
**Road vehicles — Diagnostics on
Controller Area Networks (CAN) —**

**Part 3:
Implementation of unified diagnostic
services (UDS on CAN)**

*Véhicules routiers — Diagnostic sur gestionnaire de réseau de
communication (CAN) —*

*Partie 3: Mise en œuvre des services de diagnostic unifiés (SDU sur
CAN)*



Reference number
ISO 15765-3:2004(E)

AVA Documents
Document Sharing Hub
© ISO 2004

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO 2004

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword	v
Introduction	vi
1 Scope	1
2 Normative references	1
3 Terms, definitions and abbreviated terms	2
4 Conventions	2
5 Unified diagnostic services (UDS) applicability to OSI model	2
6 Application and session layers	2
6.1 Application layer services	2
6.2 Application layer protocol	2
6.3 Application layer and diagnostic session management timing	2
6.3.1 General	2
6.3.2 Application layer timing parameter definitions	4
6.3.3 Session layer timing parameter definitions	6
6.3.4 Client and server timer resource requirements	6
6.3.5 Detailed timing parameter descriptions	9
6.3.6 Error handling	27
7 Network layer interface	29
7.1 General information	29
7.2 FlowControl N_PCI parameter definition	29
7.3 Mapping of A_PDU onto N_PDU for message transmission	29
7.4 Mapping of N_PDU onto A_PDU for message reception	29
8 Standardized diagnostic CAN identifiers	30
8.1 Legislated 11 bit OBD CAN identifiers	30
8.2 Legislated 29 bit OBD CAN identifiers	30
8.3 Enhanced diagnostics 29 bit CAN identifiers	30
8.3.1 General information	30
8.3.2 Structure of 29 bit CAN identifier	31
8.3.3 Structure of address	33
8.3.4 Message retrieval	35
8.3.5 Routing	36
9 Diagnostic services implementation	40
9.1 Unified diagnostic services overview	40
9.2 Diagnostic and communication control functional unit	42
9.2.1 DiagnosticSessionControl (10 hex) service	42
9.2.2 ECUReset (11 hex) service	42
9.2.3 SecurityAccess (27 hex) service	43
9.2.4 CommunicationControl (28 hex) service	43
9.2.5 TesterPresent (3E hex) service	43
9.2.6 SecuredDataTransmission (84 hex) service	44
9.2.7 ControlDTCSetting (85 hex) service	44
9.2.8 ResponseOnEvent (86 hex) service	44
9.2.9 LinkControl (87 hex) service	47
9.3 Data transmission functional unit	47
9.3.1 ReadDataByIdentifier (22 hex) service	47
9.3.2 ReadMemoryByAddress (23 hex) service	47
9.3.3 ReadScalingDataByIdentifier(24 hex) service	48

9.3.4	ReadDataByPeriodicIdentifier (2A hex) service	48
9.3.5	DynamicallyDefineDataIdentifier (2C hex) service.....	54
9.3.6	WriteDataByIdentifier (2E hex) service	54
9.3.7	WriteMemoryByAddress (3D hex) service	54
9.4	Stored data transmission functional unit	54
9.4.1	ReadDTCInformation (19 hex) service	54
9.4.2	ClearDiagnosticInformation (14 hex) service	56
9.5	Input/Output control functional unit.....	56
9.5.1	InputOutputControlByIdentifier (2F hex) service.....	56
9.6	Remote activation of routine functional unit.....	56
9.6.1	RoutineControl (31 hex) service	56
9.7	Upload/Download functional unit	57
9.7.1	RequestDownload (34 hex) service	57
9.7.2	RequestUpload (35 hex) service	57
9.7.3	TransferData (36 hex) service	57
9.7.4	RequestTransferExit (37 hex) service	57
10	Non-volatile server memory programming process.....	58
10.1	General information	58
10.2	Detailed programming sequence.....	61
10.2.1	Programming phase #1 — Download of application software and/or application data.....	61
10.2.2	Programming phase #2 — Server configuration.....	66
10.3	Server reprogramming requirements.....	69
10.3.1	Programmable servers and their categories	69
10.3.2	Requirements for all servers to support programming.....	70
10.3.3	Requirements for programmable servers to support programming	70
10.3.4	Software, data identification and fingerprints	74
10.3.5	Server routine access	77
10.4	Non-volatile server memory programming message flow examples	78
10.4.1	General information	78
10.4.2	Programming phase #1 — Pre-Programming step.....	78
10.4.3	Programming phase #1 — Programming step	79
10.4.4	Programming phase #1 — Post-Programming step	86
Annex A (normative)	Network configuration dataIdentifier definitions	87
Bibliography.....		92

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 15765-3 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 15765 consists of the following parts, under the general title *Road vehicles — Diagnostics on Controller Area Networks (CAN)*:

- *Part 1: General information*
- *Part 2: Network layer services*
- *Part 3: Implementation of unified diagnostic services (UDS on CAN)*
- *Part 4: Requirements for emissions-related systems*

Introduction

This part of ISO 15765 has been established in order to enable the implementation of unified diagnostic services, as specified in ISO 14229-1, on controller area networks (UDS on CAN).

To achieve this, it is based on the Open Systems Interconnection (OSI) Basic Reference Model specified in ISO/IEC 7498 and ISO/IEC 10731, which structures communication systems into seven layers. When mapped on this model, the services specified by ISO 15765 are divided into

- unified diagnostic services (layer 7), specified in this part of ISO 15765,
- network layer services (layer 3), specified in ISO 15765-2,
- CAN services (layers 1 and 2), specified in ISO 11898,

in accordance with Table 1.

Table 1 — Enhanced and legislated OBD diagnostic specifications applicable to the OSI layers

Open Systems Interconnection (OSI) layers	Vehicle manufacturer enhanced diagnostics	Legislated on-board diagnostics (OBD)
Diagnostic application	User defined	ISO 15031-5
Application layer	ISO 15765-3	ISO 15031-5
Presentation layer	N/A	N/A
Session layer	ISO 15765-3	N/A
Transport layer	N/A	N/A
Network layer	ISO 15765-2	ISO 15765-4
Data link layer	ISO 11898-1	ISO 15765-4
Physical layer	User defined	ISO 15765-4

Road vehicles — Diagnostics on Controller Area Networks (CAN) —

Part 3: Implementation of unified diagnostic services (UDS on CAN)

1 Scope

This part of ISO 15765 specifies the implementation of a common set of unified diagnostic services (UDS), in accordance with ISO 14229-1, on controller area networks (CAN) in road vehicles as specified in ISO 11898. It gives the diagnostic services and server memory programming requirements for all in-vehicle servers connected to a CAN network and external test equipment. It does not specify any requirement for the in-vehicle CAN bus architecture.

2 Normative references

The following referenced documents are indispensable for the application 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 14229-1, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements*

ISO 11898-1, *Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling*

ISO 11898-2, *Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit*

ISO 11898-3, *Road vehicles — Controller area network (CAN) — Part 3: Low-speed, fault-tolerant, medium dependent interface¹⁾*

ISO 15031-6, *Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics — Part 6: Diagnostic trouble code definitions¹⁾*

ISO 15765-1, *Road vehicles — Diagnostics on controller area network (CAN) — Part 1: General information*

ISO 15765-2, *Road vehicles — Diagnostics on controller area network (CAN) — Part 2: Network layer service¹⁾*

ISO 15765-4, *Road vehicles — Diagnostics on controller area network (CAN) — Part 4: Requirements for emissions-related systems¹⁾*

SAE J1939-21, *Recommended practice for a serial control and communications vehicle network — Data link layer²⁾*

1) To be published.

2) Society of Automotive Engineers standard.

3 Terms, definitions and abbreviated terms

For the purposes of this document, the terms and definitions given in ISO 14229-1, ISO 15765-1 and ISO 15765-2 and the following abbreviated terms apply.

DA	destination address
ID	identifier
DLC	data length code
GW	gateway
LSB	least significant bit
MSB	most significant bit
NA	network address
SA	source address
SM	subnet mask
TOS	type of service

4 Conventions

This part of ISO 15765 is based on conventions defined in ISO 14229-1, which are guided by OSI Service Conventions (see ISO/TR 8509) as they apply for diagnostic services.

5 Unified diagnostic services (UDS) applicability to OSI model

See Figure 1.

