EPSG Draft Standard 310

Ethernet POWERLINK

Conformance Test Specification

Version 1.0.10

(B&R Industrial Automation GmbH)

2023

B&R Industrial Automation GmbH

POWERLINK-Office
B&R Straße 1
5142 Eggelsberg
Austria

powerlink.office@br-automation.com
www.br-automation.com/en/technologies/powerlink/

The EPSG Draft Standard 310 “Ethernet Powerlink Conformance Test Specification” has been provided by Ethernet POWERLINK Standardisation Group (hereinafter referred to as “EPSG”). As a consequence of the EPSG being dissolved from March 31st, 2023, B&R Industrial Automation GmbH will – as the formal successor of EPSG regarding the rights and content – make the Ethernet Powerlink Conformance Test Specification available as open source on it’s own website subject to the conditions mentioned in the disclaimer under clause Pre. 1 of this document. B&R Industrial Automation GmbH especially disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other EPSG Standard document.

Pre. 1 Disclaimer

Use of this EPSG Standard is wholly voluntary. The EPSG disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other EPSG Standard document.
The EPSG does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. EPSG Standards documents are supplied “AS IS”.
The existence of an EPSG Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the EPSG Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Users are cautioned to check to determine that they have the latest edition of any EPSG Standard.
In publishing and making this document available, the EPSG is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the EPSG undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other EPSG Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.
Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the EPSG, the group will initiate action to prepare appropriate responses. Since EPSG Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the EPSG and its members are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.
Comments for revision of EPSG Standards are welcome from any interested party, regardless of membership affiliation with the EPSG. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be sent to the address given on the page before.

Pre. 1.1 Patent notice

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. B&R shall not be responsible for identifying patents for which a license may be required by an EPSG standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Pre. 2 History

Vers.	Date	Author / Filename		short description
0.9.0	2009-12-07			Final draft version
1.0.0	2012-03-12	David Puffer	B&R	Created
		SRS_POWERLINKConformance_Spec_V1.0.0.doc		
1.0.1	2012-03-20	David Puffer	B&R	Adapted Testcase_3.2.9.T1
		SRS_POWERLINKConformance_Spec_V1.0.1.doc		
1.0.2	2012-05-07	David Puffer	B&R	Inserted/Modified NMT-State commands
		SRS_POWERLINKConformance_Spec_V1.0.2.doc		
1.0.3	2012-07-27	David Puffer	B&R	Added testcases for SDO NIL- and unknown commands
		SRS_POWERLINKConformance_Spec_V1.0.3.doc		
1.0.4	2012-11-13	David Puffer	B&R	Modified existing advanced testcases and added new ones.
		SRS_POWERLINKConformance_Spec_V1.0.4.doc		
1.0.5	2013-02-29	David Puffer	B&R	Added advanced testcases (extension compatibility)
		EPSG WDP 310 V-1-0-5.doc		
1.0.6	2014-03-13	David Puffer	B&R	Removed incomplete test chapters, added new chapter “Configuration Tests”, cosmetic changes. Change document status to DSP.
		EPSG DSP 310 V-1-0-6.doc		
1.0.7	2014-05-19	David Puffer	B&R	Added additional test cases. Cosmetic changes.
		EPSG DSP 310 V-1-0-7.doc		
1.0.8	2015-05-08	David Puffer	B&R	Turn into draft standard.
		EPSG DS 310 V-1-0-8.doc		
1.0.9	2023-06-05	Stephan Kirchmayer	B&R	© B&R due to dissolution of EPSG
		EPSG 310 V-1-0-9 DS.doc		
1.0.10	2024-06-04	Wolfgang Burreiner	B&R	Added additional information about Testcase 3.2.6.T8_1-3 (SDO NIL-command)
		EPSG 310 V-1-0-10 DS.doc		

Pre. 3 Content

Pre. 4 Tables

Pre. 5 Figures

Pre. 6 Definitions and Abbreviations

Pre. 6.1 Definitions

Pre. 6.2 Abbreviations

DUT	Device under test
MN	Managing node
CN	Controlled node

Pre. 7 References

[1] EPSG Draft Standard 301 (EPSG DS 301), Ethernet POWERLINK, Communication Profile Specification

[2] EPSG Draft Standard 311 (EPSG DS 311) Ethernet POWERLINK, XML Device Description Specification

[3] EPSG Draft Standard 302-C (EPSG DS 302-C) Ethernet POWERLINK, Part C: PollResponse Chaining

[4] EPSG Draft Standard 302-B (EPSG DS 302-B) Ethernet POWERLINK, Part B: Multiple-ASnd

1 Introduction

1.1 Scope

This document describes the

Conformance Test Specification

for the

EPSG Draft Standard 301

titled

Ethernet POWERLINK Communication Profile Specification

Version 1.2.0

These tests test all communication specific issues without testing the functionality of the DUT.

The structure of the document reflects the sequence of tests to be performed on the DUT.

1.2 General

Test cases testing the correct functioning of extensions to EPSG DS 301 are specified in extension documents to this document.

The POWERLINK Conformance Test shall be the authoritative source for certification-relevant testcases. The release notes of each version shall state which test cases specified in this document, are implemented in the respective version of the POWERLINK Conformance Test.

2 Test Setup

This chapter specifies the requirements on the manufacturer to accomplish the certification.

2.1 Test Prerequisites

A potential Device under Test shall be loaded with an application that runs as soon as the device is powered-on. This application may be trivial, but shall be fully described both in documentation and, along with the communication profile, the XML Device Description file (XDD).

If the XDD is not standard-conform, then the test cannot proceed.

2.2 Test Sequence

The tests described within this document, are divided into 3 major blocks:

1. XDD-Tests: Before any other tests can be conducted on the DUT, its XDD-File shall pass all tests concerning XML-Wellformedness, XML-Validity and semantic correctness.
2. Bootup- and Basic-Tests: This block covers basic tests concerning the bootup-phase and operation of the DUT. Main sections are as follows:
   - Device-identification: Checks whether the DUT really identifies itself with the same values (vendor-id, device-id, feature-flags etc.) that are defined in the XDD.
   - NMT state-machine: Bootup-process of the DUT, its NMT-States and its reaction to NMT-Commands.
   - SDO-Stack: Checks whether the SDO-Stack works correctly, i.e. SDO operations via ASnd, UDP/IP, PDO.
   - NMT-State commands: Check the correct functioning of NMT-State commands.
   - Object-Dictionary: Check object-dictionary for correct values and behavior.
3. Advanced tests: This block covers tests for testing correct error-handling capabilities of the DUT, as well as tests concerning cyclic data transfer (PDO).

2.3 Test Sub-Sequences and Test Failure

In order to enable the applicant to narrow down a possible Conformance Test failure, the tests have

been divided into sub-sequences labelled as follows:

example: TEST 3.2.9.T1.1

The numbers before the letter “T” represent the chapter under which the test can be found. The first number after the letter “T” represents a running number for the test within the chapter, a test can be divided into sub-tests by appending “.” and a subsequent running number.

Tests are written as fall-through and end on <PASSED> , <FAILED> or <NOT_SUPPORTED>. There can be multiple <FAILED>’s within a test, each <FAILED> is given a reference label as follows:

example: FAIL 3.2.9.T1.1.F1

The letters up to “F” represent the identifier for the test (see above), followed by the identifier for the ocurred <FAILED>. If a <FAILED> clause is deleted from the test it is also deleted from the Conformance Test Software. If a <FAILED> clause is added in a new Conformance Test Version it receives the next highest running number within the test. This labelling shall also be reflected in the Conformance Test Software logs and screen output.

3 Test Description and Specification

3.1 XDD/XDC-Check

3.1.1 XDD/XDC Validation

3.1.1.1 Description

In order to carry out tests, the conformance-test tool shall provide means to import and check XDD/XDC files for XML-validity.

3.1.1.2 Action/Response Sequences

User imports an XDD/XDC-file, application performs tests on validity and informs the user about occurred errors or successful import.

3.1.1.3 Functional Requirements

REQ-1: XDD-Check shall provide a menu-entry and a button to import XDD/XDC files.

REQ-2: XDD-Check shall provide a file-selection dialog upon starting the import. Invalid input filenames shall trigger an error-message and abort the operation.

REQ-3: XDD-Check shall provide a possibility to configure the path to the XML-Scheme file used for validation of the input-file.
This configured path shall be stored in the application-configuration file.

REQ-4: XDD-Check shall provide functionality to test the input file for:

- Whether it is well-formed XML.
- Valid according to the configured XML-Scheme.

REQ-5: XDD-Check shall return successful/failed import/validation status of the input-file.

3.1.2 XDD/XDC OD-Validation

3.1.2.1 Description

The OD of the POWERLINK-Device shall contain the objects defined as being mandatory or conditional by the POWERLINK-Specification. Furthermore all object-attributes in die OD shall have values within the range defined in the POWERLINK-Specification.

Subobjects need to be checked for their existence, respectively their attribute-values as well.

Pre-Condition: Successful termination of test 3.1.1.

3.1.2.2 Action/Response Sequences

The application processes a list of all objects (including their attributes) defined by the POWERLINK-Specification (range 1000-1FFF) and tests for their existence in the XDD/XDC. If an object is defined as being mandatory by the POWERLINK-Specification and does not exist in the XDD-file, validation fails. If it does exist, its attributes and subobjects (and their attributes) are checked for conformance with the defined ranges in the specification.

Attributes of an Object to be checked are: see 7.5.2.4.1 of the XML Device Description.

Attributes of a SubObject to be checked are: see 7.5.2.4.1.1 of the XML Device Description.

If an object is defined as being optional and it does not exist, it is ignored, otherwise its attributes are checked (see above).

If an object is defined as being conditional (existence depending on another object) and it is supposed to exist but doesn't, validation fails. If it does exist, its attributes are checked (see above).

3.1.2.3 Functional Requirements

REQ-1: XDD-Check shall parse a list of objects which are defined in the POWERLINK-Specification.

REQ-2: XDD-Check shall test each object-id according to section 3.1.2.2 and act accordingly.

3.1.3 XDD/XDC Semantics

3.1.3.1 Description

The XDD/XDC-file shall be tested for semantic validity. Performed validations shall be expressed in XML, using the Schematron Validation Language. Ruleset files are delivered together with the POWERLINK Conformance Test.

Pre-Condition: Successful termination of test 3.1.2.

3.1.3.2 Action/Response Sequences

Application tests XDD/XDC-file for semantic validity using an XML-defined rule-set.

3.1.3.3 Functional Requirements

REQ-1: XDD-Check shall provide a possibility to configure the path to the XML-Scheme file used for the XML rule-set file.

This configured path shall be stored in the application-configuration file.

REQ-2: XDD-Check shall provide a possibility to configure the path to an XML rule-set used for testing the semantic correctness of the XDD/XDC input-file.

This configured path shall be stored in the application-configuration file.

REQ-3: XDD-Check shall validate a rule-set defined in an XML-file (see REQ-2) according to the defined XML-Scheme of REQ-1. An error-message shall be displayed if the file is not well-formed or not valid.

REQ-4: XDD-Check shall parse the rule-set defined in REQ-2 and report an error-message if the POWERLINK-Device's XDD/XDC violates any defined rule.

3.2 Bootup- and Basic-Tests

Tests contained in this chapter, test the ability of a POWERLINK CN to boot up correctly into the OPERATIONAL-State. Furthermore it is tested, that while running in OPERATIONAL, the device-application is not capable of affecting the communication state negatively (i.e. causing the CN to leave OPERATIONAL). Additional tests include support for certain NMT-Commands.

What is explicitly excluded, are timing-related tests.

3.2.1 CS_PRE_OPERATIONAL_1

3.2.1.1 Device-Identity

3.2.1.1.1 Description

During the state PRE_OPERATIONAL_1 the POWERLINK-MN starts to query the configured CN's with SoA IdentRequest-Frames. The IdentResponse-Frames of the CN shall be checked for conformance with the specification.

3.2.1.1.2 Action/Response Sequences

Parameter:

t_1 = D_NMT_BootTimeNotActive

t_2 = C_NMT_STATE_TOLERANCE(5) * cycleTime

t_3 = AsyncSlotTimeout_U32

Pre-Condition:

MN is in state MS_PRE_OPERATIONAL_1

TEST 3.2.1.T1

```
{power up device}(nodeID)
{wait}(t1)
/* device should be in CS_NOT_ACTIVE state now */
{wait}(t2)
/* device should be in CS_PRE_OPERATIONAL_1 state now */
{send SoA IdentRequest}(nodeID)
{wait}(t3)
on no message received
    <FAIL> 3.2.1.T1.F1
on IdentResponse fields not specification-conform
    <FAIL> 3.2.1.T1.F2-F18
```

Post-Condition:

MN and CN are in state PRE_OPERATIONAL_1

3.2.1.1.3 Functional Requirements

3.2.1.T1.F1: IdentResponse-Frame shall be received by the MN

3.2.1.T1.F2: IdentResponse[stat] shall be CS_PRE_OPERATIONAL_1.

3.2.1.T1.F3: IdentResponse[eplv] shall conform to object 1F83 of the XDD's object-dictionary.

3.2.1.T1.F4: IdentResponse[feat] shall be identical to object 1F82 of the XDD's object-dictionary.

3.2.1.T1.F5: IdentResponse[mtu] shall be equal to object 1F98.08 of the XDD's object-dictionary (and between 300-1500).

3.2.1.T1.F6: IdentResponse[pis] shall be: equal to defaultValue of object 1F98.04 of the XDD's object-dictionary if it is given and >= 36; 36 (from specification and default) otherwise.

3.2.1.T1.F7: IdentResponse[pos] shall be: equal to defaultValue of object 1F98.05 of the XDD's object-dictionary if it is given and >= 36; 36 (from specification and default) otherwise.

3.2.1.T1.F8: IdentResponse[rst] shall be: equal to defaultValue of object 1F98.03 of the XDD's object-dictionary if it is given, otherwise, "rst" cannot be validated.

3.2.1.T1.F9: IdentResponse[dt] shall be: equal to defaultValue of object 1000 of the XDD's object-dictionary if it is given, otherwise, "dt" cannot be validated.

3.2.1.T1.F10: IdentResponse[vid] shall be: equal to defaultValue of object 1018.01 of the XDD's object-dictionary if it is given, otherwise, "vid" cannot be validated.

3.2.1.T1.F11: IdentResponse[prdc] shall be: equal to defaultValue of object 1018.02 of the XDD's object-dictionary if it is given, otherwise, "prdc" shall be set to 0.

3.2.1.T1.F12: IdentResponse[rno] shall be: equal to defaultValue of object 1018.03 of the XDD's object-dictionary if it is given, otherwise, "rno" shall be set to 0.

3.2.1.T1.F13: IdentResponse[sno] Check disabled, because serial-number cannot be verified.

3.2.1.T1.F14: IdentResponse[vcd] and IdentResponse[vct] shall be set to 0 by default and before CFM was able to write this information.

3.2.1.T1.F15: IdentResponse[ad] and IdentResponse [at] shall exist.

3.2.1.T1.F16: IdentResponse[ipa] shall be equal to 192.168.100.x where x is the nodeID of the CN.

3.2.1.T1.F17: IdentResponse[snm] shall be 255.255.255.0.

3.2.1.T1.F18: IdentResponse[hn] shall be: equal to defaultValue of object 1F9A of the XDD's object-dictionary if it is given, otherwise, "hn" shall be equal to the default

hostname according to EPSG DS301, which is the nodeId followed by a dash and the vendorId. nodeId and vendorId shall be hex-encoded.

3.2.1.2 Status-Transition

3.2.1.2.1 Description

CN shall change its state from CS_PRE_OPERATIONAL_1 to CS_PRE_OPERATIONAL_2 upon MN changing its state to MS_PRE_OPERATIONAL_2.

3.2.1.2.2 Action/Response Sequences

Parameter:

t_1 = Configurable timeout for CN changing its state from CS_PRE_OPERATIONAL_1 to CS_PRE_OPERATIONAL_2.

t_2 = AsyncSlotTimeout_U32

MAX_COUNT = 5

Pre-Condition:

MN and CN are in state PRE_OPERATIONAL_1

TEST 3.2.1.T2

```
/*MN switches to MS_PRE_OPERATIONAL_2 and CN shall follow*/
{repeat}
    {wait}(t1)
    {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
    {wait}(t2)
{until (message received && reported state == CS_PRE_OPERATIONAL_2) or
MAX_COUNT times}
{if count > 1 && count <= MAX_COUNT}
    <FAIL> 3.2.1.T2.F1 /* device changed state, but not within timeout */
{else if count > MAX_COUNT}
    {if no message received}
        <FAIL> 3.2.1.T2.F2 /* device did not answer at all */
    {else}
        <FAIL> 3.2.1.T2.F3 /* device did not change state */
```

Post-Condition:

MN and CN are in state PRE_OPERATIONAL_2

3.2.1.2.3 Functional Requirements

3.2.1.T2.F1: Device shall change its state from CS_PRE_OPERATIONAL_1 to CS_PRE_OPERATIONAL_2 upon MN switching to MS_PRE_OPERATIONAL_2 within t1.

