INTERNATIONAL STANDARD

ISO 14229-8

First edition
2020-02

Road vehicles — Unified diagnostic services (UDS) —

Part 8: UDS on Clock eXtension Peripheral Interface (UDSonCXPI)

Véhicules routiers — Services de diagnostic unifiés (SDU) —

Partie 8: Partie 8: SDU sur l'interface périphérique d'extension d'horloge (UDSonCXPI)

Reference number
ISO 14229-8:2020(E)

Contents

Page

Foreword ..... **v**

Introduction ..... **vi**

1 Scope **1**

2 Normative references **1**

3 Terms and definitions **1**

4 Abbreviated terms **2**

5 Conventions **2**

6 SIP – Service interface parameters **2**

6.1 SIP – General 2
6.2 SIP — Data type definitions 2
6.3 SIP — A_Mtype, message type 3
6.4 SIP — A_TAtype, target address type 3
6.5 SIP — A_TA, target address 3
6.6 SIP — A_SA, source address 3
6.7 SIP — A_Length, length of A_PDU 4
6.8 SIP — A_Data, protocol data unit 4
6.9 SIP — A_SCT, sequence count 4
6.10 SIP — A_Result, result 4
6.11 SIP — ev_wakeup_ind, event wake-up indication (optional) 4
6.12 SIP — cmd_wakeup_req, command wake-up request 5
6.13 SIP — NMInfo, network management information 5

7 APP – Application **5**

7.1 APP – General 5
7.2 APP – Definition of diagnostic classes 6
7.2.1 APP – Overview 6
7.2.2 APP – Diagnostic class I 6
7.2.3 APP – Diagnostic class II 6
7.2.4 APP – Diagnostic class III 6
7.3 APP – CXPI master node requirements – Master node fault management, sensor reading, I/O control 7
7.4 APP – CXPI slave node requirements 7
7.4.1 APP – General 7
7.4.2 APP – Error indications 7
7.5 APP – CXPI measurement and control data diagnostics 7
7.5.1 APP – Master handling of slave failure status measurement and control data 7
7.5.2 APP – Slave node current failure status support 7
7.6 APP – Network management (optional) 8
7.7 APP – CXPI master node gateway application 8
7.7.1 APP – General 8
7.7.2 APP – CXPI master gateway number of subnets 8
7.7.3 APP – CXPI master gateway address routing table 8
7.7.4 APP – CXPI master gateway all nodes request message handling 9
7.7.5 APP – Round trip of all node addressing with functional NAD 9
7.7.6 APP – Round trip of all node addressing with node-specific NADs 10

8 AL – Application layer **11**

8.1 AL – Client to CXPI slave node(s) communication 11
8.2 AL – Overview of UDSonCXPI services and applicability to diagnostic classes 11
8.3 AL – CommunicationControl (28_{16}) service 12
8.4 AL – UDSonCXPI services 13
8.4.1 AL – Supported functions 13
8.4.2 AL – Master node receive buffer length 14

8.4.3 AL – Message length is exceeded 14
8.5 AL – Protocol 14
8.6 AL – Timing 14
8.6.1 AL – General 14
8.6.2 AL – Timing parameter values 14
8.6.3 AL – Server timing performance requirements 14
8.6.4 AL – SuppressPosRspMsgIndicationBit 15
8.7 AL – Response pending 15
8.8 AL – CXPI slave node configuration services 16
8.8.1 AL – CXPI node configuration 16
8.8.2 AL – Slave node model 16
8.8.3 AL – WriteDataByIdentifier – AssignNodeAddress 20
8.8.4 AL – WriteDataByIdentifier – NodeDataDump 22
8.8.5 AL – ReadDataByIdentifier – NodeProductIdentification 23
8.8.6 AL – ReadDataByIdentifier – NodeSerialNumberIdentification 24
8.8.7 AL – ReadDataByIdentifier – NodeConfigurationFileAvailability 25
8.8.8 AL – WriteDataByIdentifier – SaveConfiguration 27
8.8.9 AL – WriteDataByIdentifier – AssignFrameIdentifierRange 28

9 PL – Presentation layer 29

10 SL – Session layer 29
10.1 SL – General 29
10.2 SL – A_Data and T_Data service interface parameter mapping 29

11 TL – Transport layer 30
11.1 TL – Service primitive interface adaptation – General information 30
11.2 TL – CXPI transport layer interface adaptation 30
11.2.1 TL – Mapping of session layer to transport layer service primitives 30
11.2.2 TL – Mapping of T_Data service primitive interface parameters 30

12 NL – Network layer 31
12.1 NL – Service primitive interface adaptation 31
12.1.1 NL – General information 31
12.1.2 NL – CXPI network layer interface adaptation 31
12.2 NL – CXPI master node 32
12.2.1 NL – Network layer 32
12.2.2 NL – Dynamic NAD assignment 32
12.2.3 NL – NodeIdentificationNumber 32
12.3 NL – Master message routing 32
12.3.1 NL – General 32
12.3.2 NL – Diagnostic request message routing 33
12.3.3 NL – Diagnostic response message routing 33
12.3.4 NL – Master node transport protocol support 33
12.4 NL – CXPI slave node 33
12.4.1 NL – General 33
12.4.2 NL – Node configuration handling 33
12.4.3 NL – Slave node diagnostic class II 34
12.4.4 NL – Slave node diagnostic class II – Fixed node address 34
12.4.5 NL – Slave node diagnostic class II – Ignore NAD $7E_{16}$ as broadcast 34
12.4.6 NL – Slave diagnostic class III – Network layer 34
12.4.7 NL – Slave diagnostic class III – Fixed node address 34
12.4.8 NL – Slave diagnostic class III – Accept NAD $7E_{16}$ as broadcast 34

13 DLL – Data link layer 34

Annex A (normative) **DID parameter definitions 35**

Annex B (informative) **Guideline for $P2_{CAN\_Client}$ setting 36**

Bibliography 43

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

A list of all parts in the ISO 14229 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document has been created in order to enable the implementation of unified diagnostic services, as specified in ISO 14229-1, on the clock extension peripheral interface (CXPI) networks (UDSonCXPI).

To achieve this, it is based on the Open Systems Interconnection (OSI) Basic Reference Model specified in ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers.

Figure 1 illustrates the document references from ISO 14229-1, ISO 14229-2 and ISO 20794 (all parts). This document uses only a subset of the diagnostic services defined in ISO 14229-1 (see Table 2).

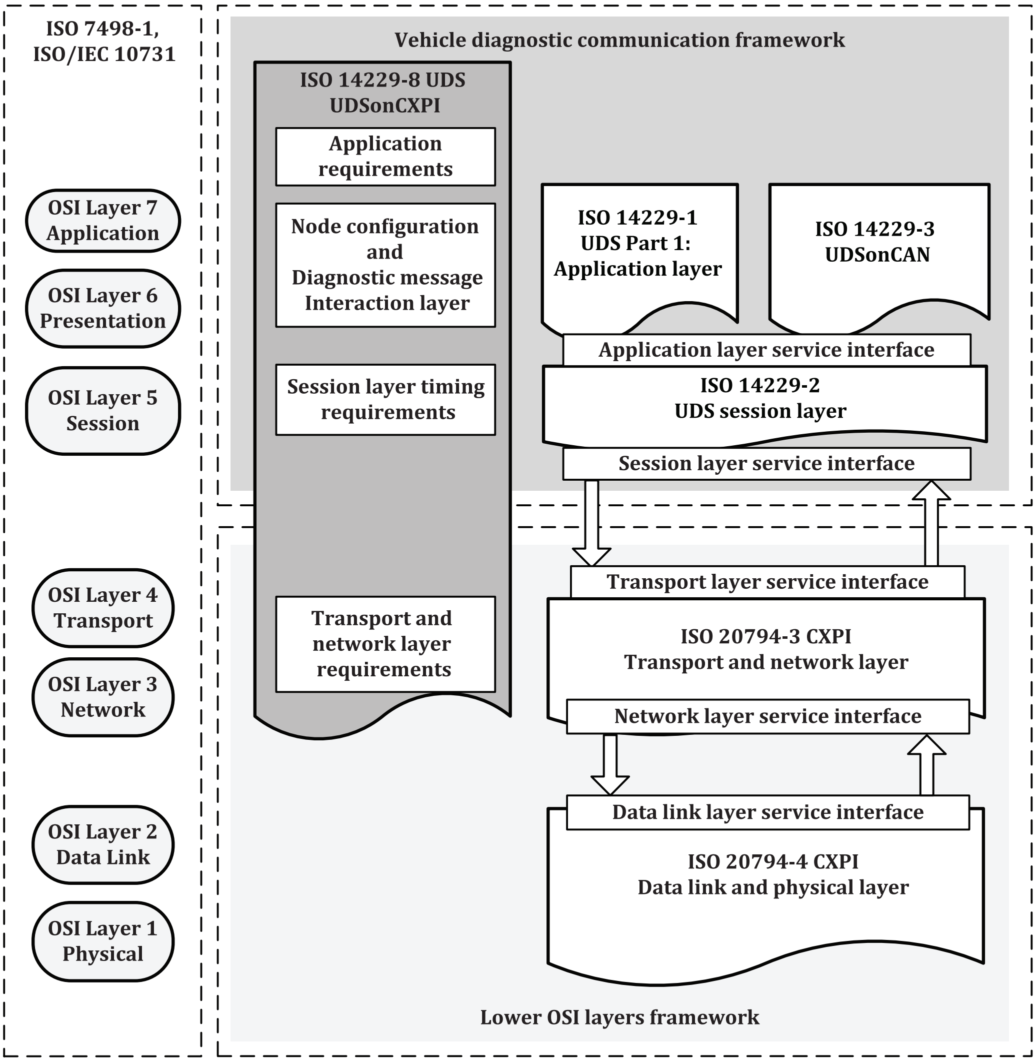

Figure 1 — UDSonCXPI documents reference according to OSI model

Road vehicles — Unified diagnostic services (UDS) —

Part 8: UDS on Clock eXtension Peripheral Interface (UDSonCXPI)

1 Scope

This document specifies the implementation of a common set of unified diagnostic services (UDS) on clock extension peripheral interface networks in road vehicles. The UDSonCXPI diagnostics defines methods to implement diagnostic data transfer between a client and the CXPI slave nodes via the CXPI master node.

This document specifies support of three different diagnostic classes for CXPI slave nodes.

This document references ISO 14229-1 and ISO 14229-2 and specifies implementation requirements of the UDSonCXPI communication protocol for mainly HMI (Human Machine Interface), but not limited to, electric/electronic systems of road vehicles. UDSonCXPI defines how to implement the diagnostic data transfer between a client and CXPI slave nodes via CXPI master node.

NOTE UDSonCXPI does not specify any requirement for the in-vehicle CXPI bus architecture.

This document refers to information contained in ISO 14229-1, ISO 14229-2 and ISO 20794 (all parts).

This document does not include any redundant information of the above-mentioned documents.

It focuses on

— additional requirements specific to the implementation of UDSonCXPI network, and

— specific restrictions in the implementation of UDSonCXPI network.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 7498-1, *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*

ISO 14229-1, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Application layer*

ISO 14229-2, *Road vehicles — Unified diagnostic services (UDS) — Part 2: Session layer services*

ISO 14229-3, *Road vehicles — Unified diagnostic services (UDS) — Part 3: Unified diagnostic services on CAN implementation (UDSonCAN)*

ISO 20794 (all parts), *Road vehicles — Clock extension peripheral interface (CXPI)*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 14229-1, ISO 14229-2, ISO 20794 (all parts), ISO/IEC 7498-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

4 Abbreviated terms

AE	address extension
APP	application
C	conditional
Cvt	convention
GW	gateway
M	mandatory
ID	identifier
N/A	not applicable
NAD	node address for slave nodes
P2	server response time
SA	source address
SF	single frame
TA	target address
U	user optional

5 Conventions

This document is based on the conventions discussed in the OSI Service Conventions (ISO/IEC 10731:1994) as they apply for diagnostic services.

6 SIP – Service interface parameters

6.1 SIP – General

The following subclauses specify the service interface parameters and data types, which are used by the application and application layer services.

6.2 SIP — Data type definitions

This requirement specifies the data type definitions of the UDSonCXPI service interface parameters.

REQ	0.1 SIP – Data type definitions
The data types shall be in accordance to:	

REQ	0.1 SIP – Data type definitions
— `Enum` = 8-bit enumeration — `Unsigned Byte` = 8-bit unsigned numeric value — `Unsigned Word` = 16-bit unsigned numeric value — `Byte Array` = sequence of 8-bit aligned data — `2-bit Bit String` = 2-bit binary coded — `8-bit Bit String` = 8-bit binary coded — `16-bit Bit String` = 16-bit binary coded	

6.3 SIP — A_Mtype, message type

This requirement specifies the message type parameter values of the CXPI service interface.

REQ	0.2 SIP – A_Mtype, message type
The `A_Mtype` parameter shall be of data type `Enum` and shall be used to identify the message type and range of address information included in a service call. Range: [`NormalCom, DiagNodeCfg`]	

6.4 SIP — A_TAtype, target address type

This requirement specifies the target address type parameter values of the CXPI service interface.

REQ	0.3 SIP – A_TAtype, target address type
The `A_TAtype` parameter shall be of data type `Enum` and shall be used to encode the communication model used by the communicating peer entities. Two communication models are specified: '1 to 1' communication, called physical addressing, and '1 to n' communication, called functional addressing. Range: [`physical, functional`]	

6.5 SIP — A_TA, target address

This requirement specifies the target address parameter values of the CXPI service interface.

REQ	0.4 SIP – A_TA, target address
The `A_TA` parameter shall be of data type `Unsigned Byte` and shall be used to identify the target address of the information to be transmitted. Range: [`0116 to FF16`]	

6.6 SIP — A_SA, source address

This requirement specifies the source address parameter values of the CXPI service interface.

REQ	0.5 SIP – A_SA, source address
The `A_SA` parameter shall be of data type `Unsigned Byte` and shall be used to identify the source address of the received information. Mtype = DiagNodeCfg: Range = [00_{16} to FC_{16}]	

6.7 SIP — A_Length, length of A_PDU

This requirement specifies the length of PDU parameter value of the CXPI service interface.

REQ	0.7 SIP – A_Length, length of PDU
The A_Length parameter shall be of data type Unsigned Byte and shall contain the length of the A_PDU to be requested transmission/notified reception.	

6.8 SIP — A_Data, protocol data unit

This requirement specifies the protocol data unit parameter values of the CXPI service interface.

