific delay has been removed but a similar behavior can still be obtained by increasing all shutdown timers (configuration parameter `NmGlobalCoordinatorTime`). Special care must be taken when cyclic networks (such as FlexRay) are used when this increased delay time should be quantified to the synchronization indication periodicity of those networks.

## 7.2.6 Synchronization Use Case 1 - Synchronous command

This Use Case focuses on how to synchronize the point in time where the different networks are released.

This results in the fastest possible total shutdown of all networks, but with the downside that the networks will not enter "bus-sleep mode" at the same time.

**Rationale:** One example of this Use Case is when several CAN networks shall be kept alive as long as any CAN-node is requesting one of the networks; but when all nodes are ready to go to sleep it does not matter if "bus-sleep mode" is entered at the same time for the different networks.

Since the Use Case does not consider any cyclic behavior of the networks, the synchronization parameter `NmSynchronizingNetwork` shall be set to `FALSE` for all networks and no **bus specific NM** shall be configured to invoke the `Nm_Synchronization-Point` callback.

To achieve the fastest possible shutdown, the shutdown timer parameter `NmGlobalCoordinatorTime` needs to be set to `0.0`.

### 7.2.7 Synchronization Use Case 2 - Synchronous initiation

This Use Case is an extension of Use Case 1, but here consideration is taken to the fact that for some networks the request to release the network will only be acted upon at specific points in time. This Use Case will command a simultaneous shutdown like in Use Case 1, but will wait until a point in time suitable for the synchronizing network.

**Rationale:** One example of this Use Case is when one FlexRay network and several CAN networks where the time when all networks are active shall be maximized, but the networks shall still be put to sleep as fast as possible.

Since this Use Case shall consider the cyclic behavior of a selected network, one of the networks shall have its synchronization parameter `NmSynchronizingNetwork` set to `TRUE` while the other networks shall have this parameter set to `FALSE`. The synchronizing network's **bus specific NM** shall also be configured to invoke the `Nm_SynchronizationPoint` callback at suitable points in time where the shutdown shall be initiated.

To achieve the fastest possible shutdown, the shutdown timer parameter `NmGlobalCoordinatorTime` needs to be set to `0.0`.

### 7.2.8 Synchronization Use Case 3 - Synchronous network sleep

This Use Case will focus on synchronizing the point in time where the different networks enters "bus-sleep mode". It will wait for indication from a synchronizing network, and then delay the network releases of all networks based on timing values so that the transition from "network mode" (or "prepare bus-sleep mode") into "bus-sleep mode" is as synchronized as possible.

**Rationale:** One example of this Use Case is when one FlexRay network and several CAN networks shall stop communicating at the same time.

Since this Use Case shall consider the cyclic behavior of a selected network, of the networks - preferably the cyclic one - shall have its synchronization parameter `NmSynchronizingNetwork` set to `TRUE` while the other networks shall have this parameter set to `FALSE`. The synchronizing network's **bus specific NM** shall also be configured to invoke the `Nm_SynchronizationPoint` callback at suitable points in time where the shutdown shall be initiated.

To calculate the shutdown timer **TSHUTDOWN\_CHANNEL** of each network, specific knowledge of each networks timing behavior must be obtained.

For all networks, **TSHUTDOWN\_CHANNEL** must be calculated, this is the minimum time it will take the network to enter "bus-sleep mode". For non-cyclic networks (such as CAN), the time shall be measured from the point in time when the network is released until it enters "bus-sleep mode". For cyclic networks (such as FlexRay) the time shall also include the full range from the synchronization indication made just before



the network is released. For Generic **BusNms** the time is given by the configuration parameter `NmGenericBusNmShutdownTime`.

For the synchronizing network, **TSYNCHRONIZATION\_INDICATION** must be determined. This is the time between any two consecutive calls made by that **bus specific NM** to `Nm_SynchronizationPoint`.

The `NmGlobalCoordinatorTime` shall be the total time that is needed for the `co-ordination algorithm`. This includes the shutdown time of nested sub-busses. Start with setting `NmGlobalCoordinatorTime` to the same value as **TSHUTDOWN\_CHANNEL** for the synchronizing network. If the **TSHUTDOWN\_CHANNEL** for any other network is greater than `NmGlobalCoordinatorTime`, extend `NmGlobalCoordinatorTime` with **TSYNCHRONIZATION\_INDICATION** repeatedly until `NmGlobalCoordinatorTime` is equal to, or larger than any **TSHUTDOWN\_CHANNEL**.

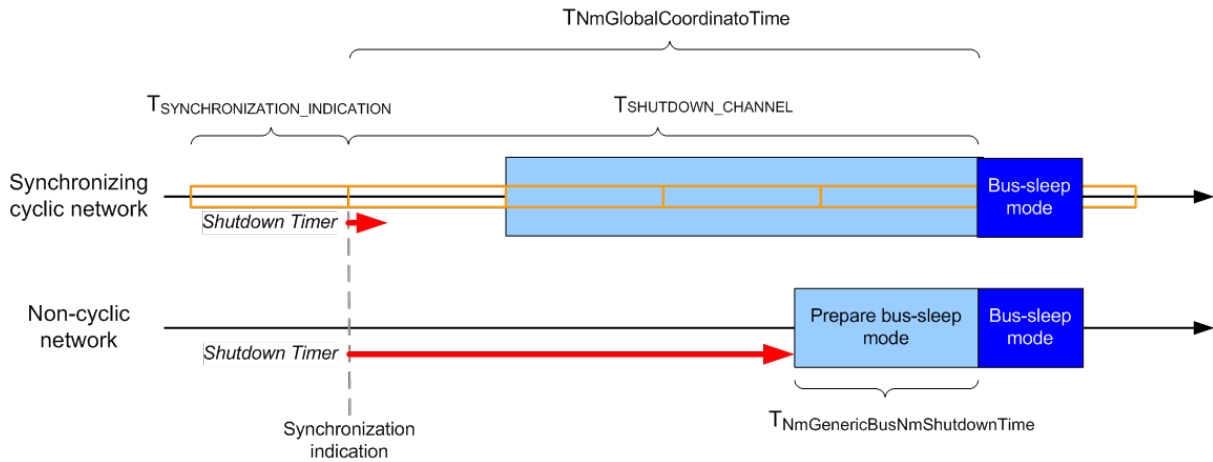
The shutdown delay timer for each network shall be calculated as `NmGlobalCoordinatorTime - TSHUTDOWN_CHANNEL` for that network.

For the cyclic networks this parameter must then be increased slightly in order to make sure that the network release will occur between to synchronization indications, slightly after `Nm_SynchronizationIndication` (would) have been called. The amount of time to extend the timer depends on the implementation and configuration of the **bus specific NM** but should be far smaller than **TSYNCHRONIZATION\_INDICATION**.

### 7.2.8.1 Examples

In the first case ( [Figure 7.4](#)), the synchronizing network holds the largest **TSHUTDOWN\_CHANNEL**, which will therefore equal the `NmGlobalCoordinatorTime`. For the synchronizing network, the shutdown delay timer will be `NmGlobalCoordinatorTime - TSHUTDOWN_CHANNEL`, which is zero, but then a small amount of time is added to make sure that the Nm will wait to release the network between the two synchronization points.

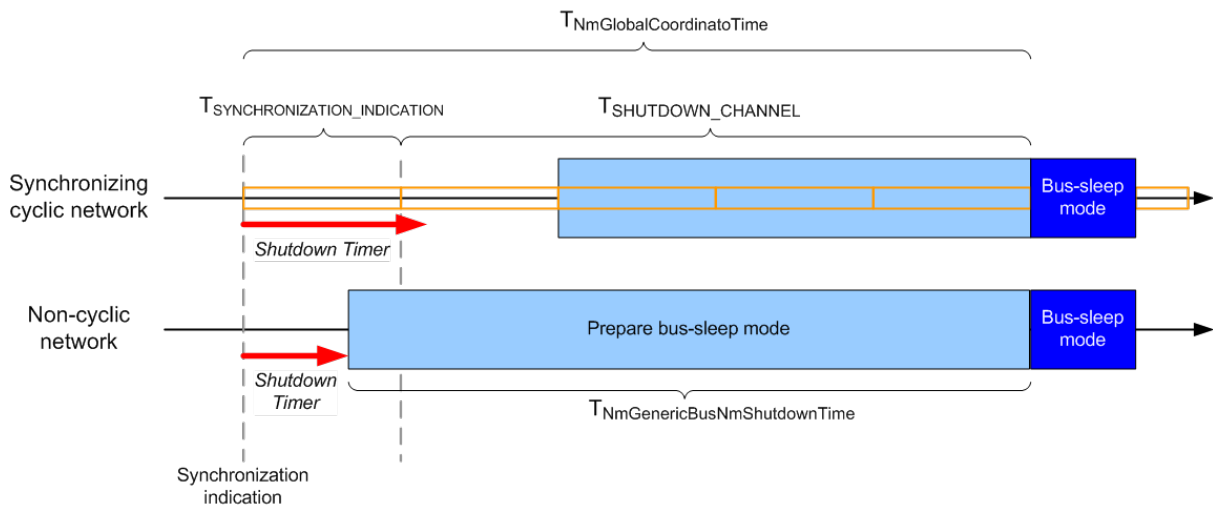
For the Non-cyclic network, the shutdown delay timer will simply be `NmGlobalCoordinatorTime - TSHUTDOWN_CHANNEL`.



**Figure 7.4: Timing example one**

In the second case ( [Figure 7.5](#)), the non-cyclic network takes very long time to shut down and therefore holds the largest **T<sub>SHUTDOWN\_CHANNEL</sub>**. The **NmGlobalCoordinatorTime** has now been obtained by taking the synchronizing network's (slightly shorter) **T<sub>SHUTDOWN\_CHANNEL</sub>** adding **T<sub>SYNCHRONIZATION\_INDICATION</sub>** once to this value.

For the synchronizing network, the shutdown timer will be **NmGlobalCoordinatorTime - T<sub>SHUTDOWN\_CHANNEL</sub>**, with a small amount of time added to make sure that the **Nm** will wait to release the network between the two synchronization points. For the Non-cyclic network, the shutdown timer will simply be **NmGlobalCoordinatorTime - T<sub>SHUTDOWN\_CHANNEL</sub>**.



**Figure 7.5: Timing example two**

## 7.3 Wakeup and abortion of the coordinated shutdown

Nm is not responsible for normal wakeup of the node or the networks this will be done by the COM Manager (**ComM**).

### 7.3.1 External network wakeup

For both *Basic functionality* and *NM Coordination functionality*, **Nm** will forward wakeup indications from the networks (indicated by the bus specific NMs calling the callback `Nm_NetworkStartIndication`) to the **ComM** by calling `ComM_Nm_NetworkStartIndication()`. **ComM** will then call `Nm_PassiveStartUp`, which will be forwarded by Nm to the corresponding interface of the bus specific NM.

Processing of wake-up events for channels in bus-sleep (related to transceiver and controller state) will be handled by **EcuM** and **ComM**. No interaction of the **Nm** apply here. **Nm** will get the network request from **ComM** as stated above, depending on the wake-up validation and the respective communication needs.

**[SWS\_Nm\_00245]** [If the **ComM** calls `Nm_PassiveStartUp()` for a network that is part of a coordinated cluster of networks, the **Nm** coordinator functionality shall treat this call as if the **ComM** had called `Nm_NetworkRequest()`. In case `BusNmType` is not set to `NM_BUSNM_LOCALNM` the **Nm** shall forward a call of `<BusNm>_NetworkRequest` to the lower layer and accordingly, the network shall be counted as requested by the NM coordinator.] (*SRS\_Nm\_02536*)

**Note:** In other words: Calls of `Nm_PassiveStartUp` for networks that are part of a cluster of coordinated networks shall be "translated" to / handled as calls of `Nm_NetworkRequest`.

### 7.3.2 Coordinated wakeup

Depending on the configuration, **ComM** can start multiple networks based on the indication from one network. It is recommended to configure the **ComM** to automatically start all network of a `NM Coordination Cluster` if one of the networks indicates network start, but this is not always necessary. Since the wakeup of network is outside the scope of **Nm**, this is independent of if the *NM Coordination functionality* is used or not.

### 7.3.3 Abortion of the coordinated shutdown

If the *NM Coordination functionality* is activated and `coordinated shutdown` has been initiated on an `NM Coordination Cluster`, dependent on the coordinator algorithm configuration it might take time before each included bus is actually released. If any node on one of the coordinated buses changes its state and starts requesting

the network before all networks are released, race conditions can occur in the `coordination algorithm`. This can happen in four ways:

1. A node on a network that has not yet been released and is still in 'network mode' starts requesting the network again. This will be detected by the **bus specific NM** which will inform **Nm** by calling `Nm_RemoteSleepCancellation`.
2. A node on a network that has already been released and has indicated "prepare bus-sleep mode" but not "bus-sleep mode" starts requesting the network again. This will be detected by the **bus specific NM** that will automatically change state to "network mode" and inform **Nm** by calling `Nm_NetworkMode`.
3. The **ComM** requests the network on any of the networks in the `NM Coordination Cluster`.
4. The coordinator which actively coordinates this network sends `Nm` message with cleared Ready-Sleep Bit. This will be detected by the Bus spec NM (only on passively coordinated channels) and forwarded to the NM by calling `Nm_CoordReadyToSleepCancellation`.

The generic approach is to abort the shutdown and start requesting the networks again. However, networks that have already gone into "bus-sleep mode" shall not be automatically woken up; this must be requested explicitly by **ComM**.

**[SWS\_Nm\_00181]** [The `coordinated shutdown` shall be aborted if any network in that `NM Coordination Cluster`,

- indicates `Nm_RemoteSleepCancellation` or
- indicates `Nm_NetworkMode` or
- indicates `Nm_CoordReadyToSleepCancellation`
- or the **ComM** request one of the networks with `Nm_NetworkRequest` or `Nm_PassiveStartUp`.

]([SRS\\_Nm\\_02537](#))

**Note:** `Nm_NetworkStartIndication` is not a trigger to abort the `coordinated shutdown`, as this is handled by the upper layer.

**[SWS\_Nm\_00182]** [If the `coordinated shutdown` is aborted, `NM Coordinator` shall call `ComM_Nm_RestartIndication` for all networks that already indicated "bus sleep".]([SRS\\_Nm\\_02537](#))

**Rationale:** Since **Nm** cannot take decision to wake networks on its own, this must be decided by **ComM** just as in the (external) wakeup case.

**[SWS\_Nm\_00183]** [If the `coordinated shutdown` is aborted, `NM Coordinator` shall in case `BusNmType` is not set `NM_BUSNM_LOCALNM` request the network from the **<busNm's>** for the networks that have not indicated "bus sleep". In case `BusNmType` is set to `NM_BUSNM_LOCALNM` **Nm** shall inform **ComM** about network startup by calling `ComM_Nm_NetworkMode()`].([SRS\\_Nm\\_02537](#))

[SWS\_Nm\_00185] [If the `coordination algorithm` has been aborted, all conditions that guard the initiation of the `coordinated shutdown` shall be evaluated again.] (*SRS\_Nm\_02537*)

**Rationale:** When a `coordinated shutdown` has been aborted, in most cases there are now networks in that NM Coordination Cluster that do not longer indicate that network sleep is possible, and thus the NM Coordinator must keep all presently non-sleeping networks awake. There can be cases where none of the conditions have been changed, which will only lead to a re-initiation of the `coordinated shutdown`.

[SWS\_Nm\_00235] [If a `coordinated shutdown` has been aborted and Nm receives `E_NOT_OK` on a `<BusNm>_NetworkRequest`, that network shall not be considered awake when the conditions for initiating a coordinated shutdown are evaluated again.] (*SRS\_Nm\_02537*)

**Rationale:** Any `<BusNm>` that needs to be re-requested during an aborted `coordinated shutdown` have previously been released, both by **ComM** and by **Nm**. It is the responsibility of the `<BusNm>` to inform the **ComM** (through **Nm**) that the network really has been released and therefore the **ComM** will have knowledge of the network state even though the error response on `Nm_NetworkRequest` never reached the **ComM** directly.

[SWS\_Nm\_00236] [If a `coordinated shutdown` has been initiated and Nm receives `E_NOT_OK` on a `<BusNm>_NetworkRelease`, the shutdown shall be immediately aborted. For all networks that have not entered "bus-sleep mode", Nm shall request the networks. This includes the network that indicated an error for `<BusNm>_NetworkRelease`. As soon as this has been done, the conditions for initiating coordinated shutdown can be evaluated again. This applies also to networks that were not actively participating in the current coordinated shutdown.] (*SRS\_Nm\_02537*)

**Rationale:** If a network cannot be released, it shall immediately be requested again to synchronize the states between the NM Coordinator in the Nm and the `<BusNm>`. The `coordinated shutdown` will eventually be initiated again as long as the problem with the `<BusNm>` persists. It is up to the `<BusNm>` to report any problems directly to the **DEM** and/or Default Error Tracer so the NM Coordinator shall only try to release the networks until it is successful.

## 7.4 Synchronized PNC shutdown functionality

The synchronized PNC shutdown is a functionality which is a cooperation of **ComM**, **Nm** and `<Bus>Nm` to ensure a synchronized PNC shutdown at almost the same point in time across the whole PN topology. A synchronized PNC shutdown is handled by ECUs in role of a top-level PNC coordinator or intermediate PNC coordinator and where the PNC gateway is enabled. If **ComM** of an ECU in the role of a top-level PNC coordinator detects that a PNC is released, the **ComM** requests a synchronized PNC shutdown by calling `Nm_RequestSynchronizedPncShutdown` per `ComMChannel` and `ComMPnc`. The Nm forwards the call immediately to the affected

**<Bus>Nms.** The **<Bus>Nms** store all requests and handle them in the context of the **<Bus>Nm\_Mainfunction**. Therefore, the **<Bus>Nm** has to aggregate the request for a synchronized PNC shutdown into a byte array (**PN info**), where each bit represent a particular PNC. The PN info is created and transmitted per **<Bus>NM-Channel**.