*3.2.1.T2.F2: Device shall respond to StatusRequests.

*3.2.1.T2.F3: Device shall change its state from CS_PRE_OPERATIONAL_1 to CS_PRE_OPERATIONAL_2 upon MN switching to MS_PRE_OPERATIONAL_2.

3.2.2 CS_PRE_OPERATIONAL_2

3.2.2.1 PRes Validity

3.2.2.1.1 Description

In CS_PRE_OPERATIONAL_2, the CN shall be queried by the MN with Preq-Frames continuously. All PRes-Frames received from the CN shall not have their RD-Flag set.

3.2.2.1.2 Action/Response Sequences

Parameter:

t_1 = PResMaxLatency_U32

t_2 = signal propagation time to CN and back to MN (to be estimated)

Pre-Condition:

MN and CN are in state PRE_OPERATIONAL_2

TEST 3.2.1.T1

TEST 3.2.2.T1

```
on (1F82 bit 0 == 1)
    {send SoC}
    {send PReq}(nodeID)
    {wait}(t1 + t2)
    {receive PRes-Frame}
    on no message received
        <FAIL> 3.2.2.T1.F1
    on (message received && PRes[RD] == 1)
        <FAIL> 3.2.2.T1.F2
on (1F82 bit 0 == 0)
    {send SoC}
    {send PReq}(nodeID)
    {wait}(t1 + t2)
    {receive PRes-Frame}
```

```
on message received
    <FAIL> 3.2.2.T1.F3
```

Post-Condition:

MN and CN are in state PRE_OPERATIONAL_2

3.2.2.1.3 Functional Requirements

3.2.2.T1.F1: CN shall respond with PRes on a received PReq.

3.2.2.T1.F2: RD-flag of PRes-Frames received from the CN shall not be set.

3.2.2.T1.F3: If CN does not support isochronous communication, it shall not answer on PReq-Frames.

3.2.2.2 Status-Transition

3.2.2.2.1 Description

Upon reception of the NMTEnableReadyToOperate command from the MN, CN shall initiate a state-transition to CS_READY_TO_OPERATE. It is not defined how long this may take, the CN changes its state when it is “ready” to do so.

The wait-time for the state-change shall therefore be defined as configurable variable.

3.2.2.2.2 Action/Response Sequences

Parameter:

t_1 = 1s (configurable within application, this is the time the MN waits for the CN to change its state to READY_TO_OPERATE)

t_2 = AsyncSlotTimeout_U32

MAX_COUNT = 5

Pre-Condition:

MN and CN are in state PRE_OPERATIONAL_2

TEST 3.2.2.T2

```
{repeat}
    {send ASnd NMTEnableReadyToOperate}(nodeID)
    {wait}(t1)
    {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
    {wait}(t2)
{until (message received && reported state == CS_READY_TO_OPERATE) or
MAX_COUNT times}
{if count > 1 && count <= MAX_COUNT}
    <FAIL> 3.2.2.T2.F1 /* device changed state, but not within timeout */
{else if count > MAX_COUNT}
    {if no message received}
        <FAIL> 3.2.2.T2.F2 /* device did not answer at all */
    {else}
        <FAIL> 3.2.2.T2.F3 /* device did not change state */
```

Post-Condition:

MN and CN are in state READY_TO_OPERATE

3.2.2.2.3 Functional Requirements

3.2.2.T2.F1: Device shall change its state from CS_PRE_OPERATIONAL_2 to CS_READY_TO_OPERATE, upon MN sending command NMTEnableReadyToOperate within t_1.

3.2.2.T2.F2 Device shall respond to StatusRequests.

3.2.2.T2.F3 Device shall change its state from CS_PRE_OPERATIONAL_2 to CS_READY_TO_OPERATE upon MN sending command NMTEnableReadyToOperate.

3.2.3 CS_READY_TO_OPERATE

3.2.3.1 PRes Validity

3.2.3.1.1 Description

In CS_READY_TO_OPERATE, the CN shall be queried by the MN with Preq-Frames continuously. All PRes-Frames received from the CN shall not have their RD flag set, but correspond to the requirements defined by the PDO-Mapping.

3.2.3.1.2 Action/Response Sequences

Parameter:

t_1 = PResMaxLatency_U32

t_2 = signal propagation time to CN and back to MN (to be estimated)

Pre-Condition:

MN and CN are in state READY_TO_OPERATE

TEST 3.2.1.T1

TEST 3.2.3.T1

```
on (1F82 bit 0 == 1)
    {send SoC}
    {send PReq}(nodeID)
    {wait}(t1 + t2)
    {receive PRes-Frame}
    on no message received
        <FAIL> 3.2.3.T1.F1
```

```
    on (message received && PRes-Frame not specification-conform)
        <FAIL> 3.2.3.T1.F2, F4-F10
on (1F82 bit 0 == 0)
    {send SoC}
    {send PReq}(nodeID)
    {wait}(t1 + t2)
    {receive PRes-Frame}
    on message received
        <FAIL> 3.2.3.T1.F3
```

Post-Condition:

MN and CN are in state READY_TO_OPERATE

3.2.3.1.3 Functional Requirements

3.2.3.T1.F1: Upon MN sending a PReq-Frame, CN shall answer with a PRes-Frame within the timeout.

3.2.3.T1.F2: PRes[RD] shall be equal to 0h

3.2.3.T1.F3: If CN does not support isochronous communication, it shall not answer on PReq-Frames.

3.2.3.T1.F4: PRes[src] shall be equal to nodeID

3.2.3.T1.F5: PRes[dest] shall be equal to FFh

3.2.3.T1.F6: PRes[MS] shall be equal to 0h

3.2.3.T1.F7: PRes[mtyp] shall be equal to 04h

3.2.3.T1.F8: PRes[pdov] shall be equal to PDO_TxCommParam_00h_REC.MappingVersion_U8

3.2.3.T1.F9: PRes[size] shall be >= 0 and <= min(1F98.05, C_DLL_ISOCHR_MAX_PAYL(1490))

3.2.3.T1.F10: PRes[stat] shall be equal to 6Dh (CS_READY_TO_OPERATE)

3.2.3.2 Status-Transition

3.2.3.2.1 Description

CN shall change its status to CS_OPERATIONAL upon reception of the NMTStartNode command from the MN.

3.2.3.2.2 Action/Response Sequences

Parameter:

t_1 = C_NMT_STATE_TOLERANCE(5) * cycleTime

t_2 = AsyncSlotTimeout_U32

MAX_COUNT = 5

Pre-Condition:

MN and CN are in state READY_TO_OPERATE

TEST 3.2.3.T2

```
{repeat}
    {send ASnd NMTStartNode }(nodeID)
    {wait}(t1)
    {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
    {wait}(t2)
{until (message received && reported state == CS_OPERATIONAL) or MAX_COUNT
times}
{if count > 1 && count <= MAX_COUNT}
    <FAIL> 3.2.3.T2.F1 /* device changed state, but not within timeout */
{else if count > MAX_COUNT}
    {if no message received}
        <FAIL> 3.2.3.T2.F2 /* device did not answer at all */
    {else}
        <FAIL> 3.2.3.T2.F3 /* device did not change state */
```

Post-Condition:

MN and CN are in state OPERATIONAL

3.2.3.2.3 Functional Requirements

3.2.3.T2.F1: Device shall change its state from CS_READY_TO_OPERATE to CS_OPERATIONAL within t_1, upon MN sending command NMTStartNode.

3.2.3.T2.F2: Device shall respond to StatusRequests.

3.2.3.T2.F3: Device shall change its state from CS_READY_TO_OPERATE to CS_OPERATIONAL upon MN sending command NMTStartNode.

3.2.4 CS_OPERATIONAL

3.2.4.1 General Behaviour

3.2.4.1.1 Description

In CS_OPERATIONAL PRes-Frames received from the CN can be declared valid (RD flag set) and shall correspond to the requirements defined by the PDO-Mapping.

3.2.4.1.2 Action/Response Sequences

Parameter:

t_1 = PResMaxLatency_U32

t_2 = signal propagation time to CN and back to MN (to be estimated)

t_3 = AsyncSlotTimeout_U32

Pre-Condition:

MN and CN are in state OPERATIONAL

TEST 3.2.1.T1

TEST 3.2.4.T1

```
/* Identical to TEST 3.2.3.T1 */
```

Post-Condition:

MN and CN are in state OPERATIONAL

3.2.4.1.3 Functional Requirements

3.2.4.T1.F1: Upon MN sending a PReq-Frame, CN shall answer with a PRes-Frame within the timeout.

3.2.4.T1.F2: PRes[src] shall be equal to nodeID

3.2.4.T1.F3: If CN does not support isochronous communication, it shall not answer on PReq-Frames.

3.2.4.T1.F4: PRes[dest] shall be equal to FFh

3.2.4.T1.F5: PRes[MS] shall be equal to 0h

3.2.4.T1.F6: PRes[mtyp] shall be equal to 04h

3.2.4.T1.F7: PRes[pdov] shall be equal to PDO_TxCommParam_00h_REC.MappingVersion_U8

3.2.4.T1.F8: PRes[size] shall be >= 0 and <= min(1F98.05, C_DLL_ISOCHR_MAX_PAYL(1490))

3.2.4.T1.F9: PRes[stat] shall be equal to FDh (CS_OPERATIONAL)

3.2.4.2 Status-Transitions

3.2.4.2.1 Description

With CN being in CS_OPERATIONAL, it shall be verified that it enters state CS_PRE_OPERATIONAL_2 on receiving NMT-Command NMTEnterPreOperational2 and CS_STOPPED on receiving NMT-Command NMTStopNode.

3.2.4.2.2 Action/Response Sequences

Parameter:

t_1 = AsyncSlotTimeout_U32

t_2 = C_NMT_STATE_TOLERANCE(5) * cycleTime

MAX_COUNT = 5

Pre-Condition:

MN and CN are in state OPERATIONAL

TEST 3.2.4.T2 moved to TEST 3.2.7.T5_1

TEST 3.2.4.T3

```
{repeat}
    {send ASnd NMTStopNode}(nodeID)
    {wait}(t2)
    {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
    {wait}(t1)
{until (message received && reported state == CS_STOPPED) or MAX_COUNT
times}
{if count > 1 && count <= MAX_COUNT}
    <FAIL> 3.2.4.T3.F1 /* device changed state, but not within timeout */
{else if count > MAX_COUNT}
    {if no message received}
        <FAIL> 3.2.4.T3.F2 /* device did not answer at all */
    {else}
        <FAIL> 3.2.4.T3.F3 /* device did not change state */
```

Post-Condition:

CN is in state CS_STOPPED

3.2.4.2.3 Functional Requirements

3.2.4.T3.F1: Device shall change its state from CS_OPERATIONAL to CS_STOPPED within t_2, upon MN sending command NMTStopNode.

3.2.4.T3.F2: Device shall respond to StatusRequests sent after switching the CN to CS_STOPPED.

3.2.4.T3.F3: Device shall change its state from CS_OPERATIONAL to CS_STOPPED upon MN sending command NMTStopNode.

3.2.5 CS_STOPPED

3.2.5.1 General Behaviour

3.2.5.1.1 Description

In CS_STOPPED, the CN shall not respond to PReq-Frames sent by the MN, but it shall respond to StatusRequests.

3.2.5.1.2 Action/Response Sequences

Parameter:

t_1 = PResMaxLatency_U32

t_2 = signal propagation time to CN and back to MN (to be estimated)

t_3 = AsyncSlotTimeout_U32

Pre-Condition:

MN is in state MS_READY_TO_OPERATE

CN is in state CS_STOPPED

TEST 3.2.1.T1

TEST 3.2.4.T3

TEST 3.2.5.T1

```
{send SoC}
{send PReq}(nodeID)
{wait}(t1 + t2)
on message received and 1F82 bit 0 == 1 // isochronous node
    <FAIL> 3.2.5.T1.F1
on message received and 1F82 bit 0 == 0 // async-only node
    <FAIL> 3.2.5.T1.F2
{send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
{wait}(t3)
on no message received
    <FAIL> 3.2.5.T1.F3
on (message received && reported state != CS_STOPPED)
    <FAIL> 3.2.5.T1.F4
```

Post-Condition:

CN is in state CS_STOPPED

3.2.5.1.3 Functional Requirements

3.2.5.T1.F1: If CN supports isochr. communication, it shall not respond to PReq-Frames while being in state CS_STOPPED.

3.2.5.T1.F2: If CN is async-only, it shall not respond to PReq-Frames.

3.2.5.T1.F3: CN shall respond to StatusRequests sent by MN.

3.2.5.T1.F4: CN shall report state CS_STOPPED.

3.2.5.2 Status-Transitions

3.2.5.2.1 Description

In CS_STOPPED, the CN shall change its state to CS_PRE_OPERATIONAL_2 upon receiving NMT-Command NMTEnterPreOperational2.

3.2.5.2.2 Action/Response Sequences

Parameter:

t_1 = C_NMT_STATE_TOLERANCE(5) * cycleTime

t_2 = AsyncSlotTimeout_U32

Pre-Condition:

MN is in state CS_READY_TO_OPERATE

CN is in state CS_STOPPED

TEST 3.2.5.T2

```
{repeat}
    {send ASnd NMTEnterPreOperational2}(nodeID)
    {wait}(t1)
    {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
    {wait}(t2)
{until (message received && reported state == CS_PRE_OPERATIONAL_2) or
MAX_COUNT times}
{if count > 1 && count <= MAX_COUNT}
    <FAIL> 3.2.5.T2.F1 /* device changed state, but not within timeout */
{else if count > MAX_COUNT}
    {if no message received}
        <FAIL> 3.2.5.T2.F2 /* device did not answer at all */
    {else}
        <FAIL> 3.2.5.T2.F3 /* device did not change state */
```

Post-Condition:

CN is in state CS_PRE_OPERATIONAL_2

3.2.5.2.3 Functional Requirements

3.2.5.T2.F1: Device shall change its state from CS_STOPPED to CS_PRE_OPERATIONAL_2 within t_1, upon MN sending command NMTEnterPreOperational2.

3.2.5.T2.F2: Device shall respond to StatusRequests sent after switching the CN to CS_PRE_OPERATIONAL_2.

3.2.5.T2.F3: Device shall change its state from CS_STOPPED to CS_PRE_OPERATIONAL_2 upon MN sending command NMTEnterPreOperational2.

3.2.6 SDO-Tests

3.2.6.1 General SDO-Tests

3.2.6.1.1 Description

Tests described in this chapter are general in nature. Following chapters will define the basic and extended SDO-Tests being referenced in this chapter.

If there are tests specific to SDO by ASnd, SDO by UDP/IP or SDO by PDO, they are to be defined in this chapter, since basic and extended tests will be executed for all three SDO-Transfer modes.

3.2.6.1.2 Action/Response Sequences

Parameter:

-

Pre-Condition:

Device passed 3.2.1

CN is in CS_PRE_OPERATIONAL_1

MN is in MS_PRE_OPERATIONAL_1

```
/* Run tests for SDO by ASnd */
/* Bring CN into CS_PRE_OPERATIONAL_1 */
on (SDO by ASnd)
    {run Basic SDO-Tests}
    {run Extended SDO-Tests}

/* Run tests for SDO by UDP/IP */
/* Bring CN into CS_PRE_OPERATIONAL_1 */
on (SDO by UDP/IP)
    {run Basic SDO-Tests}
    {run Extended SDO-Tests}

/* Run tests for SDO by PDO */
/* Bring CN into CS_OPERATIONAL */
on (SDO by PDO)
    {run Basic SDO-Tests}
    {run Extended SDO-Tests}
```

Post-Condition:

CN is in CS_PRE_OPERATIONAL_1

MN is in MS_PRE_OPERATIONAL_1

3.2.6.1.3 Functional Requirements

If device supports SDO by ASnd it shall pass basic and extended tests.

If device supports SDO by UDP/IP it shall pass basic and extended tests.

If device supports SDO by PDO it shall pass basic and extended tests.

3.2.6.2 Basic SDO-Tests

3.2.6.2.1 Description

SDO-Tests in this chapter shall test basic functionality of all available SDO-Transfer modes. These are as following:

- SDO-Commands Read-By-Index and Write-By-Index shall be successful
- Reading/Writing of a non-existent index or subIndex shall result in corresponding SDO-Abort codes
- Check reaction of device upon receiving the NIL-Command and a non-existing command.