REQ	0.8 SIP – A_Data, protocol data unit
The A_Data parameter shall be of data type Byte Array and shall contain the packet data (A_Data) content of the request or response packet to be transmitted/received. Range: [00_{16} to FF_{16}]	

6.9 SIP — A_SCT, sequence count

This requirement specifies the sequence count parameter values of the CXPI service interface.

REQ	0.9 SIP – A_SCT, sequence count
The A_SCT parameter shall be of data type 2-bit Numeric and shall contain the sequence count information in the response field to be transmitted/received. Range: [0_{02} to 1_{12}]	

6.10 SIP — A_Result, result

This requirement specifies the result parameter values of the CXPI service interface.

REQ	0.10 SIP – A_Result, result
The A_Result parameter shall be of data type 16-bit Bit String and shall contain the status relating to the outcome of a service execution. If two or more errors are discovered at the same time, then the transport or network layer entity shall set the appropriate error bit in the Result parameter. Range: [OK, DLL_Arb_Lost, Err_DLL_Byte, Err_DLL_CRC, Err_DLL_DLC, Err_DLL_DLCext, Err_DLL_Parity, Err_DLL_Framing, Err_NL_TIMEOUT_A, Err_TL_Ptype, Err_TL_PCI_DL_Value, Err_APP_SCT] Range: [0_2 to 1_2] (1-bit per result)	

6.11 SIP — ev_wakeup_ind, event wake-up indication (optional)

This requirement specifies the event wake-up indication parameter values of the CXPI service interface.

REQ	0.11 SIP – ev_wakeup_ind, event wake-up indication (optional)
The ev_wakeup_ind parameter shall be of data type Enum and shall include the event wake-up indication information. Table 1 describes the network management values. Range: [ev_wakeup_pulse_detect, ev_dominant_pulse_detect, ev_clk_detect, ev_clk_loss]	

Table 1 — ev_wakeup_ind, event wake-up indication (optional)

Enum values	Description
`ev_wakeup_pulse_detect`	This service interface parameter value indicates the reception of the wake-up pulse event from the lower OSI layers. This parameter is optional if cmd_wake-up_pulse_on in Table 2 is (optional).
`ev_dominant_pulse_detect`	This service interface parameter value indicates the reception of the dominant pulse event from the lower OSI layers. This parameter is optional if cmd_wakeup_pulse_on in Table 2 is (optional).
`ev_clk_detect`	This service interface parameter value indicates the reception of the clock detection event from the lower OSI layers.
`ev_clk_loss`	This service interface parameter value indicates the reception of the clock loss event from the lower OSI layers.

6.12 SIP — cmd_wakeup_req, command wake-up request

This requirement specifies the command wake-up request parameter values of the CXPI service interface.

REQ	**0.12 SIP – cmd_wakeup_req, command wake-up request**
The `cmd_wakeup_req` parameter shall be of data type `Enum` and shall include the wake-up request command information to wake-up a CXPI node. Table 2 specifies the `cmd_wakeup_req` values. Range: [`cmd_clk_generator_on, cmd_clk_generator_off, cmd_wakeup_pulse_on`]	

Table 2 — cmd_wakeup_req values

Enum values	Description
`cmd_clk_generator_on`	This service interface parameter value commands the clock generator to turn on the clock transmission to the lower OSI layers.
`cmd_clk_generator_off`	This service interface parameter value commands the clock generator to turn off the clock transmission to the lower OSI layers.
`cmd_wakeup_pulse_on (optional)`	This service interface parameter value commands the transmission of the wake-up pulse to the lower OSI layers.

6.13 SIP — NMInfo, network management information

This requirement specifies the network management information parameter values of the CXPI service interface.

REQ	**0.13 SIP – NMInfo, network management information**
The `NMInfo` parameter shall be of data type `2-bit Bit String` and shall contain the NetMngt information in the response field. 00_2 = [no request for `wakeup_ind, sleep_ind` prohibition] 01_2 = [no request for `wakeup_ind, sleep_ind` permission] 10_2 = [request for `wakeup_ind, sleep_ind` prohibition] 11_2 = [request for `wakeup_ind, sleep_ind` permission]	

7 APP – Application

7.1 APP – General

The subclauses define how the diagnostic services, as defined in ISO 14229-1, apply to CXPI.

To allow a common implementation of application layer and session layer for the ISO 20794 series and other communications, this document uses the session layer protocol as defined in ISO 14229-2 and focuses on necessary modifications and interfaces to adopt it to the ISO 20794 series.

The subfunction parameter definitions consider that the most significant bit is used for the suppressPosRspMsgIndicationBit parameter as defined in ISO 14229-1.

It is the vehicle manufacturer's responsibility to setup the CXPI master and slave nodes to exchange UDSonCXPI information according to the ISO 20794 series documents.

7.2 APP – Definition of diagnostic classes

7.2.1 APP – Overview

Architectural, diagnostic communication performance and transport protocol requirements of slave nodes are accommodated by classifying diagnostic services functionality into three diagnostic classes.

Therefore, a diagnostic class is assigned to each slave node according to its level of diagnostic functionality and complexity. See Table 4 for further detail.

7.2.2 APP – Diagnostic class I

Diagnostic class I slave nodes are smart and simple devices like intelligent sensors and actuators, requiring none or very low amount of diagnostic functionality. Actuator control, sensor reading and fault memory handling is done by the master node, using measurement and control data carrying messages. Therefore, specific diagnostic support for these tasks is not required. Fault indication is always measurement and control data based.

7.2.3 APP – Diagnostic class II

A diagnostic class II slave node is similar to a diagnostic class I slave node, but it provides node identification support. The extended node identification is normally required by vehicle manufacturers. Test equipment or master nodes use ISO 14229-1 diagnostic services to request the extended node identification information. Actuator control, sensor reading and fault memory handling is done by the master node, using measurement and control data carrying messages. Therefore, specific diagnostic support for these tasks is not required. Fault indication is always measurement and control data based.

7.2.4 APP – Diagnostic class III

Diagnostic class III slave nodes are devices with enhanced application functions, typically performing their own local information processing (e.g. function controllers, local sensor/actuator loops). The slave nodes execute tasks beyond the basic sensor/actuator functionality, and therefore require extended diagnostic support. Direct actuator control and raw sensor data is often not exchanged with the master node, and therefore not included in measurement and control data carrying messages. ISO 14229-1 diagnostic services for I/O control, sensor value reading and parameter configuration (beyond node configuration) are required.

Diagnostic class III slave nodes have internal fault memory, along with associated reading and clearing services. Optionally, reprogramming (flash/NVRAM reprogramming) of the slave node is possible. This requires an implementation of a boot loader and necessary diagnostic services to unlock the device to initiate downloads and transfer data, etc.

The primary difference between diagnostic class II and diagnostic class III is the distribution of diagnostic capabilities to both, CXPI master node and the CXPI slave node, while for a diagnostic class III CXPI slave node no diagnostic application features of the CXPI slave node are implemented in the CXPI master node.

7.3 APP – CXPI master node requirements – Master node fault management, sensor reading, I/O control

Diagnostic class I and diagnostic class II slave nodes provide measurement and control data-based fault information and sensor, I/O access via measurement and control data carrying messages. The CXPI master node is responsible to handle the slave nodes measurement and control data faults and handle the associated DTCs. The CXPI master node serves UDS requests directly to the client/tester and acts as a diagnostic application layer gateway. UDS services provide access to the measurement and control data on the CXPI bus.

Diagnostic class III slave nodes are independent diagnostic entities. The CXPI master node does not implement diagnostic services for the diagnostic capabilities of its diagnostic class III slave nodes.

7.4 APP – CXPI slave node requirements

7.4.1 APP – General

Slave nodes are typically electronic devices that are not involved in a complex data communication. Also, their need of distributing diagnostic data is low.

7.4.2 APP – Error indications

REQ	8.1 APP – Error indications
Slave nodes shall transmit diagnostic information such as error indications in measurement and control data carrying messages.	

Node configuration services use frame ID $1F_{16}$ (same as the ID of master request field) and $5F_{16}$ (same as the ID of slave response field). Node configuration can be performed by the master node. The NAD is used as the target address in a diagnostic request message and as the source address in a diagnostic response message.

Slave nodes are only accessable by the external test equipment via the CXPI master node network connected to the diagnostic connector.

There is a one-to-many mapping between a physical slave node and a logical slave node and it is addressed using the target address (NAD) value. This means that one physical slave node may be composed of several logical slave nodes.

7.5 APP – CXPI measurement and control data diagnostics

7.5.1 APP – Master handling of slave failure status measurement and control data

REQ	8.2 APP – Master handling of slave failure status measurement and control data
A failure status measurement and control data shall be assigned for each failure that would result in a separate DTC in the master node.	

This information is used to indicate a failure of one of the components to the master node's application, which can then store the associated DTC. There should be one measurement and control data per replaceable component to simplify repair and maintenance of the vehicle.

7.5.2 APP – Slave node current failure status support

Measurement and control data diagnostics are implemented by slave nodes (diagnostic class I and II), which do not implement a fault memory and the diagnostic protocol to directly access this fault memory from an external test tool.

There are two types of failure transmission via measurement and control data carrying messages:

a) Type 1 failure information is periodically transmitted and encoded into an existing measurement and control data (e.g. upper values of measurement and control data range used to indicate specific failure conditions) by the slave node. A type 1 use case-specific failure defined by vehicle manufacturers is not part of this document.

b) Type 2 failure information is not periodically transmitted for components which do not generate a measurement and control data that is periodically transmitted (e.g. slave node internal failure). Additional measurement and control data-based failure transmission shall be implemented for type 2 failures (i.e. if a slave node is capable of locally detecting faults which are not transmitted via the associated measurement and control data in carrying messages already).

REQ	8.3 APP – Slave node current failure status support
Each slave node shall transmit the failure status information that is monitored by the slave node to the master node via measurement and control data carrying messages. The status information shall contain the current failure status of the slave nodes' components. The measurement and control data shall support the states as defined in Table 3.	

Table 3 — Measurement and control data fault states

Test result	Description
no test result	No test execution available, default, initialization value
failed	---
passed	---

If a slave node implements more than one independent function, a status measurement and control data can be assigned to each function. In this case only the failing function could be disabled by the application.

7.6 APP – Network management (optional)

Network management involves wake-up and sleep functionality. The wake-up/sleep function is an optional feature. This function manages the start and stop of transmission and reception of the message by each node.

7.7 APP – CXPI master node gateway application

7.7.1 APP – General

The CXPI master node gateway application supports bidirectional CAN to CXPI communication as well as multiple client backbone network handling on the CXPI subnetwork(s).

7.7.2 APP – CXPI master gateway number of subnets

REQ	8.4 APP – CXPI master gateway number of subnets
A CXPI master node gateway shall have no more than 8 subnets with a maximum of 15 CXPI slave nodes connected to each subnet.	

7.7.3 APP – CXPI master gateway address routing table

Each connected subnet requires an address routing table.

REQ	8.5 APP – Master node – Address routing table
The CXPI master node gateway shall implement an address routing table including CXPI target addresses (NADs) and a supported message size for each CXPI slave node on a subnet.	

REQ	8.6 APP – Master node – Verification of request message length
The message length of each request received from a client (on-board/off-board test equipment) shall be verified by the CXPI master node gateway application according to the CXPI slave node source address and the supported message length in the address routing table. If the received frame length is ≤ the supported message size of the target CXPI slave node then the message shall be transmitted to the CXPI slave node. If the received frame length is > the supported message size of the target CXPI slave node then the message shall not be transmitted to the CXPI slave node.	

7.7.4 APP – CXPI master gateway all nodes request message handling

REQ	8.7 APP – Master node – All nodes request message handling
The client shall send a request message to the CXPI master gateway on the backbone network including a CXPI all nodes request message which the master node dispatches to the CXPI subnet using the $7E_{16}$ all nodes target address (NAD) as specified in 7.7.6.	

7.7.5 APP – Round trip of all node addressing with functional NAD

Figure 2 shows the round trip of an all node addressing with functional NAD. The test equipment (client) sends a single message CAN message with an all node addressing with functional NAD to the CXPI master gateway. The CXPI master gateway routes the message onto the CXPI network with the functional all node NAD. Each CXPI slave node processes the request message and sends the response message to the CXPI master gateway. The CXPI master gateway forwards the response message to the test equipment (client).

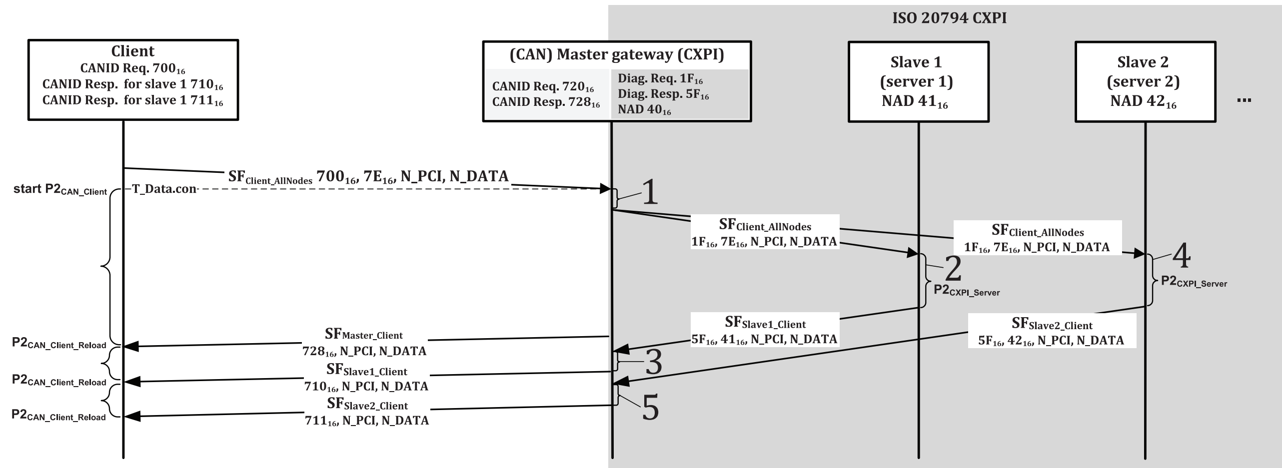

Figure 2 — Round trip of all node addressing with functional NAD

Key

1 CXPI master gateway has received a CAN request message ($SF_{Client\_AllNodes}$) from the client. CXPI master gateway verifies that the N_AE (if 11-Bit CAN ID) or N_TA (if 29-Bit CAN ID) included in the CAN request message is listed in the address routing table (NAD $7E_{16}$) for the target CXPI subnet. CXPI master gateway sends the "all nodes" CXPI request message onto the CXPI network.