If a PN shutdown message is received by an ECU in the role of an intermediate PNC coordinator, the **<Bus>Nms** forward the information by writing the corresponding ComSignals and indicate the reception of a PN shutdown message by calling the callback function **Nm\_ForwardSynchronizedPncShutdown**. The callback function will immediately forward the indication to **ComM** by calling **ComM\_Nm\_ForwardSynchronizedPncShutdown**. **ComM** will immediately request a synchronized PNC shutdown of all actively PNC coordinated (coordinated by a PNC gateway) ComMChannels. The requests for a synchronized PNC shutdown are forwarded to all affected **<Bus>Nms** via Nm and handled in the same way as described in the previous section.

If a subordinated PNC node receives a top-level PNC coordinator Nm frame, then it will handle the frame as a usual NM message (update the local PN info and reset PN reset time).

**[SWS\_Nm\_00506]{DRAFT}** [If function **Nm\_RequestSynchronizedPncShutdown** is called, **NmSynchronizedPncShutdownEnabled** is set to TRUE and **Nm\_StandardBusType** is set to a type other than **NM\_BUSNM\_LOCALNM**, the function shall immediately forward the call to the affected **<Bus>Nm** by calling **<Bus>Nm\_RequestSynchronizedPncShutdown**. Otherwise the function shall return with **E\_NOT\_OK**.] ([SRS\\_Nm\\_02543](#))

**[SWS\_Nm\_00507]{DRAFT}** [The indication of a received PN shutdown message via callback function **Nm\_ForwardSynchronizedPncShutdown** shall be forwarded to **ComM** with the corresponding ComMChannel by calling **ComM\_Nm\_ForwardSynchronizedPncShutdown (<ComMChannel>)**.] ([SRS\\_Nm\\_02544](#))

## 7.5 Prerequisites of bus specific Network Management modules

This chapter gives an overview of the API calls that are used for the *Basic functionality*, the *NM Coordination functionality* and *synchronized PNC shutdown functionality* as well as information on the expected behavior of the **bus specific NM** for both functionalities.

For specific requirements of the interfaces and the configuration parameters for enabling/disabling the API's, refer to chapter 8.

### 7.5.1 Prerequisites for basic functionality

The **Nm** only acts as a forwarding layer between the **ComM** and the **bus specific NM** for the *basic functionality*.

All API calls made from the upper layer shall be forwarded to the corresponding API call of the lower layer. All callbacks of **Nm** invoked by the lower layer shall be forwarded to the corresponding callback of the upper layer.

The *Basic functionality* provides the following API calls to the ComM:

- [Nm\\_NetworkRequest](#) - [SWS\_Nm\_00032]
- [Nm\\_NetworkRelease](#) - [SWS\_Nm\_00046]
- [Nm\\_PassiveStartUp](#) - [SWS\_Nm\_00031]

**Note:** This implies that the **bus specific NM** provides the corresponding functions `<BusNm>_NetworkRequest`, `<BusNm>_NetworkRelease` and `<BusNm>_PassiveStartUp`.

The *Basic functionality* forwards the following API callbacks to the **ComM**:

- [Nm\\_NetworkStartIndication](#) - [SWS\_Nm\_00154]
- [Nm\\_NetworkMode](#) - [SWS\_Nm\_00156]
- [Nm\\_BusSleepMode](#) - [SWS\_Nm\_00162]
- [Nm\\_PrepareBusSleepMode](#) - [SWS\_Nm\_00159]

**Note:** This implies that the **ComM** provides the corresponding callback functions `ComM_Nm_NetworkStartIndication`, `ComM_Nm_NetworkMode`, `ComM_Nm_BusSleepMode` and `ComM_Nm_PrepareBusSleepMode`.

The **Nm** provides a number of API calls to the upper layers that are not used by **ComM**. These are provided for OEM specific extensions of the NM stack and are not required by any AUTOSAR module. They shall be forwarded to the corresponding API calls provided by the **bus specific NMs**.

The *Basic functionality* provides the following API calls to any OEM extension of an upper layer:

- [Nm\\_DisableCommunication](#) - [SWS\_Nm\_00033]
- [Nm\\_EnableCommunication](#) - [SWS\_Nm\_00034]
- [Nm\\_SetUserData](#) - [SWS\_Nm\_00035]
- [Nm\\_GetUserData](#) - [SWS\_Nm\_00036]
- [Nm\\_GetPduData](#) - [SWS\_Nm\_00037]
- [Nm\\_RepeatMessageRequest](#) - [SWS\_Nm\_00038]
- [Nm\\_GetNodeIdentifier](#) - [SWS\_Nm\_00039]

- [Nm\\_GetLocalNodeIdentifier](#) - [SWS\_Nm\_00040]
- [Nm\\_CheckRemoteSleepIndication](#) - [SWS\_Nm\_00042]
- [Nm\\_GetState](#) - [SWS\_Nm\_00043]

**Note:** This implies that the **bus specific NM** optionally provides the corresponding functions.

## 7.5.2 Prerequisites for NM Coordinator functionality

The [coordination algorithm](#) makes use of the following interfaces of the **bus specific NM**:

- [<BusNm>\\_NetworkRequest](#) - [SWS\_Nm\_00119]
- [<BusNm>\\_NetworkRelease](#) - [SWS\_Nm\_00119]
- [<BusNm>\\_RequestBusSynchronization](#) - [SWS\_Nm\_00119]
- [<BusNm>\\_CheckRemoteSleepIndication](#) - [SWS\_Nm\_00119]

**Note:** All NM networks configured to be part of a coordinated cluster of the *NM coordinator functionality* must have the corresponding Bus NM configured to be able to actively send out NM messages (e.g. `CANNM_PASSIVE_MODE_ENABLED = false`). As a result of this configuration restriction, all **BusNm** used by the *coordinator functionality* of the Nm module must provide the API [<BusNm>\\_NetworkRequest](#).

**Note:** Any configuration where a network is part of a coordinated cluster of networks where the corresponding **BusNm** is configured as passive is invalid.

**Note:** The [<BusNm>\\_RequestBusSynchronization](#) is called by **Nm** immediately before [<BusNm>\\_NetworkRelease](#) in order to allow non-synchronous networks to synchronize before the network is released. For some networks, this call has no meaning. The **bus specific NM** shall still provide this interface in order to support the generality of the *NM Coordinator functionality*, but can choose to provide an empty implementation.

**Rationale:** The [<BusNm>\\_CheckRemoteSleepIndication](#) is never explicitly mentioned in the [coordination algorithm](#). Its use is dependent on the implementation.

The [coordination algorithm](#) requires that the following callbacks of the **Nm** can be invoked by the **bus specific NM**:

- [Nm\\_NetworkStartIndication](#) - [SWS\_Nm\_00154]
- [Nm\\_NetworkMode](#) - [SWS\_Nm\_00156]
- [Nm\\_BusSleepMode](#) - [SWS\_Nm\_00162]
- [Nm\\_PrepareBusSleepMode](#) - [SWS\_Nm\_00159]



- [Nm\\_SynchronizeMode](#) - [SWS\_Nm\_91002]
- [Nm\\_RemoteSleepIndication](#) - [SWS\_Nm\_00192]
- [Nm\\_RemoteSleepCancellation](#) - [SWS\_Nm\_00193]
- [Nm\\_SynchronizationPoint](#) - [SWS\_Nm\_00194]

**Note:** The [Nm\\_NetworkStartIndication](#), [Nm\\_NetworkMode](#), [Nm\\_BusSleepMode](#) and [Nm\\_PrepareBusSleepMode](#) are used by the [coordination algorithm](#) to keep track of the status of the different networks and to handle aborted shutdown (see Chapter 7.3.3).

**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** when it detects that all other nodes on the network (except for itself) is ready to go to "bus-sleep mode". Some implementations will also make use of the API call `<BusNm>_CheckRemoteSleepIndication`.

**Note:** A **bus specific NM** which is included in a coordination cluster must monitor its bus to identify when all other nodes on the network is ready to go to sleep. When this occurs, the **bus specific NM** shall call the callback [Nm\\_RemoteSleepIndication](#) of **Nm**. (See [SWS\_Nm\_00192]).

**Note:** After a **bus specific NM** which is included in a coordination cluster has signaled to **Nm** that all other nodes on the network is ready to go to sleep (See [SWS\_Nm\_00192]), it must continue monitoring its bus to identify if any node starts requesting the network again, implying that the bus is no longer ready to go to sleep. When this occurs, the **bus specific NM** shall call the callback [Nm\\_RemoteSleepCancellation](#) of **Nm**. (See [SWS\_Nm\_00193]).

**Note:** The Remote Sleep Indication and Cancellation functionality is further specified in the respective bus specific NM.

**Rationale:** The [Nm\\_SynchronizationPoint](#) shall be called by the **bus specific NM** in order to inform the [coordination algorithm](#) of a suitable point in time to initiate the [coordinated shutdown](#). For cyclic networks this is typically at cycle boundaries. For non-cyclic networks this must be defined by other means. Each *NM Coordination Cluster* can be configured to make use of synchronization indications or not (See [SWS\_Nm\_00172]), and if they are used, the coordination algorithm filters indications and only acts on indications from networks that are configured as synchronizing networks.

**Note:** Please note for implementation of `<bus>Nm`: Cyclic networks invoke the [Nm\\_SynchronizationPoint](#) repeatedly when no other nodes request the network. The invocation is typically made at boundaries in the **bus specific NM** protocol when changes in the **NM** voting will occur.

It is assumed that any call to `<BusNm>_ReleaseNetwork` made between two of these `Nm_SynchronizationPoint` will be acted upon at the same point in time as the next `Nm_SynchronizationPoint` would have been invoked.

**Rationale:** The synchronization indication shall start when `Nm_RemoteSleepIndication` has been notified and continue until either the network has been released (`<BusNm>_NetworkRelease`) or the `Nm_RemoteSleepCancellation` is called.

**Note:** For the use case of coordinating Flexray-channel A + B if there is no other Network inside the `NM Cluster`, hence, if an `NM Coordinator` contains only one `NM Channel`, the `NmActiveCoordinator` for this `NmChannelConfig` needs to be set to `TRUE` and the `NmChannelSleepMaster` needs to be set to `FALSE` to allow the channel to coordinate itself. Note: The Value of "`NmSynchronizingNetwork`" is only relevant if this network is in the same coordination cluster with other networks.

### 7.5.3 Prerequisites for synchronized PNC shutdown functionality

The synchronized PNC shutdown functionality makes use of the following interface of the bus specific NM:

- `<BusNm>_RequestSynchronizedPncShutdown` - [SWS\_Nm\_00166][SWS\_Nm\_00506]

The synchronized PNC shutdown functionality requires that the following callback of the Nm can be invoked by the bus specific NM:

- `Nm_ForwardSynchronizedPncShutdown` - [SWS\_Nm\_91004]

The synchronized PNC shutdown functionality provides the following API, to be called by the ComM:

- `Nm_RequestSynchronizedPncShutdown` - [SWS\_Nm\_91005]

### 7.5.4 Configuration of global parameters for bus specific networks

The **Nm**'s configuration contains parameters that regulate support of optional features found in the **bus specific NMs**. Since **Nm** is only a pass-through interface layer regarding features that are not used by the *NM Coordinator functionality*, enabling these in **Nm**'s configuration will in many cases only enable the pass-through of the controlling API functions and the callback indications from the bus specific layers.

Many of the parameters defined for NM are used only as a source for global configuration of all bus specific NM modules. Corresponding parameters of the bus specific NMs are derived from these parameters.

## 7.6 NM\_BUSNM\_LOCALNM

**[SWS\_Nm\_00483]** [If BusNmType is NM\_BUSNM\_LOCALNM and ComM requests Nm\_PassiveStartUp() or Nm\_NetworkRequest() then Nm shall inform ComM about start of network by calling ComM\_Nm\_NetworkMode().

Rationale : Buses of type NM\_LOCAL\_NM which are coordinated do not have a network management message but are synchronized e.g. by a master - slave concept like LIN). These Bus-Types are always directly started on request by ComM but the shutdown will be done by coordinator algorithm.]()

## 7.7 Multicore Distribution

In its role as central module dealing with different network types the Nm interaction spans across partitions in case the Com-Stack is distributed and so shall provide required multi-core features to ensure a clean architecture and keep the network dependent clusters free of multi-partition (multi-core) addons.

**[SWS\_Nm\_00484]** [The Nm module shall apply appropriate mechanisms to allow calls of its APIs from other partitions than its main function, e.g. by providing a Nm satellite.] ([SRS\\_BSW\\_00459](#))

**[SWS\_Nm\_00485]** [Nm shall interact with <Bus>Nm (i.e. call <Bus>Nm APIs) only in the partition, where the respective <Bus>Nm module is assigned to.] ([SRS\\_BSW\\_00459](#))

**[SWS\_Nm\_00486]** [The Nm kernel shall be assigned to the same partition as ComM kernel in order to keep the interaction between these two modules on an intra-partition basis.] ([SRS\\_BSW\\_00459](#))

**Note:** Even though the basic software (and the Com-Stack in particular) is distributed across several partitions, ComM [9] and Nm Masters should reside in the same partition in order to keep mode interfaces between the two modules simple (for further information see chapter Master/Satellite-approach in [10, Guide to BSW Distribution]).

## 7.8 Additional Functionality

### 7.8.1 Nm\_CarWakeUpIndication

**[SWS\_Nm\_00252]** [If the <bus>Nm calls Nm\_CarWakeUpIndication and NmCarWakeUpCallout is defined, the NM Interface shall call the callout function defined by NmCarWakeUpCallout with nmNetworkHandle as parameter.] ([SRS\\_Nm\\_02503](#))

**[SWS\_Nm\_00285]** [If the <bus>Nm calls Nm\_CarWakeUpIndication and NmCarWakeUpCallout is not defined, the NM Interface shall call the function

BswM\_Nm\_CarWakeUpIndication with nmNetworkHandle as parameter.](SRS\_Nm\_02503)

**Note:** The application, called by NmCarWakeUpCallout, 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 drops the CWU request if the request is not repeated within a specific time.

**Note:** The callout is declared as specified within SWS\_BSW\_00039 and SWS\_BSW\_00135.

### 7.8.2 Nm\_StateChangeNotification

[SWS\_Nm\_00249] [If NmStateReportEnabled is set to TRUE and NmStateReportSignalRef is configured, when one of the state transitions mentioned in Table 7.1 occur, Nm\_StateChangeNotification shall call Com\_SendSignal(uint8, Com\_SignalIdType, const void\*) for the signal referenced by NmStateReportSignalRef with the value according to Table 7.1](SRS\_Nm\_00051)

**Note:** The transmitted signal has to be at least a 6 bit signal in Com that should be part of the NM message.

[SWS\_Nm\_00487] [When Nm\_StateChangeNotification is called to report a change of the Nm state, Nm shall call BswM\_Nm\_StateChangeNotification() with the reported current state.](SRS\_Nm\_00051)

| 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)     |

**Table 7.1: Network Management States**

[SWS\_Nm\_00501]{DRAFT} [If NmDynamicPncToChannelMappingEnabled is set to TRUE and Nm\_StateChangeNotification is called, then ComM\_Nm\_RepeatMessageLeftIndication() shall be called if nmPreviousState is set to NM\_STATE\_REPEAT\_MESSAGE.](/)

## 7.9 Error classification

Section 7.2 "Error Handling" of the document [2, SWS\_BSW General] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

### 7.9.1 Development Errors

[SWS\_Nm\_00232] [

| <i>Type of error</i>  | <i>Related error code</i> | <i>Error value</i> |
|---|---------------------------|--------------------|
| API service used without Nm interface initialization              | NM_E_UNINIT               | 0x00               |
| API Service called with wrong parameter but not with NULL-pointer | NM_E_INVALID_CHANNEL      | 0x01               |
| API service called with a NULL pointer                            | NM_E_PARAM_POINTER        | 0x02               |

]([SRS\\_BSW\\_00327](#), [SRS\\_BSW\\_00337](#), [SRS\\_BSW\\_00385](#), [SRS\\_BSW\\_00386](#))

### 7.9.2 Runtime Errors

There are no runtime errors.

### 7.9.3 Transient Faults

There are no transient faults.

### 7.9.4 Production Errors

There are no production errors.

### 7.9.5 Extended Production Errors

There are no extended production errors.