3.2.6.2.2 Action/Response Sequences

Parameter:

C_1 = NIL command

C_2 = Non-existing command

O_1 = Writeable and readable object

O_2 = Non-existent index

O_3 = Non-existent subIndex

O_4 = Read-only index

O_5 = Write-only index

Pre-Condition:

T2_1-3 depend on T1

Tx_1 depend on T2_1

Tx_2 depend on T2_2

Tx_3 depend on T2_3

TEST 3.2.6.T1

```
/* Test general SDO-Capabilities */
{read XDD device description entry} ("SDOServer")
on (SDOServer == false)
    <FAIL> 3.2.6.T1.F1
```

TEST 3.2.6.T2_1-3

```
/* Test whether CN supports requested SDO-Transfertype */
{read Ident-Response feature-flags}
on (SDO by ASnd == false)
    <FAIL> 3.2.6.T2_1.F1
on (SDO by UDP/IP == false)
    <NOT_SUPPORTED> 3.2.6.T2_1.F1
on (SDO by PDO == false)
    <NOT_SUPPORTED> 3.2.6.T2_3.F1
```

TEST 3.2.6.T3_1-3

```
/* Test SDO Read-By-Index */
{SDO Read-By-Index}(O1)
on no message received
    <FAIL> 3.2.6.T3_1-3.F1
on SDO-Abort
    <FAIL> 3.2.6.T3_1-3.F2
/* Test SDO Write-By-Index */
{SDO Write-By-Index}(O1)
on no message received
    <FAIL> 3.2.6.T3_1-3.F3
on SDO-Abort
    <FAIL> 3.2.6.T3_1-3.F4
```

TEST 3.2.6.T4_1-3

```
/* Test reading/writing a non-existent index */
{SDO Read-By-Index}(O2)
on no message received
    <FAIL> 3.2.6.T4_1-3.F1
on success
    <FAIL> 3.2.6.T4_1-3.F2
on SDO-Abort code != 0602 0000h
    <FAIL> 3.2.6.T4_1-3.F3
{SDO Write-By-Index}(O2)
on no message received
    <FAIL> 3.2.6.T4_1-3.F4
on success
    <FAIL> 3.2.6.T4_1-3.F5
on SDO-Abort code != 0602 0000h
    <FAIL> 3.2.6.T4_1-3.F6
```

TEST 3.2.6.T5_1-3

```
/* Test reading/writing a non-existent subIndex */
{SDO Read-By-Index}(O3)
on no message received
    <FAIL> 3.2.6.T5_1-3.F1
on success
    <FAIL> 3.2.6.T5_1-3.F2
on SDO-Abort code != 0609 0011h
    <FAIL> 3.2.6.T5_1-3.F3
{SDO Write-By-Index}(O3)
on no message received
    <FAIL> 3.2.6.T5_1-3.F4
on success
    <FAIL> 3.2.6.T5_1-3.F5
on SDO-Abort code != 0609 0011h
    <FAIL> 3.2.6.T5_1-3.F6
```

TEST 3.2.6.T6_1-3

```
/* Test writing a read-only index */
{SDO Write-By-Index}(O4)
on no message received
    <FAIL> 3.2.6.T6_1-3.F1
on success
    <FAIL> 3.2.6.T6_1-3.F2
on SDO-Abort code != 0601 0002h
    <FAIL> 3.2.6.T6_1-3.F3
```

TEST 3.2.6.T7_1-3

```
/* Note: There is currently no mandatory object/subobject of accessType
 * write-only in the 0x1000-0x1FFF range.
 */
/* Test reading a write-only index */
{SDO Read-By-Index}(O5)
on no message received
    <FAIL> 3.2.6.T7_1-3.F1
on success
    <FAIL> 3.2.6.T7_1-3.F2
on SDO-Abort code != 0601 0001h
    <FAIL> 3.2.6.T7_1-3.F3
```

TEST 3.2.6.T8_1-3

```
/* Test reaction to NIL-Command */
{send C1}
wait 5 seconds
on Command-Layer ACK received within 5 seconds
    <FAIL> 3.2.6.T8_1-3.F1
on SDO channel closed within 5 seconds
    <FAIL> 3.2.6.T8_1-3.F1
{trigger SDO-Abort}
{resend manipulated NIL command}
{SDO Read-By-Index}(1018/01)
on (no response)
    <FAIL> 3.2.6.T8_1-3.F1
on (incorrect value)
    <FAIL> 3.2.6.T8_1-3.F1
```

TEST 3.2.6.T9_1-3

```
/* Test reaction to NIL-Command */
{send C1}
on !Sequence-Layer ACK
    <FAIL> 3.2.6.T9_1-3.F1
```

TEST 3.2.6.T10_1-3

```
/* Non-existing commands shall trigger an error-message */
{send C2}
on no message received
    <FAIL> 3.2.6.T10_1-3.F1
on success
    <FAIL> 3.2.6.T10_1-3.F2
on Abort-Code != 0504 0001h
    <FAIL> 3.2.6.T10_1-3.F3
```

Post-Condition:

No post-condition

3.2.6.2.3 Functional Requirements

3.2.6.T1.F1: Device-XDD states that SDO-Protocol is not supported (GeneralFeatures/@SDOServer == false).

3.2.6.T2_1-3.F1: Ident-Response of CN indicates that SDO-Transfertype <SDO-Transfertype> is not supported.

3.2.6.T3_1-3.F1: CN did not answer.

3.2.6.T3_1-3.F2: SDO Read-By-Index for readable object shall succeed, but failed with Abort-Code.

3.2.6.T3_1-3.F3: CN did not answer.

3.2.6.T3_1-3.F4: SDO Write-By-Index for writeable object shall succeed, but failed with Abort-Code.

3.2.6.T4_1-3.F1: CN did not answer.

3.2.6.T4_1-3.F2: SDO Read-By-Index non-existing object shall not succeed.

3.2.6.T4_1-3.F3: SDO Read-By-Index non-existing object shall fail with correct SDO abort-code.

3.2.6.T4_1-3.F4: CN did not answer.

3.2.6.T4_1-3.F5: SDO Write-By-Index non-existing object shall not succeed.

3.2.6.T4_1-3.F6: SDO Write-By-Index non-existing object shall fail with correct SDO abort-code.

3.2.6.T5_1-3.F1: CN did not answer.

3.2.6.T5_1-3.F2: SDO Read-By-Index non-existing subobject shall not succeed.

3.2.6.T5_1-3.F3: SDO Read-By-Index non-existing subobject shall fail with correct SDO abort-code.

3.2.6.T5_1-3.F4: CN did not answer.

3.2.6.T5_1-3.F5: SDO Write-By-Index non-existing subobject shall not succeed.

3.2.6.T5_1-3.F6: SDO Write-By-Index non-existing subobject shall fail with correct SDO abort-code.

3.2.6.T6_1-3.F1: CN did not answer.

3.2.6.T6_1-3.F2: SDO Write-By-Index a read-only object shall not succeed.

3.2.6.T6_1-3.F3: SDO Write-By-Index a read-only object shall fail with correct SDO abort-code.

3.2.6.T7_1-3.F1: CN did not answer.

3.2.6.T7_1-3.F2: SDO Read-By-Index a write-only object shall not succeed.

3.2.6.T7_1-3.F3: SDO Read-By-Index a write-only object shall fail with correct SDO abort-code.

3.2.6.T8_1-3.F1: Reception of a NIL-Command by the CN shall not result in a Command-Layer ACK being sent to the MN or closing the SDO channel within 5 seconds. An incorrect value in object 1018:01 (VendorID), or no answer by SDO Read will result in a test failure.

3.2.6.T9_1-3.F1: Reception of a NIL-Command by the CN shall result in a Sequence-Layer ACK being sent to the MN.

3.2.6.T10_1-3.F1: Sending of a non-existing command to the CN shall trigger a response.

3.2.6.T10_1-3.F2: Sending of a non-existing command to the CN shall not succeed.

3.2.6.T10_1-3.F3: Sending of a non-existing command to the CN shall fail with the correct SDO abort-code.

3.2.7 Plain/Extended NMT state commands

3.2.7.1 Description

The ability of a CN to react on NMT state commands resetting its NMT-State is essential to ensure correct operation. For each of the states:

- CS_PRE_OPERATIONAL_1
- CS_PRE_OPERATIONAL_2
- CS_READY_TO_OPERATE
- CS_OPERATIONAL
- CS_STOPPED

It shall be tested, whether the NMT state commands NMTSwReset, NMTResetNode, NMTResetCommunication and NMTResetConfiguration (as well as their extended versions, if supported) can successfully reset the CN (TEST 3.2.7.T1 shall be applied to each of these states).

MN shall be in MS_OPERATIONAL while executing this test. After reset CN shall bootup to CS_PRE_OPERATIONAL_2. To test correct behavior in CS_PRE_OPERATIONAL_1, MN shall be in state MS_PRE_OPERATIONAL_1.

In a second test, the NMTStopNode(Ex) command shall be tested on each of the states:

- CS_PRE_OPERATIONAL_2
- CS_READY_TO_OPERATE
- CS_OPERATIONAL

Upon receiving this command, the CN shall change its state to CS_STOPPED.

Further tests, test the NMTEnterPreOperational2(Ex), NMTStartNodeEx and NMTEnableReadyToOperateEx-Commands.

3.2.7.2 Action/Response Sequences

Parameter:

t_1 = AsyncSlotTimeout_U32

t_2 = D_NMT_BootTimeNotActive_U32

t_3 = C_NMT_STATE_TOLERANCE(5) * cycleTime

Pre-Condition:

MN is in state MS_OPERATIONAL to test all CN-States except CS_PRE_OPERATIONAL_1

MN is in state MS_PRE_OPERATIONAL_1 to test CS_PRE_OPERATIONAL_1

CN is in state CS_PRE_OPERATIONAL1 || CS_PRE_OPERATIONAL_2 ||

CS_READY_TO_OPERATE || CS_OPERATIONAL || CS_STOPPED

TEST 3.2.7.T1

```
{foreach (unicast, broadcast)(NMTSwReset, NMTResetNode,
NMTResetCommunication, NMTResetConfiguration)}
    {send ASnd with NMTCommandID}(nodeID)
        {wait}(t2 + t3 + t3)
        if (MN-State == MS_PRE_OPERATIONAL_1)
            {repeat}
                /* CN should now be in CS_PRE_OPERATIONAL_1 */
                {send SoA RequestedServiceID IDENT_REQUEST}(nodeID)
                {wait}(t1)
            {until (message received &&
                    reported state == CS_PRE_OPERATIONAL_1) or
                   MAX_COUNT times}
            {if count > 1 && count <= MAX_COUNT}
                <FAIL> 3.2.7.T1.F1 /*device changed state, but not within
            timeout*/
            {else if count > MAX_COUNT}
                {if no message received}
                    <FAIL> 3.2.7.T1.F2 /* device did not answer at all */
                {else}
                    <FAIL> 3.2.7.T1.F3 /* device did not change state */
        else if (MN-State == MS_OPERATIONAL)
            /* wait for CN to switch from CS_PRE_OPERATIONAL_1 to
             CS_PRE_OPERATIONAL_2 */
            {wait}(t3)
            {repeat}
                {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
                {wait}(t1)
            {until (message received &&
                    reported state == CS_PRE_OPERATIONAL_2) or
                    MAX_COUNT times}
            {if count > 1 && count <= MAX_COUNT}
                <FAIL> 3.2.7.T1.F4 /*device changed state, but not within
                timeout*/
            {else if count > MAX_COUNT}
                {if no message received}
                    <FAIL> 3.2.7.T1.F5 /* device did not answer at all */
                {else}
                    <FAIL> 3.2.7.T1.F6 /* device did not change state */
```

Pre-Condition:

TEST 3.2.1.T1

TEST 3.2.7.T2_1

```
/* MN shall be in MS_PRE_OPERATIONAL_2 */
/* CN shall be in CS_PRE_OPERATIONAL_2 */
{repeat}
    {send ASnd with NMTStopNode}(nodeID)
    {wait}(t3)
    {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
    {wait}(t1)
{until (message received && reported state == CS_STOPPED)
 or MAX_COUNT times}
{if count > 1 && count <= MAX_COUNT}
    <FAIL> 3.2.7.T2_1.F1 /*device changed state, but not within timeout */
{else if count > MAX_COUNT}
    {if no message received}
        <FAIL> 3.2.7.T2_1.F2 /* device did not answer at all */
    {else}
        <FAIL> 3.2.7.T2_1.F3 /* device did not change state */
```

TEST 3.2.7.T2_2

```
/* Identical to TEST 3.2.7.T2_1 with following exceptions: */
 *
 * - MN shall be in MS_READY_TO_OPERATE.
 * - CN shall be in CS_READY_TO_OPERATE.
 */
```

TEST 3.2.7.T3_1-2

```
/* Identical to TEST 3.2.7.T2_1-2 with following exceptions: */
 *
 * - Sending NMT-Command NMTStopNodeEx instead of NMTStopNode.
 * - Returns <NOT_SUPPORTED> with error F4 if node does not support
 *   extended NMT-Commands (0x1F82, bit 5 == 0)
 */
```

TEST 3.2.7.T3_3

```
/* Identical to TEST 3.2.7.T2_1 with following exceptions: */
 *
 * - MN and CN shall be in OPERATIONAL.
 * - Sending NMT-Command NMTStopNodeEx instead of NMTStopNode.
 * - Returns <NOT_SUPPORTED> with error F4 if node does not support
 *   extended NMT-Commands (0x1F82, bit 5 == 0)
 */
```

Post-Condition:

CN is in state CS_STOPPED

TEST 3.2.7.T4

```
/* Identical to TEST 3.2.7.T2_1 with following exceptions: */
 *
 * - MN shall be in MS_READY_TO_OPERATE.
 * - CN shall be in CS_STOPPED.
 * - Sending NMT-Command NMTEnterPreOperational2Ex.
 * - Returns <NOT_SUPPORTED> with error F4 if node does not support
 *   extended NMT-Commands (0x1F82, bit 5 == 0)
 */
```

TEST 3.2.7.T5_1

```
/* Identical to TEST 3.2.7.T2_1 with following exceptions: */
 *
 * - MN shall be in MS_OPERATIONAL.
 * - CN shall be in CS_OPERATIONAL.
 * - Sending NMT-Command NMTEnterPreOperational2.
 */
```

TEST 3.2.7.T5_2

```
/* Identical to TEST 3.2.7.T2_1 with following exceptions: */
 *
 * - MN shall be in MS_OPERATIONAL.
 * - CN shall be in CS_OPERATIONAL.
 * - Sending NMT-Command NMTEnterPreOperational2Ex.
 * - Returns <NOT_SUPPORTED> with error F4 if node does not support
 *   extended NMT-Commands (0x1F82, bit 5 == 0)
```

*/

Post-Condition:

CN is in state CS_PRE_OPERATIONAL_2

3.2.7.3 Functional Requirements

3.2.7.T1.F1: Device shall change its state to CS_PRE_OPERATIONAL_1 within $t_1 + t_3$, upon MN sending command NMTSwReset, NMTResetNode, NMTResetCommunication, or NMTResetConfiguration.

3.2.7.T1.F2: Device shall respond to IdentRequests received after being reset by the MN.

3.2.7.T1.F3: Device shall change its state to CS_PRE_OPERATIONAL_1 upon MN sending SoA-Frame (IDENT_REQUEST).

3.2.7.T1.F4: Device shall change its state to CS_PRE_OPERATIONAL_2 within $t_1 + 2\ t_3$, upon MN sending command NMTSwReset, NMTResetNode, NMTResetCommunication, or NMTResetConfiguration.

3.2.7.T1.F5: Device shall respond to StatusRequests received after being reset by the MN.

3.2.7.T1.F6: Device shall change its state to CS_PRE_OPERATIONAL_2 upon MN sending SoC- and SoA-Frames.

3.2.7.T2-5.F1: Device shall change its state to <*NMT-State*> within $t_1 + t_3$, upon MN sending command <*NMT-State command*>.

3.2.7.T2-5.F2: Device shall respond to StatusRequests after MN issues command <*NMT-State command*>.

3.2.7.T2-5.F3: Device shall change its state to <*NMT-State*> upon MN sending command <*NMT-State command*>.

3.2.7.T3(4)(5_2).F4: Testcase is not supported since device does not support extended NMT-State commands (Featureflags, Bit 5 is not set).

3.2.8 NMT Info-Services

3.2.8.1 Description

Each command will be tested, regardless of whether its feature-object in the object dictionary indicates support for it. If actual support for the command and feature-object differ, test shall fail. Even if command is not supported, CN shall remain in CS_OPERATIONAL.

This test will be executed with MN and CN being both in OPERATIONAL.