2 CXPI slave node #1 (server 1) has received the CXPI request message from the CXPI master gateway. CXPI slave node #1 starts the $P2_{CXPI\_Server}$ timer. CXPI slave node #1 prepares the response message. CXPI slave node #1 sends the response message to the CXPI master gateway and stops the $P2_{CXPI\_Server}$ timer.

3 CXPI master gateway has received a CXPI response message (SF_{Slave1_Client}) from CXPI slave node #1 (NAD 41_{16}). CXPI master gateway performs a transport protocol layer message routing into the CAN response message format and sends the response message (SF_{Slave1_Client}) to the client.

4 CXPI slave node #2 (server 2) has received the CXPI request message from the CXPI master gateway. CXPI slave node #2 starts the $P2_{CXPI\_Server}$ timer. CXPI slave node #2 prepares the response message. CXPI slave node #2 sends the response message to the CXPI master gateway and stops the $P2_{CXPI\_Server}$ timer.

5 CXPI master gateway has received a CXPI response message (SF_{Slave2_Client}) from CXPI slave node #2 (NAD 42_{16}). CXPI master gateway performs a transport protocol layer frame routing into the CAN response message format and sends the response message (SF_{Slave2_Client}) to the client.

7.7.6 APP – Round trip of all node addressing with node-specific NADs

Figure 3 shows the round trip of a single frame CAN request message including an all node addressing with functional NAD (NAD $7E_{16}$), which is sent to each CXPI slave node by the CXPI master gateway. The CXPI master node serializes individual request messages and waits for the response message of the CXPI slave node before it sends the next request message to the next CXPI slave node. This methodology requires a single response buffer in the CXPI master node.

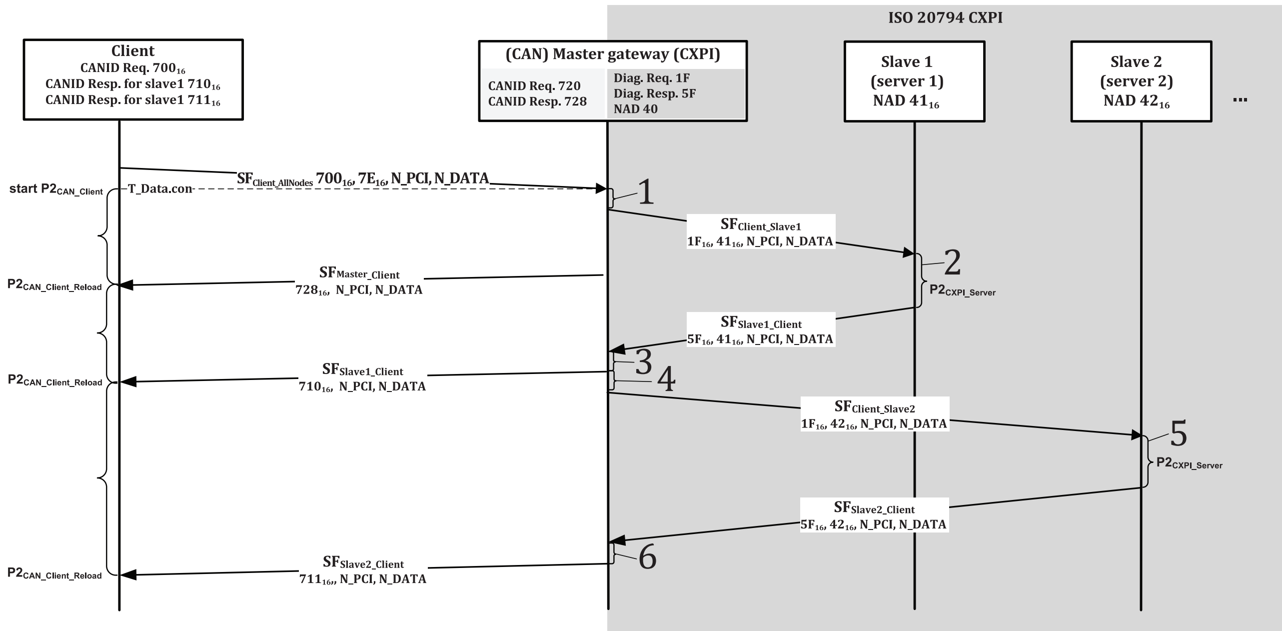

Figure 3 — Round trip of all node addressing with node-specific NADs

Key

1 CXPI master gateway has received a CAN request message ($SF_{Client\_AllNodes}$) from the client. CXPI master gateway verifies that the N_AE (if 11-Bit CAN ID) or N_TA (if 29-Bit CAN ID) included in the CAN request message is listed in the address routing table (NAD $7E_{16}$) for the target CXPI subnet. CXPI master gateway sends the CXPI request message (SF_{Client_Slave1}) to the target CXPI slave node (NAD 41_{16}).

2 CXPI slave node #1 (server 1) has received the CXPI request message from the CXPI master gateway. CXPI slave node #1 starts the $P2_{CXPI\_Server}$ timer. CXPI slave node #1 prepares the response message. CXPI slave node #1 sends the response message to the CXPI master gateway and stops the $P2_{CXPI\_Server}$ timer.

3 CXPI master gateway has received a CXPI response message (SF_{Slave1_Client}) from CXPI slave node #1 (NAD 41_{16}). CXPI master gateway performs a transport protocol layer frame routing into the CAN response message format and sends the response message (SF_{Slave1_Client}) to the client.

4 CXPI master gateway checks all of the slave nodes. If next slave node is found then it sends the request message (SF_{Client_Slave2}) to the target CXPI slave node (NAD 42_{16}).

5 CXPI slave node #2 (server 2) has received the CXPI request message from the CXPI master gateway. CXPI slave node #2 starts the $P2_{CXPI\_Server}$ timer. CXPI slave node #2 prepares the response message. CXPI slave node #2 sends the response message to the CXPI master gateway and stops the $P2_{CXPI\_Server}$ timer.

6 CXPI master gateway has received a CXPI response message (SF_{Slave2_Client}) from CXPI slave node #2 (NAD 42_{16}). CXPI master gateway performs a transport protocol layer frame routing into the CAN response message format and sends the response message (SF_{Slave2_Client}) to the client.

8 AL – Application layer

8.1 AL – Client to CXPI slave node(s) communication

The CXPI application layer protocol supports bidirectional CXPI communication.

8.2 AL – Overview of UDSonCXPI services and applicability to diagnostic classes

The purpose of Table 4 is to reference all unified diagnostic services as they are applicable for an implementation of UDSonCXPI. The table contains the sum of all applicable services. Certain applications using this document to implement UDSonCXPI may restrict the number of useable services and may categorize them in certain application areas/diagnostic sessions (default session, programming session, etc.). The restriction of data length for all diagnostic services due to the data link layer applies (see 8.4).

NOTE The ISO 20794 series supports different diagnostic classes for CXPI slave nodes. It is the vehicle manufacturer's responsibility to specify which diagnostic services of ISO 14229-1 are implemented in a CXPI slave node.

Table 4 — Overview of applicable ISO 14229-1 unified diagnostic services and data ranges

Diagnostic service name[a] (see ISO 14229-1)	Restriction for service application support for		UDSonCXPI with slave node diagnostic class		
	single node addressing	all node addressing	I	II	III
Implementation requirements					
Hardware implementation	N/A	N/A	selection	N/A	
Small MCU implementation	N/A	N/A	selection		N/A
High-performance MCU implementation	N/A	N/A	selection		
Diagnostic and communication management functional unit					
DiagnosticSessionControl	available	available	mandatory		
ECUReset	available	N/A	mandatory		
SecurityAccess	available	N/A	optional		

[a] Services that are not described in Table 4 are not supported.

Table 4 *(continued)*

Diagnostic service name[a] (see ISO 14229-1)	Restriction for service application support for		UDSonCXPI with slave node diagnostic class		
	single node addressing	all node addressing	I	II	III
CommunicationControl	available	available	mandatory		
TesterPresent	available	available	mandatory		
ResponseOnEvent	available	N/A	optional		
ControlDTCSetting	available	available	optional		
SecuredDataTransmission	available	N/A	optional		
Data transmission functional unit					
ReadDataByIdentifier	available	available	mandatory		
ReadMemoryByAddress	available	N/A	optional		
ReadScalingDataByIdentifier	available	N/A	optional		
ReadDataByPeriodIdentifier	available	N/A	optional		
DynamicallyDefineDataIdentifier	available	N/A	optional		
WriteDataByIdentifier	available	available	mandatory		
WriteMemoryByAddress	available	N/A	optional		
Stored data transmission functional unit					
ClearDiagnosticInformation	available	available	optional		mandatory
ReadDTCInformation	available	available	optional		mandatory
Input/Output control functional unit					
InputOutputControlByIdentifier	available	N/A	N/A		mandatory
Remote activation of routine functional unit					
RoutineControl	available	available	N/A		mandatory
Upload/Download functional unit					
RequestDownload	available	N/A	optional		mandatory
RequestUpload	available	N/A	optional		mandatory
TransferData	available	N/A	optional		mandatory
RequestTransferExit	available	N/A	optional		mandatory
RequestFileTransfer	available	N/A	optional		mandatory
[a] Services that are not described in Table 4 are not supported.					

8.3 AL – CommunicationControl (28_{16}) service

REQ	7.1 AL – Master node CXPI cluster specific activation and de-activation
The CXPI master node shall implement the service CommunicationControl as specified in ISO 14229-1 to allow for CXPI cluster specific activation and de-activation of message type "normal communication" (see Table 5).	

Table 5 — Request message subFunction parameter definition

bit 6-0	Description	Cvt	Mnemonic
00_{16}	enableRxAndTx	M	ERXTX
01_{16}	enableRxAndDisableTx	M	ERXDTX
02_{16}	disableRxAndEnableTx	M	DRXETX
03_{16}	disableRxAndTx	M	DRXTX
04_{16}	enableRxAndDisableTxWithEnhancedAddressInformation	M	ERXDTXWEAI
05_{16}	enableRxAndTxWithEnhancedAddressInformation	M	ERXTXWEAI

REQ	7.2 AL – communicationType and subnetNumber
The CXPI master node shall implement the communicationType and subnetNumber of the service CommunicationControl as specified in Table 6.	

Table 6 — communicationType and subnetNumber definition

Bit coding	Bit value	Description	Cvt	Mnemonic
0 to 1	0_{16}	ISOSAEReserved	M	---
	1_{16}	nomalCommunicationMessages	M	NCM
	2_{16}	networkManagementCommunicationMessages	M	NWMCM
	3_{16}	networkManagementCommunicationMessages and nomalCommunicationMessages	M	NWMCM-NCM
2 to 3	0_{16} to 3_{16}	ISOSAEReserved	M	ISOSAERESRVD
4 to 7	0_{16}	Disable/Enable specified communicationType	U	DISENSCT
	1_{16} to E_{16}	Disable/Enable specific subnet identified by subnet number	U	DISENSSIVSN
	F_{16}	Disable/Enable network which request is received on (Receiving node (server))	U	DENWRIRO

REQ	7.3 AL – nodeIdentificationNumber
The CXPI master node shall implement the nodeIdentificationNumber of the service CommunicationControl as specified in Table 7.	

Table 7 — nodeIdentificationNumber definition

A_Data byte	Parameter name	Byte value	Description	Cvt	Mnemonic
#4	nodeIdentificationNumber (high byte)	00_{16}	Reserved	M	---
		01_{16} to FF_{16}	**nodeIdentificationNumberCXPI Subnet** Identify the CXPI subnet(s).	M	NINCS
#5	nodeIdentificationNumber (low byte)	00_{16}	Reserved	M	---
		01_{16} to FF_{16}	**nodeIdentificationNumberNAD** The NAD (slave node address) of the node in the CXPI subnet	M	NINNAD

8.4 AL – UDSonCXPI services

8.4.1 AL – Supported functions

This document uses the application layer services as defined in ISO 14229-1 for client-server based systems to perform functions such as test, inspection, monitoring, diagnosis or programming of on-board vehicle servers.

8.4.2 AL – Master node receive buffer length

The message length of UDSonCXPI messages is specified in ISO 20794-3. The message buffer is controlled by the data link layer.

REQ	7.4 AL – Master node receive buffer length
The vehicle manufacturer shall specify the maximum amount of receive buffer in the master node based on the CXPI slave node supported data length.	

8.4.3 AL – Message length is exceeded

Certain diagnostic services, for example ReadDTCInformation might exceed the message length restriction (depends on the number of DTCs to be reported).

REQ	7.5 AL – Message length exceeded
If the message length is exceeded the node shall apply the negative response handling as specified in ISO 14229-1 for each concerned service.	

8.5 AL – Protocol

This document uses the application layer protocol as defined in ISO 14229-1.

8.6 AL – Timing

8.6.1 AL – General

The subclauses specify the application and session layer timing parameters and how those apply to the client and the server.

8.6.2 AL – Timing parameter values

It is the vehicle manufacturer's responsibility to provide documentation as guidance for the external tool suppliers to calculate the $P2_{Client}$ time-out value according to Annex B. This annex includes example calculations of $P2_{Client}$.

8.6.3 AL – Server timing performance requirements

REQ	7.6 AL – Server timing performance requirements
The server (ECU) shall meet the diagnostic communication performance timings as specified in Table 8.	

Table 8 — Diagnostic communication timings

Parameter	Description	Minimum value / performance requirement	Maximum value / time-out
$P2_{CXPI\_Server}$	Time between reception of the last message of a diagnostic request on the CXPI bus and the CXPI slave node being able to provide data for a response. The CXPI slave node should not respond over the maximum value.	0 ms	500 ms
$P2^*_{CXPI\_Server}$	Time between sending a negative response message with NRC 78_{16} and the CXPI-slave being able to provide data for a response.	0 ms	2 000 ms

8.6.4 AL – SuppressPosRspMsgIndicationBit

REQ	7.7 AL – SuppressPosRspMsgIndicationBit
It is the system designer's responsibility to assure that in case the client does not require a response message (suppressPosRspMsgIndicationBit = TRUE ('1')) and the server might need more time than $P2_{Server}$ to process the request message, that the client shall insert sufficient time between subsequent requests.	

Figure 4 illustrates the transmission timing sequence chart from DoCAN backbone bus to UDSonCXPI.