## 8 API specification

### 8.1 Imported types

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

[SWS\_Nm\_00117] [

| <i>Module</i>  | <i>Header File</i> | <i>Imported Type</i> |
|----------------|--------------------|----------------------|
| Com            | Com.h              | Com_SignalIdType     |
| ComStack_Types | ComStack_Types.h   | NetworkHandleType    |
|                | ComStack_Types.h   | PNCHandleType        |
| Std            | Std_Types.h        | Std_ReturnType       |
|                | Std_Types.h        | Std_VersionInfoType  |

]([SRS\\_BSW\\_00301](#))

### 8.2 Type definitions

#### 8.2.1 Nm\_ModeType

[SWS\_Nm\_00274] [

|                      |  |   |                        |
|----------------------|--|---|------------------------|
| <b>Name</b>          | Nm_ModeType                                  |   |                        |
| <b>Kind</b>          | Enumeration                                  |   |                        |
| <b>Range</b>         | 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           |
| <b>Description</b>   | Operational modes of the network management. |   |                        |
| <b>Available via</b> | NmStack_types.h                              |   |                        |

]([SRS\\_Nm\\_00044](#))

#### 8.2.2 Nm\_StateType

[SWS\_Nm\_00275] [

|                      |   |      |                        |
|----------------------|---|------|------------------------|
| <b>Name</b>          | Nm_StateType                                    |      |                        |
| <b>Kind</b>          | Enumeration                                     |      |                        |
| <b>Range</b>         | NM_STATE_UNINIT                                 | 0x00 | Uninitialized State    |
|                      | NM_STATE_BUS_SLEEP                              | 0x01 | Bus-Sleep State        |
|                      | NM_STATE_PREPARE_BUS_SLEEP                      | 0x02 | Prepare-Bus State      |
|                      | NM_STATE_READY_SLEEP                            | 0x03 | Ready Sleep State      |
|                      | NM_STATE_NORMAL_OPERATION                       | 0x04 | Normal Operation State |
|                      | NM_STATE_REPEAT_MESSAGE                         | 0x05 | Repeat Message State   |
|                      | NM_STATE_SYNCHRONIZE                            | 0x06 | Synchronize State      |
|                      | NM_STATE_OFFLINE                                | 0x07 | Offline State          |
| <b>Description</b>   | States of the network management state machine. |      |                        |
| <b>Available via</b> | NmStack_types.h                                 |      |                        |

](SRS\_Nm\_00050)

### 8.2.3 Nm\_BusNmType

[SWS\_Nm\_00276] [

|                      |                    |   |   |
|----------------------|--------------------|---|---|
| <b>Name</b>          | Nm_BusNmType       |   |   |
| <b>Kind</b>          | Enumeration        |   |   |
| <b>Range</b>         | NM_BUSNM_CANNM     | – | CAN NM type                                   |
|                      | NM_BUSNM_FRNM      | – | FR NM type                                    |
|                      | NM_BUSNM_UDPNM     | – | UDP NM type                                   |
|                      | NM_BUSNM_GENERICNM | – | Generic NM type                               |
|                      | NM_BUSNM_UNDEF     | – | NM type undefined; it shall be defined as FFh |
|                      | NM_BUSNM_J1939NM   | – | SAE J1939 NM type (address claiming)          |
|                      | NM_BUSNM_LOCALNM   | – | Local NM Type                                 |
| <b>Description</b>   | BusNm Type         |   |   |
| <b>Available via</b> | NmStack_types.h    |   |   |

](SRS\_Nm\_00044, SRS\_Nm\_00154, SRS\_Nm\_02515)

### 8.2.4 Nm\_ConfigType

[SWS\_Nm\_00282] [

|                      |  |   |
|----------------------|--|---|
| <b>Name</b>          | Nm_ConfigType                                  |   |
| <b>Kind</b>          | Structure                                      |   |
| <b>Elements</b>      | implementation specific                        |   |
|                      | <b>Type</b>                                    | - |
|                      | <b>Comment</b>                                 | - |
| <b>Description</b>   | Configuration data structure of the Nm module. |   |
| <b>Available via</b> | Nm.h   |   |

]([SRS\\_BSW\\_00414](#))

## 8.3 Function definitions

### 8.3.1 Standard services provided by NM Interface

#### 8.3.1.1 Nm\_Init

[SWS\_Nm\_00030] [

|                           |  |  |
|---------------------------|--|--|
| <b>Service Name</b>       | Nm_Init  |  |
| <b>Syntax</b>             | <pre>void Nm_Init (     const Nm_ConfigType* ConfigPtr )</pre> |  |
| <b>Service ID [hex]</b>   | 0x00   |  |
| <b>Sync/Async</b>         | Synchronous  |  |
| <b>Reentrancy</b>         | Non Reentrant  |  |
| <b>Parameters (in)</b>    | ConfigPtr  | Pointer to the selected configuration set. |
| <b>Parameters (inout)</b> | None   |  |
| <b>Parameters (out)</b>   | None   |  |
| <b>Return value</b>       | None   |  |
| <b>Description</b>        | Initializes the NM Interface.                                  |  |
| <b>Available via</b>      | Nm.h   |  |

]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00344](#), [SRS\\_BSW\\_00358](#), [SRS\\_BSW\\_00405](#), [SRS\\_BSW\\_00414](#))

[SWS\_Nm\_00127] [Caveats of `Nm_Init`: This service function has to be called after the initialization of the respective bus interface.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00283] [The Configuration pointer `ConfigPtr` shall always have a `NULL_PTR` value.]([SRS\\_BSW\\_00414](#))

**Note:** The Configuration pointer `ConfigPtr` is currently not used and shall therefore be set `NULL_PTR` value.



### 8.3.1.2 Nm\_PassiveStartUp

[SWS\_Nm\_00031] [

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Nm_PassiveStartUp   |  |
| <b>Syntax</b>             | Std_ReturnType Nm_PassiveStartUp (<br>NetworkHandleType NetworkHandle<br>)  |  |
| <b>Service ID [hex]</b>   | 0x01  |  |
| <b>Sync/Async</b>         | Asynchronous  |  |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise   |  |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel   |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Passive start of network management has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | This function calls the <BusNm>_PassiveStartUp function in case NmBusType is not set to NM_BUSNM_LOCALNM (e.g. CanNm_PassiveStartUp function is called for NM_BUSNM_CANNM). |  |
| <b>Available via</b>      | Nm.h  |  |

] ([SRS\\_Nm\\_00046](#), [SRS\\_Nm\\_00051](#), [SRS\\_Nm\\_00151](#), [SRS\\_Nm\\_02513](#), [SRS\\_Nm\\_02536](#))

[SWS\_Nm\_00128] [Caveats of Nm\_PassiveStartUp: The <BusNm> and the Nm itself are initialized correctly.] ([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00488] [If the pre-processor switch NmDevErrorDetect is set to TRUE, the function Nm\_PassiveStartUp shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter NetworkHandle is not a configured network handle.] ([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.1.3 Nm\_NetworkRequest

[SWS\_Nm\_00032] [

|                         |  |  |
|-------------------------|--|--|
| <b>Service Name</b>     | Nm_NetworkRequest  |  |
| <b>Syntax</b>           | Std_ReturnType Nm_NetworkRequest (<br>NetworkHandleType NetworkHandle<br>) |  |
| <b>Service ID [hex]</b> | 0x02   |  |
| <b>Sync/Async</b>       | Asynchronous   |  |
| <b>Reentrancy</b>       | Non-reentrant for the same NetworkHandle, reentrant otherwise              |  |





|                           |  |  |
|---------------------------|--|--|
| <b>Parameters (in)</b>    | NetworkHandle  | Identification of the NM-channel   |
| <b>Parameters (inout)</b> | None   |  |
| <b>Parameters (out)</b>   | None   |  |
| <b>Return value</b>       | Std_ReturnType   | E_OK: No error<br>E_NOT_OK: Requesting of bus communication has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | This function calls the <BusNm>_NetworkRequest (e.g. CanNm_NetworkRequest function is called if channel is configured as CAN) function in case NmBusType is not set to NM_BUSNM_LOCALNM. |  |
| <b>Available via</b>      | Nm.h   |  |

]([SRS\\_Nm\\_00046](#), [SRS\\_Nm\\_00047](#), [SRS\\_Nm\\_00051](#), [SRS\\_Nm\\_02513](#))

**[SWS\_Nm\_00129]** [Caveats of [Nm\\_NetworkRequest](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

**[SWS\_Nm\_00130]** [If [Nm\\_NetworkRequest](#) is called with a network handle where NmPassiveModeEnabled is set to TRUE it shall not execute any functionality and return with E\_NOT\_OK. If NmDevErrorDetect is set to TRUE then it shall raise the error NM\_E\_INVALID\_CHANNEL in this case.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00489]** [If the pre-processor switch NmDevErrorDetect is set to TRUE, the function [Nm\\_NetworkRequest](#) shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter NetworkHandle is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.1.4 Nm\_NetworkRelease

**[SWS\_Nm\_00046]** [

|                           |  |   |
|---------------------------|--|---|
| <b>Service Name</b>       | Nm_NetworkRelease  |   |
| <b>Syntax</b>             | Std_ReturnType Nm_NetworkRelease (<br>NetworkHandleType NetworkHandle<br>) |   |
| <b>Service ID [hex]</b>   | 0x03   |   |
| <b>Sync/Async</b>         | Asynchronous   |   |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise              |   |
| <b>Parameters (in)</b>    | NetworkHandle  | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None   |   |
| <b>Parameters (out)</b>   | None   |   |
| <b>Return value</b>       | Std_ReturnType   | E_OK: No error<br>E_NOT_OK: Releasing of bus communication has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |





|                      |   |
|----------------------|---|
| <b>Description</b>   | This function calls the <BusNm>_NetworkRelease bus specific function in case NmBusType is not set to NM_BUSNM_LOCALNM (e.g. CanNm_NetworkRelease function is called if channel is configured as CAN). |
| <b>Available via</b> | Nm.h  |

]([SRS\\_Nm\\_00048](#), [SRS\\_Nm\\_00051](#))

**[SWS\_Nm\_00131]** [Caveats of [Nm\\_NetworkRelease](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

**[SWS\_Nm\_00132]** [If [Nm\\_NetworkRelease](#) is called with a network handle where NmPassiveModeEnabled is set to TRUE it shall not execute any functionality and return with E\_NOT\_OK. If NmDevErrorDetect is set to TRUE then it shall raise the error NM\_E\_INVALID\_CHANNEL in this case.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00490]** [If the pre-processor switch NmDevErrorDetect is set to TRUE, the function [Nm\\_NetworkRelease](#) shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.2 Communication control services provided by NM Interface

The following services are provided by NM Interface to allow the Diagnostic Communication Manager (DCM) to control the transmission of NM Messages.

**Note:** To run the [coordination algorithm](#) correctly, it has to be ensured that NM PDU transmission ability is enabled before the ECU is shut down. If <BusNm>\_NetworkRelease is called while NM PDU transmission ability is disabled, the ECU will shut down after NM PDU transmission ability has been re-enabled again. Therefore the ECU can also shut down in case of race conditions (e.g. diagnostic session left shortly before enabling communication) or a wrong usage of communication control.

#### 8.3.2.1 Nm\_DisableCommunication

**[SWS\_Nm\_00033]** [

|                         |  |
|-------------------------|--|
| <b>Service Name</b>     | Nm_DisableCommunication  |
| <b>Syntax</b>           | Std_ReturnType Nm_DisableCommunication (<br>NetworkHandleType NetworkHandle<br>) |
| <b>Service ID [hex]</b> | 0x04   |
| <b>Sync/Async</b>       | Asynchronous   |





|                           |  |  |
|---------------------------|--|--|
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise  |  |
| <b>Parameters (in)</b>    | NetworkHandle  | Identification of the NM-channel   |
| <b>Parameters (inout)</b> | None   |  |
| <b>Parameters (out)</b>   | None   |  |
| <b>Return value</b>       | Std_ReturnType   | E_OK: No error<br>E_NOT_OK: Disabling of NM PDU transmission ability has failed.<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | Disables the NM PDU transmission ability. For that purpose <BusNm>_DisableCommunication shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM (e.g. CanNm_DisableCommunication function is called if channel is configured as CAN). |  |
| <b>Available via</b>      | Nm.h   |  |

]([SRS\\_Nm\\_02513](#), [SRS\\_Nm\\_02512](#))

[SWS\_Nm\_00133] [Caveats of [Nm\\_DisableCommunication](#): The <BusNm> and the Nm itself are initialized correctly.] ([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00134] [Configuration of [Nm\\_DisableCommunication](#): This function is only available if [NmComControlEnabled](#) is set to TRUE.] ([SRS\\_Nm\\_00150](#))

[SWS\_Nm\_00286] [If [Nm\\_DisableCommunication](#) is called with a network handle where [NmPassiveModeEnabled](#) is set to TRUE it shall not execute any functionality and return with E\_NOT\_OK. If [NmDevErrorDetect](#) is set to TRUE then it shall raise the error NM\_E\_INVALID\_CHANNEL in this case.] ([SRS\\_Nm\\_00150](#))

[SWS\_Nm\_00491] [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_DisableCommunication](#) shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter [NetworkHandle](#) is not a configured network handle.] ([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.2.2 Nm\_EnableCommunication

[SWS\_Nm\_00034] [

|                           |   |                                  |
|---------------------------|---|----------------------------------|
| <b>Service Name</b>       | Nm_EnableCommunication  |                                  |
| <b>Syntax</b>             | Std_ReturnType Nm_EnableCommunication (<br>NetworkHandleType NetworkHandle<br>) |                                  |
| <b>Service ID [hex]</b>   | 0x05  |                                  |
| <b>Sync/Async</b>         | Asynchronous  |                                  |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise                   |                                  |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None  |                                  |





|                         |  |   |
|-------------------------|--|---|
| <b>Parameters (out)</b> | None   |   |
| <b>Return value</b>     | Std_ReturnType   | E_OK: No error<br>E_NOT_OK: Enabling of NM PDU transmission ability has failed.<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>      | Enables the NM PDU transmission ability. For that purpose <BusNm>_EnableCommunication shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. CanNm_EnableCommunication function is called if channel is configured as CAN). |   |
| <b>Available via</b>    | Nm.h   |   |

]([SRS\\_Nm\\_00047](#), [SRS\\_Nm\\_02512](#))

**[SWS\_Nm\_00135]** [Caveats of [Nm\\_EnableCommunication](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

**[SWS\_Nm\_00136]** [Configuration of [Nm\\_EnableCommunication](#): This function is only available if [NmComControlEnabled](#) is set to TRUE.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00287]** [If [Nm\\_EnableCommunication](#) is called with a network handle where [NmPassiveModeEnabled](#) is set to TRUE it shall not execute any functionality and return with E\_NOT\_OK. If [NmDevErrorDetect](#) is set to TRUE then it shall raise the error NM\_E\_INVALID\_CHANNEL in this case.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00492]** [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_EnableCommunication](#) shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.3 Partial Network services provided by NM Interface

#### 8.3.3.1 Nm\_RequestSynchronizedPncShutdown

**[SWS\_Nm\_91005]**{DRAFT} [

|                         |  |  |
|-------------------------|--|--|
| <b>Service Name</b>     | Nm_RequestSynchronizedPncShutdown (draft)  |  |
| <b>Syntax</b>           | Std_ReturnType Nm_RequestSynchronizedPncShutdown (<br>NetworkHandleType NetworkHandle,<br>PNCHandleType PncId<br>) |  |
| <b>Service ID [hex]</b> | 0x24   |  |
| <b>Sync/Async</b>       | Synchronous  |  |
| <b>Reentrancy</b>       | Non Reentrant for the same NetworkHandle, reentrant otherwise  |  |
| <b>Parameters (in)</b>  | NetworkHandle  | Identification of the NM-channel   |
|                         | PncId  | Identification of the Pnc which is requested for a synchronized shutdown across the PNC network topology |





|                           |   |  |
|---------------------------|---|--|
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Request for a synchronized PNC shutdown has failed, e.g.<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | This function forward the request for a synchronized PNC shutdown of a particular PNC given by PncId to the affected <Bus>Nm by calling <Bus>Nm_RequestSynchronizedPncShutdown. The function call is only valid if NmStandardBusType is not set to NM_BUSNM_LOCALNM (e.g. CanNm_RequestSynchronizedPncShutdown function is called for NM_BUSNM_CANNM).<br><b>Tags:</b> atp.Status=draft |  |
| <b>Available via</b>      | Nm.h  |  |

]([SRS\\_Nm\\_02545](#), [SRS\\_Nm\\_02543](#))

**[SWS\_Nm\_00508]{DRAFT}** [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_RequestSynchronizedPncShutdown](#) shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4 Extra services provided by NM Interface

The following services are provided by NM Interface for OEM specific extensions of the NM stack and are not required by any AUTOSAR module.

#### 8.3.4.1 Nm\_SetUserData

**[SWS\_Nm\_00035]** [

|                           |  |   |
|---------------------------|--|---|
| <b>Service Name</b>       | Nm_SetUserData   |   |
| <b>Syntax</b>             | Std_ReturnType Nm_SetUserData (<br>NetworkHandleType NetworkHandle,<br>const uint8* nmUserDataPtr<br>) |   |
| <b>Service ID [hex]</b>   | 0x06   |   |
| <b>Sync/Async</b>         | Synchronous  |   |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise  |   |
| <b>Parameters (in)</b>    | NetworkHandle  | Identification of the NM-channel              |
|                           | nmUserDataPtr  | User data for the next transmitted NM message |
| <b>Parameters (inout)</b> | None   |   |
| <b>Parameters (out)</b>   | None   |   |





|                      |   |   |
|----------------------|---|---|
| <b>Return value</b>  | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Setting of user data has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>   | Set user data for NM messages transmitted next on the bus. For that purpose <BusNm>_SetUserData shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. Can Nm_SetUserData function is called if channel is configured as CAN). |   |
| <b>Available via</b> | Nm.h  |   |

]([SRS\\_Nm\\_02503](#))

**[SWS\_Nm\_00137]** [Caveats of `Nm_SetUserData`: The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

**[SWS\_Nm\_00138]** [Configuration of `Nm_SetUserData`: This function is only available if `NmUserDataEnabled` is set to TRUE.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00288]** [If `Nm_SetUserData` is called with a network handle where `NmPassiveModeEnabled` is set to TRUE it shall not execute any functionality and return with E\_NOT\_OK. If `NmDevErrorDetect` is set to TRUE then it shall raise the error NM\_E\_INVALID\_CHANNEL in this case.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00241]** [Configuration of `Nm_SetUserData`: If `NmComUserDataSupport` is TRUE the API `Nm_SetUserData` shall not be available.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00493]** [If the pre-processor switch `NmDevErrorDetect` is set to TRUE, the function `Nm_SetUserData` shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter `NetworkHandle` is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.2 Nm\_GetUserData

**[SWS\_Nm\_00036]** [

|                           |  |   |
|---------------------------|--|---|
| <b>Service Name</b>       | Nm_GetUserData   |   |
| <b>Syntax</b>             | <pre>Std_ReturnType Nm_GetUserData (     NetworkHandleType NetworkHandle,     uint8* nmUserDataPtr )</pre> |   |
| <b>Service ID [hex]</b>   | 0x07   |   |
| <b>Sync/Async</b>         | Synchronous  |   |
| <b>Reentrancy</b>         | Reentrant  |   |
| <b>Parameters (in)</b>    | NetworkHandle  | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None   |   |
| <b>Parameters (out)</b>   | nmUserDataPtr  | Pointer where user data out of the last successfully received NM message shall be copied to |





|                      |   |   |
|----------------------|---|---|
| <b>Return value</b>  | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Getting of user data has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>   | Get user data out of the last successfully received NM message. For that purpose <BusNm>_GetUserData shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. CanNm_GetUserData function is called if channel is configured as CAN). |   |
| <b>Available via</b> | Nm.h  |   |

]([SRS\\_Nm\\_02504](#))

