

# **OSEK/VDX**

## **NM test plan**

Version 2.0

April 26th, 1999

This document is an official release and replaces all previously distributed documents. The OSEK group retains the right to make changes to this document without notice and does not accept any liability for errors. All rights reserved. No part of this document may be reproduced, in any form or by any means, without permission in writing from the OSEK/VDX steering committee.

## **What is OSEK/VDX?**

OSEK/VDX is a joint project of the automotive industry. It aims at an industry standard for an open-ended architecture for distributed control units in vehicles.

A real-time operating system, software interfaces and functions for communication and network management tasks are thus jointly specified.

The term OSEK means "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" (Open systems and the corresponding interfaces for automotive electronics).

The term VDX means „Vehicle Distributed eXecutive“. The functionality of OSEK operating system was harmonized with VDX. For simplicity OSEK will be used instead of OSEK/VDX in the document.

## **OSEK partners:**

Adam Opel AG, BMW AG, Daimler-Benz AG, IIT University of Karlsruhe, Mercedes-Benz AG, Robert Bosch GmbH, Siemens AG, Volkswagen AG.

GIE.RE. PSA-Renault (Groupement d'intérêt Economique de Recherches et d'Etudes PSA-Renault).

## **Motivation:**

- High, recurring expenses in the development and variant management of non-application related aspects of control unit software.
- Incompatibility of control units made by different manufacturers due to different interfaces and protocols.

## **Goal:**

Support of the portability and reusability of the application software by:

- Specification of interfaces which are abstract and as application-independent as possible, in the following areas: real-time operating system, communication and network management.
- Specification of a user interface independent of hardware and network.
- Efficient design of architecture: The functionality shall be configurable and scaleable, to enable optimal adjustment of the architecture to the application in question.
- Verification of functionality and implementation of prototypes in selected pilot projects.

## **Advantages:**

- Clear savings in costs and development time.
- Enhanced quality of the control units software of various companies.
- Standardized interfacing features for control units with different architectural designs.
- Sequenced utilization of the intelligence (existing resources) distributed in the vehicle, to enhance the performance of the overall system without requiring additional hardware.
- Provides absolute independence with regards to individual implementation, as the specification does not prescribe implementation aspects.

## **OSEK conformance testing**

OSEK conformance testing aims at checking conformance of products to OSEK specifications. Test suites are thus specified for implementations of OSEK operating system, communication and network management.

Work around OSEK conformance testing is supported by the MODISTARC project sponsored by the Commission of European Communities. The term MODISTARC means "Methods and tools for the validation of OSEK/VDX based DISTRIBUTED ARChitectures".

This document has been drafted by the COM/NM project group of MODISTARC:

|                     |                           |
|---------------------|---------------------------|
| Harald Heinecke     | BMW AG                    |
| Wolfgang Kremer     | BMW AG                    |
| Benoit Caillaud     | INRIA                     |
| Dirk John           | IIT, Karlsruhe University |
| Yevgeny Shakuro     | Motorola GmbH             |
| Barbara Ziker       | Motorola GmbH             |
| Jean-Paul Cloup     | Peugeot Citroën S.A.      |
| Jean-Emmanuel Hanne | Peugeot Citroën S.A.      |
| Samuel Boutin       | Renault S.A.              |
| Patrick Palmieri    | Siemens Automotive SA     |
| Didier Stunault     | Thomson-CSF Detexis       |

# TABLE OF CONTENTS

|  |           |
|--|-----------|
| <b>1. INTRODUCTION</b>   | <b>5</b>  |
| 1.1. Scope   | 5         |
| 1.2. References  | 5         |
| 1.3. Abbreviations   | 6         |
| <b>2. TEST PURPOSES STRUCTURE</b>                                | <b>7</b>  |
| 2.1. Description   | 7         |
| 2.2. Detailed structure  | 9         |
| 2.2.1. Direct NM   | 9         |
| 2.2.2. Indirect NM   | 10        |
| <b>3. TEST PURPOSES</b>  | <b>11</b> |
| <b>3.1. Test purposes of Direct NM</b>                           | <b>11</b> |
| 3.1.1. Service test group  | 11        |
| 3.1.1.1. Configuration Management                                | 13        |
| 3.1.1.2. Operating Modes and Operating Mode Management           | 14        |
| 3.1.1.3. Data field management                                   | 16        |
| 3.1.2. Protocol test group                                       | 18        |
| 3.1.2.1. NMInit and NMReset                                      | 19        |
| 3.1.2.2. NMNormal  | 22        |
| 3.1.2.3. NMLimpHome  | 26        |
| 3.1.2.4. NMBusSleep  | 30        |
| <b>3.2. Test purposes of Indirect NM</b>                         | <b>30</b> |
| 3.2.1. Service test group  | 30        |
| 3.2.1.1. Configuration Management                                | 31        |
| 3.2.1.2. Operating Modes and Operating Mode Management           | 32        |
| 3.2.2. Protocol test group - One global time-out TOB             | 33        |
| 3.2.2.1. Handling of StartNM and StopNM                          | 34        |
| 3.2.2.2. NMNormal  | 35        |
| 3.2.2.3. NMLimphome  | 37        |
| 3.2.3. Protocol test group - One monitoring time-out per message | 39        |
| 3.2.3.1. Handling of StartNM, StopNM and InitConfig              | 41        |
| 3.2.3.2. NMNormal  | 42        |
| 3.2.3.3. NMLimphome  | 44        |
| 3.2.3.4. NMBusSleep  | 48        |

# 1. Introduction

## 1.1. Scope

This document specifies a test plan for services and protocols of the OSEK NM as defined in specification document [4]. It applies to conformance test suites for testing implementations which claim conformance to the OSEK NM specification.

According to the Conformance Methodology [1], definition of conformance tests is a two-stage process. This test plan document corresponds to the first step. It specifies a list of test purposes extracted from the NM specification. In the second step, test cases will be derived from the test purposes to build up the OSEK NM conformance test suite. Basically, a test case specifies the sequence of interactions between a tester and the NM implementation in order to verify a test purpose of this document. However, it is possible to have individual test cases that address multiple test purposes and likewise multiple test cases that address the same test purpose.

According to the Conformance Methodology this document follows the principle of black box testing. Relevant interfaces are the NM API and the NM messages (NMPDUs). Interfaces between COM and NM have been considered not mandatory and they are therefore not tested.

As OSEK NM implementations can operate either the Direct OSEK NM or the Indirect OSEK NM, the list of test purposes is divided into two parts accordingly. Inside both categories, the test purposes are organised according to a tree structure described in Chapter 2.

## 1.2. References

- [1] OSEK/VDX Conformance Testing Methodology - Version 1.0 - 19 December 1997.
- [2] OSEK/VDX Operating System - Version 2.0 - revision 1 - 15 October 1997.
- [3] OSEK/VDX Communication - Version 2.1 - revision 1 - 17th June 1998.
- [4] OSEK Network Management - Concept and Application Programming Interface - Version 2.50 - 31th of May 1998.
- [5] ISO/IEC 9646-1 - Information technology, Open Systems Interconnection, Conformance testing methodology and framework, *part 1 : General Concepts*, 1992.
- [6] ISO/IEC 9646-3 - Information technology, Open Systems Interconnection, Conformance testing, methodology and framework, *part 3 : The Tree and Tabular Combined Notation (TTCN)*, 1992.

### 1.3. Abbreviations

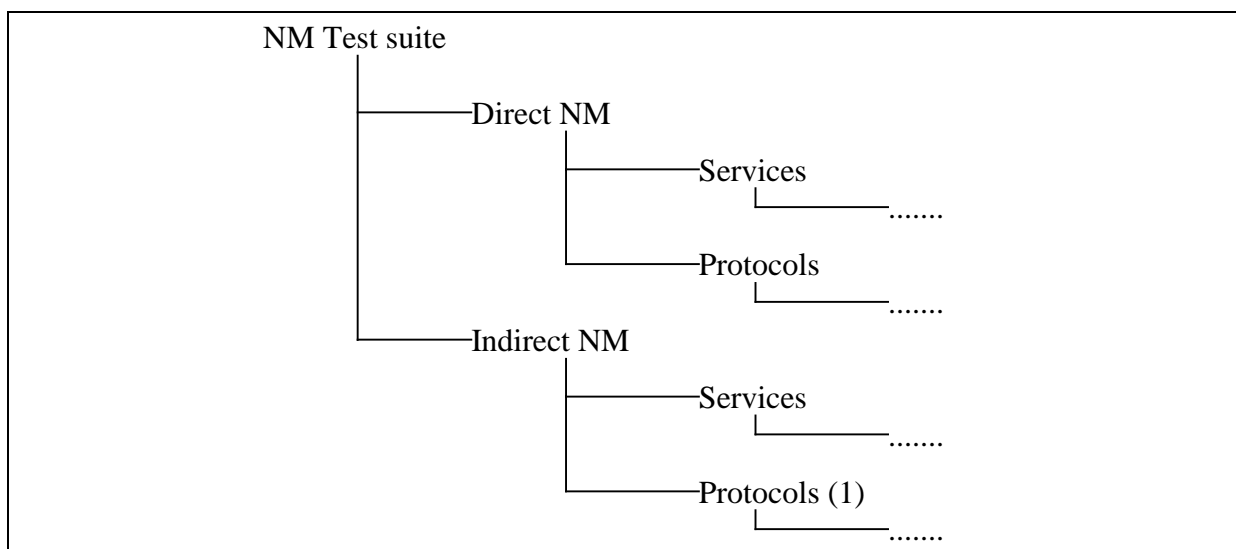
|      |  |
|------|--|
| API  | Application Programming Interface      |
| COM  | Communication                          |
| ISO  | International Standard Organization    |
| NM   | Network Management                     |
| OS   | Operating System                       |
| PDU  | Protocol Data Unit                     |
| SDL  | Specification and Description Language |
| TOB  | Time-out for OBServation               |
| TTCN | Tree and Tabular Combined Notation     |

## 2. Test purposes structure

### 2.1. Description

The test purposes for the OSEK NM services and protocols are arranged in groups and subgroups following a hierarchical structure. This organisation is common to direct and indirect NM and it follows the NM specification structure. It intends to facilitate cross-checking with the specification and verification of completeness. It does not preclude a different approach for test cases organisation inside the test suite.

The tree structure of NM test purposes is illustrated in Figure 1 below:



- (1) For Indirect NM, the protocol subtree consists of two distinct branches corresponding respectively to “One global time-out TOB” specification and “One monitoring time-out per message” specification.

Figure 1 Hierarchy of NM test purposes

The service tests are subdivided on a per service basis. There is at least one test for each API in order to demonstrate that all implemented services can be successfully called by an application.

The protocol tests are subdivided according to protocol states and substates defined in the NM specification. They intend to verify that the NM implementation behaves as specified in all implemented (sub)states. They also check all transitions between the different (sub)states.

Both the service and the protocol test purposes include verification of:

- valid behaviour: the implementation is actually waiting for the stimuli received from the tester,
- inopportune actions: unexpected stimuli are sent by the tester, e.g. a ring message while the NM is not expecting such a message.

There is no test of invalid behaviour, that is sending stimuli with bad parameters. The NM specification does not specify the format of protocol messages and of API parameters. Therefore, it is not possible to set bad values that could apply to every implementation.

Test purposes are brought together into tables corresponding to the leaves of the tree structure. Each table is made up of four columns providing:

- a reference number,
- the test assertion,
- the paragraph or picture of the NM specification from which the assertion was extracted,
- the specification variant needing to be implemented for the test purpose to be verified.

Each test assertion contains:

- the stimulus to be sent to verify the test purpose and if necessary the NM specification state needing to be reached before sending the stimulus,
- the action that shall be performed by the implementation to verify the test purpose and the subsequent output that should be observed by the tester. Note that the output can be "nothing" in which case the tester shall verify that the implementation did not send anything.



## 2.2. Detailed structure

### 2.2.1. Direct NM

|           |   |
|-----------|---|
| DirectNM  |   |
| Services  |   |
|           | Configuration Management                                  |
|           | InitIndDeltaConfig  |
|           | InitConfig  |
|           | GetConfig   |
|           | CmpConfig   |
|           | SelectDeltaConfig   |
|           | Operating Modes and Operating Mode Management             |
|           | StartNM   |
|           | StopNM  |
|           | GotoMode  |
|           | Network Status and GetStatus                              |
|           | CmpStatus   |
|           | SelectDeltaStatus   |
|           | InitIndDeltaStatus  |
|           | SilentNM  |
|           | TalkNM  |
|           | Data Field Management                                     |
|           | InitIndRingData   |
|           | “Using of RingData allowed” information of network status |
|           | ReadRingData  |
|           | TransmitRingData  |
| Protocols |   |
|           | NMInit and NMReset  |
|           | Transitions to NMReset                                    |
|           | InitReset   |
|           | ResetActive state   |
|           | NMNormal  |
|           | Transitions to NMNormal                                   |
|           | NormalStandardNM  |
|           | NormalActive state  |
|           | Transitions from/to NormalActive                          |
|           | NormalActivePrepSleep state                               |
|           | Transitions from/to Normal(Active/Passive)PrepSleep       |
|           | Transitions to NMTwbsNormal                               |
|           | NMTwbsNormal state  |
|           | NMLimpHome  |
|           | Transitions to NMLimpHome                                 |
|           | InitLimpHome  |
|           | LimpHomeActive state                                      |
|           | LimpHomePassive state                                     |
|           | Transitions from/to LimpHomeActive                        |
|           | Transitions from/to LimpHome(Active/Passive)PrepSleep     |
|           | Transitions to NMTwbsLimpHome                             |
|           | NMTwbsLimpHome state                                      |
|           | NMBusSleep  |
|           | Transitions to NMBusSleep                                 |

Table 1 Test Purposes Structure of Direct NM

## 2.2.2. Indirect NM

|   |   |
|---|---|
| IndirectNM                                      |   |
| Services  |   |
|   | Configuration Management                      |
|   | InitIndDeltaConfig                            |
|   | InitConfig                                    |
|   | GetConfig                                     |
|   | CmpConfig                                     |
|   | SelectDeltaConfig                             |
|   | Operating Modes and Operating Mode Management |
|   | StartNM                                       |
|   | StopNM  |
|   | GotoMode                                      |
|   | GetStatus                                     |
|   | CmpStatus                                     |
|   | SelectDeltaStatus                             |
|   | InitIndDeltaStatus                            |
| Protocols - One global time-out TOB             |   |
|   | Handling of StartNM and StopNM                |
|   | User's communication management               |
|   | Configuration management                      |
|   | Network status management                     |
|   | NMNormal                                      |
|   | Configuration management                      |
|   | Network status management                     |
|   | NMLimpHome                                    |
|   | User's communication management               |
|   | Configuration management                      |
|   | Network status management                     |
| Protocols - One monitoring time-out per message |   |
|   | Handling of StartNM, StopNM and InitConfig    |
|   | User's communication management               |
|   | Configuration management                      |
|   | Network status management                     |
|   | NMNormal                                      |
|   | Configuration management                      |
|   | Network status management                     |
|   | NMLimpHome                                    |
|   | User's communication management               |
|   | Configuration management                      |
|   | Network status management                     |
|   | NMBusSleep                                    |
|   | User's communication management               |
| b   | Network status management                     |

Table 2 Test Purposes Structure of Indirect NM

## 3. Test purposes

### 3.1. Test purposes of Direct NM

This section contains a set of test purposes relevant to Direct NM services and protocols. These test purposes provide ground material for developing the TTCN test suite which will be used to evaluate conformance to direct NM specification [4].