3.2.8.2 Action/Response Sequences

Parameter:

t_1 = AsyncSlotTimeout_U32

Pre-Condition:

MN and CN are in state OPERATIONAL

TEST 3.2.8.T1

```
{foreach (NMT Info-Service)}
    {send ASnd with NMTCommandID}(nodeID)
    {during C_NMT_STATE_TOLERANCE(5) cycles}
        {send SoA RequestedServiceID STATUS_REQUEST}(nodeID)
        {wait time}(t1)
        on no message received
            <FAIL> 3.2.8.T1.F1
        on (message received && reported state != CS_OPERATIONAL)
            <FAIL> 3.2.8.T1.F2
```

Post-Condition:

CN is in state CS_OPERATIONAL

3.2.8.3 Functional Requirements

3.2.8.T1.T1.F1: A CN shall respond to StatusRequests issued by the MN within the asynchronous phase.

3.2.8.T1.T1.F2: A CN shall remain in CS_OPERATIONAL upon receiving NMTInfo-Commands, even it does not support them.

3.2.9 Object Dictionary

Tests in this chapter shall make sure that:

- All objects defined in the XDD-OD exist on the device and are initialized with their default-values.
- No object exists on the device, which is not defined in the XDD-OD.
- Read-Only objects can not be written to and writeable objects can be written to.

All tests require the device to support at least one type of SDO-Transfer and cannot be conducted if there is no SDO-Support.

Furthermore following definitions apply to all tests in this chapter:

- Functional objects: Objects whose modification trigger an operation, i.e. store/restore objects (1010h, 1011h).
- Protected objects: Objects which can only be written to or read from, if a second object enables the correct execution of this operation (i.e. PDO-Mapping and –Communication objects).

3.2.9.1 Reading OD-Entries

3.2.9.1.1 Description

All objects defined in the Ethernet POWERLINK Communication Profile (index 1000h-1FFFh) shall be iterated and being read from the device:

- If it can be read and it does not exist in the XDD-OD, test shall fail.
- If it can be read, exists in the XDD-OD and is write-only, test shall fail.
- If it can be read, exists in the XDD-OD and is defined readable, read value shall equal default value (if there is a default value defined), otherwise test shall fail.
- If it cannot be read, exists in the XDD-OD and is not write-only, test shall fail.

3.2.9.1.2 Action/Response Sequences

Parameter:

L1 = List of XDD-OD entries with defaultValues and highLimit-, lowLimit-Attributes. Entries without explicit high- and/or lowLimit-Attributes shall be supplied with their limits according to EPSG DS 301 v1.1.0, or, if these are missing as well, with their datatype-limits instead.

L2 = List of functional and protected objects, write-only objects, DOMAINS, objects valid on MN only and objects with unsupported datatypes (non-Integer, non-String, non-Boolean), extracted from the POWERLINK Communication Profile

Pre-Condition:

Device passed 3.1

Device passed 3.2.1

Device passed 3.2.6.2, **Fehler! Verweisquelle konnte nicht gefunden werden.**

MN is in MS_PRE_OPERATIONAL_1

CN is in CS_PRE_OPERATIONAL_1

TEST 3.2.9.T1

```
{for all objects in the Ethernet POWERLINK Communication Profile and not in
L2}
    {for all subObjects in the Ethernet POWERLINK Communication Profile and
    not in L2}
        {read object/subObject on device}
            on no message received
                <FAIL> 3.2.9.T1.F1
            on success
                on (object/subObject not in L1)
                    <FAIL> 3.2.9.T1.F2
                on (object/subObject in L1 && object == write-only)
                    <FAIL> 3.2.9.T1.F3
                on (object/subObject in L1 && hasDefaultValue() &&
                    values not equal)
                    <FAIL> 3.2.9.T1.F4
            on SDO-Abort
                on (object/subObject in L1 && accessType != write-only)
                    <FAIL> 3.2.9.T1.F5
                on (object/subObject in L1 && accessType == write-only &&
                    SDO-Abort code != 0601 0001h)
                    <FAIL> 3.2.9.T1.F6
                on (object not in L1 && SDO-Abort code != 0602 0000h)
                    <FAIL> 3.2.9.T1.F7
                on (subObject not in L1 && SDO-Abort code != 0609 0011h)
                    <FAIL> 3.2.9.T1.F8
```

3.2.9.1.3 Functional Requirements

3.2.9.T1.F1: CN did not answer.

3.2.9.T1.F2: Objects/Subobjects not defined in the XDD-OD shall not exist on the device.

3.2.9.T1.F3: Objects/Subobjects defined as write-only in the XDD-OD shall not be readable from the device.

3.2.9.T1.F4: Objects/Subobjects that exist in the XDD and on the device shall have identical default-values.

3.2.9.T1.F5: Non-write-only objects/subobjects defined in the XDD-OD shall be readable from the device.

3.2.9.T1.F6: Write-only objects/subobjects defined in the XDD-OD shall trigger SDO-Abort code 0601 0001h upon reading.

3.2.9.T1.F7: Objects not defined in the XDD-OD shall trigger SDO-Abort code 0602 0000h upon reading.

3.2.9.T1.F8: Subobjects not defined in the XDD-OD shall trigger SDO-Abort code 0609 0011h upon reading.

3.2.9.2 Writing OD-Entries

3.2.9.2.1 Description

All objects defined in the Ethernet POWERLINK Communication Profile (index 1000h-1FFFh) and existing in the XDD-OD shall be iterated and being written to the device.

A test-value for each object shall be written to the device. Test shall fail if:

- Writing to a writeable object fails
- Writing to a read-only object succeeds

Furthermore, the lowLimit and highLimit for default-values shall be tested, as well as values outside these limits. Test shall fail if:

- A value >= lowLimit and <= highLimit cannot be written.
- A value < lowLimit or > highLimit can be written
- A value can be written to and read back from the device, but the written and read values are not equal.

3.2.9.2.2 Action/Response Sequences

Parameter:

L1 = List of XDD-OD entries with defaultValues and highLimit-, lowLimit-Attributes. Entries without explicit high- and/or lowLimit-Attributes shall be supplied with their limits according to EPSG DS 301 v1.1.0, or, if these are missing as well, with their datatype-limits instead.

L2 = List of functional and protected objects, DOMAINS, objects valid on MN only and objects with unsupported datatypes (non-Integer, non-String, non-Boolean), extracted from the POWERLINK Communication Profile

Pre-Condition:

Device passed 3.1

Device passed 3.2.1

Device passed 3.2.6.2, **Fehler! Verweisquelle konnte nicht gefunden werden.**

MN is in MS_PRE_OPERATIONAL_1

CN is in CS_PRE_OPERATIONAL_1

TEST 3.2.9.T2

```
{for each object and subObject from 1000h-1FFFh which is in L1 and not L2}
    {write test-value to object on device}
```

```
        {if object is writeable}
            on no message received
                <FAIL> 3.2.9.T2.F1
            on no success
                <FAIL> 3.2.9.T2.F2
        {else}
            on no message received
                <FAIL> 3.2.9.T2.F1
            on success
                <FAIL> 3.2.9.T2.F3
            on no success && accessType == ro &&
            SDO-Abort code != 0601 0002h
                <FAIL> 3.2.9.T2.F4
            on no success && accessType == const &&
            SDO-Abort code != 0601 0000h
                <FAIL> 3.2.9.T2.F5
```

TEST 3.2.9.T3.1-4

```
{for each object and subObject from 1000h-1FFFh which is in L1 and not L2}
    on (object is writeable and object is integer)
        /* T3.1: upper boundary */
        {set testValue = highLimit}
        {write testValue to object on device}
            on no message received
                <FAIL> 3.2.9.T3.1.F1
            on no success
                <FAIL> 3.2.9.T3.1.F2
            on (success and object is readable)
                {read object from device and save in objectValue}
                    on no message received
                        <FAIL> 3.2.9.T3.1.F5
                    on no success
                        <FAIL> 3.2.9.T3.1.F3
                    on (testValue != objectValue)
                        <FAIL> 3.2.9.T3.1.F4
        /* T3.2: lower boundary */
        {set testValue = lowLimit}
        {write testValue to object on device}
            on no message received
                <FAIL> 3.2.9.T3.2.F1
            on no success
                <FAIL> 3.2.9.T3.2.F2
            on (success and object is readable)
                {read object from device and save in objectValue}
                    on no message received
                        <FAIL> 3.2.9.T3.2.F5
                    on no success
                        <FAIL> 3.2.9.T3.2.F3
                    on (testValue != objectValue)
                        <FAIL> 3.2.9.T3.2.F4
        /* T3.3: outside lower boundary */
        {if lowLimit > type min value}
```

```
        {set testValue = type min value }
        {write testValue to object on device}
            on no message received
                <FAIL> 3.2.9.T3.3.F1
            on SDO-Abort code != 0609 0032h
                <FAIL> 3.2.9.T3.3.F2
            on success
                <FAIL> 3.2.9.T3.3.F3
    /* T3.4: outside upper boundary */
    {if highLimit < type max value}
        {set testValue = type max value }
        {write testValue to object on device}
            on no message received
                <FAIL> 3.2.9.T3.4.F1
            on SDO-Abort code != 0609 0031h
                <FAIL> 3.2.9.T3.4.F2
            on success
                <FAIL> 3.2.9.T3.4.F3
    /* T3.5: test whether it is possible to write a 3-Byte value
    to the object, not yet implemented */
    {write 3-byte value to object}
        on no message received
            <FAIL> 3.2.9.T3.5.F1
        on success
            <FAIL> 3.2.9.T3.5.F2
```

3.2.9.2.3 Functional Requirements

3.2.9.T2.F1: CN did not answer.

3.2.9.T2.F2: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall be writeable.

3.2.9.T2.F3: Each object and subObject in the 1000-1FFF range that is defined non-writeable in the XDD-OD and which exists on the device, shall not be writeable.

3.2.9.T2.F4: Each object and subObject in the 1000-1FFF range that is defined read-only in the XDD-OD and which exists on the device, shall not be writeable. Writing shall fail with SDO-Abortcode 0601 0002h.

3.2.9.T2.F5: Each object and subObject in the 1000-1FFF range that is defined const in the XDD-OD and which exists on the device, shall not be writeable. Writing shall fail with SDO-Abortcode 0601 0002h or 0601 0000h.

3.2.9.T3.1.F1: CN did not answer.

3.2.9.T3.1.F2: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall be writeable with its highLimit-value.

3.2.9.T3.1.F3: Each object and subObject in the 1000-1FFF range that is defined readable in the XDD-OD and which exists on the device, shall be readable.

3.2.9.T3.1.F4: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall be writeable with its highLimit-value. This written value and the subsequently read value shall be equal.

3.2.9.T3.1.F5: CN did not answer.

3.2.9.T3.2.F1: CN did not answer.

3.2.9.T3.2.F2: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall be writeable with its lowLimit-value.

3.2.9.T3.2.F3: Each object and subObject in the 1000-1FFF range that is defined readable in the XDD-OD and which exists on the device, shall be readable.

3.2.9.T3.2.F4: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall be writeable with its lowLimit-value.This written value and the subsequently read value shall be equal.

3.2.9.T3.2.F5: CN did not answer.

3.2.9.T3.3.F1: CN did not answer.

3.2.9.T3.3.F2: Writing a value < lowLimit to an object/subObject shall result in SDO-Abort code 0609 0032h.

3.2.9.T3.3.F3: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall not be writeable with a value that is less than its lowLimit-value.

3.2.9.T3.4.F1: CN did not answer.

3.2.9.T3.4.F2: Writing a value > highLimit to an object/subObject shall result in SDO-Abort code 0609 0031h.

3.2.9.T3.4.F3: Each object and subObject in the 1000-1FFF range that is defined writeable in the XDD-OD and which exists on the device, shall not be writeable with a value that is greater than its highLimit-value.

3.2.9.T3.5.F1: CN did not answer.

3.2.9.T3.5.F2: It shall not be possible to write a 3-byte value to an object.

3.2.9.3 Store and Restore Parameters

3.2.9.3.1 Description

Checks if the devices store- and restore-functions work as they are supposed to.

3.2.9.3.2 Action/Response Sequences

Parameter:

C_1 = Value mask to determine support for storing (00000003h)

C_2 = Write value for store-object to execute storing (6576716173h)

C_3 = Value mask to determine support for restoring (00000001h)

C_4 = Write value for store-object to execute storing (64616F6Ch)

t_1 = C_NMT_STATE_TOLERANCE(5) * cycleTime

t_2 = D_NMT_BootTimeNotActive_U32

t_3 = Time to wait for CN to start invalid restore-process (configurable)

t_{off} = generic power-off time

O_1 = object 1006.00h

V_1, V_2 = object value buffer

L1 = List of XDD-OD entries with defaultValues and highLimit-, lowLimit-Attributes. Entries without explicit high- and/or lowLimit-Attributes shall be supplied with their limits according to EPSG DS 301 v1.1.0, or, if these are missing as well, with their datatype-limits instead.

L2 = List of functional and protected objects, read-only/write-only objects, DOMAINS, objects valid on MN only and objects with unsupported datatypes (non-Integer, non-String, non-Boolean), extracted from the POWERLINK Communication Profile

Pre-Condition (all tests):

Device passed 3.2.7.T1

Device passed 3.2.9.1, 3.2.9.2

MN is in MS_PRE_OPERATIONAL_1

CN is in CS_PRE_OPERATIONAL_1

TEST 3.2.9.T4.1

```
/* Check storing with Power OFF */
/* test storage of parameters only if it is supported (existing object
 * 1010h)
 */
{if not object 1010h exists in L1}
    <FAIL>3.2.9.T4.1.F1
{else}
    {read object(Index 1010h, Sub-Index 01h) into V1}
    on no message received
        <FAIL> 3.2.9.T4.1.F2
    on no success
        <FAIL> 3.2.9.T4.1.F3
    on (V1 & C1) == 0
        <FAIL> 3.2.9.T4.1.F4
        Exit
    on (V1 & C1) > 0
        /* Device supports storage of all parameters */
        {write all objects in L1 but not in L2 except for O1 with upper
        boundary}
           on no message received
               <FAIL> 3.2.9.T4.1.F5
               exit
           on no success
               <FAIL> 3.2.9.T4.1.F6
               exit
    on (V1 & C1) < 2
        /* Device does not save parameters automatically */
        {write object 1010h Sub-Index 01h with value C2}
        on no message received
            <FAIL> 3.2.9.T4.1.F5
            exit
        on no success
            <FAIL> 3.2.9.T4.1.F6
            exit
    {power-off Device}
    {wait}(t_off)
    {power-on device and transition into CS_PRE_OPERATIONAL_1}
    {wait}(t_1 + t_2)
    on NMT-State < CS_PRE_OPERATIONAL_1
        <FAIL> 3.2.9.T4.1.F7
        exit
    {read all objects listed in L1 but not in L2 except for O1}
        on no message received
            <FAIL> 3.2.9.T4.1.F2
            exit
        on no success
            <FAIL> 3.2.9.T4.1.F3
            exit
        on values not equal to upper boundary
            <FAIL> 3.2.9.T4.1.F8
```

```
/* Identical to TEST 3.2.9.T4.1 with following exceptions: */
/*
 * - Storing configuration-parameters instead of all parameters
 *(0x1010/0x02)
 * - Writing lower-boundary instead of upper-boundary to test-objects
 *
 * All error-numbers that apply for TEST 3.2.9.T4.1, apply for this test as
 * well
 * (with adapted testNr-part)
 */
```

TEST 3.2.9.T5.1-4

```
/* Identical to TEST 3.2.9.T4.1 with following exceptions: */
/*
 * - Restart the node through NMT-Commands (SwReset, ResetNode,
 *   ResetConfiguration, ResetCommunication) instead of power-
 *   off/wait/power-on.
 * - Wait time after reset: (2 * t1 + t2) due to node changing its state
 *   from CS_PRE_OPERATIONAL_1 to GS_INITIALISING upon reset.
 * - T5.1: Writing upper-boundary, T5.2: lower, T5.3: upper, T5.4: lower
 *
 * All error-numbers that apply for TEST 3.2.9.T4.1, apply for these tests
 * as well (with adapted testNr-part)
 */
```

TEST 3.2.9.T5.5-8

```
/* Identical to TEST 3.2.9.T5.1-4 with following exceptions: */
/*
 * - Storing configuration-parameters instead of all parameters
 *  (0x1010/0x02)
 * - T5.5: Writing upper-boundary, T5.6: lower, T5.7: upper, T5.8: lower
 *
 * All error-numbers that apply for TEST 3.2.9.T4.1, apply for these tests
 * as well (with adapted testNr-part)
 */
```

Pre-Condition:

Device passed TEST 3.2.9.T4.1

TEST 3.2.9.T6.1

```
/* check the restore parameter, this part can only be checked if restoring
is supported */
/* Check restore with Power OFF */
{if not (object 1010h and object 1011h exist inside L1)}
    <NOT_SUPPORTED> 3.2.9.T6.1.F1
{else}
    {read object(Index 1011h, Sub-Index 01h) into V1}
    on no message received
        <FAIL> 3.2.9.T6.1.F2
    on no success
        <FAIL> 3.2.9.T6.1.F3
    on (V1 & C3) == 0
        <FAIL> 3.2.9.T6.1.F4
    on (V1 & C3) == 1
        {write all objects listed in L1 but not in L2 except for O1 with
        upper-boundary}
            on no message received
                <FAIL> 3.2.9.T6.1.F5
                exit
            on no success
                <FAIL> 3.2.9.T6.1.F6
                exit
            /*
             * Determine storage capabilities, we need to know whether
             * device stores autonomously or just on command.
             */
            {read object(Index 1010h, Sub-Index 01h) into V2}
            on no message received
                <FAIL> 3.2.9.T6.1.F2
            on no success
                <FAIL> 3.2.9.T6.1.F3
            on (V2 & C1) < 2
                /* Device does not save parameters autonomously, store
```

```
    manually */
    {write object 1010h Sub-Index 01h with value C2}
        on no message received
            <FAIL> 3.2.9.T6.1.F5
            exit
        on no success
            <FAIL> 3.2.9.T6.1.F6
            exit
/* check if storage was successful */
{power-off Device}
{wait}(toff)
{power-on device and transition into CS_PRE_OPERATIONAL_1}
{wait}(t1 + t2)
on NMT-State < CS_PRE_OPERATIONAL_1
    <FAIL> 3.2.9.T6.1.F7
    exit
{read all objects listed in L1 but not in L2 except for O1}
    on no message received
        <FAIL> 3.2.9.T6.1.F2
        exit
    on no success
        <FAIL> 3.2.9.T6.1.F3
        exit
    on values not equal to upper boundary
        <FAIL> 3.2.9.T6.1.F8
        exit
/* restore the values */
{write value C4 into object with Index 1011h and Sub-Index
01h}
on no message received
    <FAIL> 3.2.9.T6.1.F5
    exit
on no success
    <FAIL> 3.2.9.T6.1.F6
    exit
/* check if restoring of objects began immediately */
{wait}(t3)
{read all objects listed in L1 but not in L2 except for O1}
    on no message received
        <FAIL> 3.2.9.T6.1.F2
        exit
    on no success
        <FAIL> 3.2.9.T6.1.F3
        exit
    on values not equal to upper boundary
        <FAIL> 3.2.9.T6.1.F9
        exit
/* perform restart of the Device to start restoring */
{power-off Device}
{wait}(toff)
{power-on device and transition into CS_PRE_OPERATIONAL_1}
{wait}(t1 + t2)
on NMT-State < CS_PRE_OPERATIONAL_1
    <FAIL> 3.2.9.T6.1.F7
```

```
    exit
{read all objects listed in L1 but not in L2 except for O1}
on no message received
    <FAIL> 3.2.9.T6.1.F2
on no success
    <FAIL> 3.2.9.T6.1.F3
on values not equal to default value
    <FAIL> 3.2.9.T6.1.F10
```

TEST 3.2.9.T6.2

```
/* Identical to TEST 3.2.9.T6.1 with following exceptions:
 * - Storing/Restoring configuration-parameters instead of all parameters
 *   (0x1010/0x02)
 * - Writing lower-boundary instead of upper-boundary
 * - Pre-Condition: Test passed TEST 3.2.9.T4.2
 *
 * All error-numbers that apply for TEST 3.2.9.T6.1, apply for this test as
 * well (with adapted testNr-part)
 */
```

TEST 3.2.9.T7.1-4

```
/* Identical to TEST 3.2.9.T6.1 with following exceptions:
 *
 * - Restart the node through NMT-Commands (SwReset, ResetNode,
 *   ResetConfiguration, ResetCommunication) instead of power-
 *   off/wait/power-on.
 * - Wait time after reset: (2 * t1 + t2) due to node changing its state
 *   from CS_PRE_OPERATIONAL_1 to GS_INITIALISING upon reset.
 * - In case of reset by ResetConfiguration, test will fail with F10 if
 *   values are not equal to upper-boundary!
 * - T7.1: Writing upper-boundary, T7.2: lower, T7.3: upper, T7.4: lower
 * - Pre-Condition: Test passed TEST 3.2.9.T5.1-4
 *
 * All error-numbers that apply for TEST 3.2.9.T6.1, apply for this test as
 * well (with adapted testNr-part)
 */
```

TEST 3.2.9.T7.5-8

```
/* Identical to TEST 3.2.9.T7.1-4 with following exceptions:
 *
 * - Storing configuration-parameters instead of all parameters
 *   (0x1010/0x02)
 * - In case of reset by ResetConfiguration, test will fail with F10 if
 *   values are not equal to default values!
 * - T7.5: Writing upper-boundary, T7.6: lower, T7.7: upper, T7.8: lower
 * - Pre-Condition: Test passed TEST 3.2.9.T5.5-8
 *
 * All error-numbers that apply for TEST 3.2.9.T7.1-4, apply for this test
 * as well (with adapted testNr-part)
 */
```

Post-Condition:

All OD-Values on device are reset to their default values (XDD)

3.2.9.3.3 Functional Requirements

3.2.9.T4(5).X.F1:	Device-XDD does not contain object 0x1010 (Storage of parameters), this testcase is not supported.
3.2.9.T4(5).X.F2:	Device did not answer on SDO-Read of object 0xXXXX/0xXX within timeout.
3.2.9.T4(5).X.F3:	SDO-Read of object 0xXXXX/0xXX failed with SDO-Abort code.
3.2.9.T4(5).X.F4:	Object 0x1010/0xXX equals 0 or does not exist, device does not support storage of given parameter-type. Testcase not supported.
3.2.9.T4(5).X.F5:	Device did not answer on SDO-Write of object 0xXXXX/0xXX within timeout.
3.2.9.T4(5).X.F6:	SDO-Write of object 0xXXXX/0xXX failed with SDO-Abort code.
3.2.9.T4(5).X.F7:	Device did not reach state CS_PRE_OPERATIOAL_1 within timeout after storage of parameters and a subsequent power-off/on or reset.
3.2.9.T4(5).X.F8:	Value of object 0xXXXX/0xXX not equal to the previously written test-value. Storage failed.
3.2.9.T6(7).X.F1:	Device-XDD does not contain object 0x1010 (Storage of parameters) or object 0x1011 (Restore of parameters), this testcase is not supported.
3.2.9.T6(7).X.F2:	Device did not answer on SDO-Read of object 0xXXXX/0xXX within timeout.
3.2.9.T6(7).X.F3:	SDO-Read of object 0xXXXX/0xXX failed with SDO-Abort code.
3.2.9.T6(7).X.F4:	Object 0x1011/0xXX equals 0 or does not exist, device does not support restore of given parameter-type. Testcase not supported.
3.2.9.T6(7).X.F5:	Device did not answer on SDO-Write of object 0xXXXX/0xXX within timeout.
3.2.9.T6(7).X.F6:	SDO-Write of object 0xXXXX/0xXX failed with SDO-Abort code.
3.2.9.T6(7).X.F7:	Device did not reach state CS_PRE_OPERATIOAL_1 within timeout after storage or restore of parameters and a subsequent power-off/on or reset.
3.2.9.T6(7).X.F8:	Value of object 0xXXXX/0xXX not equal to written test-value. Storage of parameter-type failed.
3.2.9.T6(7).X.F9:	Value of object 0xXXXX/0xXX not equal to written test-value. Parameter-Restore shall not begin immediately after initiating restore-process.
3.2.9.T6.1-2.F10:	Value of object 0xXXXX/0xXX not equal to its defaultValue. Parameter-Restore failed.

3.2.9.T7.1-4.F10: Value of object 0xXXXX/0xXX not equal to its defaultValue (for SwReset, ResetNode, ResetCommunication) or written test-value (for ResetConfiguration). Parameter-Restore failed.

3.2.9.T7.5-8.F10: Value of object 0xXXXX/0xXX not equal to its defaultValue.

Parameter-Restore failed.

3.2.9.4 OD-Behaviour after Reset

3.2.9.4.1 Description

In this test, the behaviour of the devices OD after a reset is tested. Therefore, following resets are conducted, after changing OD-entries: NMTSwReset, NMTResetNode, NMTResetCommunication, and NMTResetConfiguration.

3.2.9.4.2 Action/Response Sequences

Parameter:

t_1 = C_NMT_STATE_TOLERANCE(5) * cycleTime

t_2 = D_NMT_BootTimeNotActive_U32

O1 = object 1006.00h

L1 = List of XDD-OD entries with defaultValues and highLimit-, lowLimit-Attributes. Entries without explicit high- and/or lowLimit-Attributes shall be supplied with their limits according to EPSG DS 301 v1.1.0, or, if these are missing as well, with their datatype-limits instead.

L2 = List of functional and protected objects, read-only/write-only objects, DOMAINS, objects valid on MN only and objects with unsupported datatypes (non-Integer, non-String, non-Boolean), extracted from the POWERLINK Communication Profile

Pre-Condition:

Device shall be restored to default values

Device passed 3.2.7.T1

Device passed 3.2.9.1, 3.2.9.2

Device is in CS_PRE_OPERATIONAL_1

TEST 3.2.9.T8.1

```
/* set new values in objects */
{write all integer-objects listed in L1 but not in L2 except for O1 with
upper boundary}
    on no message received
        <FAIL> 3.2.9.T8.1.F1
        exit
    on no success
        <FAIL> 3.2.9.T8.2.F2
        exit
/* perform a reset by writing into object NMT_ResetCmd_U8 */
{write NMTSwReset to object 1F9Eh}(nodeID)
    on no message received
        <FAIL> 3.2.9.T8.1.F1
        exit
    on no success
        <FAIL> 3.2.9.T8.1.F2
        exit
{transition device into CS_PRE_OPERATIONAL_1}
on NMT-State <> CS_PRE_OPERATIONAL_1
    <FAIL> 3.2.9.T8.1.F6
    exit
/* check the changed OD-Entries */
{read all integer-objects listed in L1 but not in L2 except for O1}
    on no message received
        <FAIL> 3.2.9.T8.1.F3
    on no success
        <FAIL> 3.2.9.T8.1.F4
    on values not equal to default value
        <FAIL> 3.2.9.T8.1.F5
```

TEST 3.2.9.T8.2

```
/* Identical to TEST 3.2.9.T8.2 with following exceptions:
 *
 * - Write NMTResetNode to object 1F9Eh.
 *
 * All error-numbers that apply for TEST 3.2.9.T8.1, apply for this test as
 * well (with adapted testNr-part)
 */
```

TEST 3.2.9.T8.3

```
/* Identical to TEST 3.2.9.T8.2 with following exceptions:
 *
 * - Write NMTResetCommunication to object 1F9Eh.
 *
 * All error-numbers that apply for TEST 3.2.9.T8.1, apply for this test as
 * well (with adapted testNr-part)
 */
```

TEST 3.2.9.T8.4

```
/* Identical to TEST 3.2.9.T8.2 with following exceptions:
 *
 * - Write NMTResetConfiguration to object 1F9Eh.
 * - Error F5 will be triggered if values on device are not equal to upper
 *   boundary.
 *
 * All error-numbers that apply for TEST 3.2.9.T8.1, apply for this test as
 * well (with adapted testNr-part)
 */
```

TEST 3.2.9.T9.1-4

```
/* Identical to TEST 3.2.9.T8.1-4 with following exceptions:
 *
 * - Resets are conducted by sending NMTReset-Commands to CN.
 *
 * All error-numbers that apply for TEST 3.2.9.T8.1-4, apply for this test
 * as well (with adapted testNr-part)
 */
```

TEST 3.2.9.T10.1-4

```
/* Identical to TEST 3.2.9.T9.1-4 with following exceptions:
 *
 * - Test is only executed if DUT supports extended NMT-State commands.
 *   (NMT_FeatureFlags_U32, Bit 5) and returns NOT_SUPPORTED F7 otherwise.
 * - Extended NMTReset-Commands are being used.
 *
 * All error-numbers that apply for TEST 3.2.9.T9.1-4, apply for this test
 * as well (with adapted testNr-part)
 */
```

Post-Condition:

All objects are reset to their default values.

3.2.9.4.3 Functional Requirements

3.2.9.T8-10.1-4.F1: Device did not answer on SDO-Write of object 0xXXXX/0xXX within timeout.

3.2.9.T8-10.1-4.F2: SDO-Write of object 0xXXXX/0xXX failed with SDO-Abort code.

3.2.9.T8-10.1-4.F3: Device did not answer on SDO-Read of object 0xXXXX/0xXX within timeout.

3.2.9.T8-10.1-4.F4: SDO-Read of object 0xXXXX/0xXX failed with SDO-Abort code.

3.2.9.T8.1-3.F5: After writing objects and a subsequent reset of the CN by writing NMTSwReset, NMTResetNode or NMTResetCommunication to the CN's

	0x1F9E object, all objects in L3 but not in L2, except O1, shall be reset to their default-values.
3.2.9.T8.4.F5:	After writing objects and a subsequent reset of the CN by writing NMTResetConfiguration to the CN's 0x1F9E object, all objects in L3 but not in L2, except O1, shall still be set to their upper boundary.
3.2.9.T9.1-3.F5:	After writing objects and a subsequent reset of the CN by an NMTSwReset, NMTResetNode or NMTResetCommunication, all objects in L3 but not in L2, except O1, shall be reset to their default-values.
3.2.9.T9.4.F5:	After writing objects and a subsequent reset of the CN by an NMTResetConfiguration, all objects in L3 but not in L2, except O1, shall still be set to their upper boundary.
3.2.9.T10.1-3.F5:	After writing objects and a subsequent reset of the CN by an NMTSwResetEx, NMTResetNodeEx or NMTResetCommunicationEx, all objects in L3 but not in L2, except O1, shall be reset to their default-values.
3.2.9.T10.4.F5:	After writing objects and a subsequent reset of the CN by an NMTResetConfigurationEx, all objects in L3 but not in L2, except O1, shall still be set to their upper boundary.
3.2.9.T8-10.1-4.F6:	Device did not reach state CS_PRE_OPERATIOAL_1 within timeout after a reset.
3.2.9.T10.1-4.F7:	Testcase is not supported since device does not support extended NMT-State commands (Featureflags, Bit 5 is not set).

3.3 Advanced Tests

3.3.1 POWERLINK DLL Errors

3.3.1.1 Loss of frames

3.3.1.1.1 Description

Tests in this chapter shall test the reaction of the DUT on the loss of certain POWERLINK-Frames (Requirement: Dropping of specific frames in a specified sequence). It shall be tested, that:

- a DUT is correctly in-/decrementing its corresponding Threshold-Counter.
- a DUT is correctly executing the defined action once a Threshold-Counter reaches its Threshold, changing its NMT-State to CS_PRE_OPERATIONAL_1 and reporting the correct error-code in its Error-History, if implemented (object 0x1003).
- a DUT resets its Threshold-Counter to 0 once the Threshold has been reached.
- a DUT is not performing any Threshold-Counting if Threshold is set to 0.

Reference: section 4.7.2 of DS 301 v1.1.0.