6 Application and session layers

6.1 Application layer services

This part of ISO 15765 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.

6.2 Application layer protocol

This part of ISO 15765 uses the application layer protocol as defined in ISO 14229-1.

6.3 Application layer and diagnostic session management timing

IMPORTANT — Any N_USData.indication with <N_Result> not equal to N_OK that is generated in the server shall not result in a response message from the server application.

6.3.1 General

The following specifies the application layer and session layer timing parameters and how they are handled for the client and the server.

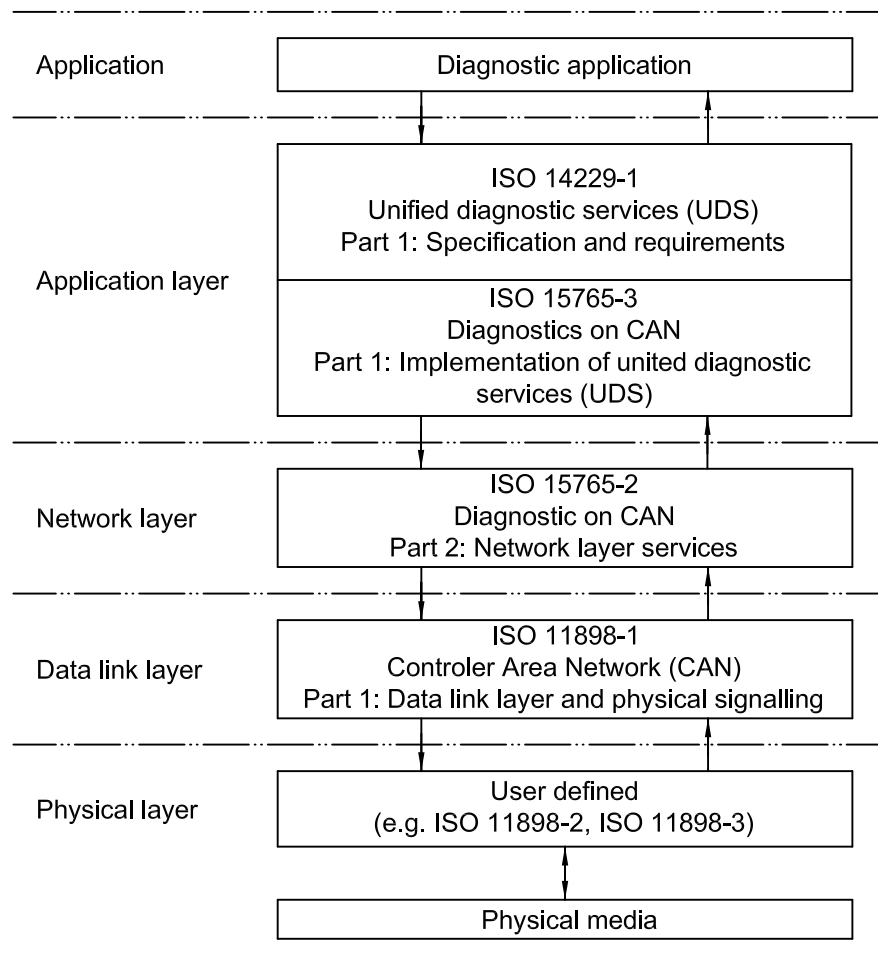


Figure 1 — Implementation of UDS on CAN in OSI model

The following communication scenarios shall be distinguished from one another:

- a) physical communication during
 - 1) default session, and
 - 2) non-default session — session handling required;
- b) functional communication during
 - 1) default session, and
 - 2) non-default session — session handling required.

For all cases, the possibility of requesting an enhanced response-timing window by the server via a negative response message, including a response code 78 hex, shall be considered.

The network layer services as defined in ISO 15765-2 are used to perform the application layer and diagnostic session management timing in the client and the server.

6.3.2 Application layer timing parameter definitions

The application layer timing parameter values for the default diagnostic session shall be in accordance with Table 2.

Table 2 — Application layer timing parameter definitions for the defaultSession

Timing parameter	Description	Type	Min.	Max.
$P2_{CAN_Client}$	Timeout for the client to wait after the successful transmission of a request message (indicated via $N_USData.con$) for the start of incoming response messages ($N_USDataFirstFrame.ind$ of a multi-frame message or $N_USData.ind$ of a SingleFrame message).	Timer reload value	$P2_{CAN_Server_max} + \Delta P2_{CAN}$	N/A ^a
$P2^*_{CAN_Client}$	Enhanced timeout for the client to wait after the reception of a negative response message with response code 78 hex (indicated via $N_USData.ind$) for the start of incoming response messages ($N_USDataFirstFrame.ind$ of a multi-frame message or $N_USData.ind$ of a SingleFrame message).	Timer reload value	$P2^*_{CAN_Server_max} + \Delta P2_{CAN_rsp}$	N/A ^b
$P2_{CAN_Server}$	Performance requirement for the server to start with the response message after the reception of a request message (indicated via $N_USData.ind$).	Performance requirement	0	50 ms
$P2^*_{CAN_Server}$	Performance requirement for the server to start with the response message after the transmission of a negative response message (indicated via $N_USData.con$) with response code 78 hex (enhanced response timing).	Performance requirement	0 ^c	5000 ms
$P3_{CAN_Client_Phys}$	Minimum time for the client to wait after the successful transmission of a physically addressed request message (indicated via $N_USData.con$) with no response required before it can transmit the next physically addressed request message (see 6.3.5.3).	Timer reload value	$P2_{CAN_Server_max}$	N/A ^d
$P3_{CAN_Client_Func}$	Minimum time for the client to wait after the successful transmission of a functionally addressed request message (indicated via $N_USData.con$) before it can transmit the next functionally addressed request message in case no response is required or the requested data is only supported by a subset of the functionally addressed servers (see 6.3.5.3).	Timer reload value	$P2_{CAN_Server_max}$	N/A ^d

^a The maximum time a client waits for a response message to start is at the discretion of the client, provided that $P2_{CAN_Client}$ is greater than the specified minimum value of $P2_{CAN_Client}$.

^b The value that a client uses for $P2^*_{CAN_Client}$ is at the discretion of the client, provided it is greater than the specified minimum value of $P2^*_{CAN_Client}$.

^c During the enhanced response timing, the minimum time between the transmission of consecutive negative messages, each with response code 78 hex, shall be $\frac{1}{2} P2^*_{CAN_Server_max}$ with a maximum tolerance of $\pm 20\%$ of $P2^*_{CAN_Server_max}$.

^d The maximum time a client waits until it transmits the next request message is at the discretion of the client, provided that for non-default sessions the $S3_{Server}$ timing is kept active in the server(s).

The parameter $\Delta P2_{CAN}$ considers any system network design-dependent delays such as delays introduced by gateways and bus bandwidth plus a safety margin (e.g. 50 % of worst case). The worst-case scenario (transmission time necessary for one “round trip” from client to server and back from server to client), based on system design, is impacted by

- the number of gateways involved,
- CAN frame transmission time (baud rate),
- CAN bus utilization, and
- the CAN device driver implementation method (polling vs interrupt) and processing time of the network layer.

The value of $\Delta P2_{CAN}$ is divided into the time to transmit the request to the addressed server and the time to transmit the response to the client:

$$\Delta P2_{CAN} = \Delta P2_{CAN_Req} + \Delta P2_{CAN_Rsp}$$

Figure 2 provides an example of how $\Delta P2_{CAN}$ can be composed.