#### 3.1.1. Service test group

This section specifies tests purposes relative to the direct NM API as defined in sections 4.4 and 4.5 of the NM specification document. Each test purpose defines both the test stimulus to be sent and the subsequent output(s) to be observed either at the NM API or on the Data Bus.

The test stimuli include API calls with different sets of input parameters and also NMPDUs causing the implementation either:

- to change the network status, or
- to change the network configuration, or
- to receive ring data.

The observable outputs are either:

- the stati returned by API calls and the associated output parameters such as network status, network configuration and ring data, or
- the tasks activations or event settings carried out by the implementation on NMPDU reception, or
- NMPDU transmissions originating from the implementation.

Each test purpose also gives information on the specification variant(s) that need to be implemented for the test purpose to be verified.

The implementation variants are identified by the following terms:

- Core means that the test purpose must be verified in any implementation,
- Active/Passive means that the test purpose must be verified only if the optional SilentNM and TalkNM services are implemented,
- BusSleep means that the test purpose must be verified only if the optional GotoMode service is implemented,
- InitConfig means that the test purpose must be verified only if the optional InitConfig service is implemented,
- CmpConfig means that the test purpose must be verified only if the optional CmpConfig service is implemented,
- SelectConfig means that the test purpose must be verified only if optional SelectDelta-Config service is implemented.
- RingData means that the test purpose must be verified only if the optional ReadRingData and TransmitRingData services are implemented,

- Status refers to API return status. It means that the test purpose must be verified only if the tested status is actually implemented (see NM specification, section 2.3 - 2<sup>nd</sup> §),
- CmpStatus means that the test purpose must be verified only if the optional CmpStatus service is implemented,
- SelectStatus means that the test purpose must be verified only if optional SelectDeltaStatus service is implemented.

The "NMStatus" variant implies a test of a network status change. There are two ways of testing this information:

1. Test by a GetStatus API call, if this optional procedure is effectively implemented,
2. Test of a task activation occurrence or an OS event occurrence, depending on the status changes selected through the SelectDeltaStatus service.

In the test purposes, the changes to network status information are expressed by the following assertions:

| Network status value |                                       | Assertion  |
|----------------------|---------------------------------------|--|
| 0                    | Present configuration not stable      | "Present configuration stable" information of network status is cleared      |
| 1                    | Present configuration stable          | "Present configuration stable" information of network status is set          |
| 0                    | No error                              | "Error, bus blocked" information of network status is cleared                |
| 1                    | Error, bus blocked                    | "Error, bus blocked" information of network status is set                    |
| 0                    | NMPassive                             | "NMActive" information of network status is cleared                          |
| 1                    | NMActive                              | "NMActive" information of network status is set                              |
| 0                    | NMOn                                  | "NMOn/NMOff" information of network status is cleared                        |
| 1                    | NMOff                                 | "NMOn/NMOff" information of network status is set                            |
| 0                    | no NMLimpHome                         | "NMLimpHome" information of network status is cleared                        |
| 1                    | NMLimpHome                            | "NMLimpHome" information of network status is set                            |
| 0                    | no NMBusSleep                         | "NMBusSleep" information of network status is cleared                        |
| 1                    | NMBusSleep                            | "NMBusSleep" information of network status is set                            |
| 0                    | no NMTwbsNormal and no NMTwbsLimpHome | "NMTwbsNormal or NMTwbsLimpHome" information of network status is cleared    |
| 1                    | NMTwbsNormal or NMTwbsLimpHome        | "NMTwbsNormal or NMTwbsLimpHome" information of network status is set        |
| 0                    | using of Ring Data allowed            | "Using of Ring Data not allowed" information of network status is cleared    |
| 1                    | using of Ring Data not allowed        | "Using of Ring Data not allowed" information of network status is set        |
| 0                    | Service GotoMode(Awake) called        | "Service GotoMode(BusSleep) called" information of network status is cleared |
| 1                    | Service GotoMode(BusSleep) called     | "Service GotoMode(BusSleep) called" information of network status is set     |

As Extended Network Status is considered implementation specific in NM specification (see § 2.2.3.3), no test relating to such a status is specified here.

### 3.1.1.1. Configuration Management

| Nr   | Assertion   | Paragraph in spec. | Affected variants   |
|--|---|--------------------|---------------------|
| Signalling specified by InitIndDeltaConfig |   |                    |                     |
| 1  | If a first alive or ring message is received from a node belonging to the current normal configuration mask and the target configuration is matched, the task specified by InitIndDeltaConfig is activated or the specified event is set. | 4.4.2.2            | Core                |
| 2  | If a first limphome message is received from a node belonging to the current limphome configuration mask and the target configuration is matched, the task specified by InitIndDeltaConfig is activated or the specified event is set.    | 4.4.2.2            | Core                |
| 3  | No task activation nor event setting happens if a first alive or ring message is received from a node not belonging to the current normal configuration mask.   | 4.4.2.2            | Core                |
| 4  | No task activation nor event setting happens if a first limphome message is received from a node not belonging to the current limphome configuration mask.  | 4.4.2.2            | Core                |
| InitConfig service                         |   |                    |                     |
| 5  | InitConfig makes the NM to start or restart the configuration management  | 4.4.2.3            | InitConfig          |
| 6  | InitConfig returns E_OK.  | 4.4.2.3            | InitConfig + Status |
| GetConfig service                          |   |                    |                     |
| 7  | GetConfig provides the current normal configuration if the ConfigKind parameter equals "Normal".  | 4.4.2.3            | Core                |
| 8  | GetConfig provides the current limphome configuration if the ConfigKind parameter equals "LimpHome".  | 4.4.2.3            | Core                |
| 9  | GetConfig returns E_OK.   | 4.4.2.3            | Status              |
| CmpConfig service                          |   |                    |                     |
| 10   | CmpConfig returns the boolean value: Status = NOT (<CMask> AND (<TestConfig> EXOR <RefConfig>)), where TestConfig is the test configuration, RefConfig is the reference configuration and CMask is the test mask.                         | 4.4.2.3            | CmpConfig           |
| SelectDeltaConfig service                  |   |                    |                     |
| 11   | SelectDeltaConfig selects a target configuration and a configuration mask to drive the signalling of changed configurations.  | 4.4.2.3            | SelectConfig        |

### 3.1.1.2. Operating Modes and Operating Mode Management

| Nr                               | Assertion  | Paragraph in spec. | Affected variants      |
|----------------------------------|--|--------------------|------------------------|
| StartNM service                  |  |                    |                        |
| 1                                | StartNM starts the local NM and leads the NM to send NM messages.  | 4.4.3.3            | Core                   |
| 2                                | StartNM causes the state transition from NMOff to NMOOn.   | 4.4.3.3            | NMStatus               |
| 3                                | StartNM returns E_OK if no error.  | 4.4.3.3            | Status                 |
| StopNM service                   |  |                    |                        |
| 4                                | StopNM stops the local NM and leads the NM to stop sending NM messages.  | 4.4.3.3            | Core                   |
| 5                                | StopNM causes the state transition from NMOOn to NMOOff.   | 4.4.3.3            | NMStatus               |
| 6                                | StopNM returns E_OK if no error.   | 4.4.3.3            | Status                 |
| GotoMode service                 |  |                    |                        |
| 7                                | GotoMode(BusSleep) serves to set the NM global operating mode to BusSleep and leads the NM to stop sending NM messages.                                    | 4.4.3.3            | BusSleep               |
| 8                                | GotoMode(Awake) serves to set back the NM global operating mode to Awake and leads the NM to restart sending NM messages.                                  | 4.4.3.3            | BusSleep               |
| 9                                | GotoMode(BusSleep) causes the state transition from NMAwake to NMBusSleep  | 4.4.3.3            | BusSleep +<br>NMStatus |
| 10                               | GotoMode(Awake) causes the state transition from NMBusSleep to NMAwake   | 4.4.3.3            | BusSleep +<br>NMStatus |
| 11                               | GotoMode returns E_OK if no error.   | 4.4.3.3            | BusSleep +<br>Status   |
| Network status (1) and GetStatus |  |                    |                        |
| 12                               | “Present configuration stable” information of network status is set if network configuration did not change during the last loop of the logical ring.      | 2.2.3.2            | NMStatus               |
| 13                               | “Present configuration stable” information of network status is cleared if the network configuration has changed during the last loop of the logical ring. | 2.2.3.2            | NMStatus               |
| 14                               | “Error, bus blocked” information of network status is set when a fatal bus error has been detected.  | 2.2.3.2            | NMStatus               |
| 15                               | “Error, bus blocked” information of network status is cleared when the fatal bus error has been repaired.  | 2.2.3.2            | NMStatus               |

|  |   |         |                           |
|--|---|---------|---------------------------|
| 16   | “NMLimphome” information of network status is set when the NM enters the LimpHome state.  | 2.2.3.2 | NMStatus                  |
| 17   | “NMLimphome” information of network status is cleared if the NM leaves the LimpHome state.  | 2.2.3.2 | NMStatus                  |
| 18   | “NMTwbsNormal or NMTwbsLimphome” information of network status is set when the NM enters the NMTwbsNormal or NMTwbsLimphome state.  | 2.2.3.2 | NMStatus                  |
| 19   | “NMTwbsNormal or NMTwbsLimphome” information of network status is cleared when the NM leaves the NMTwbsNormal or NMTwbsLimphome state.  | 2.2.3.2 | NMStatus                  |
| 20   | “Service GotoMode(BusSleep) called” information of network status is set when GotoMode(BusSleep) has been called.   | 2.2.3.2 | NMStatus                  |
| 21   | “Service GotoMode(BusSleep) called” information of network status is cleared when GotoMode(BusSleep) has not been called.   | 2.2.3.2 | NMStatus                  |
| 22   | GetStatus returns E_OK.   | 4.4.3.3 | Status                    |
| CmpStatus service                          |   |         |                           |
| 23   | CmpStatus returns the boolean value: Status = NOT (<SMask> AND (<TestStatus> EXOR <RefStatus>)), where TestStatus is the test status, RefStatus is the reference status and SMask is the test mask. | 4.4.3.3 | CmpStatus                 |
| SelectDeltaStatus service                  |   |         |                           |
| 24   | SelectDeltaStatus selects a target configuration and a status mask to drive the signalling of changed states.   | 4.4.3.3 | SelectStatus              |
| Signalling specified by InitIndDeltaStatus |   |         |                           |
| 25   | If the network status has changed and the new status matches the target status and the status mask, the task specified by InitIndDeltaStatus is activated or the specified event is set.            | 4.4.2.2 | Core                      |
| 26   | No task activation nor event setting happens if the network status has changed and the change is ignored by the status mask.  | 4.4.2.2 | Core                      |
| SilentNM service                           |   |         |                           |
| 27   | SilentNM disables the communication of the NM and leads the NM to stop sending NM messages.   | 4.5.2.1 | Active/Passive            |
| 28   | SilentNM causes the state transition from NMActive to NMPassive.  | 4.5.2.1 | Active/Passive + NMStatus |
| 29   | After a call of StartNM, the NM is always in state NMActive (it sends out NM messages).   | 4.5.2.1 | Active/Passive            |
| 30   | SilentNM returns E_OK if no error.  | 4.5.2.1 | Active/Passive + Status   |

|                |   |         |                              |
|----------------|---|---------|------------------------------|
| TalkNM service |   |         |                              |
| 31             | TalkNM enables the communication of the NM again and leads the NM to restart sending NM messages. | 4.5.2.1 | Active/Passive               |
| 32             | TalkNM causes the state transition from NMPassive to NMActive.                                    | 4.5.2.1 | Active/Passive<br>+ NMStatus |
| 33             | TalkNM returns E_OK if no error.  | 4.5.2.1 | Active/Passive<br>+ Status   |

(1) Some tests about network status are not specified in the Network Status section. They deal with the following information:

- NMPassive/NMActive (tests 28, 32),
- NMOn/NMOff (tests 2, 5),
- NMBusSleep/no NMBusSleep (tests 9, 10).

Tests about “Using of Ring Data allowed / not allowed” are defined in “Data Field Management” section.

### 3.1.1.3. Data field management

| Nr   | Assertion   | Paragraph in spec. | Affected variants      |
|--|---|--------------------|------------------------|
| Signalling specified by InitIndRingData                        |   |                    |                        |
| 1  | If a ring message is received with destination = ownstation and the logical ring runs in a stable state, the task specified by InitIndDeltaConfig is activated or the specified event is set. | 4.5.3.2 +<br>2.2.2 | RingData               |
| 2  | No task activation nor event setting happens, if a ring message is received with destination = ownstation and the configuration changed in the last loop of the logical ring.                 | 4.5.3.2 +<br>2.2.2 | RingData               |
| 3  | No task activation nor event setting happens, if a ring message is received with destination ≠ ownstation.  | 4.5.3.2 +<br>2.2.2 | RingData               |
| “Using of Ring Data not allowed” information of network status |   |                    |                        |
| 4  | If a ring message is received with destination = ownstation and the logical ring runs in a stable state, “Using of Ring Data not allowed” information of network status is cleared.           | 4.5.2.1 +<br>2.2.2 | RingData +<br>NMStatus |
| 5  | If the logical ring runs in a stable state, “Using of Ring Data not allowed” information of network status is set after ring message transmission.  | 4.5.2.1 +<br>2.2.2 | RingData +<br>NMStatus |
| ReadRingData service   |   |                    |                        |



|                          |   |                  |   |                    |   |
|--------------------------|---|------------------|---|--------------------|---|
| 6                        | Within $T_{Typ}$ period from RingData indication, ReadRingData provides the ring data received in the last ring message.  | 4.5.3.3<br>2.2.2 | + | RingData           |   |
| 7                        | If TransmitRingData is called within $T_{Typ}$ period from RingData indication, ReadRingData provides the ring data transmitted in TransmitRingData.  | 4.5.3.3<br>2.2.2 | + | RingData           |   |
| 8                        | ReadRingData returns E_OK if called within $T_{Typ}$ period from RingData indication and the network configuration remains stable.  | 4.5.3.3          |   | RingData<br>Status | + |
| 9                        | ReadRingData returns E_notOK if called within $T_{Typ}$ period from RingData indication and the network configuration has changed.  | 4.5.3.3          |   | RingData<br>Status | + |
| 10                       | ReadRingData returns E_notOK if not called within $T_{Typ}$ period from RingData indication.  | 4.5.3.3          |   | RingData<br>Status | + |
| TransmitRingData service |   |                  |   |                    |   |
| 11                       | If TransmitRingData is called within $T_{Typ}$ period from RingData indication and the network configuration remains stable, ring data provided in the service call are transmitted in the next ring message.             | 4.5.3.3<br>2.2.2 | + | RingData           |   |
| 12                       | If TransmitRingData is called within $T_{Typ}$ period from RingData indication and the network configuration has changed, the ring data fields of the last received and the next transmitted ring messages are identical. | 4.5.3.3<br>2.2.2 | + | RingData           |   |
| 13                       | If TransmitRingData is not called, the ring data fields of the last received and the next transmitted ring messages are identical.  | 4.5.3.3<br>2.2.2 | + | Core               |   |
| 14                       | TransmitRingData returns E_OK if called within $T_{Typ}$ period from RingData indication and the network configuration remains stable.  | 4.5.3.3          |   | RingData<br>Status | + |
| 15                       | TransmitRingData returns E_notOK if called within $T_{Typ}$ period from RingData indication and the network configuration has changed.  | 4.5.3.3          |   | RingData<br>Status | + |
| 16                       | TransmitRingData returns E_notOK if called outside $T_{Typ}$ period from RingData indication.   | 4.5.3.3          |   | RingData<br>Status | + |

### 3.1.2. Protocol test group

This section specifies tests purposes relative to the direct NM protocol as defined in section 2.2.8 of the NM specification document [4]. Test purposes have been mainly established from the state transition diagrams presented in the specification. They intend to verify that the NM implementation behaviour conforms to the specification. They include:

- tests of internal state activity: tests are specified to verify actions that shall be performed by the implementation while remaining in the same state,
- tests of state transitions: one test is specified for each event that leads the NM to move from a given state to another state.