3.3.1.1.2 Action/Response Sequences

Parameter:

O1 = Index 1C0Bh, Sub-Index 03h

V1 = 15, default threshold for Loss of SoC

O2 = Index 1C0Bh, Sub-Index 02h

V2 = Value of threshold-counter for Loss of SoC

O3 = Index 1C0Bh, Sub-Index 01h

V3 = Value of cumulative-counter for Loss of SoC

Pre-Condition:

TEST 3.2.9.T1

TEST 3.2.9.T2

MN is in MS_OPERATIONAL

DUT is in CS_OPERATIONAL

TEST 3.3.1.1.T1

```
/* Loss of SoC triggered */
{ write object(O1) with value V1 }
on (subObject 0x1003/0x0 exists in XDD)
    { write object 0x1003/0x0 with value 0 } // Clear history
{ write object(O3) with value 0 } // reset cumulative counter
/* start error simulation sequence */
{ drop 1 SoC } // ThresholdCnt == 8
{ send 1 SoC } // ThresholdCnt == 7
{ drop 1 SoC } // ThresholdCnt == 0 (reached V1 and was reset to 0)
/* end error simulation sequence */
on (reported CN-NMT-State != CS_PRE_OPERATIONAL_1)
    <FAIL> 3.3.1.1.T1.F1
/* wait for CN to reach CS_OPERATIONAL after being reset by MN */
{ read object(O2) into V2 }
on (V2 != 0)
    <FAIL> 3.3.1.1.T1.F2
{ read object(O3) into V3 }
on (V3 != 2)
    <FAIL> 3.3.1.1.T1.F3
on (subObject 0x1003/0x0 exists in XDD)
    // ERR_History_ADOM implemented, so error shall be written to the
    // lowest subIndex (indicating most current error).
    { read object 0x1003/0x1 }
    on (SDO abort-code == 0x06090011)
        <FAIL> 3.3.1.1.T1.F4 // no error logged
    on (error-type profile != 0x002) // communication-profile specific
        <FAIL> 3.3.1.1.T1.F5
    on (error-code != E_DLL_LOSS_SOC_TH)
        <FAIL> 3.3.1.1.T1.F6
```

TEST 3.3.1.1.T2

```
/* Loss of SoC not triggered*/
{ write object(O1) with value V1 }
{ write object(O3) with value 0 } // reset cumulative counter
/* start error simulation sequence */
{ drop 1 SoC } // ThresholdCnt == 8
{ send 2 SoC } // ThresholdCnt == 6
{ drop 1 SoC } // ThresholdCnt == 14
{ send 8 SoC } // ThresholdCnt == 6
{ drop 1 SoC } // ThresholdCnt == 14
{ send 8 SoC } // ThresholdCnt == 6
{ drop 1 SoC } // ThresholdCnt == 14
/* end error simulation sequence */
on (reported CN-NMT-State != CS_OPERATIONAL)
    <FAIL> 3.3.1.1.T2.F1
{ read object(O3) into V3 }
on (V3 != 4)
    <FAIL> 3.3.1.1.T1.F2
```

TEST 3.3.1.1.T3

```
/*
```

```
 * Loss of SoA
 * Identical to T1 with following exceptions:
 *
 * - Only executed if object 1C0Ch exists in the XDD
 *   trigger <NOT_SUPPORTED> F7 otherwise
 * - O1 = Index 1C0Ch, Sub-Index 03h
 * - O2 = Index 1C0Ch, Sub-Index 02h
 * - O3 = Index 1C0Ch, Sub-Index 01h
 * - drop/send SoA-Frames instead of SoC
 * - Trigger F6 if error-code != E_DLL_LOSS_SOA_TH
 *
 */
```

TEST 3.3.1.1.T4

```
/*
 * Loss of SoA not triggered
 * Identical to T2 with following exceptions:
 *
 * - Only executed if object 1C0Ch exists in the XDD
 *   trigger <NOT_SUPPORTED> F3 otherwise
 * - O1 = Index 1C0Ch, Sub-Index 03h
 * - O2 = Index 1C0Ch, Sub-Index 02h
 * - O3 = Index 1C0Ch, Sub-Index 01h
 * - drop/send SoA-Frames instead of SoC
 *
 */
```

TEST 3.3.1.1.T5

```
/*
 * Loss of PReq
 * Identical to T1 with following exceptions:
 *
 * - Only executed if object 1C0Dh exists in the XDD
 *   trigger <NOT_SUPPORTED> F7 otherwise
 * - O1 = Index 1C0Dh, Sub-Index 03h
 * - O2 = Index 1C0Dh, Sub-Index 02h
 * - O3 = Index 1C0Dh, Sub-Index 01h
 * - drop/send PReq-Frames instead of SoC
 * - Trigger F6 if reported error-code != E_DLL_LOSS_PREQ_TH
 *
 */
```

TEST 3.3.1.1.T6

```
/*
 * Loss of PReq not triggered
 * Identical to T2 with following exceptions:
 *
 * - Only executed if object 1C0Dh exists in the XDD
 *   trigger <NOT_SUPPORTED> F3 otherwise
 * - O1 = Index 1C0Dh, Sub-Index 03h
 * - O2 = Index 1C0Dh, Sub-Index 02h
 * - O3 = Index 1C0Dh, Sub-Index 01h
 * - drop/send PReq-Frames instead of SoC
 *
 */
```

TEST 3.3.1.1.T7

```
/* Simulate Loss of SoC with deactivated threshold counting. There shall
not be an error-reaction */
/* Set V1 = 0 */
{ write object(O1) with value V1 } // Deactivate threshold counting
on (subObject 0x1003/0x0 exists in XDD)
    { write object 0x1003/0x0 with value 0 } // Clear history
{ write object(O3) with value 0 } // reset cumulative counter
/* start error simulation sequence */
{ drop 1 SoC }
{ drop 1 SoC }
/* end error simulation sequence */
on (reported CN-NMT-State != CS_OPERATIONAL)
    <FAIL> 3.3.1.1.T7.F1
{ read object(O3) into V3 }
on (V3 != 2)
    <FAIL> 3.3.1.1.T7.F2
on (subObject 0x1003/0x0 exists in XDD)
    // ERR_History_ADOM implemented, but since threshold counting is
    // deactivated, there shall not be a corresponding error-entry.
    { read object 0x1003/0x1 }
```

```
on (error-type profile == 0x002 and error-code == E_DLL_LOSS_SOC_TH)
    <FAIL> 3.3.1.1.T1.F3
```

TEST 3.3.1.1.T8

```
/*
 * Simulate Loss of SoA with deactivated threshold counting. There shall
not
 * be an error-reaction.
 * Identical to T7 with following exceptions:
 *
 * - Only executed if object 1C0Ch exists in the XDD
 *   trigger <NOT_SUPPORTED> F4 otherwise
 * - O1 = Index 1C0Ch, Sub-Index 03h
 * - O2 = Index 1C0Ch, Sub-Index 02h
 * - O3 = Index 1C0Ch, Sub-Index 01h
 * - drop/send SoA-Frames instead of SoC
 *
 */
```

TEST 3.3.1.1.T9

```
/*
 * Simulate Loss of PReq with deactivated threshold counting. There shall
 * not be an error-reaction.
 * Identical to T7 with following exceptions:
 *
 * - Only executed if object 1C0Dh exists in the XDD
 *   trigger <NOT_SUPPORTED> F4 otherwise
 * - O1 = Index 1C0Dh, Sub-Index 03h
 * - O2 = Index 1C0Dh, Sub-Index 02h
 * - O3 = Index 1C0Dh, Sub-Index 01h
 * - drop/send PReq-Frames instead of SoC
 *
 */
```

3.3.1.1.3 Functional Requirements

3.3.1.1.T1(3)(5).F1: DUT shall fall back to CS_PRE_OPERATIONAL_1 upon error being triggered.

3.3.1.1.T1(3)(5).F2: Threshold-Counter (subIndex 2 of corresponding error-counter index) shall be reset to 0 upon error being triggered.

3.3.1.1.T1(3)(5).F3: Cumulative-Counter (subIndex 1 of corresponding error-counter index) shall be set to 2.

3.3.1.1.T1(3)(5).F4: If Error-History (0x1003) is supported, there shall be an error logged at subIndex 0x1.

3.3.1.1.T1(3)(5).F5: If Error-History (0x1003) is supported, the logged error at subIndex 0x1 shall have its profile set to 0x2 (communication profile).

3.3.1.1.T1(3)(5).F6: If Error-History (0x1003) is supported, the logged error at subIndex 0x1 shall have the correct error-code.

3.3.1.1.T3(5).F7: Error-Counter 0x1C0C (0x1C0D) not supported by device.

3.3.1.1.T2(4)(6).F1:	DUT shall stay in CS_OPERATIONAL since error shall not have been triggered.
3.3.1.1.T2(4)(6).F2:	Cumulative-Counter (subIndex 1 of corresponding error-counter index) shall be set to 4.
3.3.1.1.T4(6).F3:	Error-Counter 0x1C0C (0x1C0D) not supported by device.
3.3.1.1.T7(8)(9).F1:	DUT shall stay in CS_OPERATIONAL since threshold-counting is disabled.
3.3.1.1.T7(8)(9).F2:	Cumulative-Counter (subIndex 1 of corresponding error-counter index) shall be set to 2.
3.3.1.1.T7(8)(9).F3:	If Error-History (0x1003) is supported, there shall be no error logged at subIndex 0x1 with profile == 0x2 and a matching error-code.
3.3.1.1.T8(9).F4:	Error-Counter 0x1C0C (0x1C0D) not supported by device.

3.3.1.2 Delay of Frames

3.3.1.2.1 Description

Tests in this chapter shall test the reaction of the DUT on delayed reception of certain POWERLINK-Frames (Requirement: Delay of specified frames for a specified amount of time). It shall be tested that:

- a DUT is correctly in-/decrementing its corresponding Threshold-Counter.
- a DUT is correctly executing the defined action once a Threshold-Counter reaches its Threshold, changing its NMT-State to CS_PRE_OPERATIONAL_1 and reporting the correct error-code in its Status-Response.
- a DUT resets its Threshold-Counter to 0 once the Threshold has been reached.
- a DUT is not performing any Threshold-Counting if Threshold is set to 0.

3.3.1.2.2 Action/Response Sequences

Parameter:

O1 = Index 1C0Eh, Sub-Index 03h

V1 = 15, default threshold for SoC Jitter out of Range

O2 = Index 1C0Eh, Sub-Index 02h

V2 = Value of threshold-counter for SoC Jitter out of Range

O3 = Index 1C0Eh, Sub-Index 01h

V3 = Value of cumulative-counter for SoC Jitter out of Range

V4 = Value of 1C13 (SoC Jitter Range) from XDD

Pre-Condition:

TEST 3.2.9.T1

TEST 3.2.9.T2

MN is in MS_OPERATIONAL

DUT is in CS_ OPERATIONAL

TEST 3.3.1.2.T1

```
/* Delay SoC to trigger threshold-counting on the CN */
on not(object 1C13h and 1C0Eh exist in XDD)
    <NOT_SUPPORTED> 3.3.1.2.T1.F1
    exit
{ write object(O1) with value V1 }
```

```
    on no message received
        <FAIL> 3.3.1.2.T1.F2
    on SDO-Abort
        <FAIL> 3.3.1.2.T1.F3
/* start error simulation sequence */
{ delay 1 SoC by (V4 + 100)ns } // ThresholdCnt == 8
{ send 1 SoC } // ThresholdCnt == 7
{ delay 1 SoC by (V4 + 100)ns } // ThresholdCnt == 0 (reached V1 and was
reset to 0)
/* end error simulation sequence */
{ read object(O2) into V2}
    on no message received
        <FAIL> 3.3.1.2.T4.F4
    on SDO-Abort
        <FAIL> 3.3.1.2.T4.F5
    on (V2 != 0)
        <FAIL> 3.3.1.2.T4.F6
{ read object(O3) into V3}
    on no message received
        <FAIL> 3.3.1.2.T4.F4
    on SDO-Abort
        <FAIL> 3.3.1.2.T4.F5
    on (V3 != 2)
        <FAIL> 3.3.1.2.T4.F7
on (reported CN-NMT-State != CS_PRE_OPERATIONAL_1)
    <FAIL> 3.3.1.2.T4.F8
on (reported error-code != E_DLL_JITTER_TH)
    <FAIL> 3.3.1.2.T4.F9
```

3.3.1.2.3 Functional Requirements

3.3.1.2.T1.F1: Test is not supported since device does not implement objects 1C13h and 1C0Eh.

3.3.1.2.T1.F2: Device did not answer on SDO-Write of object 0xXXXX/0xXX within timeout.

3.3.1.2.T1.F3: SDO-Write of object 0xXXXX/0xXX failed with SDO-Abort code.

3.3.1.2.T1.F4: Device did not answer on SDO-Read of object 0xXXXX/0xXX within timeout.

3.3.1.2.T1.F5: SDO-Read of object 0xXXXX/0xXX failed with SDO-Abort code.

3.3.1.2.T1.F6: Threshold-Counter shall be reset to 0 once it reaches Threshold.

3.3.1.2.T1.F7: Cumulative-Counter shall be set to the nr. of times the error-symptom ocurred.

3.3.1.2.T1.F8: CN shall change its NMT-State to CS_PRE_OPERATIONAL_1 upon detection of this error.

3.3.1.2.T1.F9: CN shall report the correct error-code in its Status-Response.

3.3.1.3 Extension Compatibility

3.3.1.3.1 Description

Tests in this chapter shall simulate following real-world scenario: POWERLINK CN's which implement different extensions inter-operate in a single network, i.e. PRes-Chaining enabled nodes and nodes that do not support PRes-Chaining exist in the same network.

To test the correct behaviour of nodes that do not implement the extension, and at the same time prevent erronous behaviour of nodes that do implement it, following approach is chosen:

- Frames shall be sent within which certain header-fields (those affected by the extension to test) are set to unsupported values.
- Both types of CNs (those with and without the extension) should ignore the frame and stay in CS_OPERATIONAL.

3.3.1.3.2 Action/Response Sequences

Parameter:

FRAME_COUNT = 10

Pre-Condition:

MN is in MS_OPERATIONAL

DUT is in CS_OPERATIONAL

TEST 3.3.1.3.T1 // Former 3.3.1.4.T1

```
/*
 * Testing reaction on reception of frames with non-existing
 * POWERLINK message type (e.g. the Multi-ASnd extension defines the new
 * message type 0xD (AInv)).
 */
{ send NMTStopNode with POWERLINK message type 0x7F }
{ wait }(10 cycles)
on (reported CN-NMT-State != CS_OPERATIONAL)
    <FAIL> 3.3.1.3.T1.F1
```

TEST 3.3.1.3.T2 // Former 3.3.1.4.T2

```
/*
 * Testing reaction on reception of frames with non-existing
 * SoA ServiceID (e.g. the PRes-Chaining extension defines the new
 * ServiceID 0x6).
 */
{ send FRAME_COUNT SoA frames with RequestedServiceID 0x9F }
{ wait }(20 cycles)
```

```
on (reported CN-NMT-State != CS_OPERATIONAL)
    <FAIL> 3.3.1.3.T2.F1
```

TEST 3.3.1.3.T3

```
/*
 * Testing reaction on reception of frames with non-existing
 * ASnd ServiceID (e.g. the PRes-Chaining extension defines the new
 * ServiceID 0x6).
 */
{ send NMTStopNode with ServiceID 0x9F }
{ wait }(10 cycles)
on (reported CN-NMT-State != CS_OPERATIONAL)
    <FAIL> 3.3.1.3.T3.F1
```

3.3.1.3.3 Functional Requirements

3.3.1.3.T1.F1: DUT does not support unknown POWERLINK message types (0x7F).

3.3.1.3.T2.F1: DUT does not support unknown SoA RequestedServiceIDs (0x9F).

3.3.1.3.T3.F1: DUT does not support unknown ASnd ServiceIDs (0x9F).

3.3.1.4 MAC / POWERLINK Addressing

3.3.1.4.1 Description

Tests in this chapter shall test whether a DUT processes received POWERLINK-Frames with correct addressing information and ignores frames with non-specification conform addressing information.

Following tests shall be conducted for POWERLINK **ASnd**-Frames:

- T1: a DUT must process frames with Multicast ASnd MAC address and their own POWERLINK node id.
- T2: a DUT must process frames with Multicast ASnd MAC address and POWERLINK broadcast node id (255).
- T3: a DUT must process frames with their own Unicast-MAC address and their own POWERLINK node id.
- T4: a DUT must process frames with Broadcast-MAC address and their own POWERLINK node id.
- T5: a DUT must process frames with Broadcast-MAC address and POWERLINK broadcast node id (255).
- T6: a DUT must **not** process frames with Broadcast-MAC address and a POWERLINK node id other than their own.
- T7: a DUT must **not** process frames with Multicast MAC address and a POWERLINK node id other than their own.
- T8: a DUT must **not** process frames with their own Unicast-MAC address and a POWERLINK node id other than their own.
- T9: a DUT must **not** process frames with an Unicast-MAC address other than their own and a POWERLINK node id other than their own.
- T10: a DUT must **not** process frames with an Unicast-MAC address other than their own and their own POWERLINK node id.