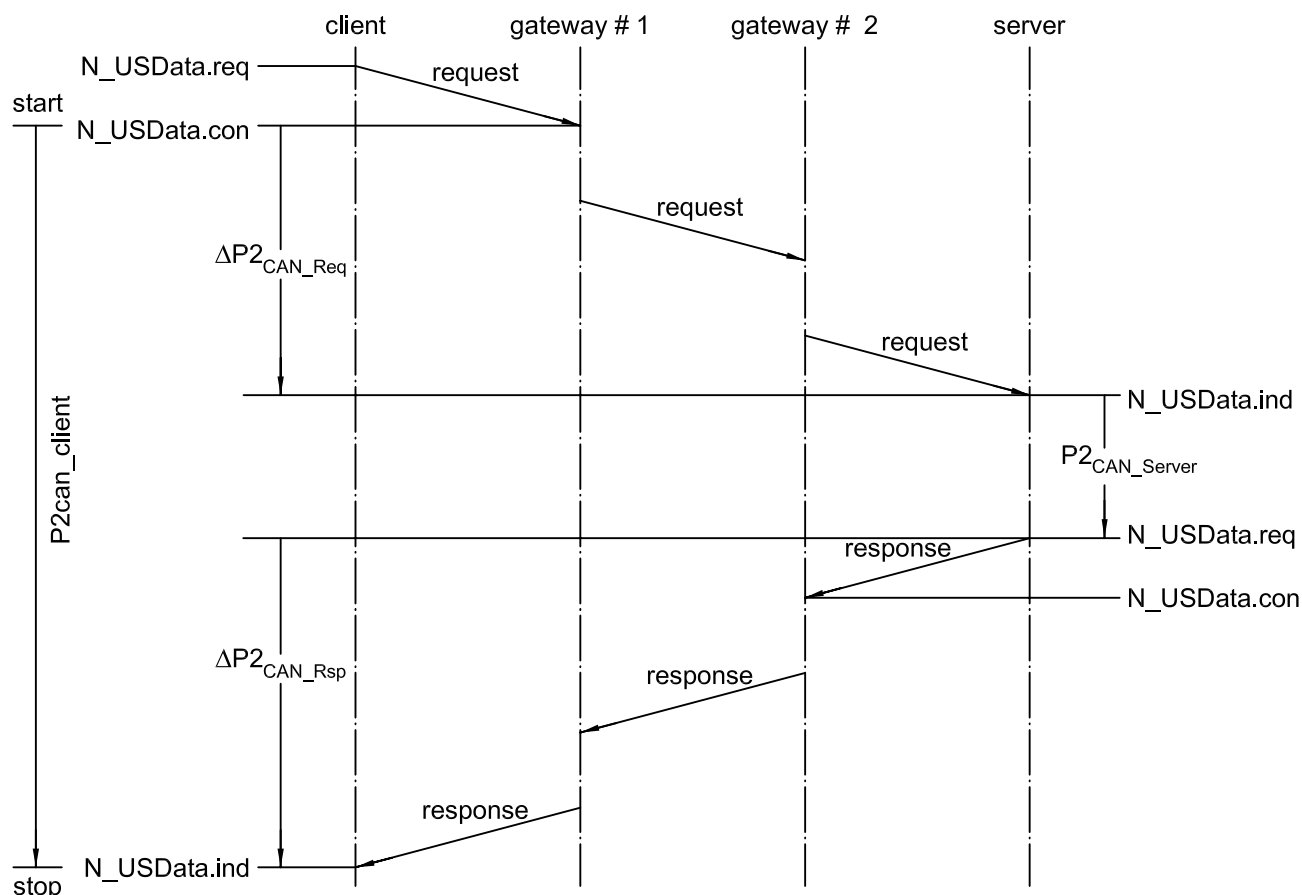


Figure 2 — Example for $\Delta P2_{CAN}$ — SingleFrame request and response message

NOTE For the sake of simplicity in describing the timing parameters, in all the figures that follow it is assumed that the client and the server are located on the same network. All descriptions and figures are presented in a time-related sequential order.

6.3.3 Session layer timing parameter definitions

When a diagnostic session other than the defaultSession is started, then a session handling is required which is achieved via the session layer timing parameter given in Table 3.

Table 3 — Session layer timing parameter definitions

Timing parameter	Description	Type	Recommended timeout ms	Timeout ms
S3 _{Client}	Time between functionally addressed TesterPresent (3E hex) request messages transmitted by the client to keep a diagnostic session other than the defaultSession active in multiple servers (functional communication) or maximum time between physically transmitted request messages to a single server (physical communication).	Timer reload value	2 000 ms	4 000 ms
S3 _{Server}	Time for the server to keep a diagnostic session other than the defaultSession active while not receiving any diagnostic request message.	Timer reload value	N/A	5 000 ms

Furthermore, the server might change its application layer timings $P2_{CAN_Server}$ and $P2^*_{CAN_Server}$ when transitioning into a non-default session in order to achieve a certain performance or to compensate restrictions which might apply during a non-default diagnostic session. The applicable timing parameters for a non-default diagnostic session are reported in the DiagnosticSessionControl positive response message in the case where a response is required to be transmitted (see service description in 9.2.1) or have to be known in advance by the client in case no response is required to be transmitted. When the client starts a non-default session functionally, then it shall adapt to the timing parameters of the responding servers.

Table 4 defines the conditions for the client and the server to start/restart its S3_{Client}/S3_{Server} timer. For the client a periodically transmitted functionally addressed TesterPresent (3E hex) request message shall be distinguished from a sequentially transmitted physically addressed TesterPresent (3E hex) request message, which is only transmitted in case of the absence of any other diagnostic request message. For the server there is no need to distinguish between that kind of TesterPresent (3E hex) handling. Furthermore, Table 4 shows that the S3_{Server} timer handling is based on the network layer service primitives, which means that the S3_{Server} timer is also restarted upon the reception of a diagnostic request message that is not supported by the server.

6.3.4 Client and server timer resource requirements

The timer resource required for the client and the server to fulfil the above given timing requirements during the default session and any non-default session shall be in accordance with Tables 5 and 6 list. During a non-default session, the additional timer resource requirements given in Table 6 shall apply for the client and the server.

Table 4 — Session layer timing start/stop conditions for the client and the server

Timing Parameter	Action	Physical and functional communication, using functionally addressed, periodically transmitted TesterPresent request message	Physical communication only, using a physically addressed, sequentially transmitted TesterPresent request message
S3 _{Client}	Initial start	N_USData.con that indicates the completion of the DiagnosticSessionControl (10 hex) request message. This is only true for if the session type is a non-default session.	N_USData.con that indicates the completion of the DiagnosticSessionControl (10 hex) request message in case no response is required.
			N_USData.ind that indicates the reception of the DiagnosticSessionControl (10 hex) response message in case a response is required.
	Subsequent start	N_USData.con that indicates the completion of the functionally addressed TesterPresent (3E hex) request message, which is transmitted each time the S3 _{Client} timer times out.	N_USData.con that indicates the completion of any request message in case no response is required.
			N_USData.ind that indicates the reception of any response message in case a response is required.
S3 _{Server}	Initial start	N_USData.con that indicates the completion of the transmission of a DiagnosticSessionControl positive response message for a transition from the default session to a non-default session, in case a response message is required.	Successful completion of the requested action of the service DiagnosticSessionControl (10 hex) for a transition from the default session to a non-default session, in case no response message is required/allowed.
	Subsequent stop	N_USDataFirstFrame.ind that indicates the start of a multi-frame request message or N_USData.ind that indicates the reception of any SingleFrame request message. If the defaultSession is active, the S3 _{Server} timer is disabled.	
	Subsequent start	N_USData.con that indicates the completion of any response message that concludes a service execution (final response message) in case a response message is required/allowed to be transmitted (this includes positive and negative response messages). A negative response with response code 78 hex does not restart the S3 _{Server} timer.	
			Completion of the requested action (service conclusion) in case no response message (positive and negative) is required/allowed.
			N_USData.ind that indicates an error during the reception of a multi-frame request message.
		See 6.3.5.4 for further details regarding the S3 _{Server} handling in the server when the server is requested to transmit unsolicited response message such as periodic data or responses based on an event.	