Each test purpose defines both the test stimulus or stimuli to be sent and the subsequent output(s) to be observed either at the NM API or on the Data Bus. Some actions can also be triggered by internal events. The test stimuli include:

- NM API procedure calls,
- NMPDUs sent to the implementation under test,
- Timer expirations (internal stimuli):  $T_{Typ}$ ,  $T_{Max}$ ,  $T_{Error}$ ,  $T_{WaitBusSleep}$ ,
- NMrxcount overflow (no reception),
- NMtxcount overflow (no transmission),
- Fatal bus error.

The  $T_{Tx}$  timer allowing to retransmit an NM message in case of rejection from the Data Link Layer has been considered implementation specific and not taken into account.

The observable outputs are as follows:

- NMPDUs sent by the implementation under test and the various PDU fields including:
  - Source node,
  - Destination node (logical successor in the ring),
  - Reserved Area of Opcode,
  - Opcode (alive, ring or limphome),
  - Sleep.ind,
  - Sleep.ack (ring message only),
  - Ring data (ring message only).
- Status of application communication (enabled or disabled),
- Information returned by API calls : network configuration, network status, ring data.

Each test purpose also gives information on the specification variant(s) that need to be implemented for the test purpose to be verified. The variants are identified by the following terms:

- Core means that the test must be verified in any case,
- Active/Passive means that the test must be verified only if the optional SilentNM and TalkNM services are implemented,
- BusSleep means that the test must be verified only if the optional GotoMode service is implemented.

### 3.1.2.1. NMInit and NMReset

| Nr                                       | Assertion  | Paragraph in spec.                            | Affected variants            |
|--|--|---|------------------------------|
| Transitions from other states to NMReset |  |   |                              |
| 1  | When StartNM is called, the NM enters the Init then the Reset state. Limphome configuration is cleared, application communication is enabled and an alive message is transmitted.  | Fig. 5 +<br>Fig. 25 +<br>Fig. 26 +<br>Fig. 27 | Core                         |
| 2  | If T <sub>Max</sub> timer expires in Normal state and the NM is active, it enters the Reset state and transmits an alive message.  | Fig. 28 +<br>Fig. 27                          | Core                         |
| 3  | If GotoMode(Awake) is called in NMTwbsNormal state and the NM is active, it enters the Reset state and transmits an alive message.   | Fig. 28 +<br>Fig. 27                          | BusSleep                     |
| 4  | If an NM message with cleared bit sleep.ind is received in NMTwbsNormal state and the NM is active, it enters the Reset state and transmits an alive message.  | Fig. 28 +<br>Fig. 27                          | BusSleep                     |
| 5  | In LimpHome state, if the NM is active, a limphome message has been transmitted, GotoMode(Awake) is called and an NM message is received, the NM enters the Reset state. Application communication is enabled and an alive message is transmitted.                               | Fig. 29 +<br>Fig. 27                          | Core                         |
| 6  | In LimpHome state, if the NM is active, a limphome message has been transmitted, GotoMode(BusSleep) is called and an NM message is received with cleared bit sleep.ack, the NM enters the Reset state. Application communication is enabled and an alive message is transmitted. | Fig. 29 +<br>Fig. 27                          | BusSleep                     |
| 7  | In LimpHome state, if the NM is passive, GotoMode(Awake) is called and an NM message is received, the NM enters the Reset state. Application communication is enabled and no NM message is transmitted.  | Fig. 29 +<br>Fig. 27                          | Active/Passive               |
| 8  | In LimpHome state, if the NM is passive, GotoMode(BusSleep) is called and an NM message is received with cleared bit sleep.ack, the NM enters the Reset state. Application communication is enabled and no NM message is transmitted.  | Fig. 29 +<br>Fig. 27                          | Active/Passive<br>+ BusSleep |
| 9  | If GotoMode(Awake) is called in BusSleep state and the NM is active, it enters the Init then the Reset state. Limphome configuration is cleared, application communication is enabled and an alive message is transmitted.   | Fig. 25 +<br>Fig. 26 +<br>Fig. 27             | BusSleep                     |

|                   |  |                                   |          |
|-------------------|--|-----------------------------------|----------|
| 10                | If an NM message is received in BusSleep state and the NM is active, it enters the Init then the Reset state. Limphome configuration is cleared, application communication is enabled and an alive message is transmitted. | Fig. 25 +<br>Fig. 26 +<br>Fig. 27 | BusSleep |
| InitReset         |  |                                   |          |
| 11                | In InitReset, the normal configuration is initialized. It contains only the local node.  | Fig. 27                           | Core     |
| ResetActive state |  |                                   |          |
| 12                | In ResetActive state, the alive message is initialized as follows: destination = ownstation, reserved area of OpCode initialized and sleep.ind cleared.  | Fig. 27                           | Core     |

Coverage of the NMReset specification by the test purposes is shown in the state diagram below. Labels in circles indicate the paths and statements covered by each test.

Notation:

- ① refers to test number 1 of NMReset table,
- ②N2 refers to test number 3 of NMLimphome table,
- ③L3 refers to test number 2 of NMNormal table,
- ④1:6 refers to test numbers 1 to 6 of NMReset table.

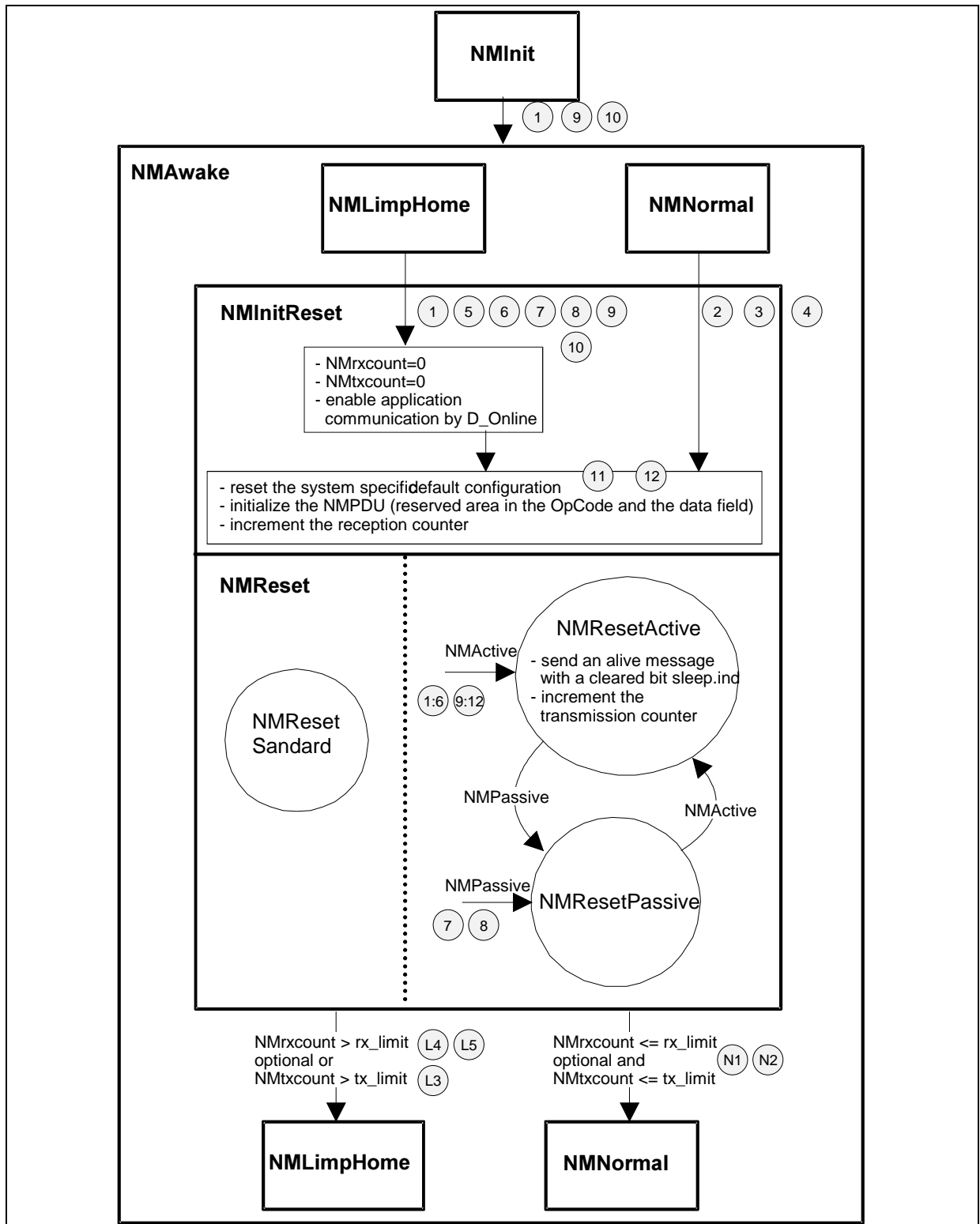


Figure 2 Test coverage of NMReset state

### 3.1.2.2. NMNormal

The NM specification defines 4 substates of NMNormal:

- NormalActive,
- NormalPassive,
- NormalActivePrepSleep,
- NormalPassivePrepSleep.

In the table below, the wording "Normal state" covers the four substates of NMNormal.

Unless otherwise specified, NM messages are sent with cleared bits sleep.ind and sleep.ack.

| Nr                                   | Assertion  | Paragraph in spec. | Affected variants |
|--------------------------------------|--|--------------------|-------------------|
| Transitions from NMReset to NMNormal |  |                    |                   |
| 1                                    | If NMtxcount <= tx_limit and NMrxcount <= rx_limit in Reset state, the NM enters the Normal state. In Active mode, it transmits a first ring message after T <sub>Typ</sub> .                | Fig. 27 + Fig. 28  | Core              |
| 2                                    | If NMtxcount <= tx_limit and NMrxcount <= rx_limit in Reset state, the NM enters the Normal state. In Passive mode, it will not transmit any message.  | Fig. 27 + Fig. 28  | Active/Passive    |
| NormalStandardNM                     |  |                    |                   |
| 3                                    | In Normal state, the NM updates the present configuration on alive message reception   | Fig. 28            | Core              |
| 4                                    | In Normal state, the NM updates the present configuration on ring message reception  | Fig. 28            | Core              |
| 5                                    | In Normal state, the NM updates the logical successor on alive message reception   | Fig. 28            | Core              |
| 6                                    | In Normal state, the NM updates the logical successor on ring message reception  | Fig. 28            | Core              |
| 7                                    | In Normal state, the NM updates the limphome configuration on limphome message reception   | Fig. 28            | Core              |
| 8                                    | In Normal state, the NM sends an alive message on ring message reception if skipped in the logical ring and if NMActive. Sleep.ind is cleared if GotoMode (BusSleep) were not called before. | Fig. 28            | Core              |
| 9                                    | In Normal state, the NM sends an alive message on ring message reception if skipped in the logical ring and if NMActive. Sleep.ind is set if GotoMode(BusSleep) were called before.          | Fig. 28            | BusSleep          |
| 10                                   | In Normal state, the NM does not send any alive message on ring message reception if skipped in the logical ring and if NMPassive.   | Fig. 28 + Fig. 38  | Active/Passive    |

|   |  |                      |                |
|---|--|----------------------|----------------|
| 11  | In Normal state, the NM does not send any alive message on ring message reception if not skipped in the logical ring.  | Fig. 28 +<br>Fig. 38 | Core           |
| NormalActive state                              |  |                      |                |
| 12  | In NormalActive state, the NM passes the ring message delayed $T_{Typ}$ if source = destination in the received message  | Fig. 28              | Core           |
| 13  | In NormalActive state, the NM passes the ring message delayed $T_{Typ}$ if destination = ownstation in the received message  | Fig. 28              | Core           |
| 14  | In NormalActive state, the NM will not pass the regular ring message if a ring message is received before $T_{Typ}$ expiration   | Fig. 28              | Core           |
| Internal transitions from/to NormalActive state |  |                      |                |
| 15  | The NM stops passing the regular ring message when SilentNM is called while in NormalActive state (transition to NormalPassive)  | Fig. 28              | Active/Passive |
| 16  | The NM passes the regular ring message again when TalkNM is called while in NormalPassive state (transition to NormalActive)   | Fig. 28              | Active/Passive |
| 17  | If GotoMode(BusSleep) is called in NormalActive state, the NM transmits the next regular ring message with set bit sleep.ind   | Fig. 28              | BusSleep       |
| 18  | If an NM message is received with cleared bit sleep.ind while in NormalActivePrepSleep state, the NM will pass the next regular ring message with set bit sleep.ind and cleared bit sleep.ack (transition to NormalActive) | Fig. 28              | BusSleep       |
| 19  | If GotoMode(Awake) is called in NormalActive-PrepSleep state, the NM transmits the next regular ring message with cleared bits sleep.ind and sleep.ack (transition to NormalActive)  | Fig. 28              | BusSleep       |
| NormalActivePrepSleep state                     |  |                      |                |
| 20  | In NormalActivePrepSleep state, the NM passes the ring message delayed $T_{Typ}$ if source = destination and sleep.ind is set in the received message. Sleep.ind and sleep.ack are set in the transmitted message.         | Fig. 28 +<br>Fig. 21 | BusSleep       |
| 21  | In NormalActivePrepSleep state, the NM passes the ring message delayed $T_{Typ}$ if destination = ownstation and sleep.ind is set in the received message. Sleep.ind and sleep.ack are set in the transmitted message.     | Fig. 28 +<br>Fig. 21 | BusSleep       |
| 22  | In NormalActivePrepSleep state, the NM will not pass the regular ring message if a ring message with set bit sleep.ind is received before $T_{Typ}$ expiration   | Fig. 28              | BusSleep       |

|  |  |         |                           |
|--|--|---------|---------------------------|
| Internal transitions from/to Normal(Active/Passive)PrepSleep state |  |         |                           |
| 23   | The NM does not pass the regular ring message when SilentNM is called while in NormalActivePrepSleep state (transition to NormalPassive PrepSleep)   | Fig. 28 | Active/Passive + BusSleep |
| 24   | The NM passes the regular ring message with set bits sleep.ind and sleep.ack when TalkNM is called while in NormalPassivePrepSleep state (transition to NormalActivePrepSleep)   | Fig. 28 | Active/Passive + BusSleep |
| 25   | In NormalPassive state, if GotoMode(BusSleep) then TalkNM are called, the NM passes the regular ring message with set bits sleep.ind and sleep.ack (test of transition to NormalPassivePrepSleep)  | Fig. 28 | Active/Passive + BusSleep |
| 26   | If an NM message is received with cleared bit sleep.ind while in NormalPassivePrepSleep state and then TalkNM is called, the NM will pass the next regular ring message with cleared bits sleep.ind and sleep.ack (transition to NormalActive) | Fig. 28 | Active/Passive + BusSleep |
| 27   | In NormalPassivePrepSleep state, if GotoMode(Awake) then TalkNM are called, the NM passes the regular ring message with cleared bits sleep.ind and sleep.ack (test of transition to NormalPassive)   | Fig. 28 | Active/Passive + BusSleep |
| Transitions from NMNormal to NMTwbsNormal state                    |  |         |                           |
| 28   | In Normal state, if GotoMode(BusSleep) is called and a ring message is received with set bit sleep.ack, the NM enters the NMTwbsNormal state and application communication is disabled   | Fig. 28 | BusSleep                  |
| 29   | When the regular ring message is transmitted with set bit sleep.ack in NormalActivePrepSleep, the NM enters the NMTwbsNormal state and application communication is disabled   | Fig. 28 | BusSleep                  |
| NMTwbsNormal state   |  |         |                           |
| 30   | In NMTwbsNormal state, the NM accepts and ignores NM messages received with set bit sleep.ind  | Fig. 34 | BusSleep                  |

Coverage of the NMReset specification by the test purposes is shown in the state diagram below. Labels in circles indicate the paths and statements covered by each test.