[SWS\_Nm\_00139] [Caveats of [Nm\\_GetUserData](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00140] [Configuration of [Nm\\_GetUserData](#): This function is only available if [NmUserDataEnabled](#) is set to TRUE.]([SRS\\_Nm\\_00150](#))

[SWS\_Nm\_00494] [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_GetUserData](#) shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.3 Nm\_GetPduData

[SWS\_Nm\_00037] [

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Nm_GetPduData   |   |
| <b>Syntax</b>             | Std_ReturnType Nm_GetPduData (<br>NetworkHandleType NetworkHandle,<br>uint8* nmPduData<br>)   |   |
| <b>Service ID [hex]</b>   | 0x08  |   |
| <b>Sync/Async</b>         | Synchronous   |   |
| <b>Reentrancy</b>         | Reentrant   |   |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | nmPduData   | Pointer where NM PDU shall be copied to.  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Getting of NM PDU data has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | Get the whole PDU data out of the most recently received NM message. For that purpose <Bus Nm>_GetPduData shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. CanNm_GetPduData function is called if channel is configured as CAN). |   |
| <b>Available via</b>      | Nm.h  |   |

]([SRS\\_Nm\\_02506](#))



[SWS\_Nm\_00141] [Caveats of `Nm_GetPduData`: The **<BusNm>** and the **Nm** itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00495] [If the pre-processor switch `NmDevErrorDetect` is set to `TRUE`, the function `Nm_GetPduData` shall raise the error `NM_E_INVALID_CHANNEL` if the parameter `NetworkHandle` is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.4 Nm\_RepeatMessageRequest

[SWS\_Nm\_00038] [

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Nm_RepeatMessageRequest   |  |
| <b>Syntax</b>             | Std_ReturnType Nm_RepeatMessageRequest (<br>NetworkHandleType NetworkHandle<br>)  |  |
| <b>Service ID [hex]</b>   | 0x09  |  |
| <b>Sync/Async</b>         | Asynchronous  |  |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise   |  |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel   |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Setting of Repeat Message Request Bit has failed<br><br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | Set Repeat Message Request Bit for NM messages transmitted next on the bus. For that purpose <code>&lt;BusNm&gt;_RepeatMessageRequest</code> shall be called in case <code>NmBusType</code> is not set to <code>NM_BUSNM_LOCALNM</code> . (e.g. <code>CanNm_RepeatMessageRequest</code> function is called if channel is configured as CAN). This will force all nodes on the bus to transmit NM messages so that they can be identified. |  |
| <b>Available via</b>      | Nm.h  |  |

]([SRS\\_Nm\\_00153](#))

[SWS\_Nm\_00143] [Caveats of `Nm_RepeatMessageRequest`: The **<BusNm>** and the **Nm** itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00289] [If `Nm_RepeatMessageRequest` is called with a network handle where `NmPassiveModeEnabled` is set to `TRUE` it shall not execute any functionality and return with `E_NOT_OK`. If `NmDevErrorDetect` is set to `TRUE` then it shall raise the error `NM_E_INVALID_CHANNEL` in this case.]([SRS\\_Nm\\_00150](#))

[SWS\_Nm\_00496] [If the pre-processor switch `NmDevErrorDetect` is set to `TRUE`, the function `Nm_RepeatMessageRequest` shall raise the error `NM_E_INVALID_CHANNEL` if the parameter `NetworkHandle` is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.5 Nm\_GetNodeIdentifier

[SWS\_Nm\_00039] [

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Nm_GetNodeIdentifier  |   |
| <b>Syntax</b>             | <pre>Std_ReturnType Nm_GetNodeIdentifier (     NetworkHandleType NetworkHandle,     uint8* nmNodeIdPtr )</pre>  |   |
| <b>Service ID [hex]</b>   | 0x0a  |   |
| <b>Sync/Async</b>         | Synchronous   |   |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise   |   |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | nmNodeIdPtr   | Pointer where node identifier out of the last successfully received NM-message shall be copied to   |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Getting of the node identifier out of the last received NM-message has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | Get node identifier out of the last successfully received NM-message. The function <BusNm>_GetNodeIdentifier shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. CanNm_GetNodeIdentifier function is called if channel is configured as CAN). |   |
| <b>Available via</b>      | Nm.h  |   |

]([SRS\\_Nm\\_02505](#))

[SWS\_Nm\_00145] [Caveats of [Nm\\_GetNodeIdentifier](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00497] [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_GetNodeIdentifier](#) shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.6 Nm\_GetLocalNodeIdentifier

[SWS\_Nm\_00040] [

|                         |   |  |
|-------------------------|---|--|
| <b>Service Name</b>     | Nm_GetLocalNodeIdentifier   |  |
| <b>Syntax</b>           | <pre>Std_ReturnType Nm_GetLocalNodeIdentifier (     NetworkHandleType NetworkHandle,     uint8* nmNodeIdPtr )</pre> |  |
| <b>Service ID [hex]</b> | 0x0b  |  |





|                           |   |   |
|---------------------------|---|---|
| <b>Sync/Async</b>         | Synchronous   |   |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise   |   |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | nmNodeIDPtr   | Pointer where node identifier of the local node shall be copied to  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Getting of the node identifier of the local node has failed<br><br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>        | Get node identifier configured for the local node. For that purpose <BusNm>_GetLocalNodeIdentifier shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. Can Nm_GetLocalNodeIdentifier function is called if channel is configured as CAN). |   |
| <b>Available via</b>      | Nm.h  |   |

]([SRS\\_Nm\\_02508](#))

[SWS\_Nm\_00147] [Caveats of [Nm\\_GetLocalNodeIdentifier](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00498] [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_GetLocalNodeIdentifier](#) shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.7 Nm\_CheckRemoteSleepIndication

[SWS\_Nm\_00042] [

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Nm_CheckRemoteSleepIndication   |   |
| <b>Syntax</b>             | <pre>Std_ReturnType Nm_CheckRemoteSleepIndication (     NetworkHandleType nmNetworkHandle,     boolean* nmRemoteSleepIndPtr )</pre> |   |
| <b>Service ID [hex]</b>   | 0x0d  |   |
| <b>Sync/Async</b>         | Synchronous   |   |
| <b>Reentrancy</b>         | Non-reentrant for the same NetworkHandle, reentrant otherwise   |   |
| <b>Parameters (in)</b>    | nmNetworkHandle   | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | nmRemoteSleepIndPtr   | Pointer where check result of remote sleep indication shall be copied to  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: Checking of remote sleep indication bits has failed<br><br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |





|                      |  |
|----------------------|--|
| <b>Description</b>   | Check if remote sleep indication takes place or not. For that purpose <BusNm>_CheckRemoteSleepIndication shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. CanNm_CheckRemoteSleepIndication function is called if channel is configured as CAN). |
| <b>Available via</b> | Nm.h   |

]([SRS\\_Nm\\_02513](#))

**[SWS\_Nm\_00149]** [Caveats of [Nm\\_CheckRemoteSleepIndication](#): The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

**[SWS\_Nm\_00290]** [If [Nm\\_CheckRemoteSleepIndication](#) is called with a network handle where NmPassiveModeEnabled is set to TRUE it shall not execute any functionality and return with E\_NOT\_OK. If NmDevErrorDetect is set to TRUE then it shall raise the error NM\_E\_INVALID\_CHANNEL in this case.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00150]** [Configuration of [Nm\\_CheckRemoteSleepIndication](#): This function is only available if NmRemoteSleepIndEnabled is set to TRUE.]([SRS\\_Nm\\_00150](#))

**[SWS\_Nm\_00499]** [If the pre-processor switch NmDevErrorDetect is set to TRUE, the function [Nm\\_CheckRemoteSleepIndication](#) shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter nmNetworkHandle is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

### 8.3.4.8 Nm\_GetState

**[SWS\_Nm\_00043]** [

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Nm_GetState   |   |
| <b>Syntax</b>             | <pre>Std_ReturnType Nm_GetState (     NetworkHandleType nmNetworkHandle,     Nm_StateType* nmStatePtr,     Nm_ModeType* nmModePtr )</pre> |   |
| <b>Service ID [hex]</b>   | 0x0e  |   |
| <b>Sync/Async</b>         | Synchronous   |   |
| <b>Reentrancy</b>         | Reentrant   |   |
| <b>Parameters (in)</b>    | nmNetworkHandle   | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | 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 |





|                      |  |  |
|----------------------|--|--|
| <b>Return value</b>  | Std_ReturnType   | E_OK: No error<br>E_NOT_OK: Getting of NM state has failed<br>NetworkHandle does not exist (development only)<br>Module not yet initialized (development only) |
| <b>Description</b>   | Returns the state of the network management. The function <BusNm>_GetState shall be called in case NmBusType is not set to NM_BUSNM_LOCALNM. (e.g. CanNm_GetState function is called if channel is configured as CAN). |  |
| <b>Available via</b> | Nm.h   |  |

]([SRS\\_Nm\\_00050](#))

[SWS\_Nm\_00151] [Caveats of Nm\_GetState: The <BusNm> and the Nm itself are initialized correctly.]([SRS\\_BSW\\_00101](#), [SRS\\_BSW\\_00416](#))

[SWS\_Nm\_00500] [If the pre-processor switch NmDevErrorDetect is set to TRUE, the function Nm\_GetState shall raise the error NM\_E\_INVALID\_CHANNEL if the parameter nmNetworkHandle is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

#### 8.3.4.9 Nm\_GetVersionInfo

[SWS\_Nm\_00044] [

|                           |  |   |
|---------------------------|--|---|
| <b>Service Name</b>       | Nm_GetVersionInfo  |   |
| <b>Syntax</b>             | void Nm_GetVersionInfo (<br>Std_VersionInfoType* nmVerInfoPtr<br>) |   |
| <b>Service ID [hex]</b>   | 0x0f   |   |
| <b>Sync/Async</b>         | Synchronous  |   |
| <b>Reentrancy</b>         | Reentrant  |   |
| <b>Parameters (in)</b>    | None   |   |
| <b>Parameters (inout)</b> | None   |   |
| <b>Parameters (out)</b>   | nmVerInfoPtr   | Pointer to where to store the version information of this module. |
| <b>Return value</b>       | None   |   |
| <b>Description</b>        | This service returns the version information of this module.       |   |
| <b>Available via</b>      | Nm.h   |   |

]([SRS\\_BSW\\_00003](#), [SRS\\_BSW\\_00407](#), [SRS\\_BSW\\_00482](#))

#### 8.3.4.10 Nm\_PnLearningRequest

[SWS\_Nm\_91003]{DRAFT} [

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Nm_PnLearningRequest (draft)  |   |
| <b>Syntax</b>             | Std_ReturnType Nm_PnLearningRequest ( NetworkHandleType NetworkHandle )   |   |
| <b>Service ID [hex]</b>   | 0x22  |   |
| <b>Sync/Async</b>         | Asynchronous  |   |
| <b>Reentrancy</b>         | Non Reentrant for the same NetworkHandle, reentrant otherwise   |   |
| <b>Parameters (in)</b>    | NetworkHandle   | Identification of the NM-channel  |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | None  |   |
| <b>Return value</b>       | Std_ReturnType  | E_OK: No error<br>E_NOT_OK: PN Learning Request has failed or is not configured for the networkHandle |
| <b>Description</b>        | Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. For that purpose <BusNm>_PnLearningRequest shall be called (e.g. CanNm_PnLearningRequest function if channel is configured as CAN). This will force all nodes to enter the PNC Learning Phase and re-enter Repeat Message Stat. This is needed for the optional Dynamic PNC-to-channel-mapping feature.<br><b>Tags:</b> atp.Status=draft |   |
| <b>Available via</b>      | Nm.h  |   |

]()

**Note:** Caveats of [Nm\\_PnLearningRequest](#): The <BusNm> and the Nm itself are initialized correctly.

**[SWS\_Nm\_00502]{DRAFT}** [[Nm\\_PnLearningRequest](#) shall only be available if [Nm-DynamicPncToChannelMappingSupport](#) is set to TRUE.]()

**[SWS\_Nm\_00503]{DRAFT}** [If [Nm\\_PnLearningRequest](#) is called with a network handle for a bus system where [NmDynamicPncToChannelMappingEnabled](#) is set to FALSE Nm shall not execute any functionality and return with E\_NOT\_OK. If [NmDevErrorDetect](#) is set to TRUE then it shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) in this case.]()

**[SWS\_Nm\_00505]{DRAFT}** [If the pre-processor switch [NmDevErrorDetect](#) is set to TRUE, the function [Nm\\_PnLearningRequest](#) shall raise the error [NM\\_E\\_INVALID\\_CHANNEL](#) if the parameter [NetworkHandle](#) is not a configured network handle.]([SRS\\_BSW\\_00323](#), [SRS\\_BSW\\_00369](#), [SRS\\_BSW\\_00386](#))

## 8.4 Call-back notifications

Callback notifications are called by the lower layer's bus-specific Network Management modules. For the Base functionality of Nm ( [section 7.1](#) ) the call-backs shall be forwarded to the upper layer's ComM. For the NM Coordinator functionality of Nm ( [section 7.2](#) ) the call-backs will provide indications used to control the NM Coordinator and the optional Dynamic PNC-to-channel-mapping feature.

[SWS\_Nm\_00028] [All callbacks of the Nm shall assume that they can run either in task or in interrupt context.] ([SRS\\_BSW\\_00333](#))

## 8.4.1 Standard Call-back notifications

### 8.4.1.1 Nm\_NetworkStartIndication

[SWS\_Nm\_00154] [

|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | Nm_NetworkStartIndication  |                                  |
| <b>Syntax</b>             | <pre>void Nm_NetworkStartIndication (     NetworkHandleType nmNetworkHandle )</pre>  |                                  |
| <b>Service ID [hex]</b>   | 0x11   |                                  |
| <b>Sync/Async</b>         | Asynchronous   |                                  |
| <b>Reentrancy</b>         | Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | 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. |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

] ([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_02513](#))

[SWS\_Nm\_00155] [The indication through callback function [Nm\\_NetworkStartIndication](#): shall be forwarded to **ComM** by calling the `ComM_Nm_NetworkStartIndication`.] ([SRS\\_Nm\\_02513](#))

### 8.4.1.2 Nm\_NetworkMode

[SWS\_Nm\_00156] [

|                         |  |                                  |
|-------------------------|--|----------------------------------|
| <b>Service Name</b>     | Nm_NetworkMode   |                                  |
| <b>Syntax</b>           | <pre>void Nm_NetworkMode (     NetworkHandleType nmNetworkHandle )</pre> |                                  |
| <b>Service ID [hex]</b> | 0x12   |                                  |
| <b>Sync/Async</b>       | Asynchronous   |                                  |
| <b>Reentrancy</b>       | Reentrant  |                                  |
| <b>Parameters (in)</b>  | nmNetworkHandle  | Identification of the NM-channel |





|                           |  |
|---------------------------|--|
| <b>Parameters (inout)</b> | None   |
| <b>Parameters (out)</b>   | None   |
| <b>Return value</b>       | None   |
| <b>Description</b>        | Notification that the network management has entered Network Mode. |
| <b>Available via</b>      | Nm.h   |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_00051](#))

**[SWS\_Nm\_00158]** [The indication through callback function `Nm_NetworkMode`: shall be forwarded to **ComM** by calling the `ComM_Nm_NetworkMode`.]([SRS\\_Nm\\_00051](#))

### 8.4.1.3 Nm\_BusSleepMode

**[SWS\_Nm\_00162]** [

|                           |   |                                  |
|---------------------------|---|----------------------------------|
| <b>Service Name</b>       | Nm_BusSleepMode   |                                  |
| <b>Syntax</b>             | <pre>void Nm_BusSleepMode (     NetworkHandleType nmNetworkHandle )</pre> |                                  |
| <b>Service ID [hex]</b>   | 0x14  |                                  |
| <b>Sync/Async</b>         | Asynchronous  |                                  |
| <b>Reentrancy</b>         | Reentrant   |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle   | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None  |                                  |
| <b>Parameters (out)</b>   | None  |                                  |
| <b>Return value</b>       | None  |                                  |
| <b>Description</b>        | Notification that the network management has entered Bus-Sleep Mode.      |                                  |
| <b>Available via</b>      | Nm.h  |                                  |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_00051](#))

**[SWS\_Nm\_00163]** [The indication through callback function `Nm_BusSleepMode`: shall be forwarded to **ComM** by calling the `ComM_Nm_BusSleepMode`.]([SRS\\_Nm\\_00051](#))

### 8.4.1.4 Nm\_PrepareBusSleepMode

**[SWS\_Nm\_00159]** [



|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | Nm_PrepareBusSleepMode   |                                  |
| <b>Syntax</b>             | <pre>void Nm_PrepareBusSleepMode (     NetworkHandleType nmNetworkHandle )</pre> |                                  |
| <b>Service ID [hex]</b>   | 0x13   |                                  |
| <b>Sync/Async</b>         | Asynchronous   |                                  |
| <b>Reentrancy</b>         | Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | Notification that the network management has entered Prepare Bus-Sleep Mode.     |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

] ([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_00051](#))

**[SWS\_Nm\_00161]** [The indication through callback function [Nm\\_PrepareBusSleepMode](#): shall be forwarded to **ComM** by calling `ComM_Nm_PrepareBusSleepMode`.] ([SRS\\_Nm\\_00051](#))

#### 8.4.1.5 NM\_SynchronizeMode

**[SWS\_Nm\_91002]** [

|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | Nm_SynchronizeMode   |                                  |
| <b>Syntax</b>             | <pre>void Nm_SynchronizeMode (     NetworkHandleType nmNetworkHandle )</pre> |                                  |
| <b>Service ID [hex]</b>   | 0x21   |                                  |
| <b>Sync/Async</b>         | Asynchronous   |                                  |
| <b>Reentrancy</b>         | Reentrant but not for the same channel                                       |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | Notification that the network management has entered Synchronize Mode.       |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

]()

#### 8.4.1.6 Nm\_RemoteSleepIndication

**[SWS\_Nm\_00192]** [

|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | Nm_RemoteSleepIndication   |                                  |
| <b>Syntax</b>             | <pre>void Nm_RemoteSleepIndication (     NetworkHandleType nmNetworkHandle )</pre>   |                                  |
| <b>Service ID [hex]</b>   | 0x17   |                                  |
| <b>Sync/Async</b>         | Asynchronous   |                                  |
| <b>Reentrancy</b>         | Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode. |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_00052](#))

[SWS\_Nm\_00277] [Configuration of `Nm_RemoteSleepIndication`: This function is only available if `NmRemoteSleepIndEnabled` is set to TRUE.] ([SRS\\_Nm\\_00150](#))

The notification that all other nodes on the network are ready to enter Bus-Sleep Mode is only needed for internal purposes of the NM Coordinator.