Table 5 — Timer resources requirements during defaultSession

Timing parameter	Client	Server
P2 _{CAN_Client}	A single timer is required for each logical communication channel (physical and functional communication), e.g. each point-to-point communication requires a separate communication channel.	N/A
P2 _{CAN_Server}	N/A	An optional timer might be required for the enhanced response timing in order to ensure that subsequent negative response messages with response code 78 hex are transmitted prior to the expiration of P2* _{CAN_Server} .
P3 _{CAN_Physical}	A single timer is required per logical physical communication channel.	N/A
P3 _{CAN_Functional}	A single timer is required per logical functional communication channel.	N/A

Table 6 — Additional timer resources requirements during non-defaultSession

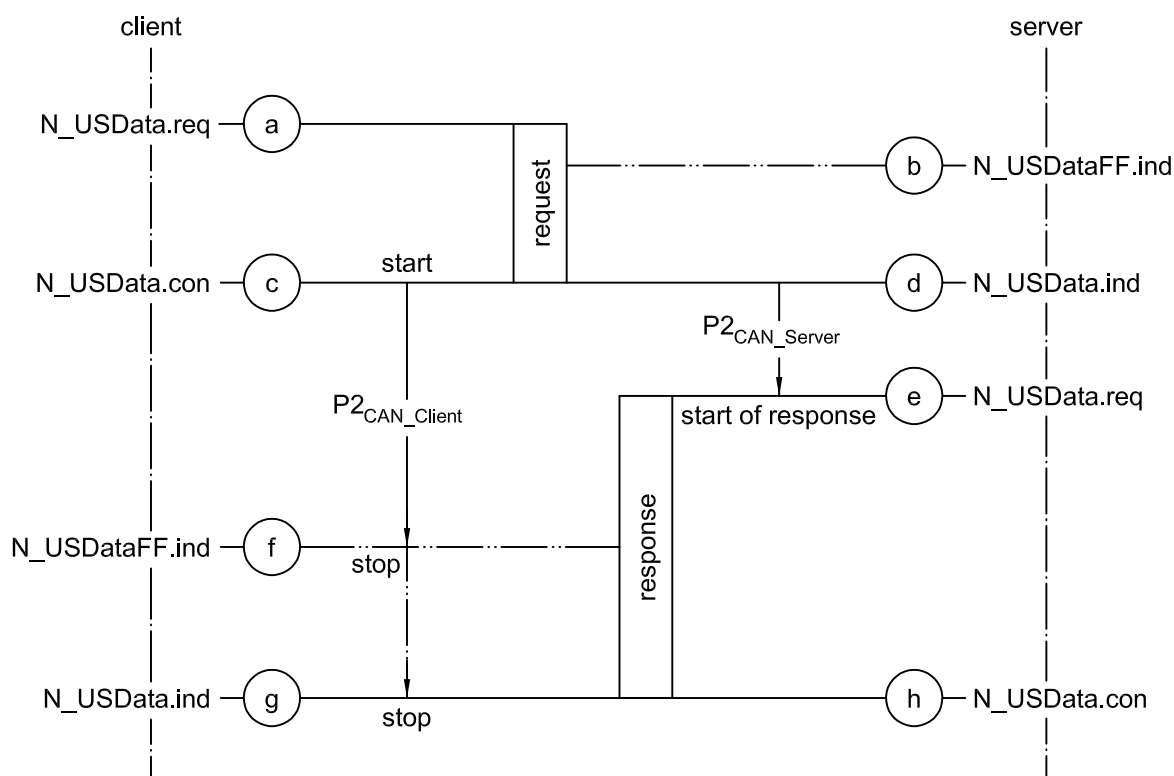
Timing Parameter	Client	Server
S3 _{Client}	A single timer is required when using a periodically transmitted, functionally addressed TesterPresent (3E) hex request message to keep the servers in a non-defaultSession. There is no need for additional timers per activated diagnostic sessions.	N/A
	A single timer is required for each point-to-point communication channel when using a sequentially transmitted, physically addressed TesterPresent (3E) hex request message to keep a single server in a non-defaultSession in case of the absence of another diagnostic request message then.	
S3 _{Server}	N/A	A single timer is required in the server, because only a single diagnostic session can be active at a time in a single server.

6.3.5 Detailed timing parameter descriptions

6.3.5.1 Physical communication

6.3.5.1.1 Physical communication during defaultSession

Figure 3 graphically depicts the timing handling in the client and the server for a physically addressed request message during the default session.

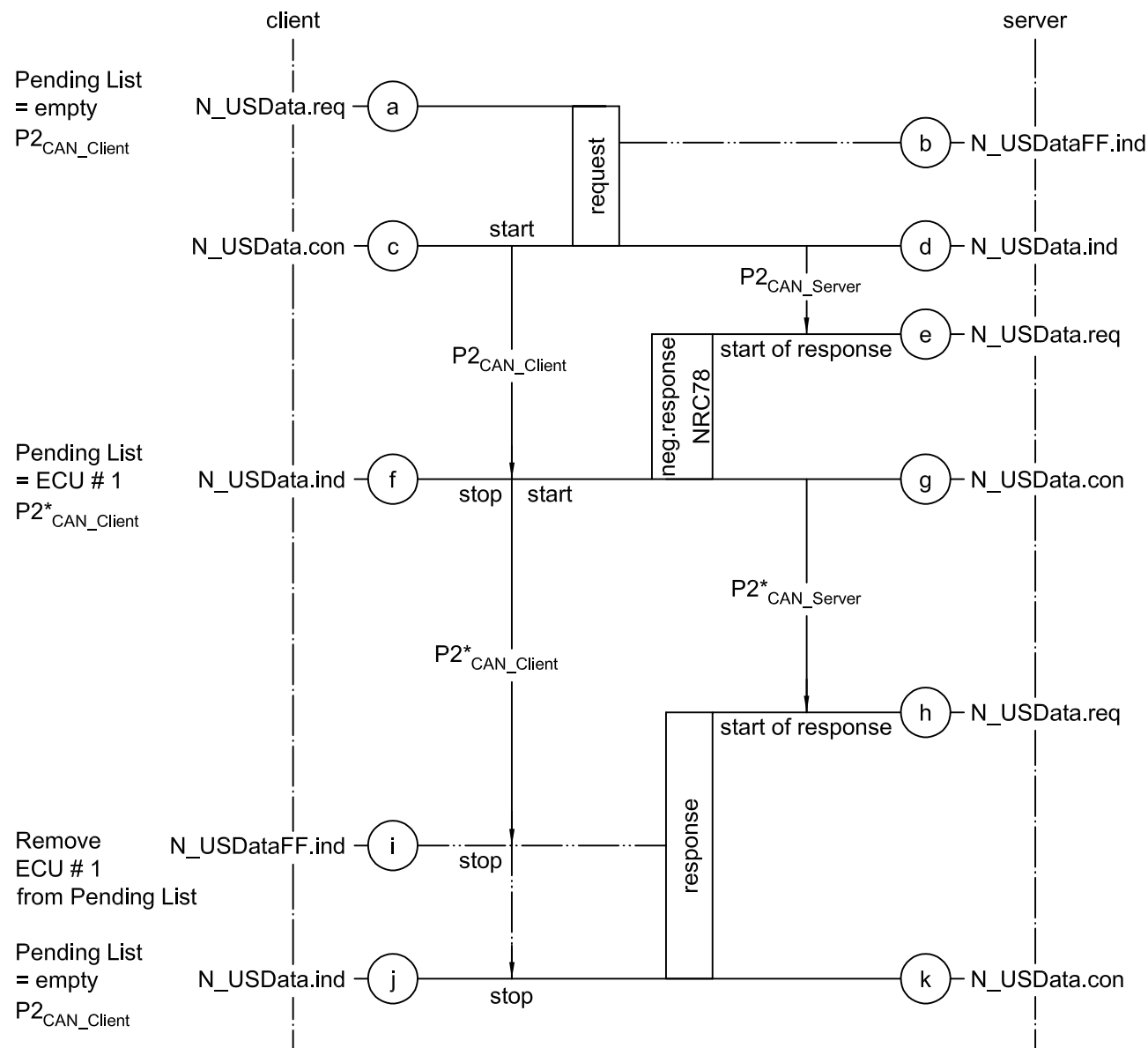


- a The diagnostic application of the client starts the transmission of the request message by issuing a $N_USData.req$ to its network layer. The network layer transmits the request message to the server. The request message can either be a single-frame message or a multi-frame message.
- b In the case of a multi-frame message, the start of the request is indicated in the server via $N_USDataFF.ind$ that is issued by its network layer.
- c The completion of the request message is indicated in the client via $N_USData.con$. When receiving the $N_USData.con$ the client starts its $P2_{CAN_Client}$ timer, using the default reload value $P2_{CAN_Client}$. The value of the $P2_{CAN_Client}$ timer shall consider any latency that is involved based on the vehicle network design (communication over gateways, bus bandwidth, etc.). For simplicity, the figure assumes that the client and the server are located on the same network.
- d The completion of the request message is indicated in the server via the $N_USData.ind$.
- e The server is required to start with its response message within $P2_{CAN_Server}$ after the reception of $N_USData.ind$. This means that, in the case of a multi-frame response message, the FirstFrame shall be sent within $P2_{CAN_Server}$ and, for single-frame response messages, that the SingleFrame shall be sent within $P2_{CAN_Server}$.
- f In the case of a multi-frame response message, the reception of the FirstFrame is indicated in the client via the $N_USDataFF.ind$ of its network layer. When receiving the FirstFrame indication, the client stops its $P2_{CAN_Client}$ timer.
- g The network layer will generate a final $N_USData.ind$ in case the complete message is received or an error occurred during the reception. In case of a single-frame response message, the reception of the SingleFrame is indicated in the client via a single $N_USData.ind$. When receiving this single frame indication, the client stops its $P2_{CAN_Client}$ timer.
- h The completion of the response message is indicated in the server via $N_USData.con$.