Notation:

- ① refers to test number 1 of NMNormal table,
- L2 refers to test number 2 of NMLimhome table,
- R3 refers to test number 3 of NMReset table,
- S1 refers to test number 1 of NMBusSleep table.

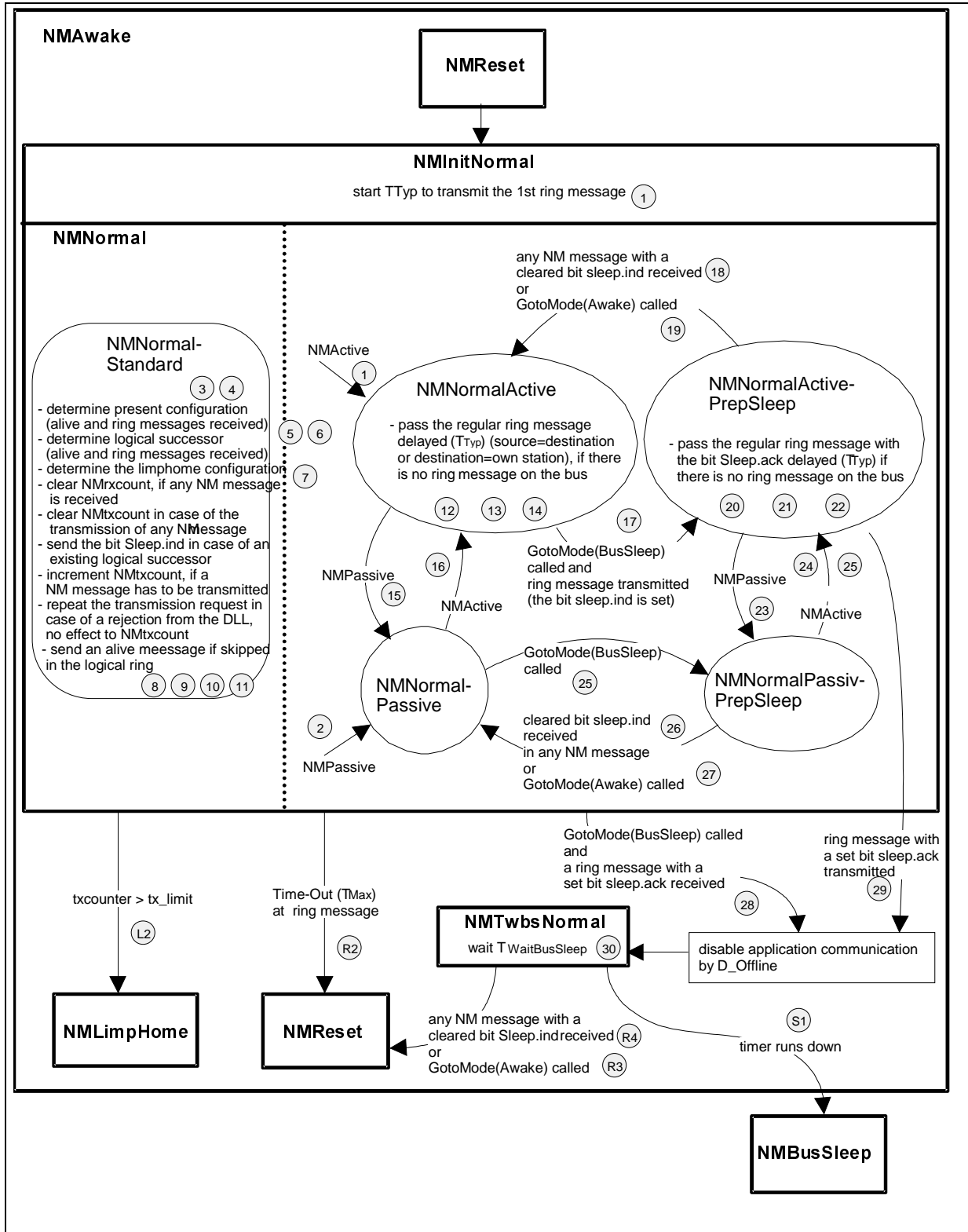


Figure 3 Test coverage of NMNormal state

### 3.1.2.3. NMLimpHome

The NM specification defines 4 substates of NMLimpHome:

- LimpHomeActive,
- LimpHomePassive,
- LimpHomeActivePrepSleep,
- LimpHomePassivePrepSleep.

In the table below, the wording "Normal state" covers the four substates of NMLimpHome.

Unless otherwise specified, LimpHome messages are sent with cleared bit sleep.ind.

| Nr  | Assertion   | Paragraph in spec. | Affected variants |
|---|---|--------------------|-------------------|
| Transitions from other states to NMLimpHome       |   |                    |                   |
| 1   | If a fatal bus error is detected in any state, the NM enters the LimpHome state and transmits a limphome message after $T_{Error}$ .              | Fig. 29            | Core              |
| 2   | If $NMtxcount > tx\_limit$ in Normal state, the NM enters the LimpHome state and transmits a limphome message after $T_{Error}$ .                 | Fig. 28 + Fig. 29  | Core              |
| 3   | If $NMtxcount > tx\_limit$ in Reset state, the NM enters the LimpHome state and transmits a limphome message after $T_{Error}$ .                  | Fig. 27 + Fig. 29  | Core              |
| 4   | If $NMrxcount > rx\_limit$ in Reset state and the NM is active, it enters the LimpHome state and transmits a limphome message after $T_{Error}$ . | Fig. 27 + Fig. 29  | Core              |
| 5   | If $NMrxcount > rx\_limit$ in Reset state and the NM is passive, it enters the LimpHome state and does not transmit any NM message.               | Fig. 27 + Fig. 29  | Active/Passive    |
| InitLimpHome                                      |   |                    |                   |
| 6   | After fatal bus error detection, application communication is disabled.   | Fig. 29            |                   |
| LimpHomeActive state                              |   |                    |                   |
| 7   | In LimpHomeActive state, the NM transmits a limphome message every $T_{Error}$ .  | Fig. 29            | Core              |
| 8   | In LimpHomeActive state, the NM enables application communication every $T_{Error}$ .   | Fig. 29            | Core              |
| LimpHomePassive state                             |   |                    |                   |
| 9   | In LimpHomePassive state, the NM enables application communication every $T_{Error}$ .  | Fig. 29            | Active/Passive    |
| Internal transitions from/to LimpHomeActive state |   |                    |                   |

|  |   |         |                           |
|--|---|---------|---------------------------|
| 10   | The NM stops sending limphome messages when SilentNM is called while in LimpHomeActive state (transition to LimpHomePassive)  | Fig. 29 | Active/Passive            |
| 11   | The NM restarts sending limphome messages if TalkNM is called while in LimpHomePassive state (transition to LimpHomeActive)   | Fig. 29 | Active/Passive            |
| 12   | If GotoMode(BusSleep) is called in LimpHomeActive state, the NM transmits the next limphome message with set bit sleep.ind. Application communication is disabled after $T_{Max}$ .   | Fig. 29 | BusSleep                  |
| 13   | If an NM message is received with cleared bit sleep.ind while in LimpHomeActivePrepSleep state, the NM will transmit the next limphome message with set bit sleep.ind (transition to LimpHome-Active)                       | Fig. 29 | BusSleep                  |
| 14   | If GotoMode(Awake) is called in LimpHomeActive-PrepSleep state, the NM transmits the next limphome message with cleared bit sleep.ind (transition to LimpHomeActive).   | Fig. 29 | BusSleep                  |
| Internal transitions from/to LimpHome(Active/Passive)PrepSleep state |   |         |                           |
| 15   | If SilentNM is called in LimpHomeActivePrepSleep state, the NM enters the LimpHomePassivePrepSleep state and application communication is disabled after $T_{Max}$ .  | Fig. 29 | Active/Passive + BusSleep |
| 16   | In LimpHomePassive state, if GotoMode(BusSleep) then TalkNM are called, the NM enters the LimpHome-PassivePrepSleep state then the LimpHomeActivePrepSleep state. Application communication is disabled after $T_{Max}$ .   | Fig. 29 | Active/Passive + BusSleep |
| 17   | In LimpHomePassive state, if GotoMode(BusSleep) then GotoMode(Awake) are called, the NM enters the LimpHomePassivePrepSleep state then the LimpHome-Passive state. Application communication is enabled after $T_{Error}$ . | Fig. 29 | Active/Passive + BusSleep |
| Other transitions between NMLimpHome and NMTwbsLimpHome              |   |         |                           |
| 18   | In LimpHome state, if GotoMode(BusSleep) is called and a ring message is received with set bit sleep.ack, the NM enters the NMTwbsLimpHome state and application communication is disabled.                                 | Fig. 29 | BusSleep                  |
| 19   | If GotoMode(Awake) is called in NMTwbsLimpHome state and the NM is active, it enters the LimpHome state and transmits a limphome message after $T_{Error}$ .  | Fig. 29 | BusSleep                  |

|                      |  |         |          |
|----------------------|--|---------|----------|
| 20                   | If an NM message with cleared bit sleep.ind is received in NMTwbsNormal state and the NM is active, it enters the Reset state and transmits a limphome message after $T_{Error}$ . | Fig. 29 | BusSleep |
| NMTwbsLimpHome state |  |         |          |
| 21                   | In NMTwbsLimpHome state, the NM accepts and ignores NM messages received with set bit sleep.ind.   | Fig. 37 | BusSleep |

Coverage of the NMReset specification by the test purposes is shown in the state diagram below. Labels in circles indicate the paths and statements covered by each test.

Notation:

- ① refers to test number 1 of NMLimphome table,
- Ⓜ<sub>R5</sub> refers to test number 5 of NMReset table,
- Ⓜ<sub>S2</sub> refers to test number 2 of NMBusSleep table.

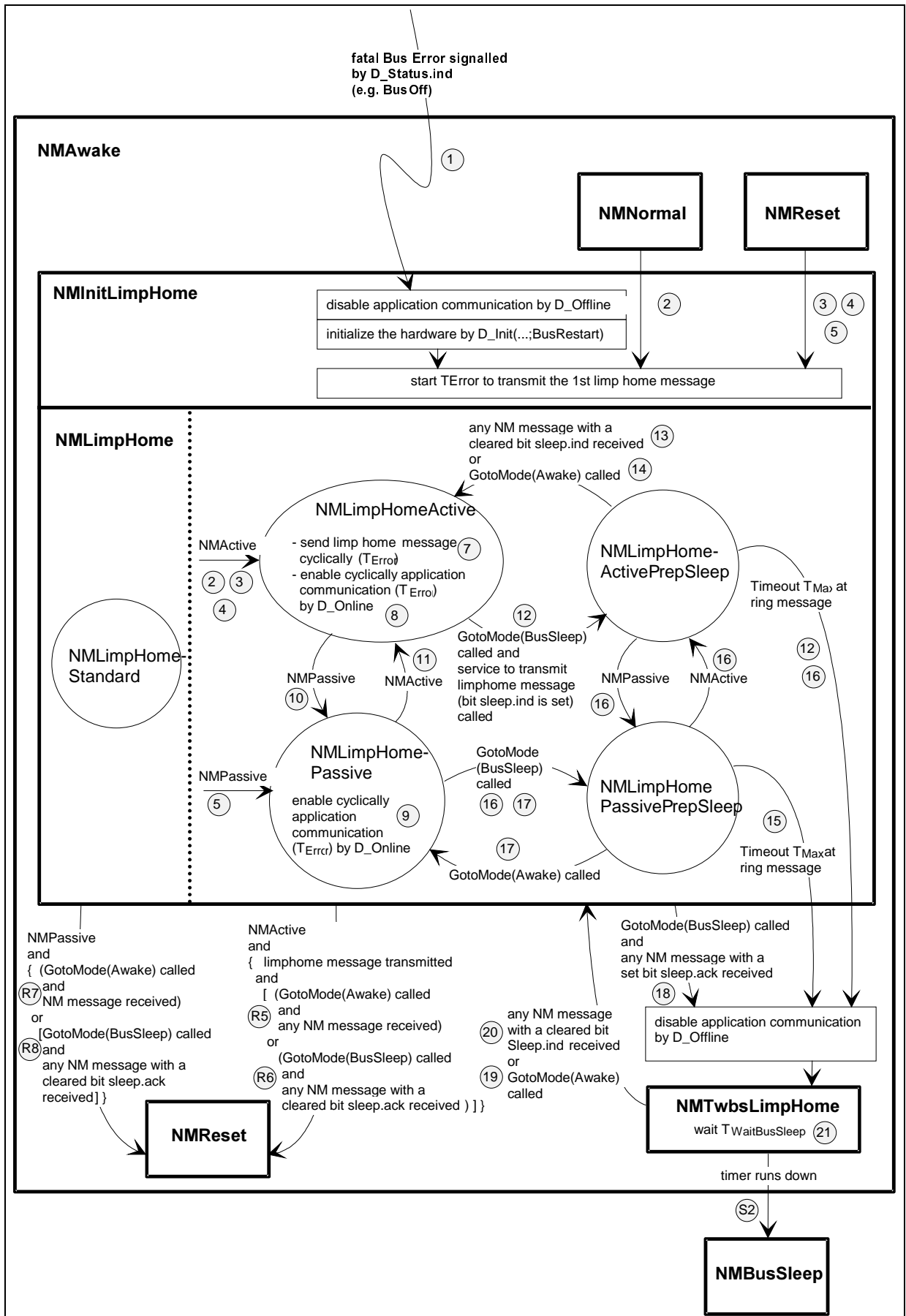


Figure 4 Test coverage of NMLimpHome state

### 3.1.2.4. NMBusSleep

| Nr  | Assertion  | Paragraph in spec. | Affected variants |
|---|--|--------------------|-------------------|
| Transitions from other states to NMBusSleep |  |                    |                   |
| 1   | If $T_{\text{WaitBusSleep}}$ timer expires in NMTwbsNormal state, the NM enters the BusSleep state and does not transmit NM messages any longer.   | Fig. 28 + Fig. 26  | BusSleep          |
| 2   | If $T_{\text{WaitBusSleep}}$ timer expires in NMTwbsLimpHome state, the NM enters the BusSleep state and does not transmit NM messages any longer. | Fig. 28 + Fig. 26  | BusSleep          |

## 3.2. Test purposes of Indirect NM

This section contains a set of test purposes relevant to Indirect NM services and protocols. These test purposes provide ground material for developing the TTCN test suite which will be used to evaluate conformance to indirect NM specification [4].