3.3.1.4.2 Action/Response Sequences

Parameter:

NMT_STATE_TOLERANCE = 10 // Use value higher than C_NMT_STATE_TOLERANCE (5) to make sure state changes of slower devices are detected as well

t_1 = NMT_STATE_TOLERANCE x cycleTime // wait time for DUT to change NMT-State

C_DLL_MULTICAST_ASND = 01-11-1E-00-00-04

DLL_BROADCAST = FF-FF-FF-FF-FF-FF

DLL_OWN_UNICAST = DUT's MAC address

DLL_FOREIGN_UNICAST = arbitrary unicast MAC address

C_ADR_BROADCAST = POWERLINK broadcast node id (255)

ADR_OWN_NODE_ID = DUT's POWERLINK node id

ADR_FOREIGN_NODE_ID = arbitrary POWERLINK node id (1-239)

Pre-Condition:

MN is in MS_OPERATIONAL

DUT is in CS_OPERATIONAL

TEST 3.3.1.4.T1

```
{ send ASnd NMTStopNode }(C_DLL_MULTICAST_ASND, ADR_OWN_NODE_ID)
{ repeat }
    { send SoA RequestedServiceID STATUS_REQUEST }(ADR_OWN_NODE_ID)
{ until ((msg received && reported state == CS_STOPPED) || (t1 elapsed)) }
on (reported NMT-State != CS_STOPPED)
    <FAIL> 3.3.1.4.T1.F1
```

TEST 3.3.1.4.T2

```
// Identical to T1 but using C_DLL_MULTICAST_ASND and C_ADR_BROADCAST
```

TEST 3.3.1.4.T3

```
// Identical to T1 but using DLL_OWN_UNICAST and ADR_OWN_NODE_ID
```

TEST 3.3.1.4.T4

```
{ send ASnd NMTStopNode }(DLL_BROADCAST, ADR_OWN_NODE_ID)
{ repeat }
    { send SoA RequestedServiceID STATUS_REQUEST }(ADR_OWN_NODE_ID)
{ until ((msg received && reported state == CS_STOPPED) || (t1 elapsed)) }
on (reported NMT-State == CS_STOPPED)
    <FAIL> 3.3.1.4.T4.F1
on (reported NMT-State != CS_OPERATIONAL)
    <FAIL> 3.3.1.4.T4.F2
```

TEST 3.3.1.4.T5

```
// Identical to T4 but using DLL_BROADCAST and C_ADR_BROADCAST
```

TEST 3.3.1.4.T6

```
// Identical to T4 but using DLL_BROADCAST and ADR_FOREIGN_NODE_ID
```

TEST 3.3.1.4.T7

```
// Identical to T4 but using C_DLL_MULTICAST_ASND and ADR_FOREIGN_NODE_ID
```

TEST 3.3.1.4.T8

```
// Identical to T4 but using DLL_OWN_UNICAST and ADR_FOREIGN_NODE_ID
```

TEST 3.3.1.4.T9

```
// Identical to T4 but using DLL_FOREIGN_UNICAST and ADR_FOREIGN_NODE_ID
```

TEST 3.3.1.4.T10

```
// Identical to T4 but using DLL_FOREIGN_UNICAST and ADR_OWN_NODE_ID
```

3.3.1.4.3 Functional Requirements

3.3.1.4.T1-3.F1: DUT did not change its NMT-State to CS_STOPPED upon reception of an NMTStopNode command. Correctly addressed ASnd-Frame was not interpreted.

3.3.1.4.T4-10.F1: DUT changed its NMT-State to CS_STOPPED upon reception of an NMTStopNode command that was sent in an incorrectly addressed ASnd-Frame.

3.3.1.4.T4-10.F2: DUT did neither stay in CS_OPERATIONAL, nor change to CS_STOPPED upon reception of an NMTStopNode command sent in an incorrectly addressed ASnd-Frame.

3.3.2 POWERLINK AL Errors

3.3.2.1 Mapping Errors

3.3.2.1.1 Description

Tests in this chapter shall test the reaction of a DUT on various errors related to the mapping of PDOs. There errors are:

- T1: A TPDO mapping is activated (by writing its corresponding NumberOfEntries object, i.e. 0x1A00/0x00) whose memory consumption exceeds the configured actual payload size limit (0x1F98/0x05).
- T2: A TPDO mapping is activated (by writing its corresponding NumberOfEntries object, i.e. 0x1A00/0x00) whose memory consumption exceeds the configured max. payload size limit (0x1F98/0x01)
- T3: An RPDO mapping is activated (by writing its corresponding NumberOfEntries object, e.g. 0x1600/0x00) whose memory consumption exceeds the configured actual payload size limit (0x1F98/0x04).
- T4: An RPDO mapping is activated (by writing its corresponding NumberOfEntries object, i.e. 0x1600/0x00) whose memory consumption exceeds the configured max. payload size limit (0x1F98/0x02)

3.3.2.1.2 Action/Response Sequences

Parameter:

O_1 = 0x1A00/0x00

O_2 = 0x1F98/0x05

O_3 = 0x1F98/0x01

O_4 = 0x1600/0x00

O_5 = 0x1F98/0x04

O_6 = 0x1F98/0x02

MAPPED_OBJECTS = No. of newly mapped TPDOs or RPDOs

Pre-Condition:

MN is in MS_OPERATIONAL

DUT is in CS_OPERATIONAL

TEST 3.3.2.1.T1

```
/*
 * Testing reaction on activation of a TPDO-Mapping that exceeds the
 * configured  payload size limit 0x1F98/0x5.
 */
on !(dynamic PDO mapping)
    <NOT_SUPPORTED> 3.3.2.1.T1.F1
    exit
on (size(TPDO-Mappable objects) <= 36bytes)
    <NOT_SUPPORTED> 3.3.2.1.T1.F2 // impossible to exceed limit
    exit
on (O3 == 36)
    <NOT_SUPPORTED> 3.3.2.1.T1.F3 // Activating a TPDO-Mapping > 36 bytes
    exit                          // would always trigger SDO-Abort
                                  // E_PDO_MAP_OVERRUN
{SDO Write-By-Index}(O1, 0) // Deactivate current (default?) TPDO mapping
                            // on DUT
on no message received
    <FAIL> 3.3.2.1.T1.F4
    exit
on SDO-Abort
    <FAIL> 3.3.2.1.T1.F5
    exit
{SDO Write-By-Index}(O2, 36) // Set PResActPayloadLimit to 36 bytes
on no message received
    <FAIL> 3.3.2.1.T1.F6
    exit
on SDO-Abort
    <FAIL> 3.3.2.1.T1.F7
    exit
{send NMTResetConfiguration}(nodeId) // Set 0x1F98/0x5 valid
{transition DUT to CS_OPERATIONAL}
{map TPDO-Mappable objects > 36 bytes on DUT} // SDO-Write to 0x1800/0xX
{SDO Write-By-Index}(O1, MAPPED_OBJECTS) // Activate new (invalid) mapping
on no message received
    <FAIL> 3.3.2.1.T1.F8
    exit
on (success && RD-Flag != 0) // EPSG DS301, v1.2.0
    <FAIL> 3.3.2.1.T1.F9
    exit
on SDO-Abort code != 0x06040042 // E_PDO_MAP_OVERRUN, EPSG DS301, v1.1.0
    <FAIL> 3.3.2.1.T1.F10
    exit
on SDO-Abort code == 0x06040042 // E_PDO_MAP_OVERRUN, EPSG DS301, v1.1.0
    <WARN> 3.3.2.1.T1.F11
```

TEST 3.3.2.1.T2

```
/*
 * Testing reaction on activation of a TPDO-Mapping that exceeds the
 * payload size limit 0x1F98/0x1.
 */
on !(dynamic PDO mapping)
    <NOT_SUPPORTED> 3.3.2.1.T2.F1
    exit
```

```
on (size(TPDO-Mappable objects) < O3 bytes)
    <NOT_SUPPORTED> 3.3.2.1.T2.F2 // Impossible to exceed limit
    exit
{SDO Write-By-Index}(O1, 0) // Deactivate current (default?) TPDO mapping
                            //on DUT
on no message received
    <FAIL> 3.3.2.1.T2.F3
    exit
on SDO-Abort
    <FAIL> 3.3.2.1.T2.F4
    exit
{map TPDO-Mappable objects > O3 bytes on DUT} // SDO-Write to 0x1800/0xX
{SDO Write-By-Index}(O1, MAPPED_OBJECTS) // Activate new (invalid) mapping
on no message received
    <FAIL> 3.3.2.1.T2.F5
    exit
on success
    <FAIL> 3.3.2.1.T2.F6
    exit
on SDO-Abort code != 0x06040042 // E_PDO_MAP_OVERRUN
    <FAIL> 3.3.2.1.T2.F7
```

TEST 3.3.2.1.T3

```
/*
 * Testing reaction on activation of an RPDO-Mapping that exceeds the
 * configured  payload size limit 0x1F98/0x4.
 */
on !(dynamic PDO mapping)
    <NOT_SUPPORTED> 3.3.2.1.T3.F1
    exit
on (size(RPDO-Mappable objects) <= 36bytes)
    <NOT_SUPPORTED> 3.3.2.1.T3.F2 // Impossible to exceed limit
    exit
{Local-Write-By-Index}(0x1AXX/0x00, 0) // Deactivate TPDO-Mapping for DUT
                                       // on MN side
{Local-Write-By-Index}(0x1F8D/0x<nodeId>, 36) // Set PReq frame-size to
                                              // minimum
{SDO Write-By-Index}(O4, 0) // Deactivate current (default?) RPDO mapping
                            // on DUT
on no message received
    <FAIL> 3.3.2.1.T3.F3
    exit
on SDO-Abort
    <FAIL> 3.3.2.1.T3.F4
    exit
{SDO Write-By-Index}(O5, 36) // Set PReqActPayloadLimit to 36 bytes
on no message received
    <FAIL> 3.3.2.1.T3.F5
    exit
on SDO-Abort
    <FAIL> 3.3.2.1.T3.F6
    exit
{send NMTResetConfiguration}(nodeId) // Set 0x1F98/0x4 valid
{transition DUT to CS_OPERATIONAL}
{map RPDO-Mappable objects > 36 bytes on DUT} // SDO-Write to 0x1600/0xXX
{SDO Write-By-Index}(O4, MAPPED_OBJECTS) // Activate new (invalid) mapping
on no message received
    <FAIL> 3.3.2.1.T3.F7
    Exit
on success
    <FAIL> 3.3.2.1.T3.F8
    exit
on SDO-Abort code != 0x06040042 // E_PDO_MAP_OVERRUN, EPSG DS301, v1.1.0
    <FAIL> 3.3.2.1.T3.F9
```

TEST 3.3.2.1.T4

```
/*
 * Testing reaction on activation of an RPDO-Mapping that exceeds the
 * payload size limit 0x1F98/0x2.
 */
on !(dynamic PDO mapping)
    <NOT_SUPPORTED> 3.3.2.1.T4.F1
    exit
on (size(RPDO-Mappable objects) <= O6 bytes)
```

```
    <NOT_SUPPORTED> 3.3.2.1.T4.F2 // Impossible to exceed limit
    exit
{SDO Write-By-Index}(O4, 0) // Deactivate current (default?) RPDO mapping
                            // on DUT
on no message received
    <FAIL> 3.3.2.1.T4.F3
    exit
on SDO-Abort
    <FAIL> 3.3.2.1.T4.F4
    exit
{map RPDO-Mappable objects > O6 bytes on DUT} // SDO-Write to 0x1600/0xXX
{SDO Write-By-Index}(O4, MAPPED_OBJECTS) // Activate new (invalid) mapping
on no message received
    <FAIL> 3.3.2.1.T4.F5
    exit
on success
    <FAIL> 3.3.2.1.T4.F6
    exit
on SDO-Abort code != 0x06040042 // E_PDO_MAP_OVERRUN, EPSG DS301, v1.1.0
    <FAIL> 3.3.2.1.T4.F7
```

3.3.2.1.3 Functional Requirements

3.3.2.1.T1.F1: Test not supported, because DUT does not support dynamic mapping.

3.3.2.1.T1.F2: Test not supported, because DUT does not provide sufficient objects which are TPDO-Mappable (0x1F98/0x05 has a minimum value of 36 bytes).

3.3.2.1.T1.F3: Test not supported, because 0x1F98/0x01 == 36 and activating a TPDO-Mapping > 36 bytes would always trigger SDO-Abort E_PDO_MAP_OVERRUN.

3.3.2.1.T1.F4: DUT did not answer on SDO-Write to 0x1A00/0x00. Deactivating TPDO-Mapping failed.

3.3.2.1.T1.F5: SDO-Abort on SDO-Write to 0x1A00/0x00. Deactivating TPDO-Mapping failed.

3.3.2.1.T1.F6: DUT did not answer on SDO-Write to 0x1F98/0x05. Setting PResActPayloadLimit failed.

3.3.2.1.T1.F7: SDO-Abort on SDO-Write to 0x1F98/0x05. Setting PResActPayloadLimit failed.

3.3.2.1.T1.F8: DUT did not answer on SDO-Write to 0x1A00/0x00. Activating new (invalid) TPDO-Mapping failed.

3.3.2.1.T1.F9: DUT accepted new (invalid) TPDO-Mapping but did not reset the RD-Flag in its Poll-Response to 0.

3.3.2.1.T1.F10: Invalid SDO-Abort code on SDO-Write to 0x1A00/0x00. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.1.T1.F11: SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN) on SDO-Write to 0x1A00/0x00. Accepted due to backwards compatibility reasons. Level WARN will turn to FAIL in future versions.

3.3.2.1.T2.F1: see 3.3.2.1.T1.F1

3.3.2.1.T2.F2: Test not supported, because DUT does not provide sufficient objects which are TPDO-Mappable to exceed 0x1F98/0x01.

3.3.2.1.T2.F3: DUT did not answer on SDO-Write to 0x1A00/0x00. Deactivating TPDO-Mapping failed.

3.3.2.1.T2.F4: SDO-Abort on SDO-Write to 0x1A00/0x00. Deactivating TPDO-Mapping failed.

3.3.2.1.T2.F5: DUT did not answer on SDO-Write to 0x1A00/0x00. Activating new (invalid) TPDO-Mapping failed.

3.3.2.1.T2.F6: DUT accepted new (invalid) TPDO-Mapping. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.1.T2.F7: Invalid SDO-Abort code on SDO-Write to 0x1A00/0x00. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.1.T3.F1: see 3.3.2.1.T1.F1

3.3.2.1.T3.F2: Test not supported, because DUT does not provide sufficient objects which are RPDO-Mappable (0x1F98/0x04 has a minimum value of 36 bytes).

3.3.2.1.T3.F3: DUT did not answer on SDO-Write to 0x1600/0x00. Deactivating RPDO-Mapping failed.

3.3.2.1.T3.F4: SDO-Abort on SDO-Write to 0x1600/0x00. Deactivating RPDO-Mapping failed.

3.3.2.1.T3.F5: DUT did not answer on SDO-Write to 0x1F98/0x04. Setting PReqActPayloadLimit failed.

3.3.2.1.T3.F6: SDO-Abort on SDO-Write to 0x1F98/0x04. Setting PReqActPayloadLimit failed.

3.3.2.1.T3.F7: DUT did not answer on SDO-Write to 0x1600/0x00. Activating new (invalid) RPDO-Mapping failed.

3.3.2.1.T3.F8: DUT accepted new (invalid) RPDO-Mapping. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.1.T3.F9: Invalid SDO-Abort code on SDO-Write to 0x1600/0x00. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.1.T4.F1: see 3.3.2.1.T1.F1

3.3.2.1.T4.F2: Test not supported, because DUT does not provide sufficient objects which are RPDO-Mappable to exceed 0x1F98/0x02.

3.3.2.1.T4.F3: DUT did not answer on SDO-Write to 0x1600/0x00. Deactivating RPDO-Mapping failed.

3.3.2.1.T4.F4: SDO-Abort on SDO-Write to 0x1600/0x00. Deactivating RPDO-Mapping failed.

3.3.2.1.T4.F5: DUT did not answer on SDO-Write to 0x1600/0x00. Activating new (invalid) RPDO-Mapping failed.

3.3.2.1.T4.F6: DUT accepted new (invalid) RPDO-Mapping. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.1.T4.F7: Invalid SDO-Abort code on SDO-Write to 0x1600/0x00. Expected SDO-Abort code 0x06040042 (E_PDO_MAP_OVERRUN).

3.3.2.2 SDO Sequence-Layer

3.3.2.2.1 Description

Tests in this chapter shall test the correct functioning of the SDO sequence layer.

- T1: Tests the correct implementation of the broken connection detection.

3.3.2.2.2 Action/Response Sequences

Parameter:

O_1 = 0x1300

O_2 = 0x1302

CYCLE_TIME = current POWERLINK cycle time in µs

ASYNC_SLOT_CONTENTION = 5 (DUT will be assigned an async. slot at least every 5th cycle)

SDO_SEQ_ACK_REQUESTS = value of O_2 if existing, 2 otherwise

SDO_SEQ_TIMEOUT = SDO_SEQ_ACK_REQUESTS x ASYNC_SLOT_CONTENTION x CYCLE_TIME [µs]

Pre-Condition:

MN is in MS_OPERATIONAL

DUT is in CS_OPERATIONAL

TEST 3.3.2.2.T1

```
/*
 * Test correct implementation of broken connection detection
 */
{SDO Write-By-Index}(O1, SDO_SEQ_TIMEOUT)
```

```
on no message received
    <FAIL> 3.3.2.2.T1.F1
    exit
on SDO-Abort
    <FAIL> 3.3.2.2.T1.F2
    exit
{send NMTResetConfiguration}(nodeId) // Set 0x1300 valid
{transition DUT to CS_OPERATIONAL}
{SDO Read-By-Index}(O1)
{receive SDO command-layer response}
{start timer}(SDO_SEQ_TIMEOUT) // don't send ACK to DUT
{while !timer_expired}
    {receive SDO command-layer frame} // ssnr, rsnr=?, rcon=2, scon=3
on (no. of received ack requests < SDO_SEQ_ACK_REQUESTS)
    <FAIL> 3.3.2.2.T1.F3
```

3.3.2.2.3 Functional Requirements

3.3.2.2.T1.F1: DUT did not answer on SDO-Write to 0x1300.