Figure 3 — Physical communication during default session

6.3.5.1.2 Physical communication during defaultSession with enhanced response timing

Figure 4 graphically depicts the timing handling in the client and the server for a physically addressed request message during the default session and the request of the server for an enhanced response timing (negative response code 78 hex handling).



a The diagnostic application of the client starts the transmission of the request message by issuing a `N_USData.req` to its network layer. The network layer transmits the request message to the server. The request message can either be a single-frame or multi-frame message.

b In the case of a multi-frame message, the start of the request is indicated in the server via `N_USDataFF.ind` that is issued by its network layer.

c The completion of the request message is indicated in the client via `N_USData.con`. When receiving the `N_USData.con`, the client starts its $P2_{CAN_Client}$ timer, using the default reload value $P2_{CAN_Client}$. The value of the $P2_{CAN_Client}$ timer shall consider any latency that is involved based on the vehicle network design (e.g. communication over gateways, bus bandwidth, etc.). For simplicity, the figure assumes that the client and the server are located on the same network.

d The completion of the request message is indicated in the server via `N_USData.ind`.

- ^e The server is required to start with its response message within $P2_{CAN_Server}$ after the reception of $N_USData.ind$. This means that, in the case of a multi-frame response message, the FirstFrame shall be sent within $P2_{CAN_Server}$ and, for single-frame response messages, that the SingleFrame shall be sent within $P2_{CAN_Server}$.
- ^f In case the server cannot provide the requested information within the $P2_{CAN_Server}$ response timing, it can request an enhanced response timing window by sending a negative response message including response code 78 hex. Upon reception of the negative response message within the client, the client network layer generates a $N_USData.ind$. The reception of a negative response message with response code 78 hex causes the client to restart its $P2_{CAN_Client}$ timer, but using the enhanced reload value $P2^*_{CAN_Client}$.
- ^g The server is required to start with its response message within the enhanced $P2_{CAN_Server}$ ($P2^*_{CAN_Server}$) following the $N_USData.con$ of the transmitted negative response message. In case the server can still not provide the requested information within the enhanced $P2^*_{CAN_Server}$, then a further negative response message including response code 78 hex can be sent by the server. This will cause the client to restart its $P2_{CAN_Client}$ timer, using the enhanced reload value $P2^*_{CAN_Client}$. For simplicity, the figure only shows a single negative response message with response code 78 hex.
- ^h Once the server can provide the requested information (positive or negative response other than response code 78 hex), it starts with its final response message.
- ⁱ In the case of a multi-frame final response message, the reception of the FirstFrame is indicated in the client via the $N_USDataFF.ind$ of the network layer. When receiving the FirstFrame indication, the client stops its $P2_{CAN_Client}$ timer.
- ^j The network layer of the client will generate a final $N_USData.ind$ in case the complete message is received or an error occurred during the reception. In the case of a single-frame response message, the reception of the SingleFrame is indicated in the client via a single $N_USData.ind$. When receiving this single-frame indication, the client stops its $P2_{CAN_Client}$ timer.
- ^k The completion of the transmission will also be indicated in the server via $N_USData.con$.

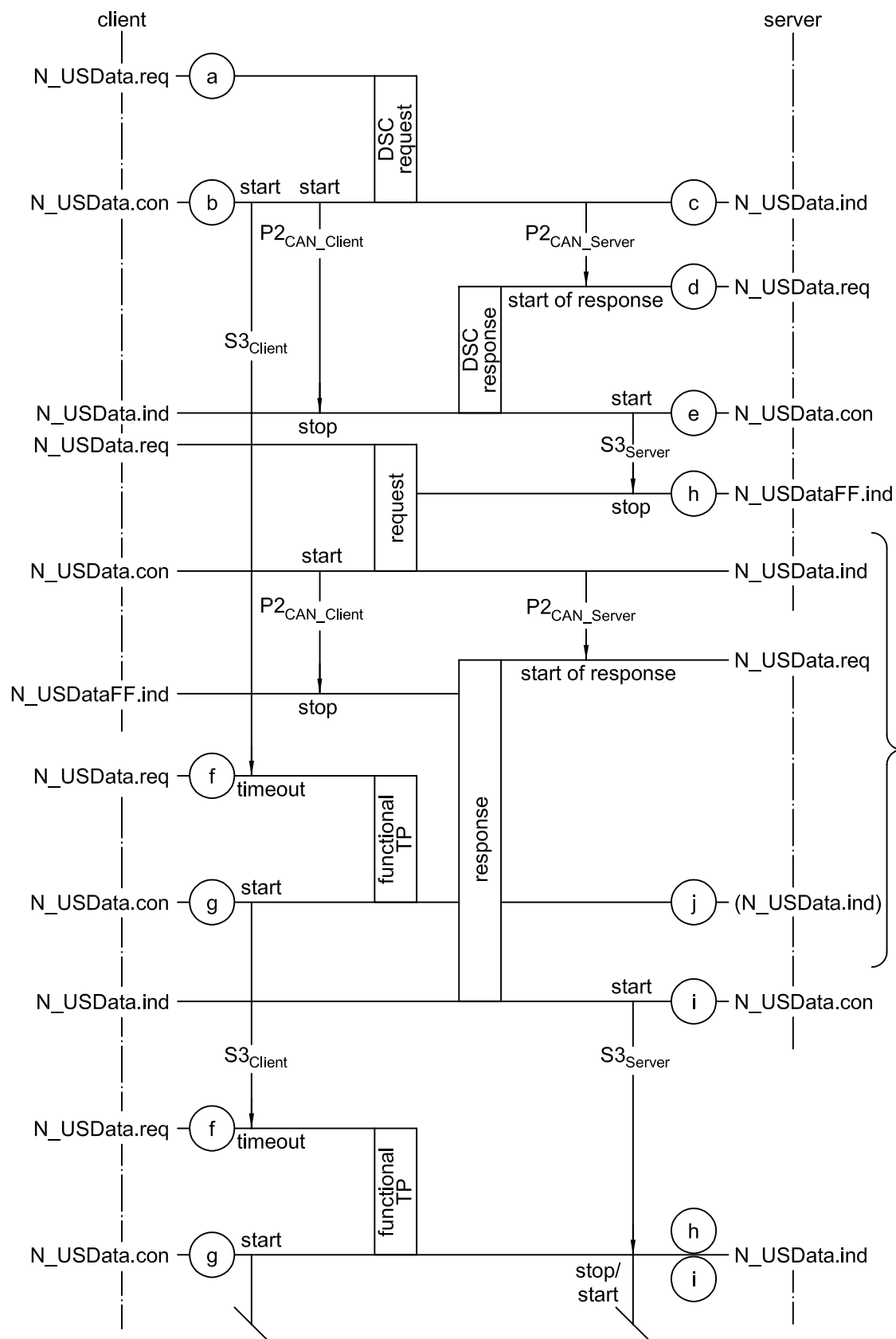
Figure 4 — Physical communication during non-default session — Enhanced response timing

6.3.5.1.3 Physical communication during a non-default session

6.3.5.1.3.1 Functionally addressed TesterPresent (3E hex) message

Figure 5 graphically depicts the timing handling in the client and the server when performing physical communication during a non-default session (e.g. programmingSession) and using a functionally addressed, periodically transmitted TesterPresent (3E hex) request message that does not require a response message from the server.

The handling of the $P2_{CAN_Client}$ and $P2_{CAN_Server}$ timing is identical to the handling as described in 6.3.5.1.1 and 6.3.5.1.2. The only exception is that the reload values on the client side and the resulting time where the server shall send its final response time might differ. This is based on the transition into a session other than the default session where different $P2_{CAN_Client}$ timing parameters might apply (see DiagnosticSessionControl (10 hex) service in 9.2.1 for details on how the timing parameters are reported to the client).