**Note:** When *NM Coordinator functionality* is disabled `Nm_RemoteSleepIndication` can be an empty function.

### 8.4.1.7 Nm\_RemoteSleepCancellation

[SWS\_Nm\_00193] [

|                           |   |                                  |
|---------------------------|---|----------------------------------|
| <b>Service Name</b>       | Nm_RemoteSleepCancellation  |                                  |
| <b>Syntax</b>             | <pre>void Nm_RemoteSleepCancellation (     NetworkHandleType nmNetworkHandle )</pre>  |                                  |
| <b>Service ID [hex]</b>   | 0x18  |                                  |
| <b>Sync/Async</b>         | Asynchronous  |                                  |
| <b>Reentrancy</b>         | Reentrant   |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle   | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None  |                                  |
| <b>Parameters (out)</b>   | None  |                                  |
| <b>Return value</b>       | None  |                                  |
| <b>Description</b>        | Notification that the network management has detected that not all other nodes on the network are longer ready to enter Bus-Sleep Mode. |                                  |
| <b>Available via</b>      | Nm.h  |                                  |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_02509](#))

[SWS\_Nm\_00278] [Configuration of [Nm\\_RemoteSleepCancellation](#): This function is only available if [NmRemoteSleepIndEnabled](#) is set to TRUE.] ([SRS\\_Nm\\_00150](#))

The notification that not all other nodes on the network are longer ready to enter Bus-Sleep Mode is only needed for internal purposes of the NM Coordinator.

**Note:** When *NM Coordinator functionality* is disabled [Nm\\_RemoteSleepCancellation](#) can be an empty function.

### 8.4.1.8 Nm\_SynchronizationPoint

[SWS\_Nm\_00194] [

|                           |   |                                  |
|---------------------------|---|----------------------------------|
| <b>Service Name</b>       | Nm_SynchronizationPoint   |                                  |
| <b>Syntax</b>             | <pre>void Nm_SynchronizationPoint (     NetworkHandleType nmNetworkHandle )</pre>   |                                  |
| <b>Service ID [hex]</b>   | 0x19  |                                  |
| <b>Sync/Async</b>         | Asynchronous  |                                  |
| <b>Reentrancy</b>         | Reentrant   |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle   | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None  |                                  |
| <b>Parameters (out)</b>   | None  |                                  |
| <b>Return value</b>       | None  |                                  |
| <b>Description</b>        | Notification to the NM Coordinator functionality that this is a suitable point in time to initiate the coordinated shutdown on. |                                  |
| <b>Available via</b>      | Nm.h  |                                  |

] ([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_02516](#))

The notification that this is a suitable point in time to initiate the [coordinated shutdown](#) is only needed for internal purposes of the NM Coordinator.

### 8.4.1.9 Nm\_CoordReadyToSleepIndication

[SWS\_Nm\_00254] [

|                         |  |  |
|-------------------------|--|--|
| <b>Service Name</b>     | Nm_CoordReadyToSleepIndication   |  |
| <b>Syntax</b>           | <pre>void Nm_CoordReadyToSleepIndication (     NetworkHandleType nmChannelHandle )</pre> |  |
| <b>Service ID [hex]</b> | 0x1e   |  |





|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Sync/Async</b>         | Synchronous  |                                  |
| <b>Reentrancy</b>         | Non Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmChannelHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_02535](#))

**[SWS\_Nm\_00255]** [Configuration of [Nm\\_CoordReadyToSleepIndication](#): Optional

If [NmCoordinatorSyncSupport](#) is set to `TRUE`, the Nm shall provide the API [Nm\\_CoordReadyToSleepIndication](#).]([SRS\\_Nm\\_00150](#))

#### 8.4.1.10 Nm\_CoordReadyToSleepCancellation

**[SWS\_Nm\_00272]** [

|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | Nm_CoordReadyToSleepCancellation   |                                  |
| <b>Syntax</b>             | <pre>void Nm_CoordReadyToSleepCancellation (     NetworkHandleType nmChannelHandle )</pre>                 |                                  |
| <b>Service ID [hex]</b>   | 0x1f   |                                  |
| <b>Sync/Async</b>         | Synchronous  |                                  |
| <b>Reentrancy</b>         | Non Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmChannelHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0. |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_02535](#))

**[SWS\_Nm\_00273]** [Configuration of [Nm\\_CoordReadyToSleepCancellation](#): Optional

If [NmCoordinatorSyncSupport](#) is set to `TRUE`, the Nm shall provide the API [Nm\\_CoordReadyToSleepCancellation](#).]([SRS\\_Nm\\_00150](#))

### 8.4.1.11 Nm\_ForwardSynchronizedPncShutdown

[SWS\_Nm\_91004]{DRAFT} [

|                           |  |
|---------------------------|--|
| <b>Service Name</b>       | Nm_ForwardSynchronizedPncShutdown (draft)  |
| <b>Syntax</b>             | <pre>void Nm_ForwardSynchronizedPncShutdown (     void )</pre>   |
| <b>Service ID [hex]</b>   | 0x23   |
| <b>Sync/Async</b>         | Synchronous  |
| <b>Reentrancy</b>         | Non Reentrant -reentrant for the same NetworkHandle, reentrant otherwise   |
| <b>Parameters (in)</b>    | None   |
| <b>Parameters (inout)</b> | None   |
| <b>Parameters (out)</b>   | None   |
| <b>Return value</b>       | None   |
| <b>Description</b>        | <p>Notification that the network management has received a PN shutdown message on a particular NM-channel. This is used to grant a nearly synchronized PNC shutdown across the entire PN topology.</p> <p><b>Tags:</b>atp.Status=draft</p> |
| <b>Available via</b>      | Nm.h   |

]([SRS\\_Nm\\_02544](#))

## 8.4.2 Extra Call-back notifications

The following call-back notifications are provided by NM Interface for OEM specific extensions of bus specific NM components and are not required by any AUTOSAR module. In the context of the Basic functionality and NM Coordinator functionality they have no specific usage.

### 8.4.2.1 Nm\_PduRxIndication

[SWS\_Nm\_00112] [

|                         |  |
|-------------------------|--|
| <b>Service Name</b>     | Nm_PduRxIndication   |
| <b>Syntax</b>           | <pre>void Nm_PduRxIndication (     NetworkHandleType nmNetworkHandle )</pre> |
| <b>Service ID [hex]</b> | 0x15   |
| <b>Sync/Async</b>       | Asynchronous   |
| <b>Reentrancy</b>       | Reentrant  |





|                           |   |                                  |
|---------------------------|---|----------------------------------|
| <b>Parameters (in)</b>    | nmNetworkHandle                                   | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None  |                                  |
| <b>Parameters (out)</b>   | None  |                                  |
| <b>Return value</b>       | None  |                                  |
| <b>Description</b>        | Notification that a NM message has been received. |                                  |
| <b>Available via</b>      | Nm.h  |                                  |

]([SRS\\_BSW\\_00359](#))

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

**[SWS\_Nm\_00164]** [Configuration of [Nm\\_PduRxIndication](#): This function is only available if [NmPduRxIndicationEnabled](#) is set to TRUE.]([SRS\\_Nm\\_00150](#))

#### 8.4.2.2 Nm\_StateChangeNotification

**[SWS\_Nm\_00114]** [

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

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_00050](#))

The notification that the state of the bus-specific NM has changed is only needed for OEM specific extensions and for the optional Dynamic PNC-to-channel-mapping feature.

**[SWS\_Nm\_00165]** [Configuration of [Nm\\_StateChangeNotification](#): This function is only available if [NmStateChangeIndEnabled](#) is set to TRUE.]([SRS\\_Nm\\_00150](#))

### 8.4.2.3 Nm\_RepeatMessageIndication

[SWS\_Nm\_00230] [

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Nm_RepeatMessageIndication  |   |
| <b>Syntax</b>             | <pre>void Nm_RepeatMessageIndication (     NetworkHandleType nmNetworkHandle,     boolean pnLearningBitSet )</pre>  |   |
| <b>Service ID [hex]</b>   | 0x1a  |   |
| <b>Sync/Async</b>         | Asynchronous  |   |
| <b>Reentrancy</b>         | Reentrant   |   |
| <b>Parameters (in)</b>    | nmNetworkHandle   | Identification of the NM-channel  |
|                           | pnLearningBitSet  | TRUE if also the Partial Network Learning Bit was received, FALSE otherwise |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | None  |   |
| <b>Return value</b>       | None  |   |
| <b>Description</b>        | Service to indicate that an NM message with set Repeat Message Re- quest Bit has been received. This is needed for node detection and the Dynamic PNC-to-channel-mapping feature. |   |
| <b>Available via</b>      | Nm.h  |   |

]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_00153](#))

The notification that an NM message with the set Repeat Message Bit has been received is only needed for OEM specific extensions and for the optional Dynamic PNC-to-channel-mapping feature.

[SWS\_Nm\_00504]{DRAFT} [If [Nm\\_RepeatMessageIndication](#) is called with [pnLearningBitSet](#) set to TRUE and [NmDynamicPncToChannelMappingEnabled](#) is set to TRUE for the provided [nmNetworkHandle](#) Nm shall call [ComM\\_Nm\\_PncLearningBitIndication](#) with the corresponding network handle.]  
()

### 8.4.2.4 Nm\_TxTimeoutException

[SWS\_Nm\_00234] [

|                         |   |  |
|-------------------------|---|--|
| <b>Service Name</b>     | Nm_TxTimeoutException   |  |
| <b>Syntax</b>           | <pre>void Nm_TxTimeoutException (     NetworkHandleType nmNetworkHandle )</pre> |  |
| <b>Service ID [hex]</b> | 0x1b  |  |
| <b>Sync/Async</b>       | Asynchronous  |  |
| <b>Reentrancy</b>       | Reentrant   |  |



△

|                           |   |   |
|---------------------------|---|---|
| <b>Parameters (in)</b>    | nmNetworkHandle   | - |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | None  |   |
| <b>Return value</b>       | None  |   |
| <b>Description</b>        | Service to indicate that an attempt to send an NM message failed. |   |
| <b>Available via</b>      | Nm.h  |   |

 ]([SRS\\_BSW\\_00359](#))

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

### 8.4.2.5 Nm\_CarWakeUpIndication

[SWS\_Nm\_00250] [

|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | Nm_CarWakeUpIndication   |                                  |
| <b>Syntax</b>             | <pre>void Nm_CarWakeUpIndication (     NetworkHandleType nmChannelHandle )</pre> |                                  |
| <b>Service ID [hex]</b>   | 0x1d   |                                  |
| <b>Sync/Async</b>         | Synchronous  |                                  |
| <b>Reentrancy</b>         | Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmChannelHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | This function is called by a <Bus>Nm to indicate reception of a CWU request.     |                                  |
| <b>Available via</b>      | Nm.h   |                                  |

 ]([SRS\\_BSW\\_00359](#), [SRS\\_Nm\\_02503](#))

[SWS\_Nm\_00251] [Configuration of [Nm\\_CarWakeUpIndication](#): Optional  
 If [NmCarWakeUpRxEnabled](#) is TRUE, The Nm shall provide the API [Nm\\_CarWakeUpIndication](#).] ([SRS\\_Nm\\_00150](#))

## 8.5 Scheduled functions

Since the Base functionality (Chapter 7.1) does not contain any logic that needs to be invoked outside the scope of call from the upper or lower layer, the main function is only needed to implement the NM Coordinator functionality (Chapter 7.2).

[SWS\_Nm\_00020] [A scheduled main function shall only contain logic related to the *NM Coordinator functionality*.] ([SRS\\_BSW\\_00373](#))



**[SWS\_Nm\_00121]** [In case the main function is called before the Nm has been initialized, the main function shall immediately return without yielding an error.] ([SRS\\_BSW\\_00450](#))

**Rationale:** In case the NM Coordinator functionality is not used and/or disabled, calling the main function shall not yield in an error, but nothing should be performed.

### 8.5.1 Nm\_MainFunction

**[SWS\_Nm\_00118]** [

|                         |   |
|-------------------------|---|
| <b>Service Name</b>     | Nm_MainFunction   |
| <b>Syntax</b>           | <pre>void Nm_MainFunction (     void )</pre>  |
| <b>Service ID [hex]</b> | 0x10  |
| <b>Description</b>      | This function implements the processes of the NM Interface, which need a fix cyclic scheduling. |
| <b>Available via</b>    | SchM_Nm.h   |

] ([SRS\\_BSW\\_00424](#), [SRS\\_BSW\\_00425](#))

**[SWS\_Nm\_00279]** [If `NmCoordinatorSupportEnabled` is set to TRUE, the `Nm_MainFunction` API shall be available.] ([SRS\\_Nm\\_00150](#))

## 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.

**[SWS\_Nm\_00119]** [

| <b>API Function</b>  | <b>Header File</b> | <b>Description</b>   |
|----------------------|--------------------|--|
| ComM_Nm_BusSleepMode | ComM_Nm.h          | 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. |





| API Function                           | Header File | Description   |
|--|-------------|---|
| ComM_Nm_ForwardSynchronizedPncShutdown | ComM_Nm.h   | If an ECU in role of an intermediate PNC coordinator receives a PN shutdown message via a <Bus>Nm, then ComM is immediately indicated via ComM_Nm_ForwardSynchronizedPncShutdown to forward the request for a synchronized PNC shutdown of the affected PNCs. Therefore, ComM will immediately release the affected PNC state machines and forward the PN information to the affected ComM Channels and the corresponding NM channels, respectively. Note: This supports a nearly synchronized PNC shutdown across the PN topology from the top-level PNC coordinator down to the subordinated PNC node.<br><b>Tags:</b> atp.Status=draft |
| ComM_Nm_NetworkMode                    | ComM_Nm.h   | Notification that the network management has entered Network Mode.  |
| ComM_Nm_NetworkStartIndication         | ComM_Nm.h   | 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_PrepareBusSleepMode            | ComM_Nm.h   | Notification that the network management has entered Prepare Bus-Sleep Mode. Reentrancy: Reentrant (but not for the same NM-Channel)  |
| ComM_Nm_RestartIndication              | ComM_Nm.h   | 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.   |

]([SRS\\_Nm\\_02515](#), [SRS\\_Nm\\_02536](#))

## 8.6.2 Optional Interfaces

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

[SWS\_Nm\_00166] [

| API Function                | Header File | Description  |
|-----------------------------|-------------|--|
| BswM_Nm_CarWakeUpIndication | BswM_Nm.h   | Function called by Nm to indicate a CarWakeUp.   |
| CanNm_PassiveStartUp        | CanNm.h     | 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.<br><br>Caveats: CanNm is initialized correctly. |





| API Function                          | Header File | Description   |
|---------------------------------------|-------------|---|
| CanNm_PnLearningRequest               | CanNm.h     | Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. This will force all nodes to enter the PNC Learning Phase. This is needed for the optional Dynamic PNC-to-channel-mapping feature.<br><b>Tags:</b> atp.Status=draft            |
| CanNm_RequestSynchronizedPnc Shutdown | CanNm.h     | Requests transmission of a NM-PDU with PNSR bit set to 1 (PN shutdown message)<br><b>Tags:</b> atp.Status=draft   |
| Com_SendSignal                        | Com.h       | The service Com_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.  |
| ComM_Nm_PncLearningBitIndication      | ComM_Nm.h   | Service to indicate that an NM message with set PNC Learning Bit has been received.<br><b>Tags:</b> atp.Status=draft  |
| ComM_Nm_RepeatMessageLeft Indication  | ComM_Nm.h   | Notification that the state of all <BusNm> has left RepeatMessage. This interface is used to indicate by the optional Dynamic PNC-to-channel-mapping feature to indicate that learning phase ends.<br><b>Tags:</b> atp.Status=draft   |
| Det_ReportError                       | Det.h       | Service to report development errors.   |
| FrNm_PassiveStartUp                   | FrNm.h      | Initiates the Passive Startup of the FlexRay NM.  |
| FrNm_PnLearningRequest                | FrNm.h      | Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. This will force all nodes to enter the PNC Learning Phase. This is needed for the optional Dynamic PNC-to-channel-mapping feature.<br><b>Tags:</b> atp.Status=draft            |
| FrNm_RequestSynchronizedPnc Shutdown  | FrNm.h      | Requests transmission of a NM-PDU with PNSR bit set to 1 (PN shutdown message)<br><b>Tags:</b> atp.Status=draft   |
| J1939Nm_PassiveStartUp                | J1939Nm.h   | Passive startup of the NM. It triggers the transition from Bus-Sleep Mode to the Network Mode without requesting the network.   |
| UdpNm_PassiveStartUp                  | UdpNm.h     | Passive startup of the AUTOSAR UdpNm. It triggers the transition from Bus-Sleep Mode or Prepare Bus Sleep Mode to the Network Mode in Repeat Message State.<br>Caveats: UdpNm is initialized correctly.   |
| UdpNm_PnLearningRequest               | UdpNm.h     | Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. This will force all nodes on the bus to enter the PNC Learning Phase. This is needed for the optional Dynamic PNC-to-channel-mapping feature.<br><b>Tags:</b> atp.Status=draft |
| UdpNm_RequestSynchronizedPnc Shutdown | UdpNm.h     | Requests transmission of a NM-PDU with PNSR bit set to 1 (PN shutdown message)<br><b>Tags:</b> atp.Status=draft   |

|(SRS\_Nm\_00150, SRS\_Nm\_02515)

### 8.6.3 Configurable Interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kind of interfaces are not fixed because they are configurable.