3.3.2.2.T1.F2: SDO-Abort on SDO-Write to 0x1300.

3.3.2.2.T1.F3: No. of received acknowledge requests during sequence layer timeout < SDO_SEQ_ACK_REQUESTS.

3.4 Configuration Tests

3.4.1 PDO

3.4.1.1 General

Tests in this section verify the mapping of objects to PDOs in both directions (from and to the DUT).

3.4.1.2 PDO1

3.4.1.2.1 Description

An TPDO-mappable object on the DUT shall be mapped and evaluated on the MN. If possible, the object should be manually changeable on the DUT (e.g. ActualPosition on an encoder).

If no TPDO-mappable object is available, this test is not applicable.

3.4.1.2.2 Action/Response Sequences

Parameter:

O_1 = TPDO-mappable object on DUT, manually changeable if possible

Pre-Condition:

MN is in MS_OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.1.T1

```
{map O1}
{verify that changes to O1 on the DUT are reflected on the MN}
on not verified
    <FAIL> 3.4.1.T1.F1
```

3.4.1.2.3 Functional Requirements

3.4.1.T1.F1: Changes to objects mapped as TPDO on the DUT shall be reflected on the MN.

3.4.1.3 PDO2

3.4.1.3.1 Description

An RPDO-mappable object on the DUT shall be mapped and evaluated on the DUT. The object should trigger a visible action on the DUT if possible (e.g. DigitialOutput indicated by an LED).

If no RPDO-mappable object is available, this test is not applicable.

3.4.1.3.2 Action/Response Sequences

Parameter:

O_1 = RPDO-mappable object on DUT, changes triggering visible action if possible

Pre-Condition:

MN is in MS_OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.1.T2

```
{map O1}
{verify that changes to O1 on the MN are reflected on the DUT}
on not verified
    <FAIL> 3.4.1.T2.F1
```

3.4.1.3.3 Functional Requirements

3.4.1.T2.F1: Changes to objects mapped as RPDO on the DUT shall be reflected on the DUT.

3.4.2 Multiplex

3.4.2.1 General

Tests in this section verify the Multiplexing feature on a DUT. If a DUT does not support Multiplexing this test is not applicable.

3.4.2.2 MUX1

3.4.2.2.1 Description

Test whether a DUT will work correctly with different multiplexed cycle lengths and assigned multiplexed cycles.

3.4.2.2.2 Action/Response Sequences

Parameter:

O_1 = Multiplexed cycle length object 0x1F98/0x7

O_2 = Assigned multiplexed cycle 0x1F9B/0x<nodeId>

V_1 = First multiplexed cycle length (e.g. 8)

V_2 = Second multiplexed cycle length (e.g. 15)

Pre-Condition:

MN is in MS_OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.2.T1

```
{write V1 to O1 on MN and DUT}
{write 1; V1/2; V1 to O2 on MN and DUT}
{verify correct behaviour of DUT, responding to PReq sent in assigned
cycle}
on not verified
    <FAIL> 3.4.2.T1.F1
{repeat test with O1 = V2 and O2 = 1; V2/2; V2}
```

3.4.2.2.3 Functional Requirements

3.4.2.T1.F1: DUT shall support different multiplexed cycle lengths and different assigned multiplexed cycles.

3.4.3 Cycle Time

3.4.3.1 General

Tests in this section verify the reduced, minimum and maximum cycle time of a DUT.

3.4.3.2 CycleTime1

3.4.3.2.1 Description

Verify the reduced cycle time of a DUT.

3.4.3.2.2 Action/Response Sequences

Parameter:

O_1 = Minimum reduced cycle time object 0x1F8A/0x4

V_1 = Minimum reduced cycle time supported by the DUT in µs

(device description entry D_NMT_MinRedCycleTime_U32, inter frame gap if not available)

Pre-Condition:

MN is in MS_PRE_OPERATIONAL_1

CN is in CS_PRE_OPERATIONAL_1

TEST 3.4.3.T1

```
{write V1 to O1 on MN}
{verify that DUT boots up to CS_PRE_OPERATIONAL_2 or higher}
on not verified
    <FAIL> 3.4.3.T1.F1
```

3.4.3.2.3 Functional Requirements

3.4.3.T1.F1: DUT shall boot up to CS_PRE_OPERATIONAL_2 or higher using the supported minimum reduced cycle time.

3.4.3.3 CycleTime2

3.4.3.3.1 Description

Verify the minimum cycle time of a DUT.

3.4.3.3.2 Action/Response Sequences

Parameter:

O_1 = Cycle time object 0x1006

V_1 = Minimum cycle time supported by the DUT in µs (D_NMT_CycleTimeMin_U32)

Pre-Condition:

MN is in MS_PRE_OPERATIONAL_1

CN is in CS_PRE_OPERATIONAL_1

TEST 3.4.3.T2

```
{write V1 to O1 on MN and CN}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.3.T2.F1
```

3.4.3.3.3 Functional Requirements

3.4.3.T2.F1: DUT shall boot up to CS_OPERATIONAL and stay operational for at least 1000 cycles using the supported minimum cycle time.

3.4.3.4 CycleTime3

3.4.3.4.1 Description

Verify the maximum cycle time of a DUT.

3.4.3.4.2 Action/Response Sequences

Parameter:

O_1 = Cycle time object 0x1006

V_1 = Maximum cycle time supported by the DUT in µs (D_NMT_CycleTimeMax_U32)

Pre-Condition:

MN is in MS_PRE_OPERATIONAL_1

CN is in CS_PRE_OPERATIONAL_1

TEST 3.4.3.T3

```
{write V1 to O1 on MN and CN}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.3.T3.F1
```

3.4.3.4.3 Functional Requirements

3.4.3.T3.F1: DUT shall boot up to CS_OPERATIONAL and stay operational for at least 1000 cycles using the supported maximum cycle time.

3.4.4 Data Load

3.4.4.1 General

Tests in this section verify the minimum and maximum possible load of isosynchronous data mapped to a PDO. If a DUT only supports static mapping, DataLoad1 and DataLoad2 are not applicable.

3.4.4.2 DataLoad1

3.4.4.2.1 Description

Verify correct behaviour of a DUT if no isochronous data is mapped.

3.4.4.2.2 Action/Response Sequences

TEST 3.4.4.T1

```
{write 0 to object 0x1A00/0x0 on CN} // deactivate TPDO mapping
{write 0 to objects 0x16xx/0x0 on CN} // deactivate RPDO mapping
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.4.T1.F1
```

3.4.4.2.3 Functional Requirements

3.4.4.T1.F1: DUT shall boot up into CS_OPERATIONAL and stay operational for at least 1000 cycles with deactivated TPDO and RPDO mapping.

3.4.4.3 DataLoad2

3.4.4.3.1 Description

Verify correct behaviour of a DUT if the minimum amount of isochronous data is mapped in both directions.

3.4.4.3.2 Action/Response Sequences

TEST 3.4.4.T2

```
{map the smallest RPDO-mappable object}
{map the smallest TPDO-mappable object}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles }
on not verified
    <FAIL> 3.4.4.T2.F1
```

3.4.4.3.3 Functional Requirements

3.4.4.T2.F1: DUT shall boot up into CS_OPERATIONAL and stay operational for at least 1000 cycles with the minimum TPDO and RPDO mapping.

3.4.4.4 DataLoad3

3.4.4.4.1 Description

Verify correct behaviour of a DUT if the maximum amount of isochronous data is mapped in both directions.

3.4.4.4.2 Action/Response Sequences

Parameter:

O_1 = IsochronousTxMaxPayload, 0x1F98/0x1

O_2 = IsochronousRxMaxPayload, 0x1F98/0x2

TEST 3.4.4.T3

```
{map the max. no. of RPDO-mappable objects} // up to value of O2 bytes
{map the max. no. of TPDO-mappable objects} // up to value of O1 bytes
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.4.T3.F1
```

3.4.4.4.3 Functional Requirements

3.4.4.T3.F1: DUT shall boot up into CS_OPERATIONAL and stay operational for at least 1000 cycles with the maximum TPDO and RPDO mapping.

3.4.5 Cross Traffic

3.4.5.1 General

Tests in this section verify the ability of the DUT to read PollResponse frames of one or several other CNs in the network.

If D_PDO_RPDOChannels_U16 is set to 1 on the DUT, none of the tests in this chapter is applicable. If it is set to 2, only CrossTraffic1 and CrossTraffic2 are applicable. For all higher values also CrossTraffic3 is applicable.

3.4.5.2 CrossTraffic1

3.4.5.2.1 Description

Verify whether a DUT correctly receives and processes PReq- and PRes-Frames from the MN.

3.4.5.2.2 Action/Response Sequences

Parameter:

O_1 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

O_2 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.5.T1

```
{map O1 to be received by the DUT via PReq}
{map O2 to be received by the DUT via PResMN}
{modify values for O1 and O2 on the MN and verify the reaction on the DUT}
on not verified
    <FAIL> 3.4.5.T1.F1
```

3.4.5.2.3 Functional Requirements

3.4.5.T1.F1: DUT shall support receiving and processing PReq- and PRes-Frames from the MN.

3.4.5.3 CrossTraffic2

3.4.5.3.1 Description

Verify whether a DUT correctly receives and processes PReq-Frames from the MN and PRes-Frames from another CN.

3.4.5.3.2 Action/Response Sequences

Parameter:

O_1 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

O_2 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.5.T2

```
{map O1 to be received by the DUT via PReq}
{map O2 to be received by the DUT via PRes from another CN}
{modify values for O1 on the MN and O2 on the other CN and verify the
reaction on the DUT}
on not verified
    <FAIL> 3.4.5.T2.F1
```

3.4.5.3.3 Functional Requirements

3.4.5.T2.F1: DUT shall support receiving and processing PReq-Frames from the MN and PRes-Frames from another CN.

3.4.5.4 CrossTraffic3

3.4.5.4.1 Description

Verify whether a DUT correctly receives and processes PReq- and PRes-Frames from the MN and PRes-Frames from another CN.

3.4.5.4.2 Action/Response Sequences

Parameter:

O_1 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

O_2 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

O_3 = RPDO-mappable object on the DUT which will trigger a visible action if possible (e.g. DigitialOutput indicated by LED).

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.5.T3

```
{map O1 to be received by the DUT via PReq}
{map O2 to be received by the DUT via PRes from the MN}
{map O3 to be received by the DUT via PRes from another CN}
{modify values for O1 and O2 on the MN and O3 on the other CN and verify the
reaction on the DUT}
on not verified
    <FAIL> 3.4.5.T3.F1
```

3.4.5.4.3 Functional Requirements

3.4.5.T3.F1: DUT shall support receiving and processing PReq- and PRes-Frames from the MN and PRes-Frames from another CN.

3.4.6 Advanced Extension Compatibility

3.4.6.1 General

Tests in this section verify that a DUT is able to cope with existing and possible future extensions of the POWERLINK protocol, even if it does not implement them itself (e.g. a device that does not support PollResponse Chaining shall still operate in a network with nodes that support and use PollResponse Chaining).

3.4.6.2 ExtensionCompatibility1

3.4.6.2.1 Description

Verify that a DUT that does not support PollResponse Chaining (EPSG DS 302-C) will work in a network with POWERLINK nodes that support and use this extension.

This test ensures that a DUT that does not support PollResponse Chaining ignores unknown SoA- and ASnd-ServiceIDs which are used by PollResponse Chaining (SyncReq- and SyncRes-Frames, service id 0x6).

Test-Setup: PollResponse Chaining MN, DUT and 3 PollResponse Chaining CNs.

3.4.6.2.2 Action/Response Sequences

Parameter:

O_1 = RPDO-mappable object on the DUT

O_2 = RPDO-mappable object on chained node CN1

O_3 = RPDO-mappable object on chained node CN2

O_4 = RPDO-mappable object on chained node CN3

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.6.T1

```
{map O1 to be received by the DUT via PReq}
{configure CN1-3 as chained stations}
{map O2 to be received by CN1 via PRes from the MN}
{map O3 to be received by CN2 via PRes from the MN}
{map O4 to be received by CN3 via PRes from the MN}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.6.T1.F1
```

3.4.6.2.3 Functional Requirements

3.4.6.T1.F1: a non PollResponse Chaining DUT shall interoperate in a network with PollResponse Chaining nodes.

3.4.6.3 ExtensionCompatibility2

3.4.6.3.1 Description

Verify that a DUT that does not support Multi-ASnd (EPSG DS 302-B) will work in a network with POWERLINK nodes that support and use this extension.

This test ensures that a DUT that does not support Multi-ASnd:

- ignores unknown POWERLINK message types which are used by Multi-ASnd (AInv-Frames, message type 0xD).
- can cope with more than 1 ASnd-Frame in the async. phase at low cycletimes.

Test-Setup: Multi-ASnd MN, DUT and 1 Multi-ASnd CNs.

3.4.6.3.2 Action/Response Sequences

Parameter:

O_1 = ASndMaxNumber, 0x1F8A/0x3 (max. no. of ASnd-Frames per async. phase)

V_1 = 2

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.6.T2

```
{configure lowest possible cycle time}
{write V1 to O1 on MN}
{ensure usage of at least 2 asynchronous slots per cycle}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.6.T2.F1
```

3.4.6.3.3 Functional Requirements

3.4.6.T2.F1: a non Multi-ASnd DUT shall interoperate in a network with Multi-ASnd nodes with at least 2 asynchronous slots being used per cycle.

3.4.7 Timeout

3.4.7.1 General

Tests in this section verify that a DUT is able to operate within the timing parameters given in its XDD file.

3.4.7.2 Timeout1

3.4.7.2.1 Description

Verify that a DUT responds to a PReq within the PRes response time given in the XDD file (object 0x1F98/0x3) at the lowest possible cycle time (feature D_NMT_CycleTime Min_U32).

3.4.7.2.2 Action/Response Sequences

Parameter:

O_1 = PResMaxLatency, 0x1F98/0x3 (PRes response time), CN

O_2 = NMT_MNCNPresTimeout, 0x1F92/0x<nodeId> (PRes timout), MN

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.7.T1

```
{configure lowest possible cycle time}
{write value of O1 to O2}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.7.T1.F1
```

3.4.7.2.3 Functional Requirements

3.4.7.T1.F1: DUT shall respond to PReq within the time given in 0x1F98/0x3.

3.4.7.3 Timeout2

3.4.7.3.1 Description

Verify that a DUT responds to an SoA within the SoA response time given in the XDD (object 0f1F98/0x6) file at the lowest possible cycle time (device description entry D_NMT_CycleTime Min_U32).

3.4.7.3.2 Action/Response Sequences

Parameter:

O_1 = ASndMaxLatency, 0x1F98/0x6 (SoA response time), CN

O_2 = AsyncSlotTimeout, 0x1F8A/0x2 (SoA timout), MN

Pre-Condition:

MN is in MS_ OPERATIONAL

CN is in CS_OPERATIONAL

TEST 3.4.7.T2

```
{configure lowest possible cycle time}
{write value of O1 to O2}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.7.T2.F1
```

3.4.7.3.3 Functional Requirements

3.4.7.T2.F1: DUT shall respond to SoA within the time given in 0x1F98/0x6.

3.4.7.4 Timeout3

3.4.7.4.1 Description

Verify that a DUT receives and processes a PReq being sent after WaitSoCPReq has elapsed (device description entry D_NMT_CNSoC2PReq_U32) at the lowest possible cycle time (device description entry D_NMT_CycleTime Min_U32).

3.4.7.4.2 Action/Response Sequences

Parameter:

O_1 = WaitSoCPReq, 0x1F8A/0x1 (wait time between SoC and first PReq), MN

V_1 = max(D_NMT_MNSoC2PReq_U32, D_NMT_CNSoC2PReq_U32)

TEST 3.4.7.T3

```
{configure lowest possible cycle time}
{write V1 to O1}
{verify that DUT boots up to CS_OPERATIONAL and stays operational for at
least 1000 cycles}
on not verified
    <FAIL> 3.4.7.T3.F1
```

3.4.7.4.3 Functional Requirements

3.4.7.T3.F1: DUT shall receive and process (and respond to) a PReq being sent after WaitSoCPReq has elapsed.