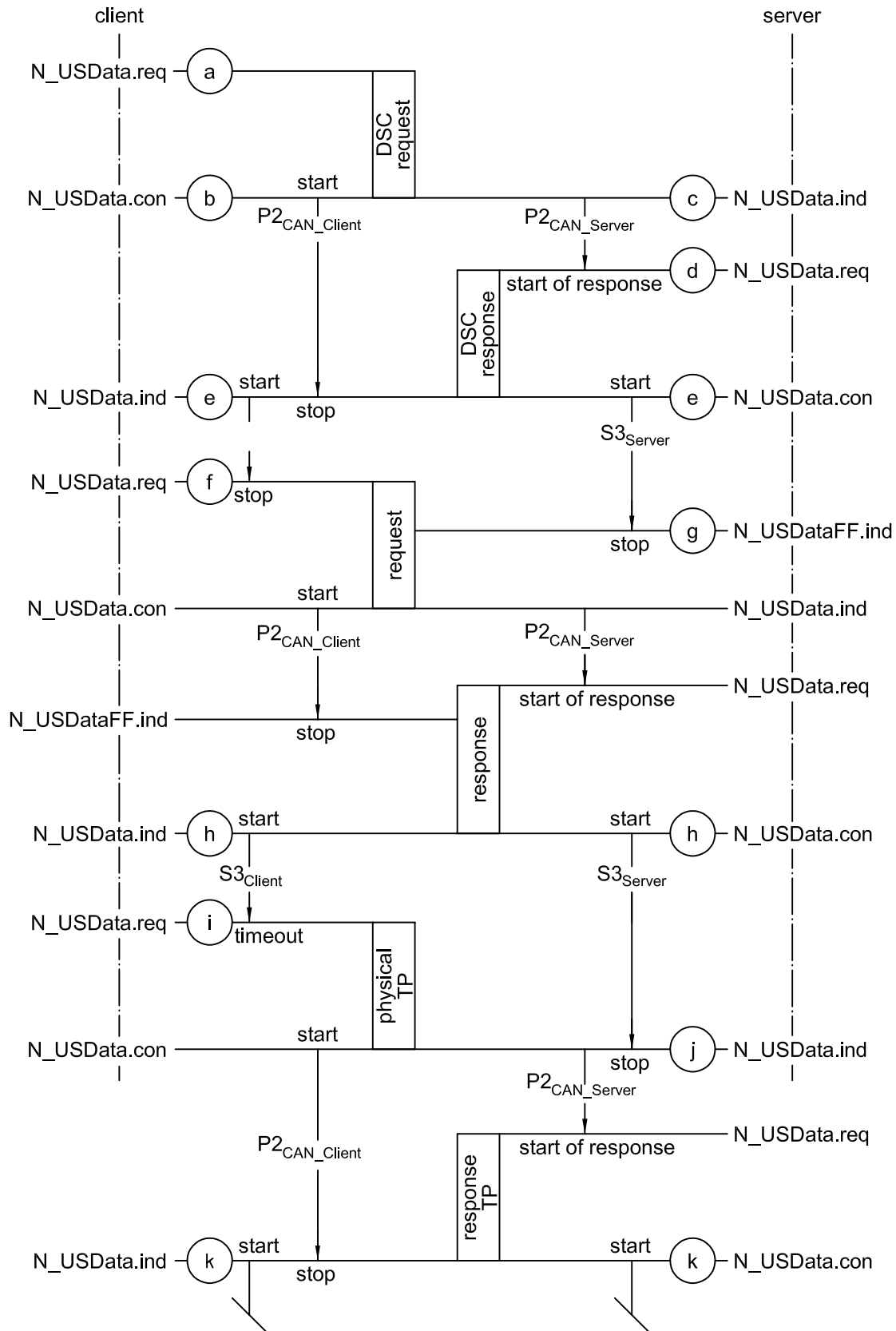
Any Tester Present that
is received during a
disabled S3server timer
will be ignored by the server

- a The diagnostic application of the client starts the transmission of the DiagnosticSessionControl (10 hex) request message by issuing a N_USData.req to its network layer. The network layer transmits the request message to the server.
- b The request message is a single-frame message. Its completion is indicated in the client via the N_USData.con. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies. The generated N_USData.con in the client causes the start of the S3_{Client} timer (session timer).
- c The completion of the request message is indicated in the server via the N_USData.ind. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.
- d For the figure given, it is assumed that the client requires a response from the server. The server shall transmit the DiagnosticSessionControl (10 hex) positive response message.
- e The completion of the transmission of the response message is indicated in the server via N_USData.con. Now the server starts its S3_{Server} timer, which keeps the activated non-default session active as long as it does not time out. It is the client's responsibility to ensure that the S3_{Server} timer is reset prior to its timeout to keep the server in the non-default session.
- f Once the S3_{Client} timer is started in the client, this causes the transmission of a functionally addressed TesterPresent (3E hex) request message, which does not require a response message, each time the S3_{Client} timer times out.
- g Upon the indication of the completed transmission of the TesterPresent (3E hex) request message via N_USData.con of its network layer, the client once again starts its S3_{Client} timer. This means that the functionally addressed TesterPresent (3E hex) request message is sent on a periodic basis every time S3_{Client} times out.
- h Any time the server is in the process of handling any diagnostic service, it stops its S3_{Server} timer.
- i When the diagnostic service is completely processed, then the server restarts its S3_{Server} timer. This means that any diagnostic service, including TesterPresent (3E hex), resets the S3_{Server} timer. A diagnostic service is meant to be in progress any time between the start of the reception of the request message (N_USDataFF.ind or N_USData.ind receive) and the completion of the transmission of the final response message, where a response message is required, or the completion of any action that is caused by the request, where no response message is required (point in time reached that would cause the start of the response message).
- j Any TesterPresent (3E hex) request message that is received during processing another request message can be ignored by the server, because it has already stopped its S3_{Server} timer and will restart it once the service that is in progress is processed completely.

Figure 5 — Physical communication during non-default session – functionally addressed TesterPresent

6.3.5.1.3.2 Physically addressed TesterPresent (3E hex) message

Figure 6 graphically depicts the timing handling in the client and the server when performing physical communication during a non-default session (e.g. programmingSession) and using a physically addressed TesterPresent (3E hex) request message that requires a response message from the server to keep the diagnostic session active in case of the absence of any other diagnostic service.