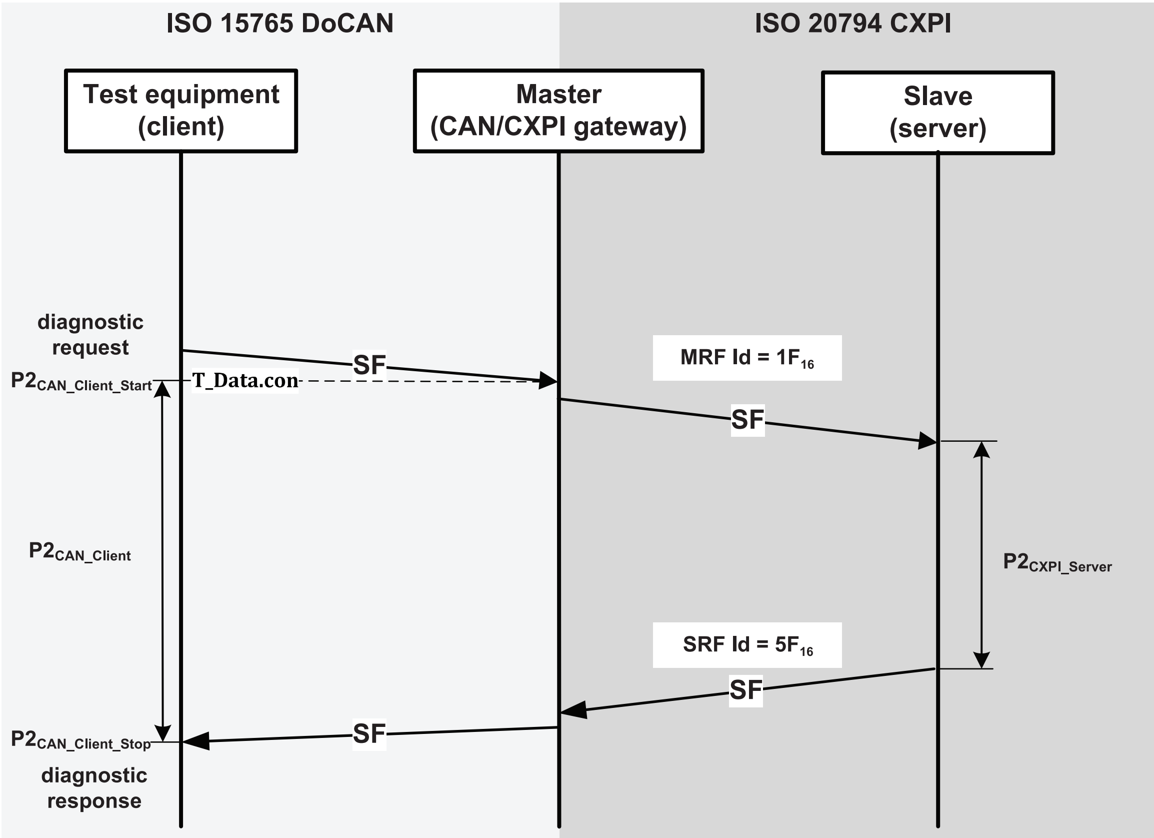

Figure 4 — Transmission timing sequence chart from DoCAN backbone bus to UDSonCXPI

8.7 AL – Response pending

Slave nodes of diagnostic class II and III support the response pending feature. In case a service, which requires more time than $P2_{CXPI\_Server}$ for processing a response pending message according to ISO 14229-1 and ISO 14229-2 is transmitted by the CXPI slave node to extend the maximum response time (see Table 9). Whenever a response pending message is used a final positive or negative response is mandatory for this request.

Table 9 — Response pending A_PDU format

A_PDU format		
DATA1	**DATA2**	**DATA3**
Response service ID: $7F_{16}$	Request SID	Negative response code: 78_{16}

8.8 AL – CXPI slave node configuration services

8.8.1 AL – CXPI node configuration

The node configuration services define how a slave node is configured. Node configuration services are used to avoid/resolve conflicts between slave nodes within a CXPI cluster built out of off-the-shelf slave nodes.

Node configuration is performed by specifying an address space, which consists of an CXPI product identification and an initial node address per slave node. Using these values, it is possible to map DIDs to all A_PDUs transmitted in the CXPI cluster.

8.8.2 AL – Slave node model

8.8.2.1 AL – Memory model

The memory layout of a slave node is shown in Figure 5.

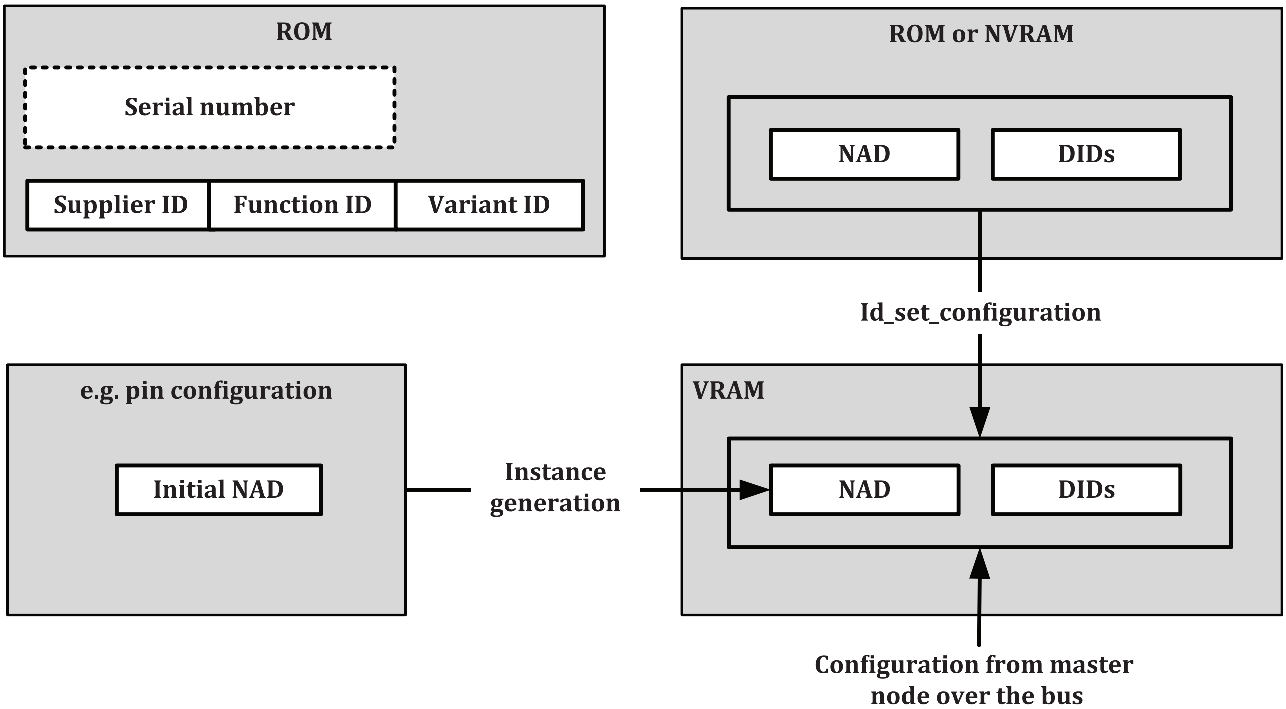

Figure 5 — Slave node memory model

VRAM (Volatile RAM) is considered a memory that is invalid after reset. The NVRAM (Non-Volatile RAM) is a memory that is maintained after reset and may be modified with internal processes (e.g. the application). ROM (Read Only Memory) is considered as a constant memory that may not be modified with internal processes (e.g. application).

8.8.2.2 AL – Slave node configuration variants

Three slave node configuration variants are defined:

a) Unconfigured slave node – After reset the slave node does not contain a valid configuration. Therefore, the master node shall configure the slave node. The configuration is stored in VRAM.

b) Preconfigured slave node – This slave node has a valid configuration after reset. The configuration is normally stored in ROM, but reconfigured data are lost after reset.

c) Full configured slave node – The slave node stores the configuration in NVRAM, so it is still active after reset.

REQ	7.1 AL – Slave node model – AL – Slave node configuration variants
All variants of the slave nodes shall support at least the mandatory configuration services (see Table 4). — It shall be the responsibility of the network designer to guarantee that the supplier ID, function ID, and configured NAD is unique. In case this cannot be achieved ahead of communication, node configuration commands shall be mandatory to resolve conflicts. — If the slave node does not contain a configuration, all messages (except the master node request message and slave node response message) in the slave node shall be marked as invalid. — If the slave node contains a configuration, all configuration requests in the slave node shall be supported.	

8.8.2.3 AL – Initial node address

Each slave node has an initial NAD list. For slave nodes that have no instance generation of the initial NAD the list contains only one entry.

REQ	7.9 AL – Slave node model – AL – Initial node address
The instance generation shall set the initial NAD based on the initial NAD list.	

The instance generation of the initial NAD is not part of this specification.

The configuration, using the writeDataByIdentifier with DID assign node address request, sets the NAD to the configured node address. If the initial node address is already equal to the configured node address, then no action is taken.

Figure 6 shows the relationship between the initial node address list, the initial node address and the configured node address.

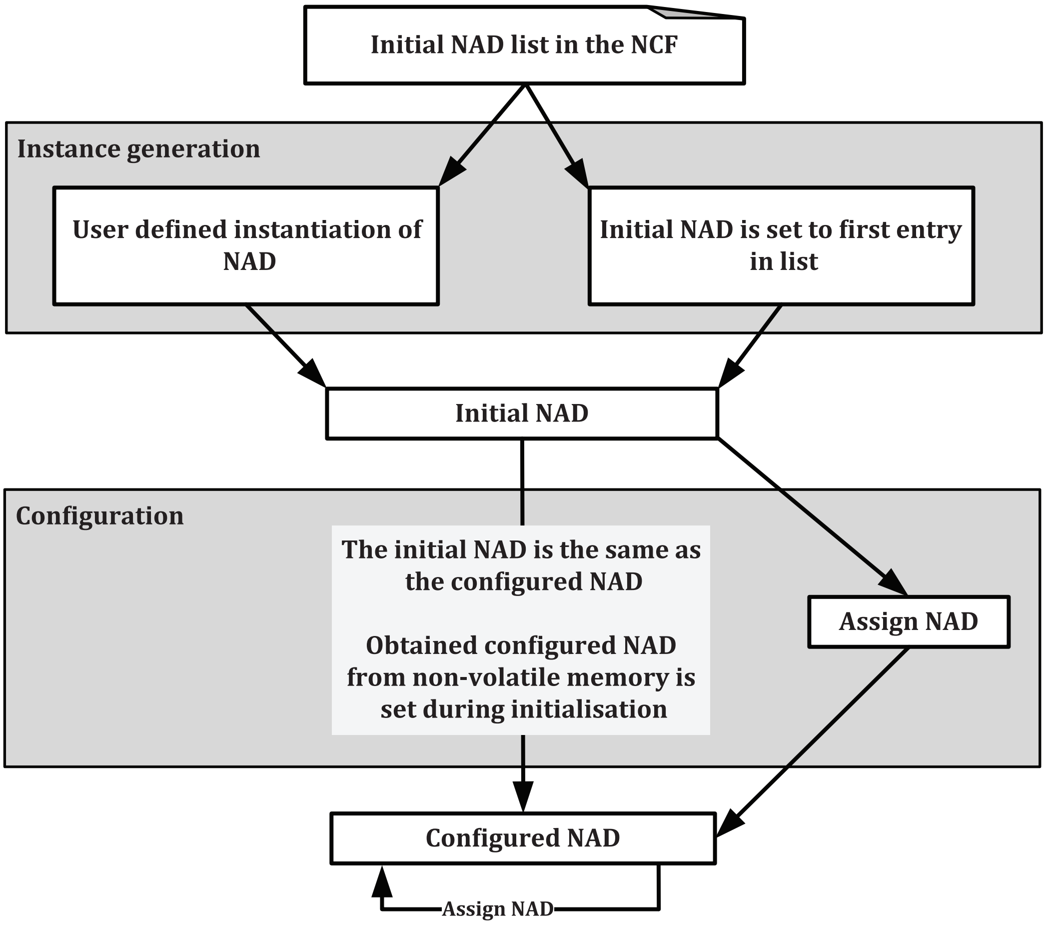

Figure 6 — NAD instantiation and configuration process

8.8.2.4 AL – A_PDU structure of node configuration and identification

8.8.2.4.1 AL – General

Requests are always sent in master node request messages and responses are always sent in slave node response messages.

REQ	7.10 AL – Slave node model – AL – A_PDU structure of node configuration and identification – AL – General
The node configuration and identification messages shall be transmitted as defined in this document.	

8.8.2.4.2 AL – Node address (NAD)

REQ	7.11 AL – Slave node model – AL – A_PDU structure of node configuration and identification – AL – Node address (NAD)
The NodeAddress shall use values as specified in Table 10.	

NAD is the address of the slave node being addressed in a request, i.e. only slave nodes have an address. NAD is also used to indicate the source of a response.

Table 10 — NAD values

NAD value		Description
decimal	hexadecimal	
0	00_{16}	Invalid node address, reserved for go-to-sleep command, see ISO 20794-2.
1 to 125	01_{16} to $7D_{16}$	Master/slave node addresses (NAD)
126	$7E_{16}$	Functional node address (functional NAD), only used for diagnostics.
127	$7F_{16}$	Slave node address broadcast (broadcast NAD)
128 to 255	80_{16} to FF_{16}	Reserved

There is a one-to-many mapping between a physical slave node and a logical slave node and it is addressed using the NAD value. This means that one physical slave node may be composed of several logical slave nodes.

Functional addressing during configuration cannot be used.

8.8.2.5 AL – Supplier ID, function ID and variant ID

Identification is used to identify a slave node.

REQ	7.12 AL – Slave node model – AL – Supplier ID, function ID and variant ID
Supplier ID, function ID, and variant ID shall satisfy the specifications in Tables 11 and 12.	
Each slave node shall have CXPI product identification, as specified in Table 11.	

The identification services define how specific slave node parameters are requested using the identification service.

Table11 — Node identification

DATA1	DATA2	DATA3	DATA4	DATA5
Supplier ID (MSB)	Supplier ID (LSB)	Function ID (MSB)	Function ID (LSB)	Variant ID

The supplier ID is a 16-bit value, with the most significant bit equal to zero. Most significant bit set to one is reserved for future extended numbering systems. The supplier ID is assigned to each supplier of the slave node which corresponds to the node setting and identification service. The supplier ID is specified in Table 12.

Table 12 — Supplier ID

Supplier ID value	Description
0000_{16} to $7FFE_{16}$	Supplier ID is assigned to each supplier.
$7FFF_{16}$	Use of the wildcards
8000_{16} to $FFFF_{16}$	Reserved for future use

The function ID is a 16-bit value assigned by each supplier. If two products differ in function, i.e. CXPI communication or physical world interaction, their function ID shall differ. For absolutely equal function, however, the function ID shall remain unchanged.