#### 8.6.3.1 NmCarWakeUpCallout

[SWS\_Nm\_00291] [

|                           |  |                                  |
|---------------------------|--|----------------------------------|
| <b>Service Name</b>       | <NmCarWakeUpCallout>   |                                  |
| <b>Syntax</b>             | <pre>void &lt;NmCarWakeUpCallout&gt; (     NetworkHandleType nmNetworkHandle )</pre> |                                  |
| <b>Service ID [hex]</b>   | 0x20   |                                  |
| <b>Sync/Async</b>         | Asynchronous   |                                  |
| <b>Reentrancy</b>         | Non Reentrant  |                                  |
| <b>Parameters (in)</b>    | nmNetworkHandle  | Identification of the NM-channel |
| <b>Parameters (inout)</b> | None   |                                  |
| <b>Parameters (out)</b>   | None   |                                  |
| <b>Return value</b>       | None   |                                  |
| <b>Description</b>        | Callout function to be called by Nm_CarWakeUpIndication()                            |                                  |
| <b>Available via</b>      | Nm_Externals.h   |                                  |

]([SRS\\_Nm\\_02504](#))

## 8.7 Version Check

For details refer to the chapter 5.1.8 "Version Check" in [2, SWS\_BSWGeneral].

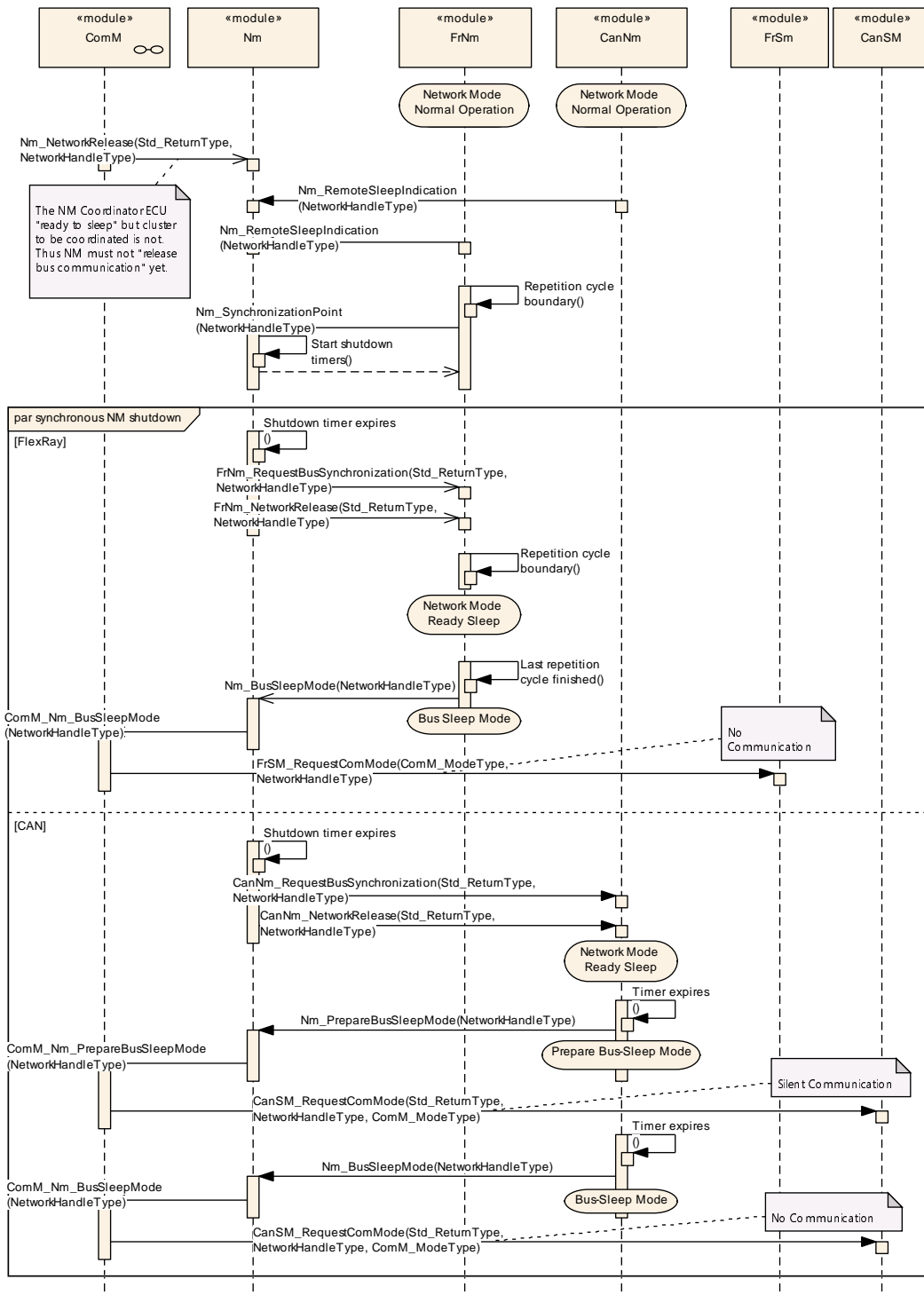
## 9 Sequence diagrams

### 9.1 Basic functionality

The role of the *Basic functionality* of the **Nm** is to act as a dispatcher of functions between the ComM and the Bus Specific NM modules. Therefore, no sequence diagram is provided.

### 9.2 Seq of NM Coordinator functionality

Figure shows the sequence diagram for the shutdown of network of the *NM Coordinator* functionality.



**Figure 9.1: Nm Coordination**

## 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. [section 10.1](#) describes fundamentals. [section 10.2](#), [section 10.3](#) and [section 10.4](#) specifies the structure (containers) and the parameters of the Nm. The [section 10.5](#) specifies published information of the Nm.

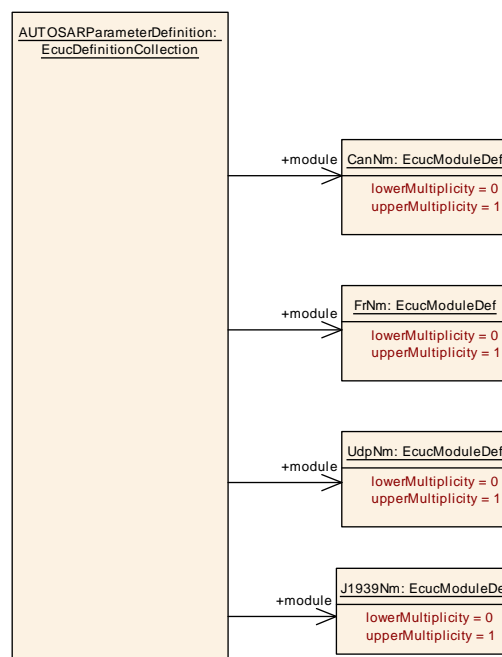


Figure 10.1: Network Management Overview

### 10.1 How to read this chapter

For details refer to the [2, chapter 10.1 “Introduction to configuration specification” in SWS\_BSWGeneral]

### 10.2 Configuration parameters

The following Chapters summarize all configuration parameters for the Nm. The detailed meanings of most parameters are described in [chapter 7](#) and [chapter 8](#). Note that the behavior and configuration of Nm is closely dependent on the behavior and configuration of the different bus specific NM modules used.

## 10.2.1 Nm

|                                   |   |   |
|-----------------------------------|---|---|
| <b>Module SWS Item</b>            | ECUC_Nm_00243                                   |   |
| <b>Module Name</b>                | Nm  |   |
| <b>Module Description</b>         | The Generic Network Management Interface module |   |
| <b>Post-Build Variant Support</b> | false   |   |
| <b>Supported Config Variants</b>  | VARIANT-LINK-TIME, VARIANT-PRE-COMPILE          |   |
| <b>Included Containers</b>        |   |   |
| <b>Container Name</b>             | <b>Multiplicity</b>                             | <b>Scope / Dependency</b>   |
| NmChannelConfig                   | 1..*  | This container contains the configuration (parameters) of the bus channel(s). The channel parameter shall be harmonized within the whole communication stack. |
| NmGlobalConfig                    | 1   | This container contains all global configuration parameters of the Nm Interface.  |

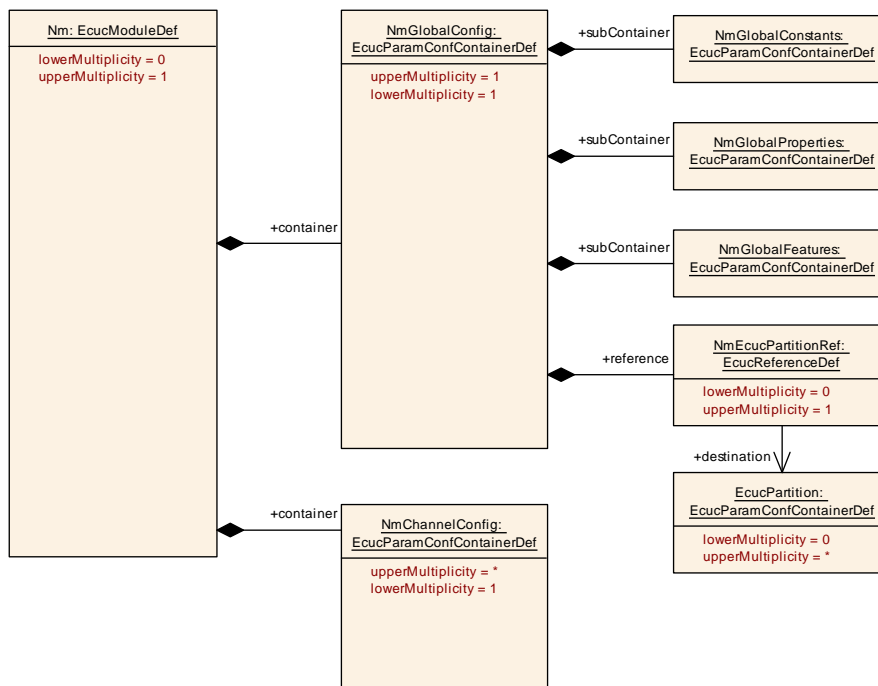


Figure 10.2: Nm configuration container overview

## 10.3 Global configurable parameters

### 10.3.1 NmGlobalConfig

|                         |                 |
|-------------------------|-----------------|
| <b>SWS Item</b>         | [ECUC_Nm_00196] |
| <b>Container Name</b>   | NmGlobalConfig  |
| <b>Parent Container</b> | Nm              |



|                                 |  |
|---------------------------------|--|
| <b>Description</b>              | This container contains all global configuration parameters of the Nm Interface. |
| <b>Configuration Parameters</b> |  |

|   |   |   |              |
|---|---|---|--------------|
| <b>Name</b>                             | NmEcucPartitionRef [ECUC_Nm_00245]                          |   |              |
| <b>Parent Container</b>                 | <a href="#">NmGlobalConfig</a>                              |   |              |
| <b>Description</b>                      | Reference to EcucPartition, where Nm module is assigned to. |   |              |
| <b>Multiplicity</b>                     | 0..1  |   |              |
| <b>Type</b>                             | Reference to EcucPartition                                  |   |              |
| <b>Post-Build Variant Multiplicity</b>  | false   |   |              |
| <b>Post-Build Variant Value</b>         | false   |   |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>                                     | X | All Variants |
|   | <b>Link time</b>  | – |              |
|   | <b>Post-build time</b>                                      | – |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>                                     | X | All Variants |
|   | <b>Link time</b>  | – |              |
|   | <b>Post-build time</b>                                      | – |              |
| <b>Scope / Dependency</b>               | scope: local  |   |              |

| <b>Included Containers</b>         |                     |                           |
|------------------------------------|---------------------|---------------------------|
| <b>Container Name</b>              | <b>Multiplicity</b> | <b>Scope / Dependency</b> |
| <a href="#">NmGlobalConstants</a>  | 1                   |                           |
| <a href="#">NmGlobalFeatures</a>   | 1                   |                           |
| <a href="#">NmGlobalProperties</a> | 1                   |                           |

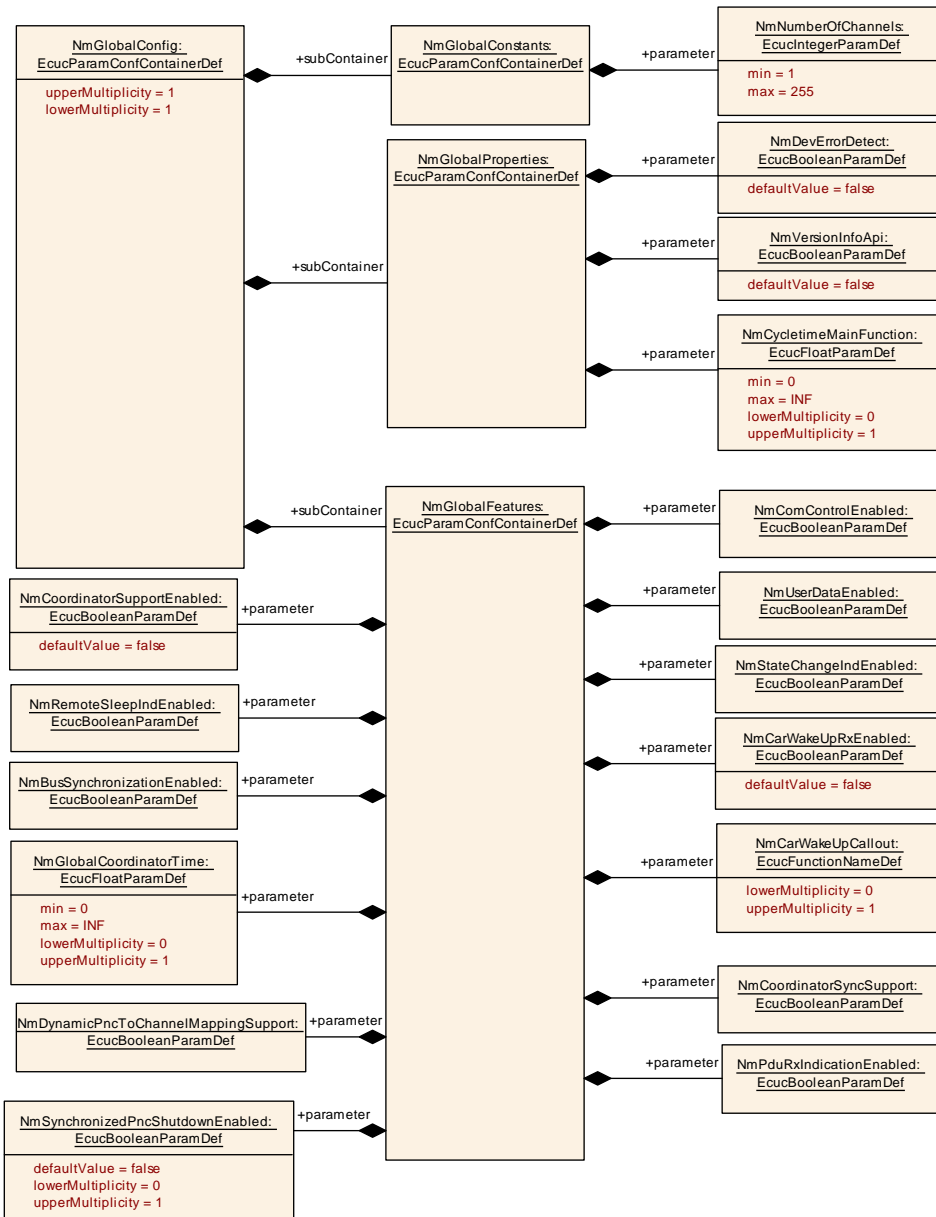


Figure 10.3: NmGlobalConfig overview

### 10.3.2 NmGlobalConstants

|                          |                                |
|--------------------------|--------------------------------|
| SWS Item                 | [ECUC_Nm_00198]                |
| Container Name           | NmGlobalConstants              |
| Parent Container         | <a href="#">NmGlobalConfig</a> |
| Description              |                                |
| Configuration Parameters |                                |

|                                  |   |   |              |
|----------------------------------|---|---|--------------|
| <b>Name</b>                      | NmNumberOfChannels [ECUC_Nm_00201]            |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalConstants</a>             |   |              |
| <b>Description</b>               | Number of NM channels allowed within one ECU. |   |              |
| <b>Multiplicity</b>              | 1   |   |              |
| <b>Type</b>                      | EcucIntegerParamDef                           |   |              |
| <b>Range</b>                     | 1 .. 255                                      |   |              |
| <b>Default Value</b>             |   |   |              |
| <b>Post-Build Variant Value</b>  | false   |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                       | X | All Variants |
|                                  | <b>Link time</b>                              | – |              |
|                                  | <b>Post-build time</b>                        | – |              |
| <b>Scope / Dependency</b>        | scope: local                                  |   |              |

No Included Containers

### 10.3.3 NmGlobalProperties

|                                 |                                |
|---------------------------------|--------------------------------|
| <b>SWS Item</b>                 | [ECUC_Nm_00199]                |
| <b>Container Name</b>           | NmGlobalProperties             |
| <b>Parent Container</b>         | <a href="#">NmGlobalConfig</a> |
| <b>Description</b>              |                                |
| <b>Configuration Parameters</b> |                                |

|   |  |   |                     |
|---|--|---|---------------------|
| <b>Name</b>                             | NmCycletimeMainFunction [ECUC_Nm_00205]  |   |                     |
| <b>Parent Container</b>                 | <a href="#">NmGlobalProperties</a>   |   |                     |
| <b>Description</b>                      | The period between successive calls to the Main Function of the NM Interface in seconds. |   |                     |
| <b>Multiplicity</b>                     | 0..1   |   |                     |
| <b>Type</b>                             | EcucFloatParamDef  |   |                     |
| <b>Range</b>                            | ]0 .. INF[   |   |                     |
| <b>Default Value</b>                    |  |   |                     |
| <b>Post-Build Variant Multiplicity</b>  | false  |   |                     |
| <b>Post-Build Variant Value</b>         | false  |   |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>   | – |                     |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>   | – |                     |

|                           |  |
|---------------------------|--|
| <b>Scope / Dependency</b> | scope: local<br>dependency: If NmCoordinatorSupportEnabled is set to TRUE, then the NmCycletimeMainFunction shall be configured. |
|---------------------------|--|

|                                  |   |   |              |
|----------------------------------|---|---|--------------|
| <b>Name</b>                      | NmDevErrorDetect [ECUC_Nm_00203]  |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalProperties</a>  |   |              |
| <b>Description</b>               | Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul> |   |              |
| <b>Multiplicity</b>              | 1   |   |              |
| <b>Type</b>                      | EcucBooleanParamDef   |   |              |
| <b>Default Value</b>             | false   |   |              |
| <b>Post-Build Variant Value</b>  | false   |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X | All Variants |
|                                  | <b>Link time</b>  | – |              |
|                                  | <b>Post-build time</b>  | – |              |
| <b>Scope / Dependency</b>        | scope: local  |   |              |

|                                  |   |   |              |
|----------------------------------|---|---|--------------|
| <b>Name</b>                      | NmVersionInfoApi [ECUC_Nm_00204]                            |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalProperties</a>                          |   |              |
| <b>Description</b>               | Pre-processor switch for enabling Version Info API support. |   |              |
| <b>Multiplicity</b>              | 1   |   |              |
| <b>Type</b>                      | EcucBooleanParamDef   |   |              |
| <b>Default Value</b>             | false   |   |              |
| <b>Post-Build Variant Value</b>  | false   |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                     | X | All Variants |
|                                  | <b>Link time</b>  | – |              |
|                                  | <b>Post-build time</b>                                      | – |              |
| <b>Scope / Dependency</b>        | scope: local  |   |              |