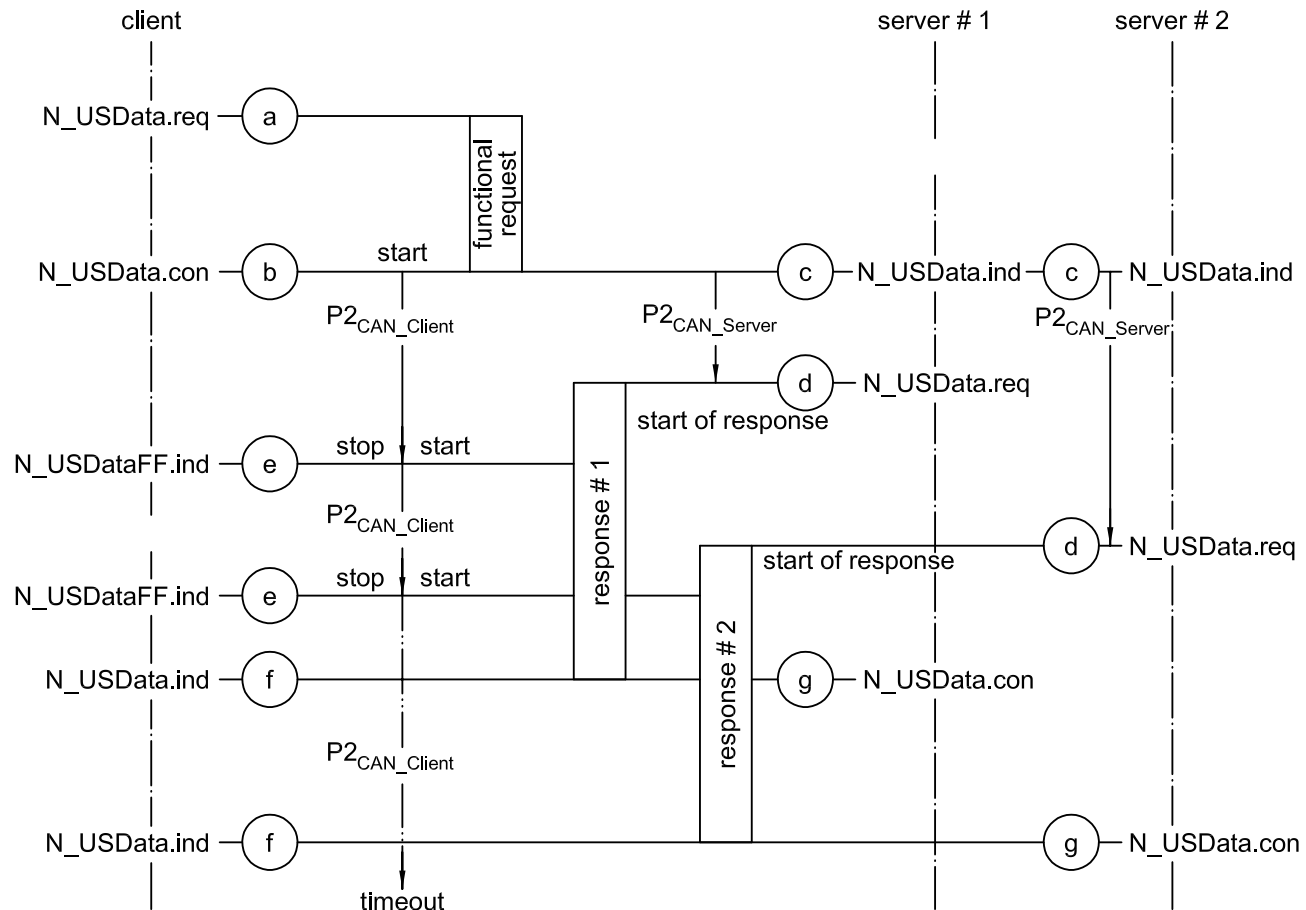
- a The diagnostic application of the client starts the transmission of the DiagnosticSessionControl (10 hex) request message by issuing a N_USData.req to its network layer. The network layer transmits the request message to the server.
- b The request message is a single-frame message. Its completion is indicated in the client via the N_USData.con. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies. The generated N_USData.con in the client does not cause the start of the S3_{Client} timer (session timer), as it would for the case of using a functionally addressed and periodically transmitted TesterPresent (3E hex) message to keep a diagnostic session alive (see 6.3.5.1.3.1).
- c The completion of the request message is indicated in the server via the N_USData.ind. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.
- d For the figure given, it is assumed that the client requires a response from the server. The server shall transmit the DiagnosticSessionControl (10 hex) positive response message.
- e The completion of the transmission of the response message is indicated in the server via N_USData.con. Now the server starts its S3_{Server} timer, which keeps the activated non-default session active as long as it does not time out. In the client, the reception of the DiagnosticSessionControl (10 hex) positive response message is indicated via N_USData.ind. This causes the start of the S3_{Client} timer. It is the client's responsibility to ensure that the S3_{Server} timer is reset prior to its timeout to keep the server in the non-default session.
- f Whenever the client transmits a request message to the server (including the TesterPresent (3E hex) message), it stops its S3_{Client} timer.
- g The reception of either a SingleFrame or a FirstFrame of the request message stops the S3_{Server} timer in the server. The completion of the request message is indicated in the server via N_USData.ind. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.
- h The completion of the response message is indicated in the client via N_USData.ind, which causes the client to start its S3_{Client}. The completion of the response message is indicated in the server via N_USData.con, which causes the server to start its S3_{Server}. In a case where the client would not require a response message, then it shall start its S3_{Client} timer when it receives confirmation of the completion of the request message, which is indicated via N_USData.con. The server would start its S3_{Server} timer when it has completed the requested action. For simplicity, the figure shows that a response is required.
- i In case the client would not send any diagnostic request message prior to the timeout of S3_{Client}, then the timeout of the S3_{Client} timer causes the client to transmit a physically addressed TesterPresent (3E hex) request message.
- j The reception of the TesterPresent (3E hex) request message is indicated in the server via N_USData.ind. This causes the server to stop its S3_{Server} timer. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.
- k The completion of the TesterPresent (3E hex) response message is indicated in the client via N_USData.ind, which causes the client to start its S3_{Client}. The completion of the TesterPresent (3E hex) response message is indicated in the server via N_USData.con, which causes the server to start its S3_{Server}. In the case where the client would not require a response message, then it shall start its S3_{Client} timer when it receives confirmation of the completion of the TesterPresent (3E hex) request message, which is indicated via N_USData.con. The server would start its S3_{Server} timer when it has completed the requested action. For simplicity, the figure shows that a response is required.

Figure 6 — Physical communication during non-default session — Physically addressed TesterPresent

6.3.5.2 Functional communication

6.3.5.2.1 Functional communication during defaultSession

Figure 7 graphically depicts the timing handling in the client and two (2) servers for a functionally addressed request message during the default session. From a server point of view, there is no difference in the timing handling compared to a physically addressed request message, but the client shall handle the timing different compared to physical communication.



a The diagnostic application of the client starts the transmission of a functionally addressed request message by issuing a N_USData.req to its network layer. The network layer transmits the request message to the servers. A functionally addressed request message shall only be a single-frame message.

b The completion of the request message is indicated in the client via N_USData.con. When receiving the N_USData.con, the client starts its $P2_{CAN_Client}$ timer, using the default reload value $P2_{CAN_Client}$. As for physical communication, the value of the $P2_{CAN_Client}$ timer shall consider any latency that is involved based on the vehicle network design (e.g. communication over gateways, bus bandwidth, etc.). For simplicity, the figure assumes that the client and the server are located on the same network.

c The completion of the request message is indicated in the servers via N_USData.ind.

d The functionally addressed servers are required to start with their response messages within $P2_{CAN_Server}$ after the reception of N_USData.ind. This means that in case of multi-frame response messages, the FirstFrame shall be sent within $P2_{CAN_Server}$ and, for single-frame response messages, that the SingleFrame shall be sent within $P2_{CAN_Server}$.

e In the case of a multi-frame response message, the reception of the FirstFrame from any server is indicated in the client via the N_USDataFF.ind of the network layer. A single-frame response message is indicated via N_USData.ind.

f When receiving the FirstFrame/SingleFrame indication of an incoming response message, the client either stops its $P2_{CAN_Client}$ where it knows the servers expected to respond and all servers have responded, or it restarts its $P2_{CAN_Client}$ timer where not all expected servers have yet responded or where the client does not know the servers expected to respond (the client awaits the start of further response messages). The network layer of the client will generate a final

N_USData.ind in case the complete message is received or an error occurred during the reception. The reception of a final N_USData.ind of a multi-frame message in the client will not have any influence on the P2_{CAN_Client} timer.