The variant ID is an 8-bit value. It shall be changed whenever the product is changed but with unaltered function. The variant ID is a property of the slave node and not the CXPI cluster.

8.8.2.6 AL – Serial number

REQ	7.13 AL – Slave node model – AL – Serial number
Serial number shall be according to the specification in Table 13.	

A slave node may have a serial number to identify a specific instance of a slave node product. The serial number is 4 byte, as specified in Table 13.

Table 13 — Serial number

DATA1	DATA2	DATA3	DATA4
MSB	...	...	LSB

8.8.2.7 AL – Wildcards

REQ	7.14 AL – Slave node model – AL – Wildcards
Wildcards shall be according to the specification in Table 14.	

To be able to leave some information unspecified the wildcard values specified in Table 14 may be used in node configuration requests. All slave nodes shall support the wildcards in requests.

Table 14 — Wildcards

Property	Wildcard value
NAD	$7F_{16}$
Supplier ID	$7FFF_{16}$
Function ID	$FFFF_{16}$

8.8.3 AL – WriteDataByIdentifier – AssignNodeAddress

8.8.3.1 AL – Service description

The WriteDataByIdentifier service with DID = AssignNodeAddress is used to resolve conflicting node addresses in CXPI clusters built using off-the-shelf slave nodes or reused slave nodes.

8.8.3.2 AL – Requirements definition – AL – WriteDataByIdentifier – AssignNodeAddress

The implementation of the AL – WriteDataByIdentifier – AssignNodeAddress service specifies the message sequence and relevant requirements. The service uses the InitialNodeAddress (or the NAD wildcard); this is to avoid the risk of losing the address of a slave node.

The response is using the InitialNodeAddress and not the NewNodeAddress.

REQ	7.15 AL – WriteDataByIdentifier – AssignNodeAddress – General requirement
The service shall only be processed by the slave node if the service is supported and the parameters' length, SupplierIdentifier and FunctionIdentifier match.	

The slave node is not responsible for checking if the NewNodeAddress is within the valid range.

REQ	7.16 AL – WriteDataByIdentifier – AssignNodeAddress – Request and response message processing
After the successful reception and processing of the AL – WriteDataByIdentifier – AssignNodeAddress physical request message, the slave node shall send an AL – WriteDataByIdentifier – AssignNodeAddress positive response message as specified in Table 15.	

REQ	7.17 AL – WriteDataByIdentifier – AssignNodeAddress – MsgParam – DID = AssignNodeAddress
The AssignNodeAddress parameter shall be used by the slave node to identify the content of the dataRecord. NAD = InitialNodeAddress or $7F_{16}$.	

REQ	7.18 AL – WriteDataByIdentifier – AssignNodeAddress – ReqMsgParam – dataRecord SupplierIdentifier
The SupplierIdentifier parameter shall be used by the slave node to compare the parameter value with the internally stored SupplierIdentifier parameter value.	

REQ	7.19 AL – WriteDataByIdentifier – AssignNodeAddress – ReqMsgParam – dataRecord FunctionIdentifier
The FunctionIdentifier parameter shall be used by the slave node to compare the parameter value with the internally stored FunctionIdentifier parameter value.	

REQ	7.20 AL – WriteDataByIdentifier – AssignNodeAddress – ReqMsgParam – dataRecord NewNodeAddress
The NewNodeAddress parameter value shall be used to request the assignment of a new address by the slave node as specified in Table A.1.	

REQ	7.21 AL – WriteDataByIdentifier – AssignNodeAddress – NegativeResponse
A negative response message shall never be sent by the slave node.	

8.8.3.3 AL – Message sequence implementation requirements – AL – WriteDataByIdentifier – AssignNodeAddress

Table 15 defines the content of the AL – WriteDataByIdentifier – AssignNodeAddress messages.

Table 15 — AL – WriteDataByIdentifier – AssignNodeAddress message definition

A_Length	A_PDU definition	REQ	Cvt
1 byte	**WriteDataByIdentifierAL – WriteDataByIdentifier – AssignNodeAddress request message SID**	7.16	M
2 byte	DID = AssignNodeAddress	7.17	M
	dataRecord = [		
2 byte	SupplierIdentifier (MSB, LSB)	7.18	M
2 byte	FunctionIdentifier (MSB, LSB)	7.19	M
1 byte	NewNodeAddress]	7.20	M
1 byte	**WriteDataByIdentifierAL – WriteDataByIdentifier – AssignNodeAddress positive response message SID**	7.16	M
2 byte	DID = AssignNodeAddress	7.17	M

8.8.4 AL – WriteDataByIdentifier – NodeDataDump

8.8.4.1 AL – Service description

The WriteDataByIdentifier service with with DID = DataDump is used for initial configuration of a slave node by the slave node supplier and the format of this message is supplier specific. This service shall only be used by supplier diagnostics and not in a running CXPI cluster, e.g. when node is implemented in a vehicle.

8.8.4.2 AL – Requirements definition – WriteDataByIdentifier – NodeDataDump

The implementation of the WriteDataByIdentifier – NodeDataDump service specifies the message sequence and relevant requirements.

REQ	7.22 AL WriteDataByIdentifier – NodeDataDump – General requirement
The service shall only be processed by the slave node if the service is supported.	

REQ	7.23 AL – WriteDataByIdentifier – NodeDataDump – Request and response message processing
After the successful reception and processing of the WriteDataByIdentifier – NodeDataDump physical request message the slave node shall send a WriteDataByIdentifier – NodeDataDump positive response message as specified in Table 16. A negative response message shall never be sent by the slave node.	

REQ	7.24 AL – WriteDataByIdentifier – NodeDataDump – MsgParam – DID = DataDump
The DataDump parameter as specified in Table A.1 shall be used by the slave node to identify the content of the dataRecord.	

REQ	7.25 AL – WriteDataByIdentifier – NodeDataDump – MsgParam – dataRecord user defined data
The dataRecord shall contain user defined data parameter values of at least 1 byte up to a maximum of 249 byte.	

REQ	7.26 AL – WriteDataByIdentifier – NodeDataDump – NegativeResponse
A negative response message shall never be sent by the slave node.	

8.8.4.3 AL – Message sequence implementation requirements – WriteDataByIdentifier – NodeDataDump

Table 16 defines the content of the WriteDataByIdentifier – NodeDataDump messages.

Table 16 — WriteDataByIdentifier – NodeDataDump message definition

Length	A_PDU definition	REQ	Cvt
1 byte	**WriteDataByIdentifier request message SID**	7.23	M
2 byte	DID = DataDump	7.24	M
	dataRecord = [		C
1 to 249 byte	user defined data (byte #1 to 249)]	7.25	
1 byte	**WriteDataByIdentifier positive response message SID**	7.23	M
2 byte	DID = DataDump	7.24	M

8.8.5 AL – ReadDataByIdentifier – NodeProductIdentification

8.8.5.1 AL – Service description

The ReadDataByIdentifier service with DID = NodeProductIdentification supports the reading of the product identification and other properties from a slave node.

8.8.5.2 AL – Requirements definition – AL – ReadDataByIdentifier – NodeProductIdentification

The implementation of the AL – ReadDataByIdentifier – NodeProductIdentification service specifies the message sequence and relevant requirements.

REQ	7.27 AL – ReadDataByIdentifier – General requirement
The service shall only be processed by the slave node if the service is supported and the parameters SupplierIdentifier and FunctionIdentifier match which those stored in the slave node.	

REQ	7.28 AL – ReadDataByIdentifier – Request and response message processing
After the successful reception and processing of the ReadDataByIdentifier – NodeProductIdentification physical request message the CXPI slave node shall send a ReadDataByIdentifier – NodeProductIdentification positive response message as specified in Table 17. If the ReadDataByIdentifier – NodeProductIdentification physical request message is not supported by the CXPI slave node it shall send a ReadDataByIdentifier negative response message as specified in Table 17.	

REQ	7.29 AL – ReadDataByIdentifier – MsgParam – DID = NodeProductIdentification
The NodeProductIdentificationparameter as specified in Table A.1 shall be used by the slave node to report the corresponding data in the positive response message.	

REQ	7.30 AL – ReadDataByIdentifier – PosRspMsgParam – dataRecord – SupplierIdentifier
The SupplierIdentifier parameter shall be used by the slave node to compare the parameter value with the internally stored SupplierIdentifier parameter value.	

REQ	7.31 AL – ReadDataByIdentifier – PosRspMsgParam – dataRecord – FunctionIdentifier
The FunctionIdentifier parameter shall be used by the slave node to compare the parameter value with the internally stored FunctionIdentifier parameter value.	

REQ	7.32 AL – ReadDataByIdentifier – PosRspMsgParam – dataRecord – Variant
The VariantIdentifier parameter shall be used by the slave node to report the internally stored VariantIdentifier parameter value.	

REQ	7.33 AL – ReadDataByIdentifier – NegRspMsgParam – NRC = subFunctionNotSupported
The negative response code parameter subFunctionNotSupported shall be reported if the physical request message or the NodeProductIdentification parameter value is not supported by the CXPI slave node.	

8.8.5.3 AL – Message sequence implementation requirements – AL – ReadDataByIdentifier – NodeProductIdentification

Table 17 defines the content of the ReadDataByIdentifier – NodeProductIdentification messages.

Table 17 — AL – ReadDataByIdentifier – NodeProductIdentification message definition

Length	A_PDU definition	REQ	Cvt
1 byte	**ReadDataByIdentifier request message SID**	7.28	M
2 byte	DID = NodeProductIdentification (MSB, LSB)	7.29	M
1 byte	**ReadDataByIdentifier positive response message SID**	7.28	M
2 byte	DID = NodeProductIdentification (MSB, LSB)	7.29	M
	dataRecord = [		M
2 byte	SupplierIdentifier (MSB, LSB)	7.28	M
2 byte	FunctionIdentifier (MSB, LSB)	7.29	M
1 byte	VariantIdentifier]	7.32	M
1 byte	**Negative response message SID**	7.28	M
1 byte	Echo of ReadDataByIdentifier request message SID	7.28	M
1 byte	Negative response code (NRC)	7.33	M

8.8.6 AL – ReadDataByIdentifier – NodeSerialNumberIdentification

8.8.6.1 AL – Service description

The ReadDataByIdentifier service with DID = NodeProductIdentification supports the reading of the product identification and other properties from a slave node.

8.8.6.2 AL – Requirements definition – AL – ReadDataByIdentifier – NodeSerialNumberIdentification

The implementation of the AL – ReadDataByIdentifier – NodeSerialNumberIdentification service specifies the message sequence and relevant requirements.

REQ	7.34 AL – ReadDataByIdentifier – General requirement
The service shall only be processed by the slave node if the service is supported and the parameters SupplierIdentifier and FunctionIdentifier match which those stored in the slave node.	

REQ	7.35 AL – ReadDataByIdentifier – Request and response message processing
After the successful reception and processing of the ReadDataByIdentifier – NodeSerialNumberIdentification physical request message the CXPI slave node shall send a ReadDataByIdentifier – NodeSerialNumberIdentification positive response message as specified in Table 18. If the ReadDataByIdentifier – NodeSerialNumberIdentification physical request message is not supported by the CXPI slave node it shall send a ReadDataByIdentifier negative response message as specified in Table 18.	

REQ	7.36 AL – ReadDataByIdentifier – MsgParam – DID = NodeSerialNumberIdentification
The NodeSerialNumberIdentification parameter as specified in Table A.1 shall be used by the slave node to report the corresponding data in the positive response message.	

REQ	7.37 AL – ReadDataByIdentifier – PosRspMsgParam – dataRecord – SerialNumber
The SerialNumber parameter shall be used by the slave node to report the serial number value internally stored in the SerialNumber parameter. The dataRecord shall contain the SerialNumber parameter value SerialNumber#1 (MSB) to SerialNumber#4 (LSB).	

REQ	7.38 AL – ReadDataByIdentifier – NegRspMsgParam – NRC = subFunctionNotSupported
The negative response code parameter subFunctionNotSupported shall be reported if the physical request message or the NodeSerialNumberIdentification parameter value is not supported by the CXPI slave node.	

8.8.6.3 AL – Message sequence implementation requirements – AL – ReadDataByIdentifier – NodeSerialNumberIdentification

Table 18 defines the content of the ReadDataByIdentifier – NodeProductIdentification messages.

Table 18 — AL – ReadDataByIdentifier – NodeSerialNumberIdentification message definition

Length	A_PDU definition	REQ	Cvt
1 byte	**ReadDataByIdentifier request message SID**	7.35	M
2 byte	DID = NodeSerialNumberIdentification (MSB, LSB)	7.36	M
1 byte	**ReadDataByIdentifier positive response message SID**	7.35	M
2 byte	DID = NodeSerialNumberIdentification (MSB, LSB)	7.36	M
	dataRecord = [		M
1 byte	SerialNumber#1 (MSB)	7.37	M
1 byte	SerialNumber#2	7.37	M
1 byte	SerialNumber#3	7.37	M
1 byte	SerialNumber#4 (LSB)]	7.37	M
1 byte	**Negative response message SID**	7.35	M
1 byte	Echo of ReadDataByIdentifier request message SID	7.35	M
1 byte	Negative response code (NRC)	7.38	M

8.8.7 AL – ReadDataByIdentifier – NodeConfigurationFileAvailability

8.8.7.1 AL – Service description

The ReadDataByIdentifier service with DID = NodeConfigurationFileAvailability supports the reading of the node configuration file availability from a slave node.

8.8.7.2 AL – Requirements definition – AL – ReadDataByIdentifier – NodeConfigurationFileAvailability

The implementation of the AL – ReadDataByIdentifier – NodeConfigurationFileAvailability service specifies the message sequence and relevant requirements.

REQ	7.39 AL – ReadDataByIdentifier – General requirement
The service shall only be processed by the slave node if the service is supported and the parameters SupplierIdentifier and FunctionIdentifier match which those stored in the slave node.	

REQ	7.40 AL – ReadDataByIdentifier – Request and response message processing
After the successful reception and processing of the ReadDataByIdentifier – NodeCfgFileAvailability physical request message the CXPI slave node shall send a ReadDataByIdentifier – NodeCfgFileAvailability positive response message as specified in Table 19. If the ReadDataByIdentifier – NodeCfgFileAvailability physical request message is not supported by the CXPI slave node it shall send a ReadDataByIdentifier negative response message as specified in Table 19.	

REQ	7.41 AL – ReadDataByIdentifier – MsgParam – DID = NodeConfigurationFileAvailability
The NodeSerialNumberIdentification parameter as specified in Table A.1 shall be used by the slave node to report the corresponding data in the positive response message.	