### 3.2.1. Service test group

This section specifies tests purposes relative to the indirect NM API as defined in sections 4.4 and 4.6 of the NM specification.

Tests of service behaviour are specified only when the service is not described in the SDL specification. Consequently, behaviour tests for StartNM, StopNM and InitConfig are defined later in Protocol test group. The tests provided in this section are relating to the status code returned by the APIs.

Each test purpose defines both the test stimulus to be sent and the subsequent output(s) to be observed at the NM API.

The test stimuli include API calls with different sets of input parameters and also monitored messages causing the implementation either:

- to change the network status or the extended network status, or
- to change the configuration or the extended configuration of supervised nodes.

The observable outputs are either:

- the stati returned by API calls and the associated output parameters such as network status and network configuration, or
- the tasks activations or event settings carried out by the implementation on monitored message reception.

Each test purpose also gives information on the specification variant(s) that need to be implemented for the test purpose to be verified.

The implementation variants are identified by the following terms:

- Core means that the test purpose must be verified in any implementation,

- BusSleep means that the test purpose must be verified only if the optional GotoMode service is implemented,
- CmpConfig means that the test purpose must be verified only if the optional CmpConfig service is implemented,
- SelectConfig means that the test purpose must be verified only if optional SelectDeltaConfig service is implemented.
- Status refers to API return status. It means that the test purpose must be verified only if the tested status is actually implemented (see NM specification, section 2.3 - 2<sup>nd</sup> §).
- NMStatus refers to network status. It means that the test purpose must be verified only if the optional GetStatus is actually implemented.
- CmpStatus means that the test purpose must be verified only if the optional CmpStatus service is implemented,
- SelectStatus means that the test purpose must be verified only if optional SelectDeltaStatus service is implemented.

In the following subsections, the tests marked with (\*) are only applicable to “One monitoring time-out per message” implementations.

### 3.2.1.1. Configuration Management

| Nr   | Assertion  | Paragraph in spec. | Affected variants |
|--|--|--------------------|-------------------|
| Signalling specified by InitIndDeltaConfig |  |                    |                   |
| 1  | If a normal configuration change selected by SelectDeltaConfig occurs and the target configuration is matched, the task specified by InitIndDeltaConfig is activated or the specified event is set.    | 4.4.2.2            | Core              |
| 2 *  | If an extended configuration change selected by SelectDeltaConfig occurs and the target configuration is matched, the task specified by InitIndDeltaConfig is activated or the specified event is set. | 4.4.2.2            | Core              |
| 3  | No task activation nor event setting happens for normal configuration changes that were not selected.  | 4.4.2.2            | Core              |
| 4 *  | No task activation nor event setting happens for extended configuration changes that were not selected.  | 4.4.2.2            | Core              |
| InitConfig service                         |  |                    |                   |
| 5 *  | InitConfig returns E_OK.   | 4.4.2.3            | Status            |
| GetConfig service                          |  |                    |                   |
| 6  | GetConfig provides the current normal configuration if the ConfigKind parameter equals “Normal”.   | 4.4.2.3            | Core              |
| 7 *  | GetConfig provides the current extended configuration if the ConfigKind parameter equals “Normal extended”.  | 4.4.2.3            | Core              |

|    |  |         |              |
|----|--|---------|--------------|
| 8  | GetConfig returns E_OK.<br>CmpConfig service   | 4.4.2.3 | Status       |
| 9  | CmpConfig returns the boolean value: Status = NOT (<CMask> AND (<TestConfig> EXOR <RefConfig>)), where TestConfig is the test configuration, RefConfig is the reference configuration and CMask is the test mask.<br>SelectDeltaConfig service | 4.4.2.3 | CmpConfig    |
| 10 | SelectDeltaConfig selects a target configuration and a configuration mask to drive the signalling of changed configurations.   | 4.4.2.3 | SelectConfig |

### 3.2.1.2. Operating Modes and Operating Mode Management

| Nr   | Assertion   | Paragraph in spec. | Affected variants    |
|--|---|--------------------|----------------------|
| StartNM service                            |   |                    |                      |
| 1  | StartNM returns E_OK if no error.   | 4.4.3.3            | Status               |
| StopNM service                             |   |                    |                      |
| 2  | StopNM returns E_OK if no error.  | 4.4.3.3            | Status               |
| GotoMode service                           |   |                    |                      |
| 3 *  | GotoMode returns E_OK if no error.  | 4.4.3.3            | BusSleep +<br>Status |
| GetStatus                                  |   |                    |                      |
| 4  | GetStatus provides the current status of the network.   | 2.2.3.2            | NMStatus             |
| 5  | GetStatus returns E_OK.   | 4.4.3.3            | NMStatus +<br>Status |
| CmpStatus service                          |   |                    |                      |
| 6  | CmpStatus returns the boolean value: Status = NOT (<SMask> AND (<TestStatus> EXOR <RefStatus>)), where TestStatus is the test status, RefStatus is the reference status and SMask is the test mask. | 4.4.3.3            | CmpStatus            |
| SelectDeltaStatus service                  |   |                    |                      |
| 7  | SelectDeltaStatus selects a target status and a status mask to drive the signalling of changed states.  | 4.4.2.3            | SelectStatus         |
| Signalling specified by InitIndDeltaStatus |   |                    |                      |
| 8  | If the network status has changed and the new status matches the target status and the status mask, the task specified by InitIndDeltaStatus is activated or the specified event is set.            | 4.4.2.2            | Core                 |



|   |  |         |      |
|---|--|---------|------|
| 9 | No task activation nor event setting happens if the network status has changed and the change is ignored by the status mask. | 4.4.2.2 | Core |
|---|--|---------|------|

### 3.2.2. Protocol test group - One global time-out TOB

This section specifies tests purposes relative to the indirect NM protocol version called “one global time-out TOB”, as defined in chapter 3 of the NM specification. Test purposes have been established from the SDL diagrams presented in the specification, according to the Conformance Methodology described in document [1]. They intend to verify that the NM implementation behaviour conforms to the specification. They include:

- tests of state activity: tests are specified to verify actions that shall be performed by the implementation on a given input,
- tests of state transitions: one test is specified for each event that leads the NM to move from a given state to another state.

Each test purpose defines both the test stimulus or stimuli to be sent and the subsequent output(s) to be observed at the NM API. Some actions can also be triggered by internal events. The test stimuli include:

- NM API procedure calls,
- Monitored messages sent to the implementation under test or transmitted by the test application,
- Timer expirations (internal stimuli): TOB,  $T_{Error}$ .
- Fatal bus error.

The observable outputs are as follows:

- Status of application communication (enabled or disabled),
- Information returned by API calls : network configuration, network status.

Each test purpose also gives information on the specification variant(s) that need to be implemented for the test purpose to be verified.

The Core variant means that the test shall be executed in any case.

The "NMStatus" variant implies a test of a network status change. There are two ways of testing this information:

1. Test by a GetStatus API call, if this optional procedure is effectively implemented,
2. Test of a task activation occurrence or an OS event occurrence, depending on the status changes selected through the SelectDeltaStatus service.

In the test purposes, the changes to network status information are expressed by the following assertions:

| Network status value |                    | Assertion   |
|----------------------|--------------------|---|
| 0                    | No error           | “Error, bus blocked” information of network status is cleared |
| 1                    | Error, Bus blocked | “Error, bus blocked” information of network status is set     |
| 0                    | NMOn               | “NMOn/NMOff” information of network status is cleared         |
| 1                    | NMOff              | “NMOn/NMOff” information of network status is set             |
| 0                    | no NMLimpHome      | “NMLimpHome” information of network status is cleared         |
| 1                    | NMLimpHome         | “NMLimpHome” information of network status is set             |

### 3.2.2.1. Handling of StartNM and StopNM

| Nr                               | Assertion  | Paragraph in spec.          | Affected variants |
|----------------------------------|--|-----------------------------|-------------------|
| User’s communication management  |  |                             |                   |
| 1                                | When StartNM is called, application communication is enabled.  | Fig. 50                     | Core              |
| Network configuration management |  |                             |                   |
| 2                                | When StartNM is called, network configuration is initialised. Own node is considered mute, remote nodes are considered absent. | Fig. 50 + Fig. 54           | Core              |
| Network status management        |  |                             |                   |
| 3                                | When StartNM is called, all network status information is cleared.   | Fig. 50 + Fig. 55 + Table 8 | NMStatus          |
| 4                                | When StopNM is called, NMOn/NMOff information of network status is set.  | Fig. 50 + Table 8           | NMStatus          |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.

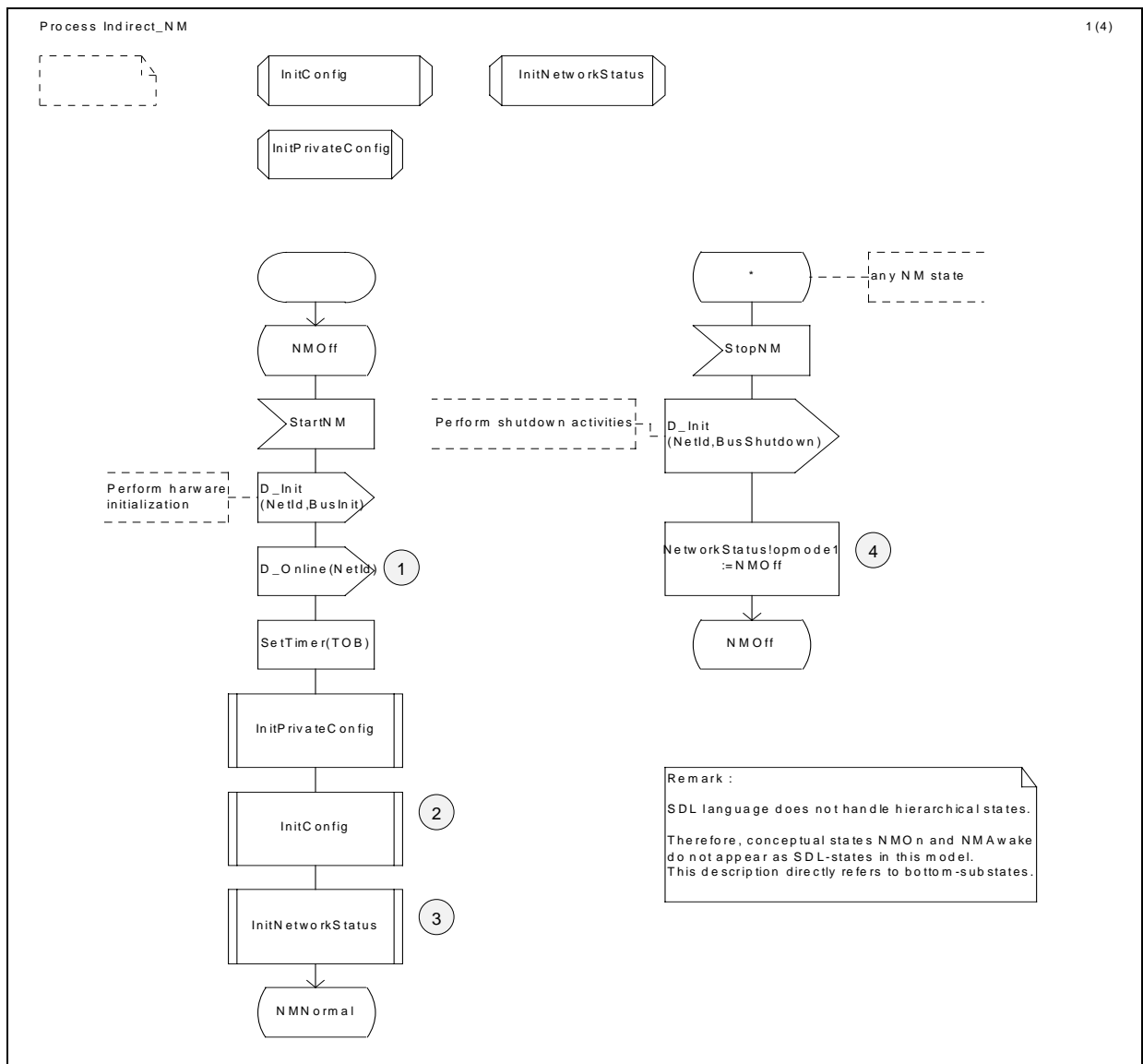


Figure 5 Test coverage of the services StartNM and StopNM

### 3.2.2.2. NMNormal

| Nr                               | Assertion  | Paragraph in spec. | Affected variants |
|----------------------------------|--|--------------------|-------------------|
| Network configuration management |  |                    |                   |
| 1                                | In NMNormal state, a supervised node is considered present if the monitored message from that node was received at least once in the last TOB period | Fig. 51            | Core              |
| 2                                | In NMNormal state, a supervised node is considered absent if the monitored message from that node was not received in the last TOB period            | Fig. 51            | Core              |

|   |  |         |      |
|---|--|---------|------|
| 3 | In NMNormal state, own node is considered not mute if the monitored application message was transmitted at least once in the last TOB period | Fig. 51 | Core |
| 4 | In NMNormal state, own node is considered mute if the monitored application message was not transmitted in the last TOB period               | Fig. 51 | Core |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.