**No Included Containers**

### 10.3.4 NmGlobalFeatures

|                                 |                                |
|---------------------------------|--------------------------------|
| <b>SWS Item</b>                 | [ECUC_Nm_00200]                |
| <b>Container Name</b>           | NmGlobalFeatures               |
| <b>Parent Container</b>         | <a href="#">NmGlobalConfig</a> |
| <b>Description</b>              |                                |
| <b>Configuration Parameters</b> |                                |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmBusSynchronizationEnabled [ECUC_Nm_00208]  |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>   |   |              |
| <b>Description</b>               | Pre-processor switch for enabling bus synchronization support of the <BusNm>s. This feature is required for NM Coordinator nodes only. |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: This parameter must be enabled if NmCoordinatorSupportEnabled is enabled.                                  |   |              |

|   |   |   |                     |
|---|---|---|---------------------|
| <b>Name</b>                             | NmCarWakeUpCallout [ECUC_Nm_00234]  |   |                     |
| <b>Parent Container</b>                 | <a href="#">NmGlobalFeatures</a>  |   |                     |
| <b>Description</b>                      | Name of the callout function to be called if Nm_CarWakeUpIndication() is called. If this parameter is not configured, the Nm will call BswM_Nm_CarWakeUpIndication. |   |                     |
| <b>Multiplicity</b>                     | 0..1  |   |                     |
| <b>Type</b>                             | EcucFunctionNameDef   |   |                     |
| <b>Default Value</b>                    |   |   |                     |
| <b>Regular Expression</b>               |   |   |                     |
| <b>Post-Build Variant Multiplicity</b>  | false   |   |                     |
| <b>Post-Build Variant Value</b>         | false   |   |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | – |                     |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | – |                     |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: only available if NmCarWakeUpRxEnabled == TRUE  |   |                     |

|                                 |   |  |  |
|---------------------------------|---|--|--|
| <b>Name</b>                     | NmCarWakeUpRxEnabled [ECUC_Nm_00235]  |  |  |
| <b>Parent Container</b>         | <a href="#">NmGlobalFeatures</a>  |  |  |
| <b>Description</b>              | Enables or disables CWU detection. FALSE - CarWakeUp not supported TRUE - CarWakeUp supported |  |  |
| <b>Multiplicity</b>             | 1   |  |  |
| <b>Type</b>                     | EcucBooleanParamDef   |  |  |
| <b>Default Value</b>            | false   |  |  |
| <b>Post-Build Variant Value</b> | false   |  |  |

|                                  |                         |   |                     |
|----------------------------------|-------------------------|---|---------------------|
| <b>Value Configuration Class</b> | <b>Pre-compile time</b> | X | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>        | X | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>  | – |                     |
| <b>Scope / Dependency</b>        | scope: local            |   |                     |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmComControlEnabled [ECUC_Nm_00210]                                  |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>                                     |   |              |
| <b>Description</b>               | Pre-processor switch for enabling the Communication Control support. |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>        | scope: local   |   |              |

|                                  |   |   |              |
|----------------------------------|---|---|--------------|
| <b>Name</b>                      | NmCoordinatorSupportEnabled [ECUC_Nm_00206]   |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>  |   |              |
| <b>Description</b>               | Pre-processor switch for enabling NM Coordinator support.   |   |              |
| <b>Multiplicity</b>              | 1   |   |              |
| <b>Type</b>                      | EcucBooleanParamDef   |   |              |
| <b>Default Value</b>             | false   |   |              |
| <b>Post-Build Variant Value</b>  | false   |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X | All Variants |
|                                  | <b>Link time</b>  | – |              |
|                                  | <b>Post-build time</b>  | – |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: Only valid if at least one NM channel exists which has NmPassiveModeEnabled set to FALSE. |   |              |

|                                  |   |   |                     |
|----------------------------------|---|---|---------------------|
| <b>Name</b>                      | NmCoordinatorSyncSupport [ECUC_Nm_00240]                  |   |                     |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>                          |   |                     |
| <b>Description</b>               | Enables/disables the coordinator synchronisation support. |   |                     |
| <b>Multiplicity</b>              | 1   |   |                     |
| <b>Type</b>                      | EcucBooleanParamDef                                       |   |                     |
| <b>Default Value</b>             |   |   |                     |
| <b>Post-Build Variant Value</b>  | false   |   |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                   | X | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>                                    | – |                     |

|                           |  |
|---------------------------|--|
| <b>Scope / Dependency</b> | scope: local<br>dependency: NmCoordinatorSyncSupport shall only be valid if NmCoordinatorSupportEnabled is TRUE. |
|---------------------------|--|

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmDynamicPncToChannelMappingSupport [ECUC_Nm_00246]  |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>   |   |              |
| <b>Description</b>               | Precompile time switch to enable the dynamic PNC-to-channel-mapping handling.<br><br>False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled<br><br><b>Tags:</b><br>atp.Status=draft |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>        | scope: local   |   |              |

|   |  |   |              |
|---|--|---|--------------|
| <b>Name</b>                             | NmGlobalCoordinatorTime [ECUC_Nm_00237]  |   |              |
| <b>Parent Container</b>                 | <a href="#">NmGlobalFeatures</a>   |   |              |
| <b>Description</b>                      | This parameter defines the maximum shutdown time of a connected and coordinated NM-Cluster. Note:This includes nested connections. |   |              |
| <b>Multiplicity</b>                     | 0..1   |   |              |
| <b>Type</b>                             | EcucFloatParamDef  |   |              |
| <b>Range</b>                            | [0 .. INF]   |   |              |
| <b>Default Value</b>                    |  |   |              |
| <b>Post-Build Variant Multiplicity</b>  | false  |   |              |
| <b>Post-Build Variant Value</b>         | false  |   |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|   | <b>Link time</b>   | – |              |
|   | <b>Post-build time</b>   | – |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X | All Variants |
|   | <b>Link time</b>   | – |              |
|   | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: NmGlobalCoordinatorTime shall only be valid if NmCoordinatorSupportEnabled is TRUE.                    |   |              |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmPduRxIndicationEnabled [ECUC_Nm_00214]                 |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>                         |   |              |
| <b>Description</b>               | Pre-processor switch for enabling the PDU Rx Indication. |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef                                      |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>                                   | – |              |
| <b>Scope / Dependency</b>        | scope: local   |   |              |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmRemoteSleepIndEnabled [ECUC_Nm_00207]  |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>   |   |              |
| <b>Description</b>               | Pre-processor switch for enabling Remote Sleep Indication support.<br>This feature is required for a Gateway or Nm Coordinator functionality.<br><br>Note that this feature should not be used if all NM channels have Passive Mode enabled. |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: If NmCoordinatorSupportEnabled == TRUE then NmRemoteSleepIndEnabled = TRUE   |   |              |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmStateChangeIndEnabled [ECUC_Nm_00215]  |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>   |   |              |
| <b>Description</b>               | Pre-processor switch for enabling the Network Management state change notification.                          |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: NmStateChangeIndEnabled = TRUE<br>if NmDynamicPncToChannelMappingEnabled == TRUE |   |              |



|   |   |   |                     |
|---|---|---|---------------------|
| <b>Name</b>                             | NmSynchronizedPncShutdownEnabled [ECUC_Nm_00249]  |   |                     |
| <b>Parent Container</b>                 | <a href="#">NmGlobalFeatures</a>  |   |                     |
| <b>Description</b>                      | Enables or disables support of synchronized PNC shutdown.<br><br>FALSE: synchronized PNC shutdown is disabled<br><br>TRUE: synchronized PNC shutdown is enabled<br><br><b>Tags:</b><br>atp.Status=draft |   |                     |
| <b>Multiplicity</b>                     | 0..1  |   |                     |
| <b>Type</b>                             | EcucBooleanParamDef   |   |                     |
| <b>Default Value</b>                    | false   |   |                     |
| <b>Post-Build Variant Multiplicity</b>  | false   |   |                     |
| <b>Post-Build Variant Value</b>         | false   |   |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | – |                     |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | – |                     |
| <b>Scope / Dependency</b>               | scope: local  |   |                     |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmUserDataEnabled [ECUC_Nm_00211]                    |   |              |
| <b>Parent Container</b>          | <a href="#">NmGlobalFeatures</a>                     |   |              |
| <b>Description</b>               | Pre-processor switch for enabling User Data support. |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef                                  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                              | X | All Variants |
|                                  | <b>Link time</b>                                     | – |              |
|                                  | <b>Post-build time</b>                               | – |              |
| <b>Scope / Dependency</b>        | scope: local   |   |              |

|                               |
|-------------------------------|
| <b>No Included Containers</b> |
|-------------------------------|

## 10.4 Channel configurable parameters

### 10.4.1 NmChannelConfig

|                                 |   |
|---------------------------------|---|
| <b>SWS Item</b>                 | [ECUC_Nm_00197]   |
| <b>Container Name</b>           | NmChannelConfig   |
| <b>Parent Container</b>         | <a href="#">Nm</a>  |
| <b>Description</b>              | This container contains the configuration (parameters) of the bus channel(s). The channel parameter shall be harmonized within the whole communication stack. |
| <b>Configuration Parameters</b> |   |

|   |  |   |              |
|---|--|---|--------------|
| <b>Name</b>                             | NmActiveCoordinator [ECUC_Nm_00236]  |   |              |
| <b>Parent Container</b>                 | <a href="#">NmChannelConfig</a>  |   |              |
| <b>Description</b>                      | This parameter indicates whether a NM channel - part of a Nm Coordination cluster - will be coordinated actively (NmActiveCoordinator = TRUE) or passively (NmActiveCoordinator = FALSE).  |   |              |
| <b>Multiplicity</b>                     | 0..1   |   |              |
| <b>Type</b>                             | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>                    |  |   |              |
| <b>Post-Build Variant Multiplicity</b>  | false  |   |              |
| <b>Post-Build Variant Value</b>         | false  |   |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|   | <b>Link time</b>   | - |              |
|   | <b>Post-build time</b>   | - |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X | All Variants |
|   | <b>Link time</b>   | - |              |
|   | <b>Post-build time</b>   | - |              |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: If the NmCoordinatorSyncSupport is set to true this feature is available. Only one channel per Coordination cluster can have NmActiveCoordinator = FALSE. This parameter is mandatory if this channel belongs to a Coordination cluster (see ECUC_Nm_00221). Value cannot be set to FALSE in case BusNmType is set to NM_BUSNM_LOCALNM (i.e. no passive coordination for this type). |   |              |

|                                  |   |   |                     |
|----------------------------------|---|---|---------------------|
| <b>Name</b>                      | NmChannelSleepMaster [ECUC_Nm_00227]  |   |                     |
| <b>Parent Container</b>          | <a href="#">NmChannelConfig</a>   |   |                     |
| <b>Description</b>               | <p>This parameter shall be set to indicate if the sleep of this network can be absolutely decided by the local node only and that no other nodes can oppose that decision.</p> <p>If this parameter is set to TRUE, the Nm shall assume that the channel is always ready to go to sleep and that no calls to Nm_RemoteSleepIndication or Nm_RemoteSleepCancellation will be made from the &lt;BusNm&gt; representing this channel.</p> <p>If this parameter is set to FALSE, the Nm shall not assume that the network is ready to sleep until a call has been made to Nm_RemoteSleepCancellation.</p> |   |                     |
| <b>Multiplicity</b>              | 1   |   |                     |
| <b>Type</b>                      | EcucBooleanParamDef   |   |                     |
| <b>Default Value</b>             |   |   |                     |
| <b>Post-Build Variant Value</b>  | false   |   |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>  | – |                     |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: If the parameter NmCoordClusterIndex is not defined, this parameter is not valid.   |   |                     |

|   |   |   |              |
|---|---|---|--------------|
| <b>Name</b>                             | NmComUserDataSupport [ECUC_Nm_00241]  |   |              |
| <b>Parent Container</b>                 | <a href="#">NmChannelConfig</a>   |   |              |
| <b>Description</b>                      | This parameter indicates whether on a NM channel user data is accessed via Com signals or by SetUserData API. |   |              |
| <b>Multiplicity</b>                     | 0..1  |   |              |
| <b>Type</b>                             | EcucBooleanParamDef   |   |              |
| <b>Default Value</b>                    | false   |   |              |
| <b>Post-Build Variant Multiplicity</b>  | false   |   |              |
| <b>Post-Build Variant Value</b>         | false   |   |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X | All Variants |
|   | <b>Link time</b>  | – |              |
|   | <b>Post-build time</b>  | – |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>   | X | All Variants |
|   | <b>Link time</b>  | – |              |
|   | <b>Post-build time</b>  | – |              |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: NmComUserDataSupport shall be equal to <Bus>NmComUserDataSupport                  |   |              |

|   |   |   |                     |
|---|---|---|---------------------|
| <b>Name</b>                             | NmCoordClusterIndex [ECUC_Nm_00221]   |   |                     |
| <b>Parent Container</b>                 | <a href="#">NmChannelConfig</a>   |   |                     |
| <b>Description</b>                      | If this parameter is undefined for a channel, the corresponding bus does not belong to an NM coordination cluster.        |   |                     |
| <b>Multiplicity</b>                     | 0..1  |   |                     |
| <b>Type</b>                             | EcucIntegerParamDef   |   |                     |
| <b>Range</b>                            | 0 .. 255  |   |                     |
| <b>Default Value</b>                    |   |   |                     |
| <b>Post-Build Variant Multiplicity</b>  | false   |   |                     |
| <b>Post-Build Variant Value</b>         | false   |   |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | – |                     |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | – |                     |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: If NmCoordClusterIndex is defined than NmPassiveModeEnabled has to be FALSE for this channel. |   |                     |

|                                  |   |   |                     |
|----------------------------------|---|---|---------------------|
| <b>Name</b>                      | NmDynamicPncToChannelMappingEnabled [ECUC_Nm_00248]   |   |                     |
| <b>Parent Container</b>          | <a href="#">NmChannelConfig</a>   |   |                     |
| <b>Description</b>               | Channel-specific parameter to enable the dynamic PNC-to-channel-mapping feature.<br><br>False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled<br><br><b>Tags:</b><br>atp.Status=draft |   |                     |
| <b>Multiplicity</b>              | 0..1  |   |                     |
| <b>Type</b>                      | EcucBooleanParamDef   |   |                     |
| <b>Default Value</b>             | false   |   |                     |
| <b>Post-Build Variant Value</b>  | false   |   |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>  | – |                     |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: Shall only be TRUE if NmDynamicPncToChannelMappingSupport is TRUE   |   |                     |

|                                  |   |   |              |
|----------------------------------|---|---|--------------|
| <b>Name</b>                      | NmPassiveModeEnabled [ECUC_Nm_00242]  |   |              |
| <b>Parent Container</b>          | <a href="#">NmChannelConfig</a>   |   |              |
| <b>Description</b>               | This parameter indicates whether a NM channel is active, e.g. can request communication and keep the bus awake, or passive, e.g. can just be woken up and kept awake by other ECUs. |   |              |
| <b>Multiplicity</b>              | 1   |   |              |
| <b>Type</b>                      | EcucBooleanParamDef   |   |              |
| <b>Default Value</b>             |   |   |              |
| <b>Post-Build Variant Value</b>  | false   |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X | All Variants |
|                                  | <b>Link time</b>  | – |              |
|                                  | <b>Post-build time</b>  | – |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: if ComMNmVariant == FULL then NmPassiveModeEnabled = FALSE; NmPassiveModeEnabled shall be equal to <Bus>NmPassiveModeEnabled                            |   |              |

|                                  |  |   |              |
|----------------------------------|--|---|--------------|
| <b>Name</b>                      | NmStateReportEnabled [ECUC_Nm_00231]   |   |              |
| <b>Parent Container</b>          | <a href="#">NmChannelConfig</a>  |   |              |
| <b>Description</b>               | Specifies if the NMS shall be set for the corresponding network. false: No NMS shall be set true: The NMS shall be set |   |              |
| <b>Multiplicity</b>              | 1  |   |              |
| <b>Type</b>                      | EcucBooleanParamDef  |   |              |
| <b>Default Value</b>             |  |   |              |
| <b>Post-Build Variant Value</b>  | false  |   |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|                                  | <b>Link time</b>   | – |              |
|                                  | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: only available if NmStatChangeIndEnabled and NmComUserDataSupport are configured to TRUE.  |   |              |

|                                  |  |   |                     |
|----------------------------------|--|---|---------------------|
| <b>Name</b>                      | NmSynchronizingNetwork [ECUC_Nm_00223]   |   |                     |
| <b>Parent Container</b>          | <a href="#">NmChannelConfig</a>  |   |                     |
| <b>Description</b>               | If this parameter is true, then this network is a synchronizing network for the NM coordination cluster which it belongs to. The network is expected to call Nm_SynchronizationPoint() at regular intervals. |   |                     |
| <b>Multiplicity</b>              | 1  |   |                     |
| <b>Type</b>                      | EcucBooleanParamDef  |   |                     |
| <b>Default Value</b>             |  |   |                     |
| <b>Post-Build Variant Value</b>  | false  |   |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>   | X | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>   | – |                     |

|                           |  |
|---------------------------|--|
| <b>Scope / Dependency</b> | scope: local<br>dependency: If the parameter NmCoordClusterIndex is not defined, this parameter is not valid. Only one network can be configured as synchronizing network (NmSynchronizingNetwork = TRUE) per coordination cluster (same NmCoordClusterIndex value per channel). NmSynchronizingNetwork can only be set to true if NmActiveCoordinator is true for all networks which have the same NmCoordClusterIndex. |
|---------------------------|--|

|                                  |  |   |                     |
|----------------------------------|--|---|---------------------|
| <b>Name</b>                      | NmComMChannelRef [ECUC_Nm_00217]             |   |                     |
| <b>Parent Container</b>          | <a href="#">NmChannelConfig</a>              |   |                     |
| <b>Description</b>               | Reference to the corresponding ComM Channel. |   |                     |
| <b>Multiplicity</b>              | 1  |   |                     |
| <b>Type</b>                      | Symbolic name reference to ComMChannel       |   |                     |
| <b>Post-Build Variant Value</b>  | false  |   |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                      | X | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>                             | X | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>                       | – |                     |
| <b>Scope / Dependency</b>        | scope: local                                 |   |                     |

|   |  |   |              |
|---|--|---|--------------|
| <b>Name</b>                             | NmStateReportSignalRef [ECUC_Nm_00232]   |   |              |
| <b>Parent Container</b>                 | <a href="#">NmChannelConfig</a>  |   |              |
| <b>Description</b>                      | Reference to the signal for setting the NMS by calling Com_SendSignal for the respective channel.            |   |              |
| <b>Multiplicity</b>                     | 0..1   |   |              |
| <b>Type</b>                             | Symbolic name reference to ComSignal   |   |              |
| <b>Post-Build Variant Multiplicity</b>  | false  |   |              |
| <b>Post-Build Variant Value</b>         | false  |   |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X | All Variants |
|   | <b>Link time</b>   | – |              |
|   | <b>Post-build time</b>   | – |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X | All Variants |
|   | <b>Link time</b>   | – |              |
|   | <b>Post-build time</b>   | – |              |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: Signal must be configured in COM. Only available if NmStateReportEnabled == true |   |              |

| Included Containers       |              |                    |
|---------------------------|--------------|--------------------|
| Container Name            | Multiplicity | Scope / Dependency |
| <a href="#">NmBusType</a> | 1            |                    |