REQ	7.42 AL – ReadDataByIdentifier – PosRspMsgParam – dataRecord – NCF version
The SerialNumber parameter shall be used by the slave node to report the serial number value internally stored in the SerialNumber parameter. The dataRecord shall contain the NodeConfigurationFileAvailability parameter value — Major version of NCF — Minor version of NCF — Sub version of NCF — SourceIdentifier[02_{16} = slave node configuration based on NCF; 01_{16}, 03_{16} to FF_{16} = Reserved]	

REQ	7.43 AL – ReadDataByIdentifier – NegRspMsgParam – NRC = subFunctionNotSupported
The negative response code parameter subFunctionNotSupported shall be reported if the physical request message or the NodeConfigurationFileAvailability parameter value is not supported by the CXPI slave node.	

8.8.7.3 AL – Message sequence implementation requirements – AL – ReadDataByIdentifier – NodeConfigurationFileAvailability

Table 19 defines the content of the ReadDataByIdentifier – NodeProductIdentification messages.

Table 19 — AL – ReadDataByIdentifier – NodeConfigurationFileAvailability message definition

Length	A_PDU definition	REQ	Cvt
1 byte	**ReadDataByIdentifier request message SID**	7.40	M
2 byte	DID = NodeConfigurationFileAvailability (MSB, LSB)	7.41	M
1 byte	**ReadDataByIdentifier positive response message SID**	7.40	M
2 byte	DID = NodeConfigurationFileAvailability (MSB, LSB)	7.41	M
	dataRecord = [		M
1 byte	Major version of NCF	7.42	M
1 byte	Minor version of NCF	7.42	M
1 byte	Sub version of NCF	7.42	M
1 byte	SourceIdentifier]	7.42	M
1 byte	**Negative response message SID**	7.40	M
1 byte	Echo of ReadDataByIdentifier request message SID	7.40	M
1 byte	Negative response code (NRC)	7.43	M

8.8.8 AL – WriteDataByIdentifier – SaveConfiguration

8.8.8.1 AL – Service description

The WriteDataByIdentifier with DID = SaveConfiguration supports the writing of the configuration in the nodes to store the current configured PIDs (Protected IDs) and PIDs located in RAM to NVRAM. The slave application gathers the current configured NAD and PIDs from the data link layer and triggers the NVRAM write routine. A configuration in the slave node may be valid even without the master node using this request (i.e. slave node does not need to wait for this request to have a valid configuration).

8.8.8.2 AL – Requirements definition – AL – WriteDataByIdentifier – SaveConfiguration

The implementation of the AL – WriteDataByIdentifier – SaveConfiguration service specifies the message sequence and relevant requirements.

REQ	**7.44 AL – WriteDataByIdentifier – SaveConfiguration – General requirement**
The service shall only be processed by the slave node if the service is supported.	

REQ	**7.45 AL – WriteDataByIdentifier – SaveConfiguration – Request and response message processing**
After the successful reception and processing of the AL – WriteDataByIdentifier – SaveConfiguration physical request message, the CXPI slave node shall send an AL – WriteDataByIdentifier – SaveConfiguration positive response message as specified in Table 20. The slave node shall not wait until the configuration is saved before a positive response is sent, for example the request is accepted by the slave node. A negative response message shall never be sent by the slave node.	

REQ	**7.46 AL – WriteDataByIdentifier – SaveConfiguration – MsgParam – DID = SaveConfiguration**
The SaveConfiguration parameter as specified in Table A.1 shall be used by the slave node as a command to store the configuration.	

REQ	**7.47 AL – WriteDataByIdentifier – SaveConfiguration – dataRecord – New version of NCF**
A new version of an NCF shall be used by the slave node as an alternative to NAD and PIDs located in RAM to NVRAM. The slave application configures NAD and PIDs to the data link layer and triggers the NVRAM write routine.	

REQ	**7.48 AL – WriteDataByIdentifier – SaveConfiguration – NegativeResponse**
A negative response message shall never be sent by the slave node.	

8.8.8.3 AL – Message sequence implementation requirements

Table 20 defines the content of the AL – WriteDataByIdentifier – SaveConfiguration messages.

Table 20 — AL – WriteDataByIdentifier – SaveConfiguration message definition

Length	A_PDU definition	REQ	Cvt
1 byte	**AL – WriteDataByIdentifier – SaveConfiguration request message SID**	7.45	M
2 byte	DID = SaveConfiguration	7.46	M
	dataRecord = [		

Table 20 *(continued)*

Length	A_PDU definition	REQ	Cvt
1 byte	New version of NCF	7.47	M
1 byte	**AL – WriteDataByIdentifier – SaveConfiguration positive response message SID**	7.45	M
2 byte	DID = SaveConfiguration	7.46	M

8.8.9 AL – WriteDataByIdentifier – AssignFrameIdentifierRange

8.8.9.1 AL – Service description

The WriteDataByIdentifier service with DID = AssignFrameIdentifierRange is used to set or disable Protected IDs (PIDs) for up to four messages. It is important to notice that the request message provides the PID including the parity bit. Furthermore, reserved PDUs with PIDs specified in ISO 20794-2 shall not be changed.

8.8.9.2 AL – Requirements definition – AL – WriteDataByIdentifier – AssignFrameIdentifierRange

The implementation of the AL – WriteDataByIdentifier – AssignFrameIdentifierRange service specifies the message sequence and relevant requirements.

REQ	7.49 AL – WriteDataByIdentifier – AssignFrameIdentifierRange – General requirement
The service shall only be processed by the slave node if the service is supported and the parameters' length, SupplierIdentifier, and FunctionIdentifier match.	

REQ	7.50 AL – WriteDataByIdentifier – AssignFrameIdentifierRange – Request and response message processing
After the successful reception and processing of the AL – WriteDataByIdentifier – AssignFrameIdentifierRange physical request message, the CXPI slave node shall send an AL – WriteDataByIdentifier – AssignFrameIdentifierRange positive response message as specified in Table 21. In case the slave cannot fulfill all 'set PID' and 'unassign PID' requests the slave node shall reject the request message and shall not sent a response. The 'do not care' is always accepted by the receiving slave node. The slave node does not validate the payload PIDs (i.e. validating the parity flags) beyond 'do not care' pattern FF_{16}, the slave node relies on that the master node sets the correct PIDs.	

REQ	7.51 AL – WriteDataByIdentifier – AssignFrameIdentifierRange – MsgParam – DID = AssignFrameIdentifierRange
The AssignFrameIdentifierRange parameter as specifiied in Table A.1 shall be used to set or disable Protected IDs (PIDs) for up to four messages.	

REQ	7.52 AL – WriteDataByIdentifier – AssignFrameIdentifierRange – ReqMsgParam – StartIndex
The StartIndex parameter shall be used by the slave node to assign the PID to the first message. The order of the list is specified in the node attributes subclause in the NCF. The first message in the list shall have index 0_{10} (zero).	

REQ	7.53 AL – WriteDataByIdentifier – AssignFrameIdentifierRange – ReqMsgParam – PID (1 to 4)
The Protected Identifier (PID) parameters (1 to 4) shall be used by the slave node to set or disable PIDs for up to four messages. The PIDs are an array of four PID values that shall be used in the configuration request. Valid PID values here are the PID values for application measurement and control data carrying messages, the 'un-assign PID' value 0_{10} (zero) and the 'do not care' value FF_{16}. The 'un-assign PID' value is used to invalidate this message for transmission on the CXPI network. The 'do not care' is used to keep the previous assigned value of this message. It is not necessary to un-assign an already set PID in a slave node to be able to set a new PID for the same message.	

8.8.9.3 AL – Message sequence implementation requirements – AL – WriteDataByIdentifier – AssignFrameIdentifierRange

Table 21 defines the content of the AL – WriteDataByIdentifier – AssignFrameIdentifierRange messages.

Table 21 — AL – WriteDataByIdentifier – AssignFrameIdentifierRange message definition

Length	A_PDU definition	REQ	Cvt
1 byte	**WriteDataByIdentifier request message SID**	7.50	M
2 byte	DID = AssignFrameIdentifierRange	7.51	M
	dataRecord = [		M
1 byte	StartIndex	7.52	M
1 byte	PID #1	7.53	M
1 byte	PID #2	7.53	M
1 byte	PID #3	7.53	M
1 byte	PID #4]	7.53	M
1 byte	**WriteDataByIdentifier positive response message SID**	7.50	M
2 byte	DID = AssignFrameIdentifierRange	7.51	M

EXAMPLE 1 A slave node supports five messages (power status, IO_1, IO_2, IO_3, IO_4). The master node application sends an AssignFrameIdentifierRange request message with the parameters start index set to 1, PID (index 1 to 4) set to (80_{16}, $C1_{16}$, 42_{16}, 00_{16}). When the slave node receives the request message it sets the PIDs to (IO_1 = 80_{16}, IO_2 = $C1_{16}$, IO_3 = 42_{16}, IO_4 = unassigned). The power status message is not affected. The slave node responds with a positive response if requested.

9 PL – Presentation layer

The presentation layer requirements are specified in ISO 14229-1.

10 SL – Session layer

10.1 SL – General

REQ	5.1 SL – General
The session layer requirements shall be implemented as specified in ISO 14229-2.	

10.2 SL – A_Data and T_Data service interface parameter mapping

REQ	5.2 SL – A_Data and T_Data service interface parameter mapping
The A_Data service interface parameters shall be mapped to the T_Data service interface parameters as specified in Table 22.	

Table 22 — A_Data to T_Data service interface parameter mapping

Application (service user) (according to ISO 14229-1)	Session layer (service provider)	A_Data.req and A_Data.ind parameter validity DiagNodeCfg with A_Length > 0			
		.req	.ind	.resp	.conf
A_Mtype	T_Ptype	X[a]	X[a]	X[a]	X[a]
A_Length	T_Length	X[a]	X[a]	X[a]	X[a]
A_ReqId	T_ReqId	X[a]	X[a]	X[a]	X[a]
A_TAtype	T_TAtype	X[a]	X[a]	X[a]	X[a]
A_TA	T_TA	X[a]	X[a]	X[a]	X[a]
A_SA	T_SA	X[a]	X[a]	X[a]	X[a]
A_Data	T_Data	X[a]	X[a]	X[a]	X[a]
A_NMInfo	T_NMInfo	X[a]	X[a]	X[a]	X[a]
A_SCT	T_SCT	X[a]	X[a]	X[a]	X[a]
A_Result	T_Result	---[b]	X[a]	---[b]	X[a]

[a] Supported = “X”.

[b] Not supported = “---”.

11 TL – Transport layer

11.1 TL – Service primitive interface adaptation – General information

This document uses the service primitive interface defined in ISO 14229-2 for the transmission and reception of diagnostic messages. This subclause defines the mapping of the session layer service primitive interface and parameters to the data link independent transport layer protocol.

NOTE The transport layer services are used to perform the application layer and diagnostic session management timing.

11.2 TL – CXPI transport layer interface adaptation

11.2.1 TL – Mapping of session layer to transport layer service primitives

Table 23 specifies the mapping interface between the session layer service primitives (see ISO 14229-2) onto the transport layer as specified in ISO 20794-3.

Table 23 — Mapping of T_Data service primitives onto T_Data service primitives

Session layer service primitives (according to ISO 14229-2)	Transport layer service primitives (according to ISO 20794-3)
S_Data.req	T_Data.req
S_Data.ind	T_Data.ind
S_Data.conf	T_Data.conf

11.2.2 TL – Mapping of T_Data service primitive interface parameters

The parameters of the session layer service primitive interface are mapped to the transport layer service primitive interface to support the diagnostic request and response message transmission between client and server, (see Table 24).

Table 24 — T_Data service interface parameter mapping

Session layer (service user) according to ISO 14229-2	Transport layer (service provider) according to ISO 20794-3	T_Data interface parameter validity DiagNodeCfg with T_Length > 0		
		.req	.ind	.conf
S_Mtype	T_Ptype	X[a]	X[a]	X[a]
S_Length	T_Length	X[a]	X[a]	X[a]
S_TAtype	T_TAtype	X[a]	X[a]	X[a]
S_TA	T_TA	X[a]	X[a]	X[a]
S_SA	T_SA	X[a]	X[a]	X[a]
S_Data	T_PDU	X[a]	X[a]	X[a]
S_NMInfo	T_NMInfo	X[a]	X[a]	X[a]
S_SCT	T_SCT	X[a]	X[a]	X[a]
S_Result	T_Result	---[b]	X[a]	X[a]

[a] Supported = “X”.

[b] Not supported = “---”.

The network layer confirmation of the successful transmission of the message (T_Data.confirm) is forwarded to the application, because it is needed in the application for starting those actions, which shall be executed immediately after the transmission of the request/response message (ECUReset, etc.).

12 NL – Network layer

12.1 NL – Service primitive interface adaptation

12.1.1 NL – General information

This document uses the service primitive interface defined in ISO 20794-3 for the transmission and reception of diagnostic messages. This subclause defines the mapping of the transport layer service primitive interface and parameters to the network layer services.

NOTE The network layer services are used to perform the diagnostic session management timing.

12.1.2 NL – CXPI network layer interface adaptation

12.1.2.1 NL – Mapping of service primitive parameters

Table 25 specifies the mapping interface between the transport layer service primitives onto the network layer as specified in ISO 20794-3.

Table 25 — Mapping of T_Data service primitives onto the N_Data service primitives

Transport layer service primitives (according to ISO 20794-3)	Network layer service primitives (according to ISO 20794-3)
T_Data.req	N_Data.req
T_Data.ind	N_Data.ind
T_Data.conf	N_Data.conf

12.1.2.2 NL – Mapping of N_Data service primitive interface parameters

The parameters of the transport layer service primitive interface are mapped to the network layer service primitive interface to support the diagnostic request and response message transmission between client and server, (see Table 26).

Table 26 — T_Data to N_Data service interface parameter mapping

Transport layer (service user) according to ISO 20794-3	Network layer (service provider) according to ISO 20794-3	N_Data interface parameter validity DiagNodeCfg with N_Length > 0			
		.req	.ind	.resp	.conf
T_Ptype	N_Ptype	X[a]	X[a]	X[a]	X[a]
T_Length	N_Length	X[a]	X[a]	X[a]	X[a]
T_TAtype	N_TAtype	X[a]	X[a]	X[a]	X[a]
T_TA	N_TA	X[a]	X[a]	X[a]	X[a]
T_SA	N_SA	X[a]	X[a]	X[a]	X[a]
T_PDU	N_PDU	X[a]	X[a]	X[a]	X[a]
T_NMInfo	N_NMInfo	X[a]	X[a]	X[a]	X[a]
T_SCT	N_SCT	X[a]	X[a]	X[a]	X[a]
T_Result	N_Result	---[b]	X[a]	---[b]	X[a]

[a] Supported = “X”.