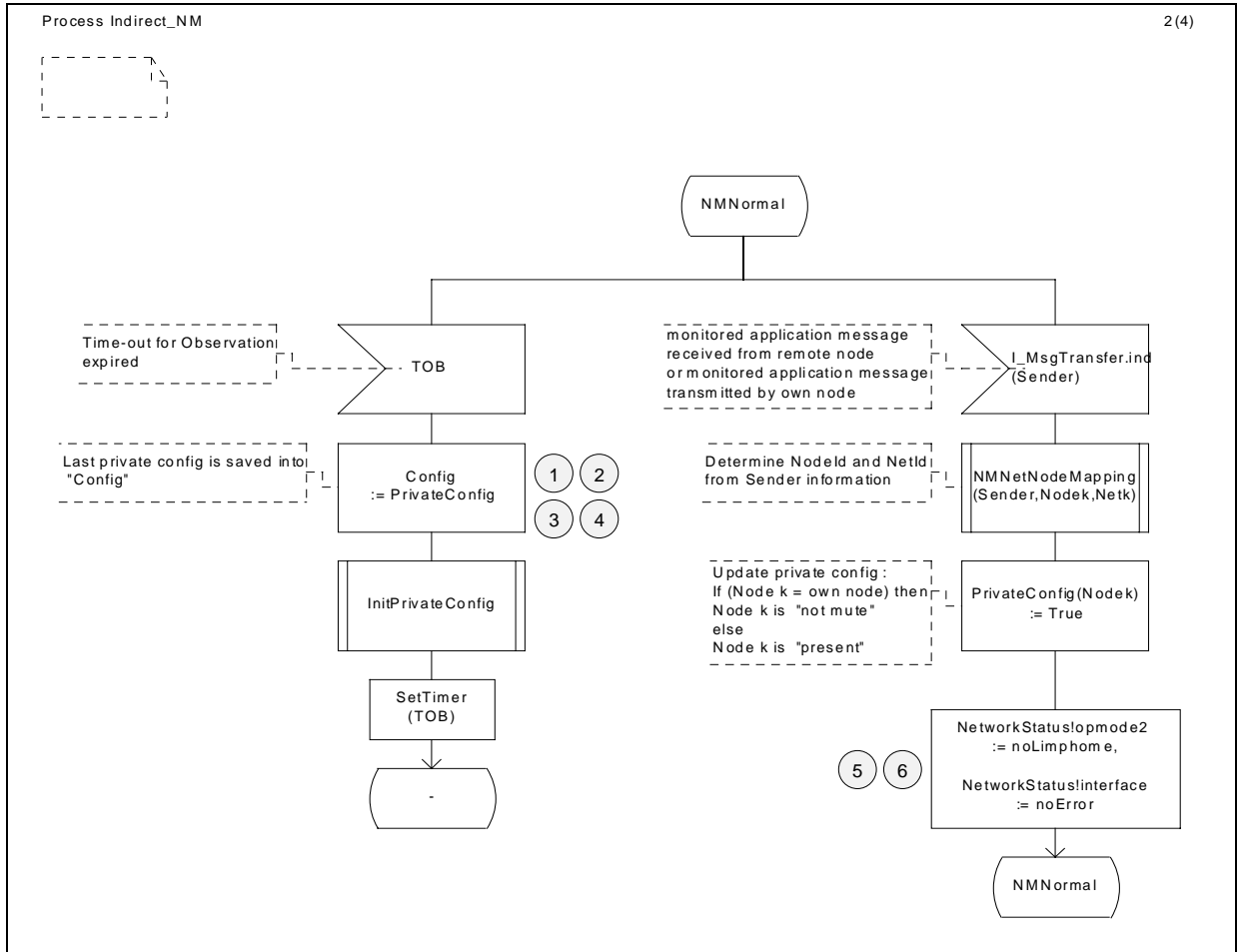
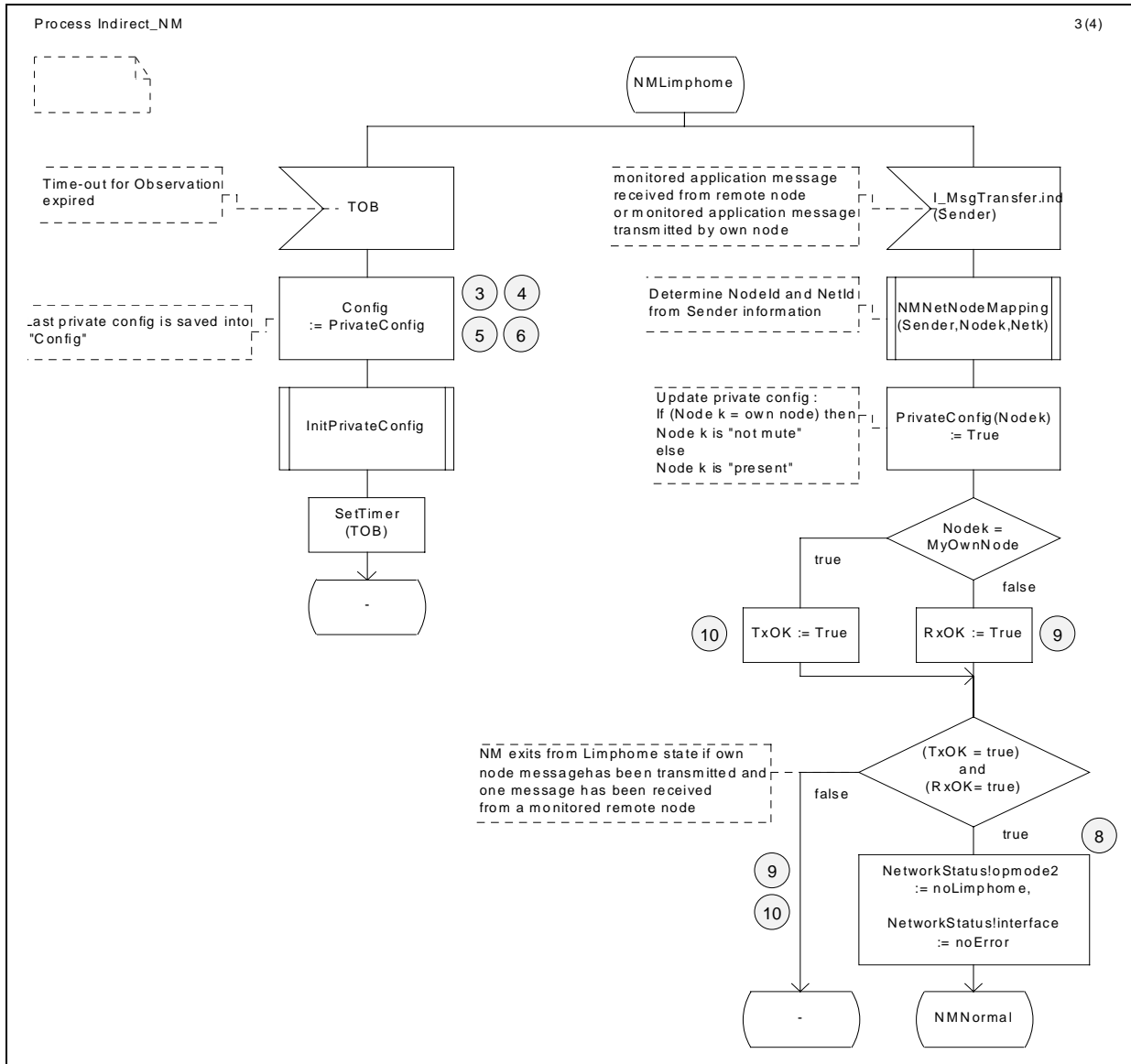


Figure 6 Test coverage of the NMNormal state

### 3.2.2.3. NMLimphome

| Nr                               | Assertion  | Paragraph in spec. | Affected variants |
|----------------------------------|--|--------------------|-------------------|
| User's communication management  |  |                    |                   |
| 1                                | If a fatal bus error is detected in any state, the NM enters the NMLimphome state and application communication is disabled.   | Fig. 53            | Core              |
| 2                                | In NMLimphome state, application communication is enabled after $T_{Error}$  | Fig. 53            | Core              |
| Network configuration management |  |                    |                   |
| 3                                | In NMLimphome state, a supervised node is considered present if the monitored message from that node was received at least once in the last TOB period   | Fig. 52            | Core              |
| 4                                | In NMLimphome state, a supervised node is considered absent if the monitored message from that node was not received in the last TOB period  | Fig. 52            | Core              |
| 5                                | In NMLimphome state, own node is considered not mute if the monitored application message was transmitted at least once in the last TOB period   | Fig. 52            | Core              |
| 6                                | In NMLimphome state, own node is considered mute if the monitored application message was not transmitted in the last TOB period   | Fig. 52            | Core              |
| Network status management        |  |                    |                   |
| 7                                | On transition to NMLimphome, "Error, bus blocked" and "NMLimphome" information of network status is set.   | Fig. 53 + Table 8  | NMStatus          |
| 8                                | In NMLimphome state, if a monitored message from a remote node is received and a monitored message from own node is transmitted, the NM enters the NMNormal state. "Error, bus blocked" and "NMLimphome" information of network status is cleared. | Fig. 52 + Table 8  | NMStatus          |
| 9                                | In NMLimphome state, if a monitored message from a remote node is received but no monitored message from own node is transmitted, "Error, bus blocked" and "NMLimphome" information of network status remains set.                                 | Fig. 52 + Table 8  | NMStatus          |
| 10                               | In NMLimphome state, if a monitored message from own node is transmitted but no monitored message from a remote node is received, "Error, bus blocked" and "NMLimphome" information of network status remains set.                                 | Fig. 52 + Table 8  | NMStatus          |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.



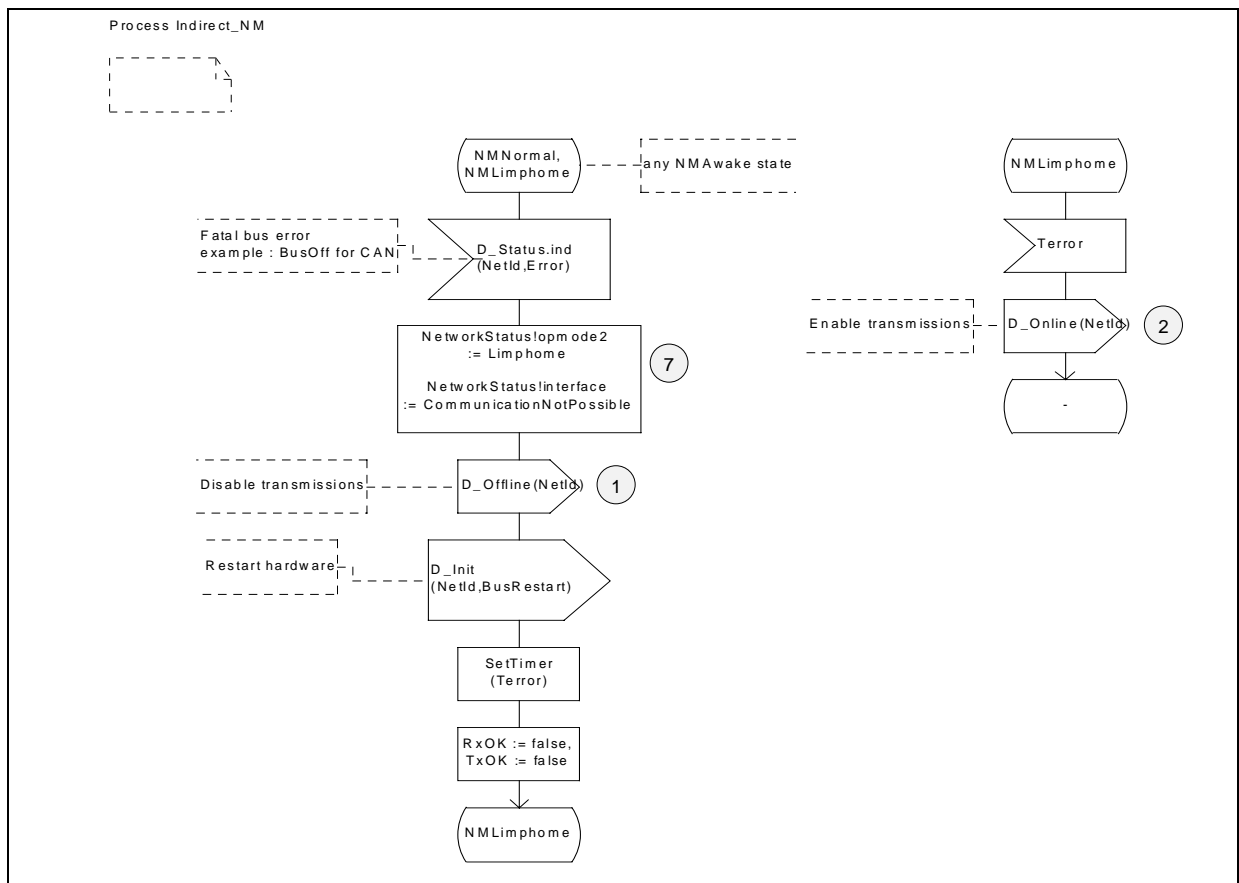


Figure 7 Test coverage of the NMLimpHome state

### 3.2.3. Protocol test group - One monitoring time-out per message

This section specifies tests purposes relative to the indirect NM protocol version called “one monitoring time-out per message”, as defined in chapter 3 of the NM specification. Test purposes have been established from the SDL diagrams presented in the specification, according to the Conformance Methodology described in document [1]. They intend to verify that the NM implementation behaviour conforms to the specification. They include:

- tests of state activity: tests are specified to verify actions that should be performed by the implementation on a given input,
- tests of state transitions: one test is specified for each event that leads the NM to move from a given state to another state.

Each test purpose defines both the test stimulus to be sent and the subsequent output(s) to be observed at the NM API. Some actions can also be triggered by internal events. The test stimuli include:

- NM API procedure calls,
- Monitored messages sent to the implementation under test or transmitted by the user application,
- Timer expirations (internal stimuli): specific time-out to each monitored message,  $T_{Error}$ ,  $T_{WaitBusSleep}$ ,
- Monitoring counter overflow (internal stimuli),
- Fatal bus error.

The observable outputs are as follows:

- Status of application communication (enabled or disabled),
- Information returned by API calls : network configuration, extended network configuration, network status, extended network status.

Each test purpose also gives information on the specification variant(s) that need to be implemented for the test purpose to be verified.

The Core variant means that the test shall be executed in any case.

The BusSleep variant means that the test must be executed only if the optional GotoMode service is implemented.

The "NMStatus" variant implies a test of a network status change. There are two ways of testing this information:

1. Test by a GetStatus API call, if this optional procedure is effectively implemented,
2. Test of a task activation occurrence or an OS event occurrence, depending on the status changes selected through the SelectDeltaStatus service.

In the test purposes, the changes to network status information are expressed by the following assertions:

| Network status value | Assertion   |
|----------------------|---|
| 0 No error           | "Error, bus blocked" information of network status is cleared |
| 1 Error, Bus blocked | "Error, bus blocked" information of network status is set     |
| 0 NMON               | "NMON/NMOff" information of network status is cleared         |
| 1 NMOff              | "NMON/NMOff" information of network status is set             |
| 0 no NMLimpHome      | "NMLimpHome" information of network status is cleared         |
| 1 NMLimpHome         | "NMLimpHome" information of network status is set             |
| 0 no NMBusSleep      | "NMBusSleep" information of network status is cleared         |
| 1 NMBusSleep         | "NMBusSleep" information of network status is set             |
| 0 no NMWaitBusSleep  | "NMWaitBusSleep" information of network status is cleared     |
| 1 NMWaitBusSleep     | "NMWaitBusSleep" information of network status is set         |

The changes to extended network status information are expressed by the following assertions:

| Extended network status value        | Assertion  |
|--------------------------------------|--|
| 00 No error                          | "Error, Communication not possible" information of network status is cleared |
| 10 Error, Communication not possible | "Error, Communication not possible" information of network status is set     |

Note that information "Error, Communication possible" is never used in the SDL specification. Therefore, there is no test purpose to check it.