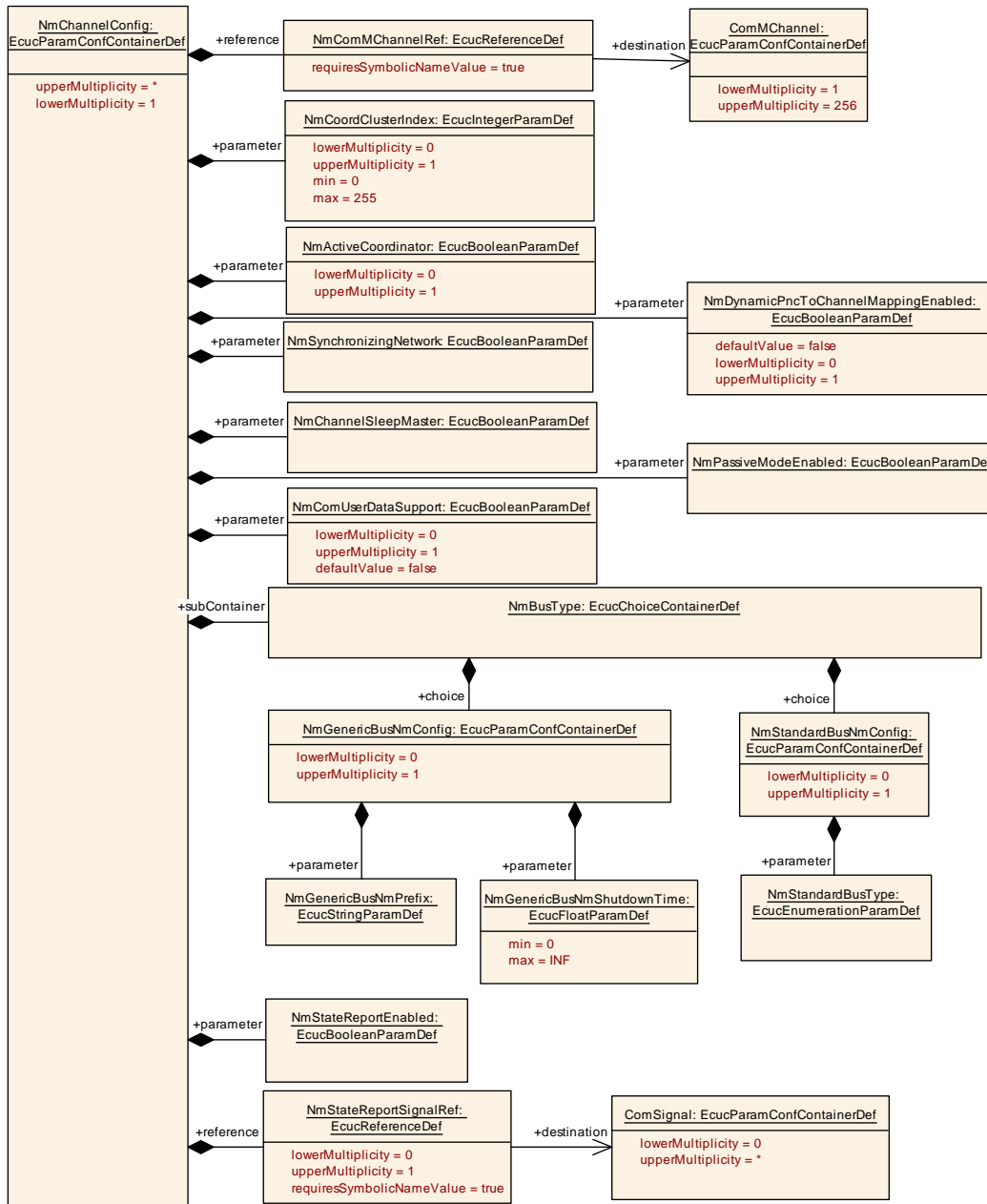


Figure 10.4: NmChannelConfig overview

### 10.4.2 NmBusType

|                                 |                                 |
|---------------------------------|---------------------------------|
| SWS Item                        | [ECUC_Nm_00218]                 |
| Container Name                  | NmBusType                       |
| Parent Container                | <a href="#">NmChannelConfig</a> |
| Description                     |                                 |
| <b>Configuration Parameters</b> |                                 |

| Container Choices                     |              |                    |
|---------------------------------------|--------------|--------------------|
| Container Name                        | Multiplicity | Scope / Dependency |
| <a href="#">NmGenericBusNmConfig</a>  | 0..1         |                    |
| <a href="#">NmStandardBusNmConfig</a> | 0..1         |                    |

### 10.4.3 NmGenericBusNmConfig

|                          |                           |
|--------------------------|---------------------------|
| SWS Item                 | [ECUC_Nm_00225]           |
| Container Name           | NmGenericBusNmConfig      |
| Parent Container         | <a href="#">NmBusType</a> |
| Description              |                           |
| Configuration Parameters |                           |

|                           |   |   |                     |
|---------------------------|---|---|---------------------|
| Name                      | NmGenericBusNmPrefix [ECUC_Nm_00219]  |   |                     |
| Parent Container          | <a href="#">NmGenericBusNmConfig</a>  |   |                     |
| Description               | The prefix which identifies the generic <BusNm>. This will be used to determine the API name to be called by Nm for the provided interfaces of the <BusNm>. This string will be used for the module prefix before the "_" character in the API call name. |   |                     |
| Multiplicity              | 1   |   |                     |
| Type                      | EcucStringParamDef  |   |                     |
| Default Value             |   |   |                     |
| Regular Expression        |   |   |                     |
| Post-Build Variant Value  | false   |   |                     |
| Value Configuration Class | Pre-compile time  | X | VARIANT-PRE-COMPILE |
|                           | Link time   | X | VARIANT-LINK-TIME   |
|                           | Post-build time   | - |                     |
| Scope / Dependency        | scope: local  |   |                     |

|                           |  |   |                     |
|---------------------------|--|---|---------------------|
| Name                      | NmGenericBusNmShutdownTime [ECUC_Nm_00239]                     |   |                     |
| Parent Container          | <a href="#">NmGenericBusNmConfig</a>                           |   |                     |
| Description               | This parameter shall be used to calculate shutdown delay time. |   |                     |
| Multiplicity              | 1  |   |                     |
| Type                      | EcucFloatParamDef  |   |                     |
| Range                     | [0 .. INF]   |   |                     |
| Default Value             |  |   |                     |
| Post-Build Variant Value  | false  |   |                     |
| Value Configuration Class | Pre-compile time   | X | VARIANT-PRE-COMPILE |
|                           | Link time  | X | VARIANT-LINK-TIME   |
|                           | Post-build time  | - |                     |
| Scope / Dependency        | scope: local   |   |                     |



No Included Containers

#### 10.4.4 NmStandardBusNmConfig

|                                 |                           |
|---------------------------------|---------------------------|
| <b>SWS Item</b>                 | [ECUC_Nm_00226]           |
| <b>Container Name</b>           | NmStandardBusNmConfig     |
| <b>Parent Container</b>         | <a href="#">NmBusType</a> |
| <b>Description</b>              |                           |
| <b>Configuration Parameters</b> |                           |

|                                  |   |                              |                     |
|----------------------------------|---|------------------------------|---------------------|
| <b>Name</b>                      | NmStandardBusType [ECUC_Nm_00220]   |                              |                     |
| <b>Parent Container</b>          | <a href="#">NmStandardBusNmConfig</a>   |                              |                     |
| <b>Description</b>               | Identifies the bus type of the channel for standard AUTOSAR <BusNm>s and is used to determine which set of API calls to be called by Nm for the <BusNm>s. Note: The Ethernet bus' NM is UdpNm ! |                              |                     |
| <b>Multiplicity</b>              | 1   |                              |                     |
| <b>Type</b>                      | EcucEnumerationParamDef   |                              |                     |
| <b>Range</b>                     | NM_BUSNM_CANNM  | CAN bus                      |                     |
|                                  | NM_BUSNM_FRNM   | FlexRay bus                  |                     |
|                                  | NM_BUSNM_J1939NM  | J1939 bus (address claiming) |                     |
|                                  | NM_BUSNM_LOCALNM  | Local Bus (e.g. LIN bus)     |                     |
|                                  | NM_BUSNM_UDPNM  | Ethernet bus (using UDP)     |                     |
| <b>Post-Build Variant Value</b>  | false   |                              |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X                            | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | X                            | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>  | -                            |                     |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: Configuring value to NM_BUSNM_LOCALNM is only allowed if NmCoordClusterIndex for the corresponding channel is defined (i.e channel is coordinated).                 |                              |                     |

No Included Containers

### 10.5 Published Information

For details refer to the chapter 10.3 “Published Information” in [2, SWS\_BSWGeneral].

## A Not applicable requirements

**[SWS\_Nm\_00999] Not applicable requirements** [These requirements are not applicable to this specification.] ([SRS\\_Nm\\_00043](#), [SRS\\_Nm\\_00052](#), [SRS\\_Nm\\_00053](#), [SRS\\_Nm\\_00054](#), [SRS\\_Nm\\_00137](#), [SRS\\_Nm\\_00142](#), [SRS\\_Nm\\_00143](#), [SRS\\_Nm\\_00144](#), [SRS\\_Nm\\_00145](#), [SRS\\_Nm\\_00146](#), [SRS\\_Nm\\_00147](#), [SRS\\_Nm\\_00148](#), [SRS\\_Nm\\_02509](#), [SRS\\_Nm\\_02510](#), [SRS\\_Nm\\_02517](#), [SRS\\_Nm\\_02518](#), [SRS\\_Nm\\_02519](#), [SRS\\_Nm\\_02520](#), [SRS\\_Nm\\_02521](#), [SRS\\_Nm\\_02522](#), [SRS\\_Nm\\_02523](#), [SRS\\_Nm\\_02524](#), [SRS\\_Nm\\_02525](#), [SRS\\_Nm\\_02526](#), [SRS\\_Nm\\_02527](#), [SRS\\_Nm\\_02528](#), [SRS\\_Nm\\_02529](#), [SRS\\_Nm\\_02530](#), [SRS\\_Nm\\_02531](#), [SRS\\_Nm\\_02532](#), [SRS\\_Nm\\_02533](#), [SRS\\_Nm\\_02534](#), [SRS\\_BSW\\_00004](#), [SRS\\_BSW\\_00005](#), [SRS\\_BSW\\_00006](#), [SRS\\_BSW\\_00007](#), [SRS\\_BSW\\_00009](#), [SRS\\_BSW\\_00010](#), [SRS\\_BSW\\_00158](#), [SRS\\_BSW\\_00160](#), [SRS\\_BSW\\_00161](#), [SRS\\_BSW\\_00162](#), [SRS\\_BSW\\_00164](#), [SRS\\_BSW\\_00167](#), [SRS\\_BSW\\_00168](#), [SRS\\_BSW\\_00170](#), [SRS\\_BSW\\_00172](#), [SRS\\_BSW\\_00300](#), [SRS\\_BSW\\_00302](#), [SRS\\_BSW\\_00304](#), [SRS\\_BSW\\_00305](#), [SRS\\_BSW\\_00306](#), [SRS\\_BSW\\_00307](#), [SRS\\_BSW\\_00308](#), [SRS\\_BSW\\_00309](#), [SRS\\_BSW\\_00310](#), [SRS\\_BSW\\_00312](#), [SRS\\_BSW\\_00314](#), [SRS\\_BSW\\_00318](#), [SRS\\_BSW\\_00321](#), [SRS\\_BSW\\_00325](#), [SRS\\_BSW\\_00328](#), [SRS\\_BSW\\_00331](#), [SRS\\_BSW\\_00334](#), [SRS\\_BSW\\_00335](#), [SRS\\_BSW\\_00336](#), [SRS\\_BSW\\_00339](#), [SRS\\_BSW\\_00341](#), [SRS\\_BSW\\_00342](#), [SRS\\_BSW\\_00343](#), [SRS\\_BSW\\_00346](#), [SRS\\_BSW\\_00347](#), [SRS\\_BSW\\_00350](#), [SRS\\_BSW\\_00351](#), [SRS\\_BSW\\_00360](#), [SRS\\_BSW\\_00361](#), [SRS\\_BSW\\_00371](#), [SRS\\_BSW\\_00374](#), [SRS\\_BSW\\_00375](#), [SRS\\_BSW\\_00377](#), [SRS\\_BSW\\_00378](#), [SRS\\_BSW\\_00379](#), [SRS\\_BSW\\_00380](#), [SRS\\_BSW\\_00383](#), [SRS\\_BSW\\_00388](#), [SRS\\_BSW\\_00389](#), [SRS\\_BSW\\_00390](#), [SRS\\_BSW\\_00392](#), [SRS\\_BSW\\_00393](#), [SRS\\_BSW\\_00394](#), [SRS\\_BSW\\_00395](#), [SRS\\_BSW\\_00397](#), [SRS\\_BSW\\_00398](#), [SRS\\_BSW\\_00399](#), [SRS\\_BSW\\_00400](#), [SRS\\_BSW\\_00401](#), [SRS\\_BSW\\_00402](#), [SRS\\_BSW\\_00403](#), [SRS\\_BSW\\_00404](#), [SRS\\_BSW\\_00406](#), [SRS\\_BSW\\_00408](#), [SRS\\_BSW\\_00409](#), [SRS\\_BSW\\_00410](#), [SRS\\_BSW\\_00411](#), [SRS\\_BSW\\_00413](#), [SRS\\_BSW\\_00415](#), [SRS\\_BSW\\_00416](#), [SRS\\_BSW\\_00417](#), [SRS\\_BSW\\_00422](#), [SRS\\_BSW\\_00423](#), [SRS\\_BSW\\_00424](#), [SRS\\_BSW\\_00426](#), [SRS\\_BSW\\_00427](#), [SRS\\_BSW\\_00428](#), [SRS\\_BSW\\_00429](#), [SRS\\_BSW\\_00432](#), [SRS\\_BSW\\_00433](#), [SRS\\_BSW\\_00437](#), [SRS\\_BSW\\_00438](#), [SRS\\_BSW\\_00439](#), [SRS\\_BSW\\_00440](#), [SRS\\_BSW\\_00441](#), [SRS\\_BSW\\_00447](#), [SRS\\_BSW\\_00448](#), [SRS\\_BSW\\_00449](#), [SRS\\_BSW\\_00451](#), [SRS\\_BSW\\_00452](#), [SRS\\_BSW\\_00453](#), [SRS\\_BSW\\_00454](#), [SRS\\_BSW\\_00456](#), [SRS\\_BSW\\_00457](#), [SRS\\_BSW\\_00458](#), [SRS\\_BSW\\_00459](#), [SRS\\_BSW\\_00460](#), [SRS\\_BSW\\_00461](#), [SRS\\_BSW\\_00462](#), [SRS\\_BSW\\_00463](#), [SRS\\_BSW\\_00464](#), [SRS\\_BSW\\_00465](#), [SRS\\_BSW\\_00466](#), [SRS\\_BSW\\_00467](#), [SRS\\_BSW\\_00469](#), [SRS\\_BSW\\_00470](#), [SRS\\_BSW\\_00471](#), [SRS\\_BSW\\_00472](#), [SRS\\_BSW\\_00473](#), [SRS\\_BSW\\_00396](#), [SRS\\_BSW\\_00477](#), [SRS\\_BSW\\_00479](#), [SRS\\_BSW\\_00480](#), [SRS\\_BSW\\_00481](#))