[b] Not supported = “---”.

12.2 NL – CXPI master node

12.2.1 NL – Network layer

REQ	3.1 NL – CXPI master node – Network layer
Each CXPI master node that communicates with slave nodes with UDS services shall implement the network layer requirements as specified in ISO 20794-3.	

12.2.2 NL – Dynamic NAD assignment

REQ	3.2 NL – CXPI master node – Dynamic NAD assignment
If dynamic NAD assignment is used on a CXPI cluster the CXPI master node shall ensure that after communication start-up all CXPI slave nodes have their NADs assigned as specified in 8.8.3. The start-up configuration time necessary to complete NAD assignment shall be documented in the diagnostic specification of the master node. NOTE This implies that diagnostic communication between external diagnostic test equipment and a CXPI slave node cannot be possible until the CXPI master node has completed the NAD assignment sequence.	

12.2.3 NL – NodeIdentificationNumber

REQ	3.3 NL – CXPI master node – NodeIdentificationNumber
The CXPI clusters connected to the CXPI master node shall be assigned a value in the range of 01_{16} – FF_{16} in the high byte of the nodeIdentificationNumber parameter to identify the CXPI subnet(s). The low byte of the nodeIdentificationNumber parameter shall be used to include the NAD (CXPI slave node address) of the node in the CXPI subnet. The nodeIdentificationNumber parameter is part of the CommunicationControl service defined in ISO 14229-1.	

12.3 NL – Master message routing

12.3.1 NL – General

The master node usually is a high-performance ECU and, in most implementations, supports the ISO 14229-1 diagnostic services. The master node and the external test equipment are connected via a backbone bus (e.g. the ISO 11898 series, the ISO 15765 series).

12.3.2 NL – Diagnostic request message routing

REQ	3.4 NL – Master message routing – Diagnostic request message routing
The master node shall receive all diagnostic requests addressed to the slave nodes from the backbone bus and route them to the appropriate CXPI cluster(s).	

12.3.3 NL – Diagnostic response message routing

REQ	3.5 NL – Master message routing – Diagnostic response message routing
The master node shall receive all diagnostic responses from the slave nodes and route them back to the backbone bus.	

12.3.4 NL – Master node transport protocol support

All diagnostic request and response messages addressed to the slave nodes can be routed in the network layer (i.e. no application layer routing).

REQ	3.6 NL – Master message routing – Master node transport protocol support
The master node shall implement the CXPI transport protocol (see ISO 20794-3) as well as the transport protocols used on the backbone busses (e.g. ISO 15765-2).	

12.4 NL – CXPI slave node

12.4.1 NL – General

This document specifies CXPI slave node network layer requirements depending on the diagnostic class.

There are no specific network layer requirements defined for a CXPI slave node which complies to diagnostic class I.

12.4.2 NL – Node configuration handling

All requests are carried in master node request messages and all responses are carried in slave node response messages. All requests and responses are using single messages only.

REQ	3.7 NL – Slave node – NL – Node configuration handling – Cancellation of pending response after reception of a valid master node requestls
The slave node shall cancel a pending response after reception of a valid master node request (except when NAD is the functional NAD).	

REQ	3.8 NL – Slave node – NL – Node configuration handling – Cancellation of pending response if a N_As_{max} timeout occurs
The slave node shall implement the "Cancellation of pending response if a N_As_{max} timeout occurs" as specified in ISO 14229-3.	

REQ	3.9 NL – Slave node – NL – Node configuration handling – Reception of an invalid master node request
The slave node shall discard the configuration information from an invalid master node request (response time out).	

REQ	3.10 NL – Slave node – NL – Node configuration handling – Use of wildcards
All services shall support the use of wildcards as specified in Table 12.	

There are no specific network layer requirements defined for a CXPI slave node which complies to diagnostic class I but it can support node configuration services as specified in 8.8.

12.4.3 NL – Slave node diagnostic class II

REQ	3.11 NL – Slave node diagnostic class II
Each diagnostic class II CXPI slave node shall implement the network layer requirements as specified in ISO 20794-3.	

12.4.4 NL – Slave node diagnostic class II – Fixed node address

REQ	3.12 NL – Slave node diagnostic class II – Fixed node address
Each diagnostic class II CXPI slave node shall be assigned a fixed node address (NAD) during system design time. NOTE For CXPI clusters using dynamic NAD assignment during network communication start-up this implies that after NAD assignment has finished, all CXPI slave nodes have their NADs assigned as specified during system design time.	

12.4.5 NL – Slave node diagnostic class II – Ignore NAD $7E_{16}$ as broadcast

REQ	3.13 NL – Slave node – Diagnostic class II: ignore NAD $7E_{16}$ as broadcast
Each diagnostic class II CXPI slave node shall ignore NAD $7E_{16}$ as broadcast (functional) address for diagnostic communication on the CXPI cluster.	

12.4.6 NL – Slave diagnostic class III – Network layer

REQ	3.14 NL – Slave node diagnostic class III – Network layer
Each diagnostic class III CXPI slave node shall implement the network layer requirements as specified in ISO 20794-3.	

12.4.7 NL – Slave diagnostic class III – Fixed node address

REQ	3.15 NL – Slave node diagnostic class III – Fixed node address
Each diagnostic class III CXPI slave node shall be assigned a fixed node address (NAD) during system design time. NOTE CXPI clusters using dynamic NAD assignment during network communication start-up implies that after NAD assignment has finished, all CXPI slave nodes have their NADs assigned as specified during system design time.	

12.4.8 NL – Slave diagnostic class III – Accept NAD $7E_{16}$ as broadcast

REQ	3.16 NL – Slave node diagnostic class III – Accept NAD $7E_{16}$ as broadcast
Each diagnostic class III CXPI slave node shall support NAD $7E_{16}$ as broadcast (functional) address for diagnostic communication on the CXPI cluster.	

13 DLL – Data link layer

This document makes use of the data link layer specification defined in ISO 20794-4 for the transmission and reception of diagnostic messages.

Annex A
(normative)

DID parameter definitions

The parameter dataIdentifier (DID) logically represents an object (e.g., NodeConfigurationFileAvailability).

Table A.1 defines the DID data-parameter definitions.

Table A.1 — DID data-parameter definitions

Byte value	Description	Cvt	Mnemonic
$FF02_{16}$	**AssignFrameIdentifierRange** This value shall be used to set or disable Protected IDs (PIDs) for up to four messages in a slave node.	M	AFRIDR
$FF03_{16}$	**SaveConfiguration** This value shall be used to reference the writing of the configuration in the nodes to store the current configured PIDs and PIDs located in RAM to NVRAM.	M	SAVECFG
$FF04_{16}$	**NodeConfigurationFileAvailability** This value shall be used to reference the reading of the node configuration file availability from a slave node.	M	NODCFGFILEAV
$FF05_{16}$	**NodeProductIdentification** This value shall be used to reference the reading of the product identification and other properties from a slave node.	M	NODPRODID
$FF06_{16}$	**DataDump** This value shall be used to reference initial configuration of a slave node by the slave node supplier.	M	DATADUMP
$FF07_{16}$	**AssignNodeAddress** This value shall be used to reference the assignment of a new node address to resolve conflicting node addresses in CXPI clusters when using off-the-shelf slave nodes or reused slave nodes.	M	ANODADDR
$F18C_{16}$	**NodeSerialNumberIdentification** This value shall be used to reference the node serial number.	M	NODSERNUMID

Annex B
(informative)

Guideline for $P2_{CAN\_Client}$ setting

B.1 General

In case the CXPI network is connected to a CAN backbone and the CXPI long frame is used the transfer data between CXPI master and client(s) the $P2_{CAN\_Client}$ becomes a large value.

This annex describes how to calculate $P2_{CAN\_Client}$ and the necessity of considering the communication rate, the number of CXPI slave nodes and the length of the CXPI data field for keeping the $P2_{CAN\_Client}$ a small value. This annex also describes the $P2_{CAN\_Client}$ setting value according to the communication rate, the number of nodes and the length of CXPI data field on limited services condition as shown in Table 4.

B.2 Description of $P2_{CAN\_Client}$ value calculation

The calculation of the $P2_{CAN\_Client}$ value is based on the following assumptions:

— CXPI master node is connected to a CAN network,

— client uses functional addressing of the diagnostic request message to the CXPI slave nodes,

— a single frame is used for the communication on the CXPI network.

Table B.1 defines the parameters of the timing sequence chart.

Table B.1 — Parameters of the timing sequence chart

Parameter	Description
$P2_{CXPI\_Server}$	The time period from the time when the CXPI slave node has received the diagnostic request until the time when the arbitration losing the CXPI slave node has sent out the last diagnostic response and the CXPI slave node has cancelled the response. $P2_{CXPI\_Server}$ = Slave handling time + CXPI time (response)
$P2_{CAN\_Client}$	See ISO 14229-2 for the definition of $P2_{Client}$.
ΔP2	See ISO 14229-2 for the definition of ΔP2.
Gateway (router) handling time (request message)	The gateway processing time is measured from the reception completion of the diagnostic request from a client to the initiation of the diagnostic request to the CXPI master node.
CAN frame transmission time	The transmission time of the diagnostic request is measured from the gateway to the CXPI master node.
Master (CAN/CXPI gateway) node handling time (request message)	The CXPI master node processing time is measured from the completion of the gateway diagnostic request message to the initiation of the diagnostic request message to the CXPI slave node.
CXPI slave node handling time	The CXPI slave node processing time from the request message reception completion to the beginning of the diagnostic response message transmission (see Table B.2).
CXPI frame transmission time (request message)	The diagnostic request message transmission time from the CXPI master node to the CXPI slave node (see Table B.2).

Table B.1 *(continued)*

Parameter	Description
CXPI frame transmission time (response message)	The total time between the transmission latency lost and the transmission time of the diagnosis response message from the CXPI slave node to the CXPI master node (see Table B.2).
Master (CAN/CXPI gateway) node handling time (response message)	The CXPI master node processing time from the response message reception completion to the initiation of the diagnostic response message transmission to gateway.
CAN time1 (response message)	The transmission time of the diagnostic response from the CXPI master node to gateway.
Gateway (router) handling time (response message)	The gateway processing time from the diagnostic response message reception completion to the initiation of the diagnostic response message transmission to the client.
CAN time2 (response message)	The diagnostic response message transmission time from the gateway to the client.

The calculation method of CXPI time (request) and CXPI time (response) is shown in Table B.2. The $P2_{CXPI\_Server}$ and $P2_{CAN\_Client}$ setting values are introduced by the calculated calculation of the CXPI time (request) and the CXPI time (response).

The transmission delay by arbitration lost in case of 50 % CXPI bus load by regular messages is illustrated in Figure B.1. The diagnostic request is delayed by one frame time length caused by arbitration lost. The diagnostic response message transmission delay is calculated based on the total time from response message sending at the same time by all CXPI slave nodes until the least prioritized diagnostic response message is initiated to be transmitted.

Table B.2 defines the CXPI time calculation method.

Table B.2 — CXPI time calculation method

Item	Formula	Unit
Normal frame time-length	(1)[a]	T_{bit}
Long frame time-length	(2)[a]	T_{bit}
Diagnostic response message transmission delay of normal frame in case of no bus load (number of CXPI slave nodes is 2)[b]	(3)	T_{bit}
Regular frame time-length in case of 50% bus load (CXPI data field byte-length is 12 byte)	(4)	T_{bit}
Diagnostic request message transmission delay of normal frame in case of 50% bus load	(5)	T_{bit}
Diagnostic response message transmission delay of normal frame in case of 50% bus load (number of CXPI slave nodes is 2)	(6)	T_{bit}
CXPI time of request message	(7)	ms
CXPI time of response message	(7)	ms

[a] "0,9" in the formula represents the sum of IBS in the maximum length of a PID field and response field.

[b] In case of a long frame, T_{pid_normal} is replaced by T_{pid_long}.

IFS (Inter Frame Space) = 20 T_{bit}.

Formula (1):

$$T_{\text{pid\_normal}} = [30 + (10 \times \text{CXPI data field length})] \times (1 + 0{,}9) \tag{1}$$

Formula (2):

$$T_{\text{pid\_long}} = [50 + (10 \times \text{CXPI data field length})] \times (1 + 0{,}9) \tag{2}$$

Formula (3):

$$T_{\text{req\_delay}} = T_{\text{pid\_normal}} + IFS + T_{\text{pid\_normal}} \tag{3}$$

Formula (4):

$$T_{\text{12byte\_normal}} = [30 \times (10 \times 12)] \times (1 + 0{,}9) \tag{4}$$

Formula (5):

$$T_{\text{req\_delay}} = T_{\text{12byte\_normal}} + IFS + T_{\text{pid\_normal}} \tag{5}$$

Formula (6):

$$T_{\text{res\_delay}} = T_{\text{12byte\_normal}} + IFS + T_{\text{pid\_normal}} + IFS + T_{\text{12byte\_normal}} + IFS + T_{\text{pid\_normal}} \tag{6}$$

Formula (7):

$$T_{\text{CXPI}} = \frac{T_{\text{res\_delay}}}{\text{bit\_rate} \times 1\,000} \tag{7}$$

Figure B.1 shows the transmission delay caused by arbitration lost.