### 3.2.3.1. Handling of StartNM, StopNM and InitConfig

| Nr                               | Assertion   | Paragraph in spec.            | Affected variants  |
|----------------------------------|---|-------------------------------|--------------------|
| User's communication management  |   |                               |                    |
| 1                                | When StartNM is called, application communication is enabled.   | Fig. 56                       | Core               |
| Network configuration management |   |                               |                    |
| 2                                | When StartNM is called, network configuration is initialised. Own node is considered mute, remote nodes are considered absent.  | Fig. 56 + Fig. 62             | Core               |
| 3                                | When StartNM is called, extended network configuration is initialised. Own node is considered static not mute, remote nodes are considered static present.  | Fig. 56 + Fig. 62             | Core               |
| 4                                | If InitConfig is called in NMNormal, NMLimpHome or NMWaitBusSleep state, extended network configuration is initialised. Own node is considered static not mute, remote nodes are considered static present. | Fig. 56 + Fig. 62             | Core (or BusSleep) |
| Network status management        |   |                               |                    |
| 5                                | When StartNM is called, all network status and extended network status information is cleared.  | Fig. 56 + Fig. 63 + Table 8,9 | NMStatus           |
| 6                                | When StopNM is called, NMon/NMoff information of network status is set.   | Fig. 56 + Table 8             | NMStatus           |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.

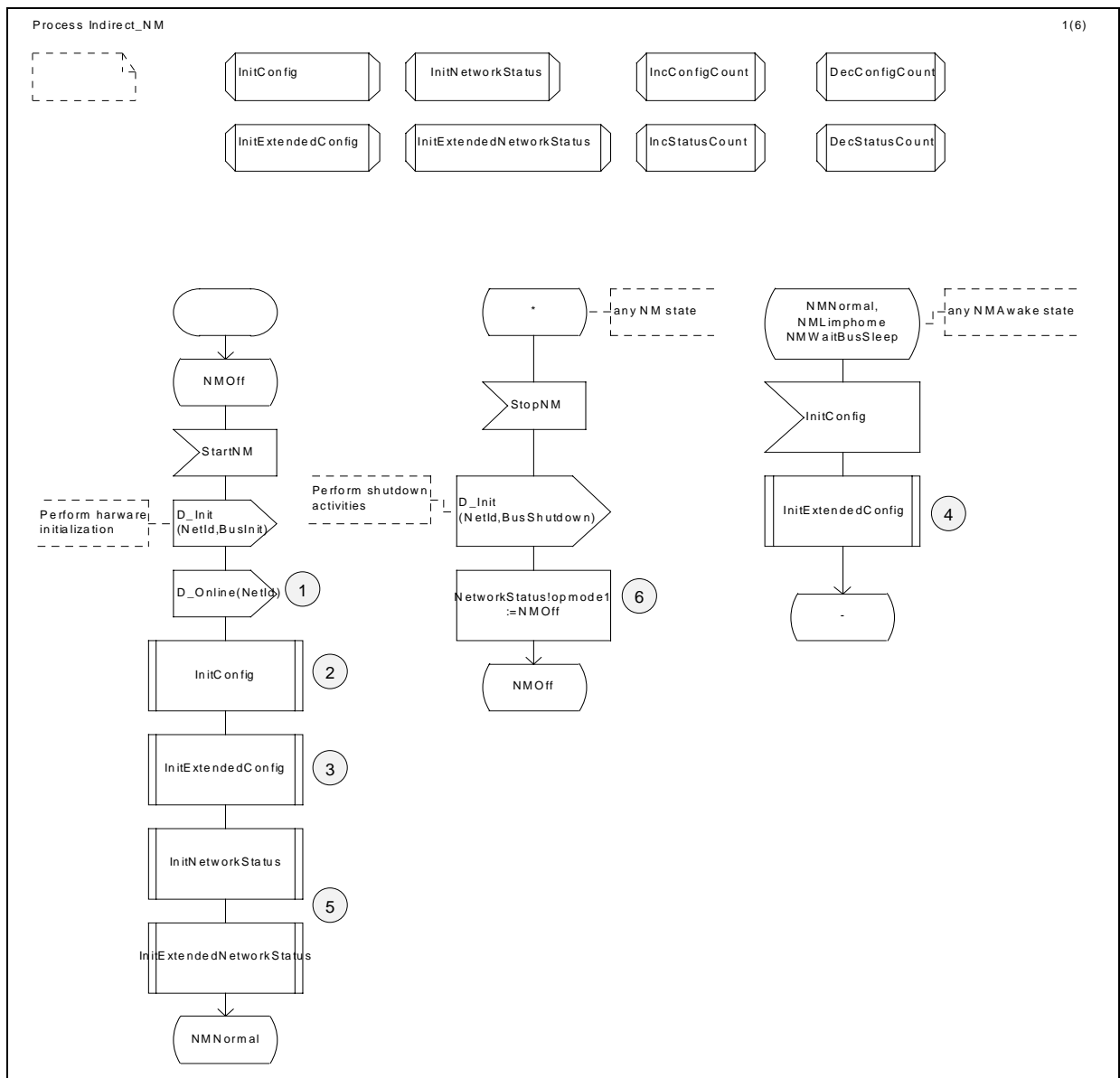


Figure 8 Test coverage of the services StartNM, StopNM and InitConfig

### 3.2.3.2. NMNormal

| Nr                               | Assertion  | Paragraph in spec. | Affected variants |
|----------------------------------|--|--------------------|-------------------|
| Network configuration management |  |                    |                   |
| 1                                | In NMNormal state, a supervised node is declared present and static present when the monitored message from that node is received.                             | Fig. 57            | Core              |
| 2                                | In NMNormal state, a supervised node is declared absent if the monitored message from that node has not been received after the dedicated monitoring time-out. | Fig. 57            | Core              |

|                           |   |                                   |          |
|---------------------------|---|-----------------------------------|----------|
| 3                         | In NMNormal state, a supervised node is declared static absent if the monitored message from that node has not been received after the dedicated monitoring time-out and if the associated counter equals the threshold.                                | Fig. 57                           | Core     |
| 4                         | In NMNormal state, a supervised node remains static present if the monitored message from that node has not been received after the dedicated monitoring time-out and if the associated counter is below the threshold.                                 | Fig. 57                           | Core     |
| 5                         | In NMNormal state, own node is declared not mute and static not mute when the monitored application message is transmitted.   | Fig. 57                           | Core     |
| 6                         | In NMNormal state, own node is declared mute if the monitored application message has not been transmitted after the dedicated monitoring time-out.   | Fig. 57                           | Core     |
| 7                         | In NMNormal state, a supervised node is declared static mute if the monitored message from that node has not been transmitted after the dedicated monitoring time-out and if the associated counter equals the threshold.                               | Fig. 57                           | Core     |
| 8                         | In NMNormal state, a supervised node remains static not mute if the monitored message from that node has not been received after the dedicated monitoring time-out and if the associated counter is below the threshold.                                | Fig. 57                           | Core     |
| Network status management |   |                                   |          |
| 9                         | When the monitored message from a supervised node is received in NMNormal state, “Error, bus blocked” and “NMLimphome” information of network status is cleared, as well as “Error, communication not possible” information of extended network status. | Fig. 57 +<br>Table 8 +<br>Table 9 | NMStatus |
| 10                        | When the monitored message from own node is transmitted in NMNormal state, “Error, bus blocked” and “NMLimphome” information of network status is cleared, as well as “Error, communication not possible” information of extended network status.       | Fig. 57 +<br>Table 8 +<br>Table 9 | NMStatus |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.

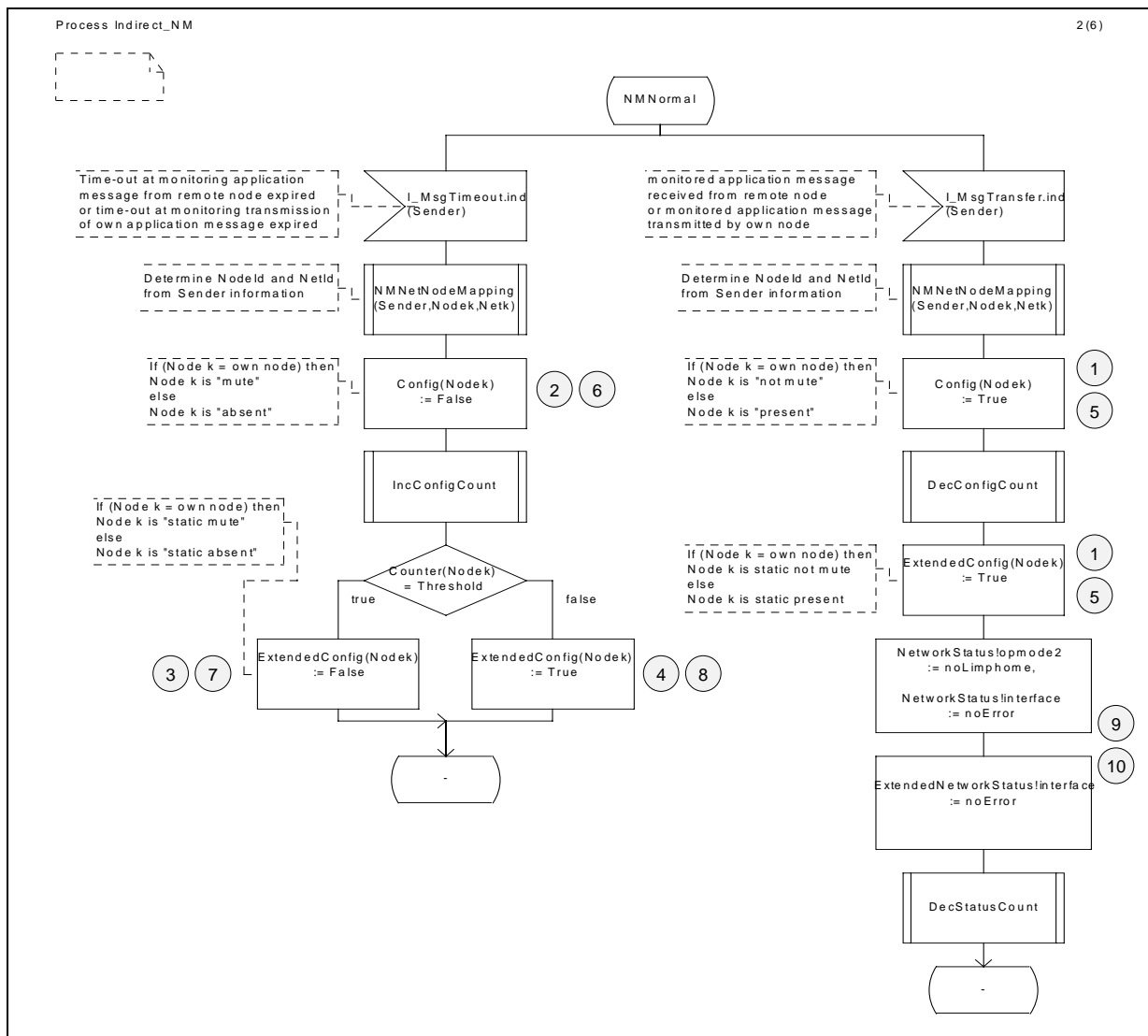


Figure 9 Test coverage of the NMNormal state

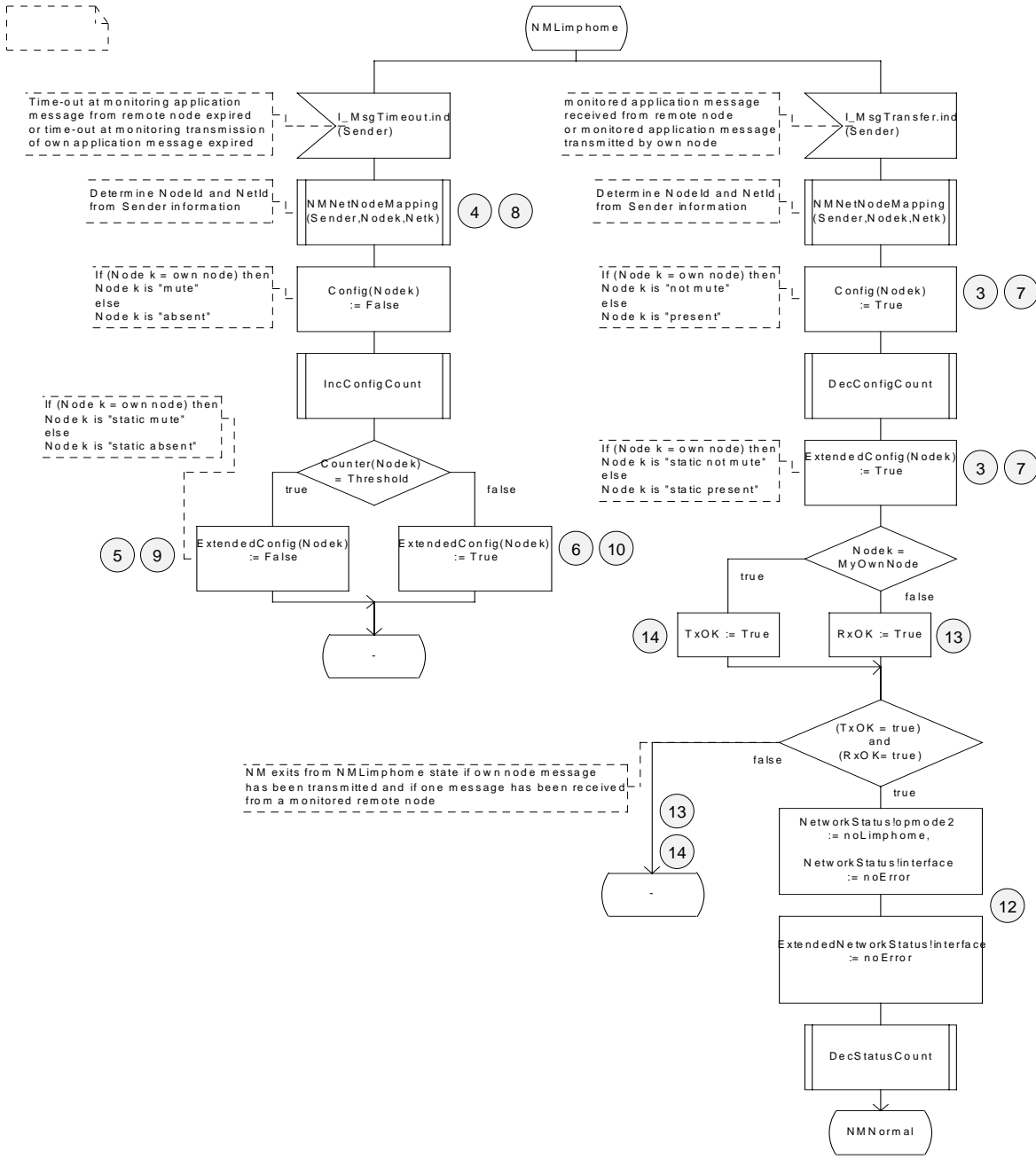
### 3.2.3.3. NMLimphome

| Nr                               | Assertion   | Paragraph in spec. | Affected variants |
|----------------------------------|---|--------------------|-------------------|
| User's communication management  |   |                    |                   |
| 1                                | If a fatal bus error is detected in NMNormal, NMLimphome state, the NM enters the NMLimphome state and application communication is disabled. | Fig. 59            | Core              |
| 2                                | In NMLimphome state, application communication is enabled after $T_{Error}$   | Fig. 59            | Core              |
| Network configuration management |   |                    |                   |
| 3                                | In NMLimphome state, a supervised node is declared present and static present when the monitored message from that node is received.          | Fig. 58            | Core              |

|                           |   |                                   |          |
|---------------------------|---|-----------------------------------|----------|
| 4                         | In NMLimphome state, a supervised node is declared absent if the monitored message from that node has not been received after the dedicated monitoring time-out.  | Fig. 58                           | Core     |
| 5                         | In NMLimphome state, a supervised node is declared static absent if the monitored message from that node has not been received after the dedicated monitoring time-out and if the associated counter equals the threshold.  | Fig. 58                           | Core     |
| 6                         | In NMLimphome state, a supervised node remains static present if the monitored message from that node has not been received after the dedicated monitoring time-out and if the associated counter is below the threshold.   | Fig. 58                           | Core     |
| 7                         | In NMLimphome state, own node is declared not mute and static not mute when the monitored application message is transmitted.   | Fig. 58                           | Core     |
| 8                         | In NMLimphome state, own node is declared mute if the monitored application message has not been transmitted after the dedicated monitoring time-out.   | Fig. 58                           | Core     |
| 9                         | In NMLimphome state, a supervised node is declared static mute if the monitored message from that node has not been transmitted after the dedicated monitoring time-out and if the associated counter equals the threshold.   | Fig. 58                           | Core     |
| 10                        | In NMLimphome state, a supervised node remains static not mute if the monitored message from that node has not been received after the dedicated monitoring time-out and if the associated counter is below the threshold.  | Fig. 58                           | Core     |
| Network status management |   |                                   |          |
| 11                        | On transition to NMLimphome, “Error, bus blocked” and “NMLimphome” information of network status is set.  | Fig. 59 +<br>Table 8              | NMStatus |
| 12                        | In NMLimphome state, if a monitored message from a remote node is received and a monitored message from own node is transmitted, the NM enters the NMNormal state. “Error, bus blocked” and “NMLimphome” information of network status is cleared, as well as “Error, communication not possible” information of extended network status. | Fig. 58 +<br>Table 8 +<br>Table 9 | NMStatus |

|    |   |                                   |                        |
|----|---|-----------------------------------|------------------------|
| 13 | In NMLimphome state, if a monitored message from a remote node is received but no monitored message from own node is transmitted, “Error, bus blocked” and “NMLimphome” information of network status remains set, as well as “Error, communication not possible” information of extended network status. | Fig. 58 +<br>Table 8 +<br>Table 9 | NMStatus               |
| 14 | In NMLimphome state, if a monitored message from own node is transmitted but no monitored message from a remote node is received, “Error, bus blocked” and “NMLimphome” information of network status remain set, as well as “Error, communication not possible” information of extended network status.  | Fig. 58 +<br>Table 8 +<br>Table 9 | NMStatus               |
| 15 | When a fatal bus error is detected, the NM is not in NMWaitBusSleep state and the status counter equals the threshold, “Error, communication not possible” information of extended network status is set.   | Fig. 59 +<br>Table 9              | NMStatus               |
| 16 | When a fatal bus error is detected, the NM is not in NMWaitBusSleep state and the status counter is below the threshold, “Error, communication not possible” information of extended network status remains cleared.  | Fig. 59 +<br>Table 9              | NMStatus               |
| 17 | When a fatal bus error is detected and the NM is not in NMWaitBusSleep state and the status counter is below the threshold, “Error, communication not possible” information of extended network status not changed.   | Fig. 59                           | NMStatus +<br>BusSleep |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.



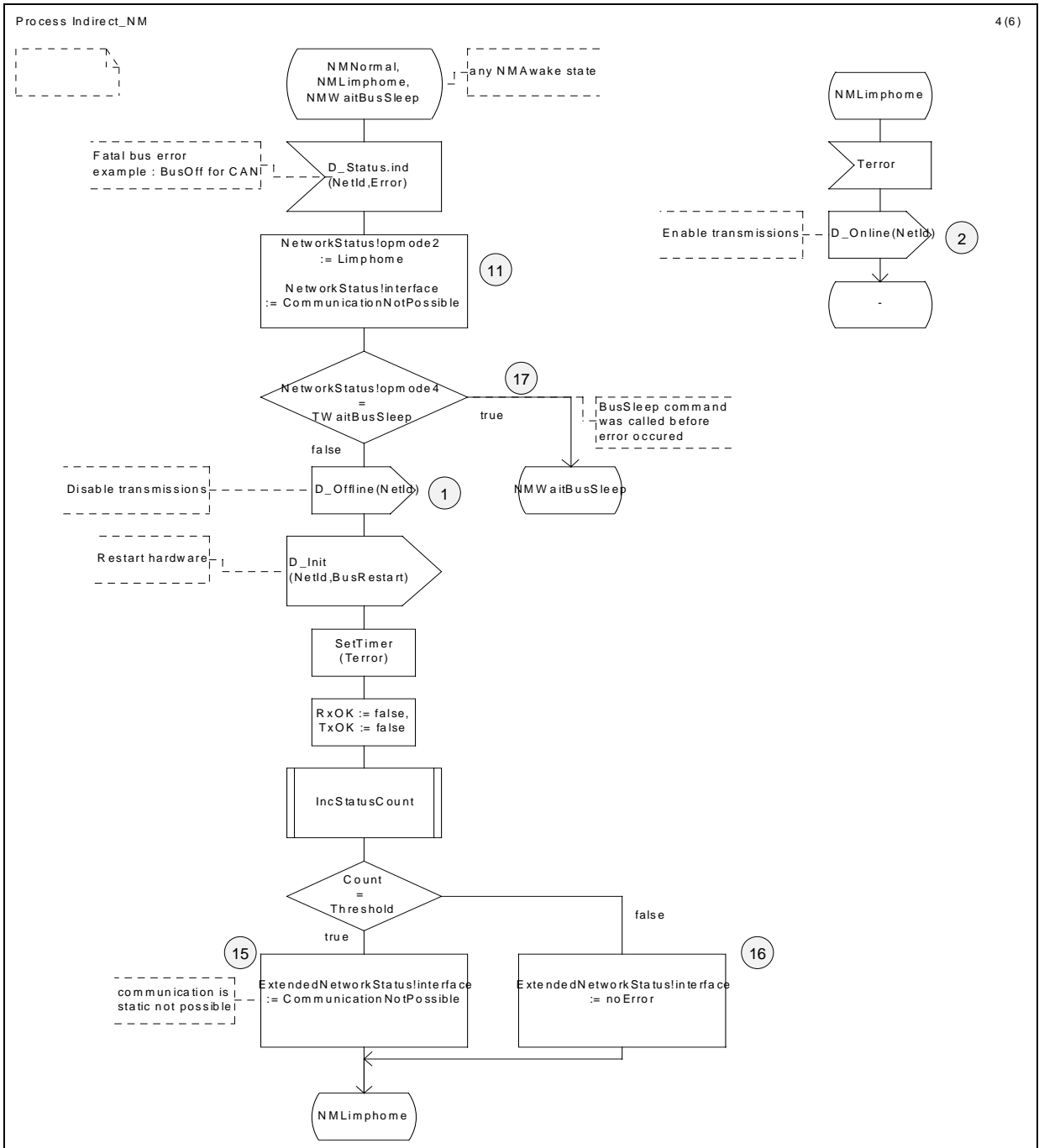


Figure 10 Test coverage of the NMLimpHome state

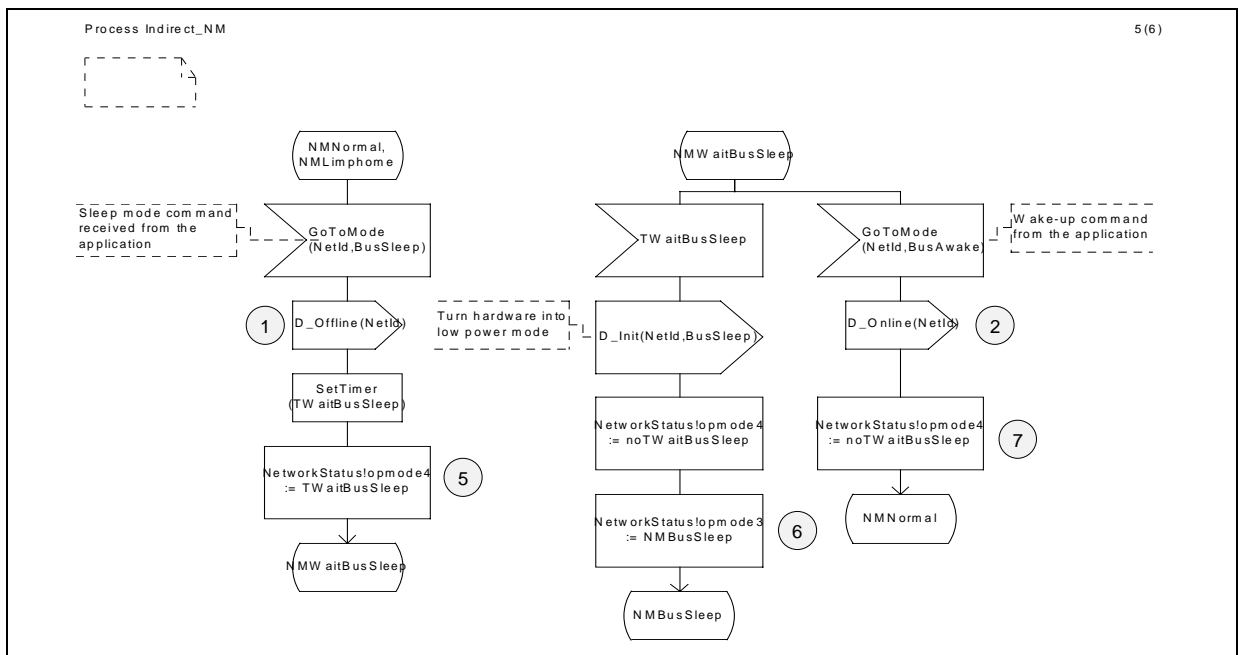
### 3.2.3.4. NMBusSleep

| Nr                              | Assertion  | Paragraph in spec. | Affected variants |
|---------------------------------|--|--------------------|-------------------|
| User's communication management |  |                    |                   |
| 1                               | If GotoMode(BusSleep) is called in NMNormal or NMLimpHome state, the NM enters the NMWaitBusSleep state and application communication is disabled. | Fig. 60            | BusSleep          |



|                           |  |                   |                       |
|---------------------------|--|-------------------|-----------------------|
| 2                         | If GotoMode(Awake) is called in NMWaitBusSleep state, it enters the Normal state and application communication is enabled.                 | Fig. 60           | BusSleep              |
| 3                         | If GotoMode(Awake) is called in NMBusSleep state, the NM enters the Normal state and application communication is enabled.                 | Fig. 61           | BusSleep              |
| 4                         | If a wake-up signal is received from the bus in NMBusSleep state, the NM enters the Normal state and application communication is enabled. | Fig. 61           | BusSleep              |
| Network status management |  |                   |                       |
| 5                         | On transition to NMWaitBusSleep, "TWaitBusSleep" information of network status is set.   | Fig. 60 + Table 8 | BusSleep + NMStatus + |
| 6                         | If TWaitBusSleep timer expires, "TWaitBusSleep" information of network status is cleared and "NMBusSleep" information is set.              | Fig. 60 + Table 8 | BusSleep + NMStatus + |
| 7                         | On transition from NMWaitBusSleep to NMNormal, "TWaitBusSleep" information of network status is cleared.                                   | Fig. 60 + Table 8 | BusSleep + NMStatus + |
| 8                         | On transition from NMBusSleep to NMNormal, "NMBusSleep" information of network status is cleared.  | Fig. 61 + Table 8 | BusSleep + NMStatus + |

Coverage of indirect NM specification by the test purposes is shown in the SDL diagrams below. Circled numbers refer to test numbers in the table above.



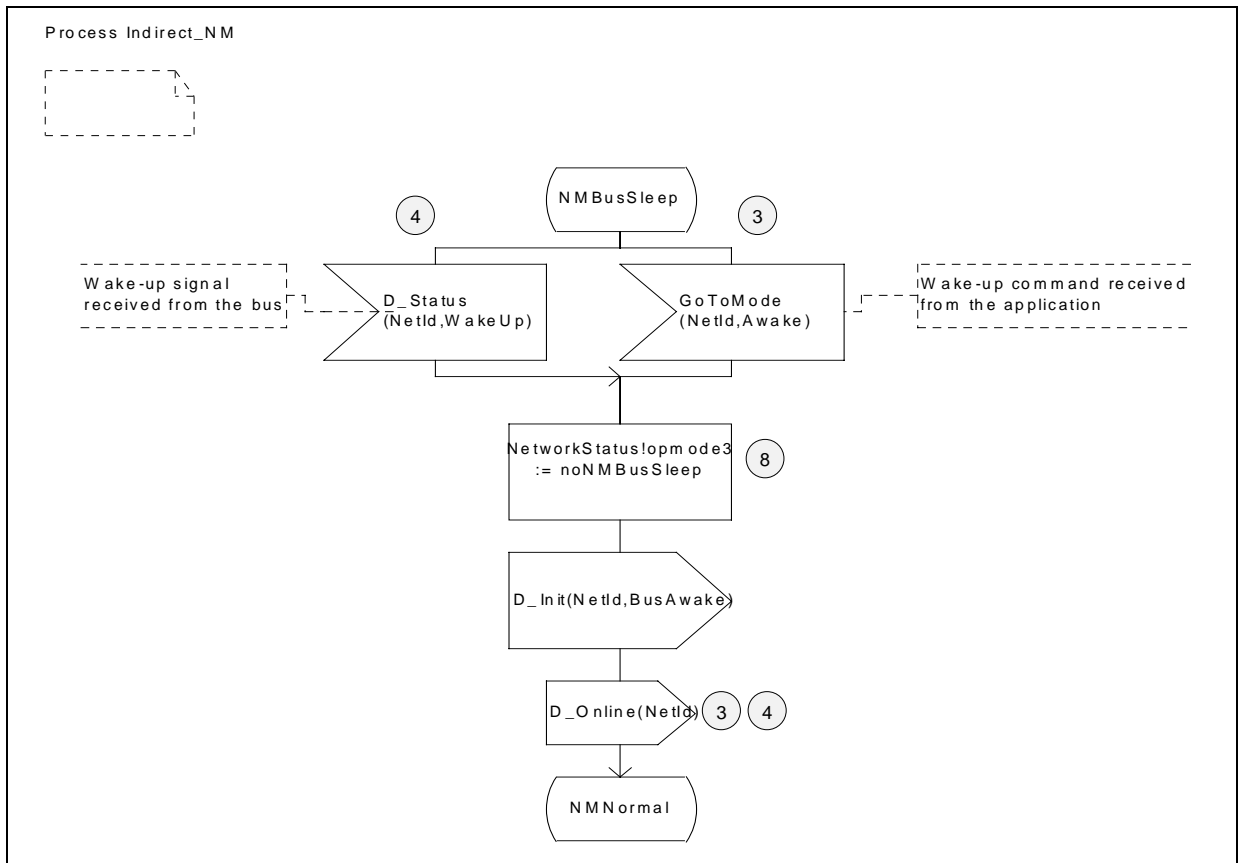


Figure 11 Test coverage of WaitBusSleep and BusSleep states