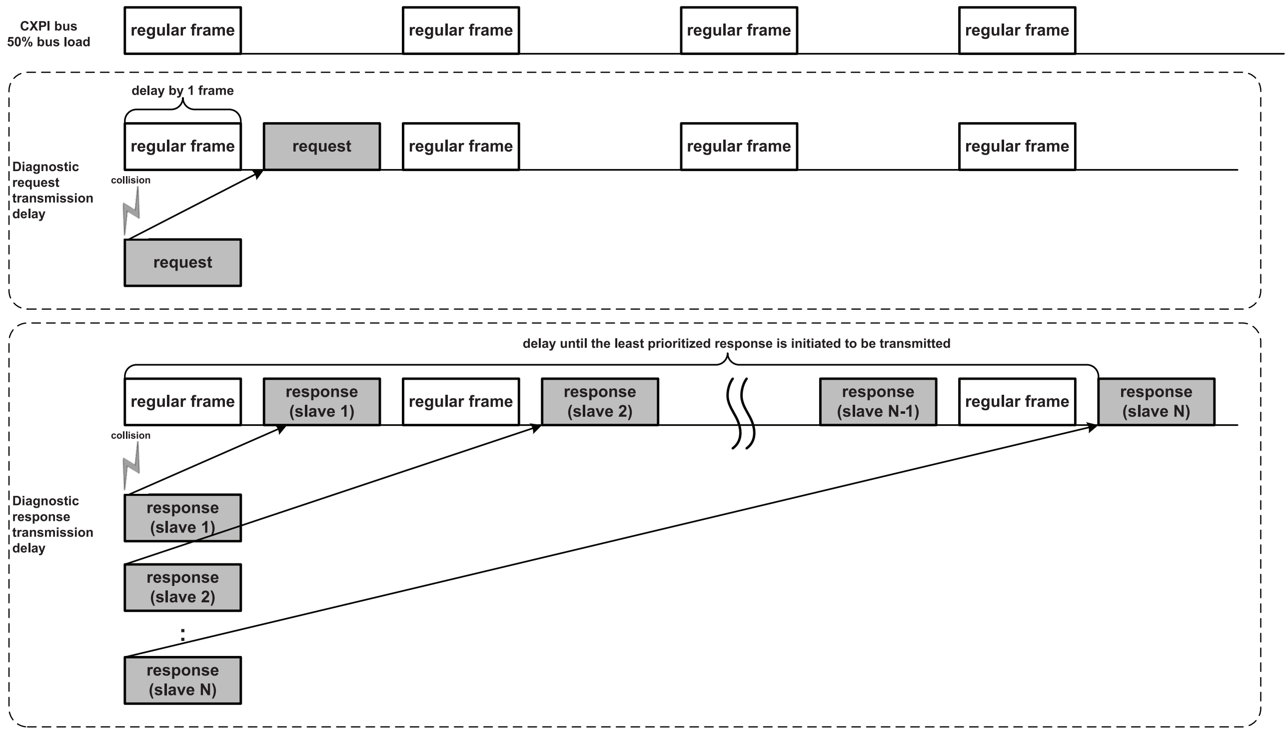

Figure B.1 — Transmission delay caused by arbitration lost

Calculation examples case #1 and case #2 are described below:

- Case #1 is an example that calculates the $P2_{CAN\_Client}$ according to the communication bit rate and the number of CXPI slave nodes on condition that the CXPI data field length satisfies "less than 500 ms $P2_{CXPI\_Server}$".
- Case #2, supposing the connection with CAN FD, calculates the $P2_{CAN\_Client}$ value for the diagnostic request of 64 byte length CXPI data field and the diagnostic response of 255 byte length CXPI data field.

Table B.3 includes the calculation condition (case #1).

Table B.3 — Calculation condition (case #1)

Item	Description
Service used for calculation	ReadDataByIdentifier (SID 22_{16}) Calculation in the case of using ReadDataByIdentifier (SID 22_{16}): CXPI Data field length of diagnostic request: 7 byte (see Table B.4) CXPI Data field length of diagnostic response: 11 byte (see Table B.5)
Relay function of master node	frame relay
Number of CXPI slave nodes	maximum 15 nodes
CXPI communication bit rate	9,6 kbit/s, 10,4 kbit/s, 19,2 kbit/s, 20 kbit/s
CXPI bus load	50 %
Handling (request)	20 ms
Handling (response)	10 ms
CAN network time[a]	20 ms
[a] See Formula (8).	

Definition of Formula:

$$T_{CAN\_network} = T_{GW\_handling\_request} + T_{CAN\_time1\_response} + T_{GW\_handling\_response} + T_{CAN\_time2\_response} \quad (8)$$

Table B.4 illustrates the ReadDataByIdentifier request message.

Table B.4 — Request message of ReadDataByIdentifier

CXPI message						
Byte #1	**Byte #2**	**Byte #3**	**Byte #4**	**Byte #5**	**Byte #6**	**Byte #7**
N_NAD	T_PCI	SID	DATA1	DATA2	DATA3	DATA4
$7E_{16}$	05_{16}	22_{16}	DID #1		DID #2	

Table B.5 illustrates the positive response message for ReadDataByIdentifier. In this example the dataRecord#1 and #2 consists of 2 data bytes.

Table B.5 — Response message of ReadDataByIdentifier

CXPI message										
Byte #1	**Byte #2**	**Byte #3**	**Byte #4**	**Byte #5**	**Byte #6**	**Byte #7**	**Byte #8**	**Byte #9**	**Byte #10**	**Byte #11**
N_NAD	T_PCI	SID	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7	DATA8
Any value	09_{16}	62_{16}	DID #1		dataRecord #1		DID #2		dataRecord #2	

The right end column of Table B.6 includes a proposed setting value of $P2_{CAN\_Client}$. The $P2_{CAN\_Client}$ value is chosen considering the CXPI data field length, the communication bit rate and the number of CXPI slave nodes associated with the SID applied for the product specification.

In case of 9,6 kbit/s, a maximum of 8 CXPI nodes are recommended be connected to keep the $P2_{CAN\_Client}$ less than 500 ms.

Table B.6 includes the calculation results (case #1).

Table B.6 — Calculation results (case #1)

# of CXPI slave nodes	Bit rate [kbit/s]	CXPI data field length (byte)		Handling time (req.)	CXPI time (req.)	Slave handling time	CXPI time (resp.)	Handling time (resp.)	CAN network time	$P2_{CXPI\_Server}$	$P2_{CAN\_Client}$	$P2_{CAN\_Client}$ setting value
		req.	resp.	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]	[ms]
1	9,6	7	11	20	43,8	50	50,4	10	20,632	100,4	194,8	500
	10,4	7	11	20	40,4	50	46,5	10	20,632	96,5	187,6	500
	19,2	7	11	20	21,9	50	25,2	10	20,632	75,2	147,7	500
	20	7	11	20	21,0	50	24,2	10	20,632	74,2	145,8	500
2	9,6	7	11	20	43,8	50	102,9	10	20,632	152,9	247,3	500
	10,4	7	11	20	40,4	50	95,0	10	20,632	145,0	236,0	500
	19,2	7	11	20	21,9	50	51,5	10	20,632	101,5	174,0	500
	20	7	11	20	21,0	50	49,4	10	20,632	99,4	171,0	500
3	9,6	7	11	20	43,8	50	153,3	10	20,632	203,3	297,7	500
	10,4	7	11	20	40,4	50	141,5	10	20,632	191,5	282,6	500
	19,2	7	11	20	21,9	50	76,7	10	20,632	126,7	199,2	500
	20	7	11	20	21,0	50	73,6	10	20,632	123,6	195,2	500
4	9,6	7	11	20	43,8	50	203,8	10	20,632	253,8	348,1	500
	10,4	7	11	20	40,4	50	188,1	10	20,632	238,1	329,1	500
	19,2	7	11	20	21,9	50	101,9	10	20,632	151,9	224,4	500
	20	7	11	20	21,0	50	97,8	10	20,632	147,8	219,4	500
...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...
7	9,6	7	11	20	43,8	50	355,0	10	20,632	405,0	499,4	1 000
	10,4	7	11	20	40,4	50	327,7	10	20,632	377,7	468,7	1 000
	19,2	7	11	20	21,9	50	177,5	10	20,632	227,5	300,0	500
	20	7	11	20	21,0	50	170,4	10	20,632	220,4	292,0	500
8	9,6	7	11	20	43,8	50	405,4	10	20,632	455,4	549,8	1 000
	10,4	7	11	20	40,4	50	374,2	10	20,632	424,2	515,2	1 000
	19,2	7	11	20	21,9	50	202,7	10	20,632	252,7	325,2	500
	20	7	11	20	21,0	50	194,6	10	20,632	244,6	316,2	500
9	9,6	7	11	20	43,8	50	455,8	10	20,632	505,8	600,2	1 000
	10,4	7	11	20	40,4	50	420,8	10	20,632	470,8	561,8	1 000
	19,2	7	11	20	21,9	50	227,9	10	20,632	277,9	350,4	1 000
	20	7	11	20	21,0	50	218,8	10	20,632	268,8	340,4	1 000
...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...
15	9,6	7	11	20	43,8	50	758,3	10	20,632	808,3	902,7	1 500
	10,4	7	11	20	40,4	50	700,0	10	20,632	750,0	841,0	1 500
	19,2	7	11	20	21,9	50	379,2	10	20,632	429,2	501,7	1 000
	20	7	11	20	21,0	50	364,0	10	20,632	414,0	485,6	1 000

Table B.7 defines the calculation condition (case #2).

Table B.7 — Calculation condition (case #2)

Item	Description
Service used for calculation	RoutineControl (SID 31_{16}) Calculation of the case using RoutineControl (SID 31_{16})
CXPI Master node relay function	message relay
Number of CXPI slave nodes	maximum 15 nodes
CXPI communication bit rate	9,6 kbit/s, 10,4 kbit/s, 19,2 kbit/s, 20 kbit/s
CXPI bus load	no load
Handling (request)	20 ms
Handling (response)	10 ms
CAN network timea	20 ms
[a] See Formula (8).	

Table B.8 defines the request message of RoutineControl.

Table B.8 — Request message of RoutineControl

CXPI message									
Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8	...	Byte #64
N_NAD	T_PCI		SID	DATA1	DATA2	DATA3	DATA4	...	DATA60
$7E_{16}$	$003D_{16}$		31_{16}	sub-function	RID#1		routineControl-OptionRecord routineControl-Option#1	...	routineControl-OptionRecord routineControl-Option#57

Table B.9 defines the positive response message for RoutineControl. As an example, suppose that the routineStatusRecord is from #1 to #247.

Table B.9 — Response message of RoutineControl

CXPI message										
Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8	Byte #9	...	Byte #255
N_NAD	T_PCI		SID	DATA1	DATA2	DATA3	DATA4	DATA5	...	DATA251
$7E_{16}$	$00FC_{16}$		71_{16}	routine-Control-Type	RID#1		routine-Info	routineStatus-Record routineStatus#1	...	routineStatus-Record routineStatus#247

The calculation result for case #2 is shown in Table B.10. As most $P2_{CXPI\_Server}$ values exceed 500 ms in case of performing the diagnostic communication with maximum CXPI data field length of CXPI, it is recommended to consider the communication bit rate, the number of CXPI slave nodes and the CXPI data field length to satisfy the $P2_{CXPI\_Server}$ requirement.

Table B.10 includes the calculation results (case #2).

Table B.10 — Calculation results (case #2)

# of CXPI slave nodes	bit rate [bit/s]	CXPI data field length (byte)		handling time (req.) [ms]	CXPI time (req.) [ms]	Slave handling time [ms]	CXPI time (resp.) [ms]	handling time (resp.) [ms]	CAN network time [ms]	$P2_{CXPI\_Server}$ [ms]	$P2_{CAN\_Client}$ [ms]	$P2_{CAN\_Client}$ Setting value [ms]
		req.	resp.									
1	9,6	64	255	20	115,0	50	433,3	10	20,632	483,3	649,0	1 000
	10,4	64	255	20	106,2	50	400,0	10	20,632	450,0	606,8	1 000
	19,2	64	255	20	57,5	50	216,7	10	20,632	266,7	374,8	500
	20	64	255	20	55,2	50	208,0	10	20,632	258,0	363,8	500

Table B.10 *(continued)*

# of CXPI slave nodes	bit rate [bit/s]	CXPI data field length (byte)		handling time (req.) [ms]	CXPI time (req.) [ms]	Slave handling time [ms]	CXPI time (resp.) [ms]	handling time (resp.) [ms]	CAN network time [ms]	$P2_{CXPI\_Server}$ [ms]	$P2_{CAN\_Client}$ [ms]	$P2_{CAN\_Client}$ Setting value [ms]
		req.	resp.									
2	9,6	64	255	20	115,0	50	868,8	10	20,632	918,8	1 084,4	1 500
	10,4	64	255	20	106,2	50	801,9	10	20,632	851,9	1 008,7	1 500
	19,2	64	255	20	57,5	50	434,4	10	20,632	484,4	592,5	1 000
	20	64	255	20	55,2	50	417,0	10	20,632	467,0	572,8	1 000
3	9,6	64	255	20	115,0	50	1 304,2	10	20,632	1 354,2	1 519,8	2 000
	10,4	64	255	20	106,2	50	1 203,8	10	20,632	1 253,8	1 410,6	2 000
	19,2	64	255	20	57,5	50	652,1	10	20,632	702,1	810,2	1 000
	20	64	255	20	55,2	50	626,0	10	20,632	676,0	781,8	1 000
4	9,6	64	255	20	115,0	50	1 739,6	10	20,632	1 789,6	1 955,2	2 500
	10,4	64	255	20	106,2	50	1 605,8	10	20,632	1 655,8	1 812,6	2 000
	19,2	64	255	20	57,5	50	869,8	10	20,632	919,8	1 027,9	1 500
	20	64	255	20	55,2	50	835,0	10	20,632	885,0	990,8	1 500
5	9,6	64	255	20	115,0	50	2 175,0	10	20,632	2 225,0	2 390,6	2 500
	10,4	64	255	20	106,2	50	2 007,7	10	20,632	2 057,7	2 214,5	2 500
	19,2	64	255	20	57,5	50	1 087,5	10	20,632	1 137,5	1 245,6	1 500
	20	64	255	20	55,2	50	1 044,0	10	20,632	1 094,0	1 199,8	1 500
6	9,6	64	255	20	115,0	50	2 610,4	10	20,632	2 660,4	2 826,0	3 000
	10,4	64	255	20	106,2	50	2 409,6	10	20,632	2 459,6	2 616,4	3 000
	19,2	64	255	20	57,5	50	1 305,2	10	20,632	1 355,2	1 463,3	2 000
	20	64	255	20	55,2	50	1 253,0	10	20,632	1 303,0	1 408,8	2 000
7	9,6	64	255	20	115,0	50	3 045,8	10	20,632	3 095,8	3 261,5	3 500
	10,4	64	255	20	106,2	50	2 811,5	10	20,632	2 861,5	3 018,3	3 500
	19,2	64	255	20	57,5	50	1 522,9	10	20,632	1 572,9	1 681,0	2 000
	20	64	255	20	55,2	50	1 462,0	10	20,632	1 512,0	1 617,8	2 000
...	...	...	...	...	...	...	...	...	...	...	...	...
15	9,6	64	255	20	115,0	50	6 529,2	10	20,632	6 579,2	6 744,8	7 000
	10,4	64	255	20	106,2	50	6 026,9	10	20,632	6 076,9	6 233,7	6 500
	19,2	64	255	20	57,5	50	3 264,6	10	20,632	3 314,6	3 422,7	4 000
	20	64	255	20	55,2	50	3 134,0	10	20,632	3 184,0	3 289,8	3 500

Bibliography

[1] ISO/IEC 10731:1994, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

[2] ISO 11898 (all parts), *Road vehicles — Controller area network (CAN)*

[3] ISO 15765 (all parts), *Road vehicles — Diagnostic communication over Controller Area Network (DoCAN)*

ICS 43.180

Price based on 43 pages