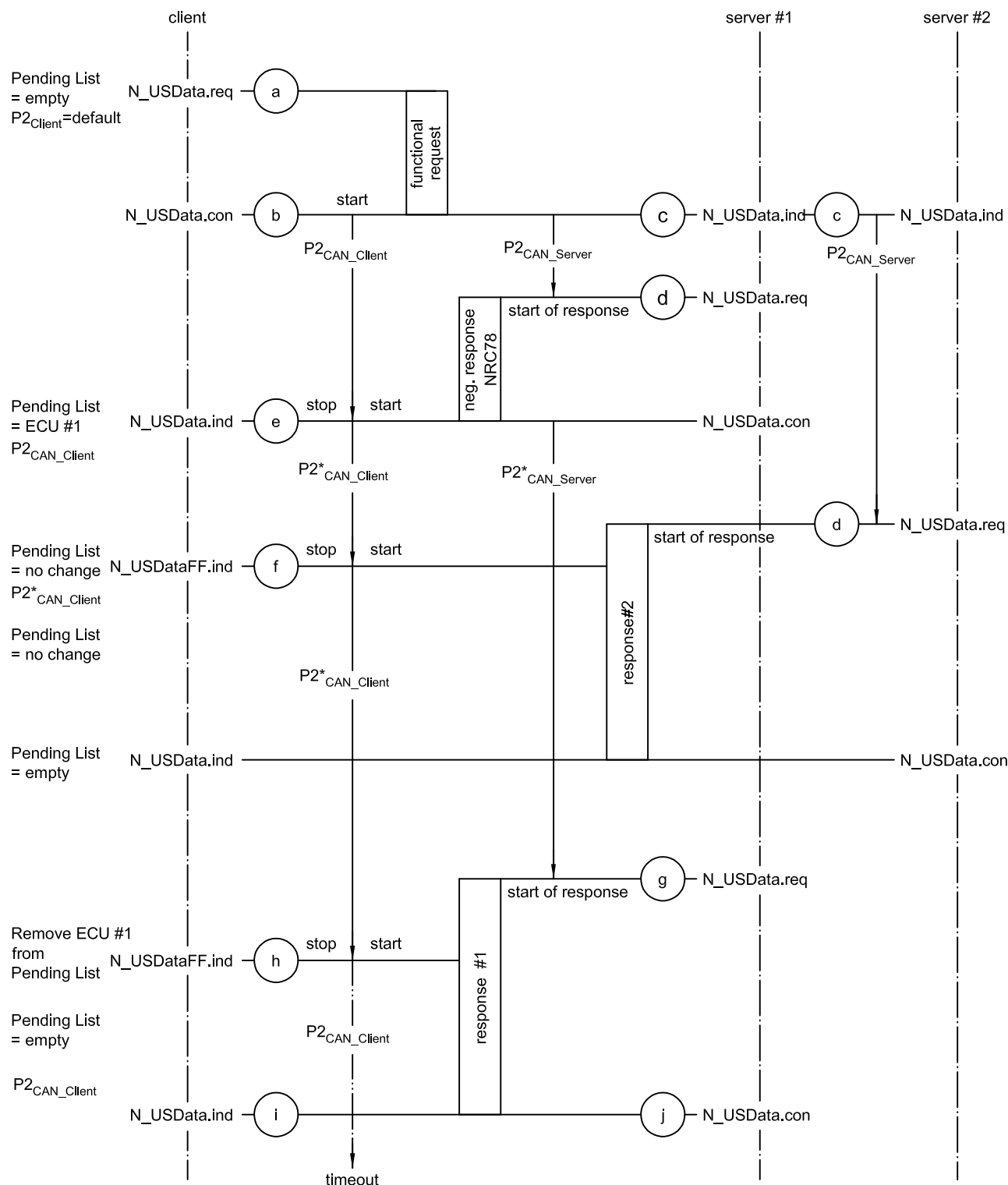
9 The completion of the transmission of the response message will also be indicated in the servers via N_USData.con.

Figure 7 — Functional communication during default session

6.3.5.2.2 Functional communication during defaultSession with enhanced response timing

Figure 8 graphically depicts the timing handling in the client and two (2) servers for a functionally addressed request message during the default session, where one server requests an enhanced response timing via a negative response message including response code 78 hex.

From a server point of view there is no difference in the timing handling compared to a physically addressed request message that requires enhanced response timing, but the client shall handle the timing differently compared to physical communication.



a The diagnostic application of the client starts the transmission of the functionally addressed request message by issuing a N_USData.req to its network layer. The network layer transmits the request message to the servers. A functionally addressed request message shall only be a single-frame message.

b The completion of the request message is indicated in the client via N_USData.con. When receiving N_USData.con, the client starts its P2_{CAN_Client} timer, using the default reload value P2_{CAN_Client}. As for physical communication, the value of the P2_{CAN_Client} timer shall consider any latency that is involved based on the vehicle network design (communication over gateways, bus bandwidth, etc.). For simplicity, the figure assumes that the client and the server are located on the same network.

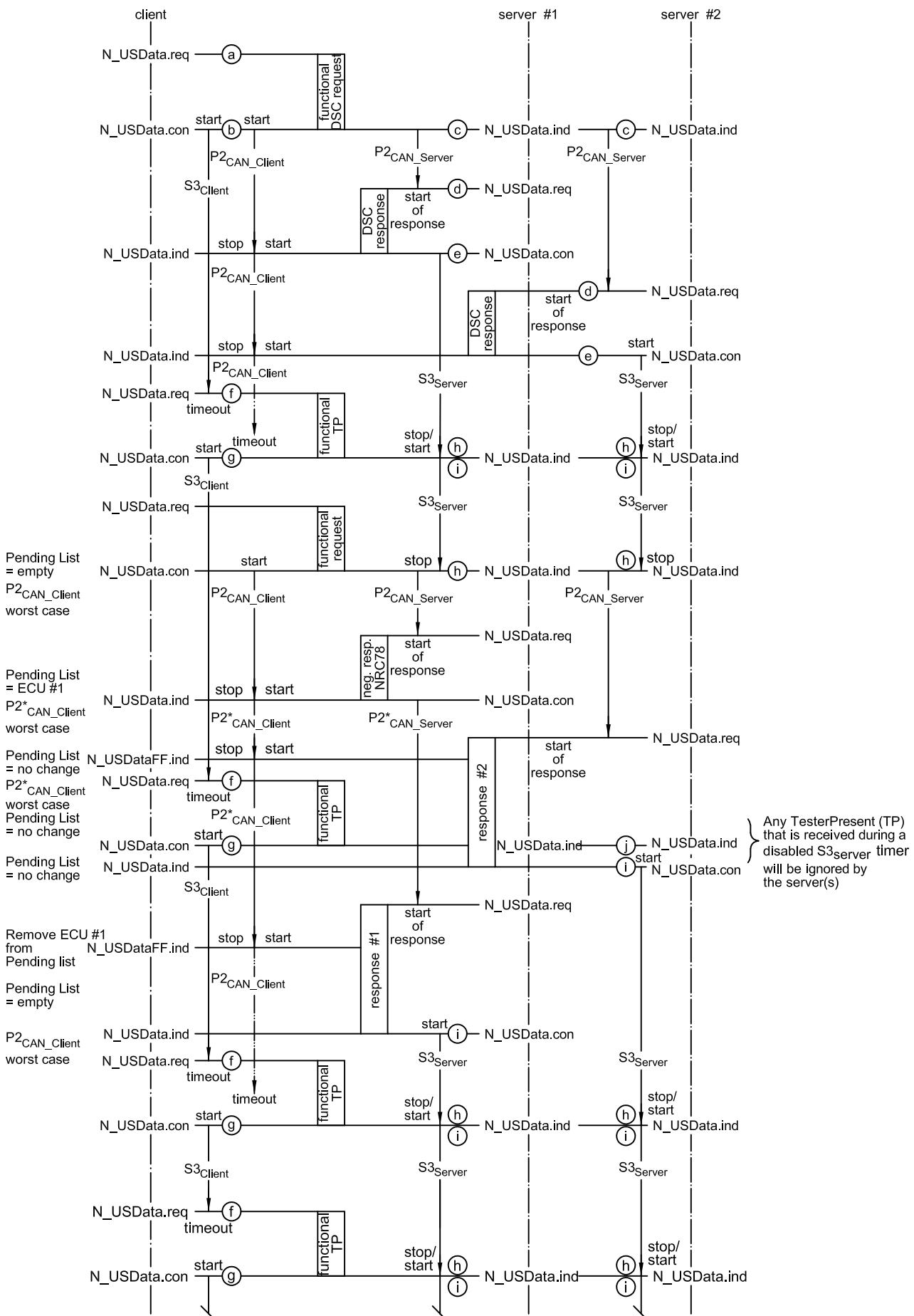
c The completion of the request message is indicated in the servers via N_USData.ind.

- d The functionally addressed servers are required to start with their response messages within $P2_{CAN_Server}$ after the reception of $N_USData.ind$. This means that, in the case of a multi-frame response message, the FirstFrame shall be sent within $P2_{CAN_Server}$ and, for single-frame response messages, that the SingleFrame shall be sent within $P2_{CAN_Server}$. In case any of the addressed servers cannot provide the requested information within the $P2_{CAN_Server}$ response timing, it can request an enhanced response timing window by sending a negative response message including response code 78 hex.
- e Upon reception of the negative response message within the client, the client network layer generates a $N_USData.ind$. The reception of a negative response message with response code 78 hex causes the client to restart its $P2_{CAN_Client}$ timer, using the enhanced reload value $P2^*_{CAN_Client}$. In addition, the client shall store a server identification in a list of pending response messages. Once a server that is stored as pending in the client starts with its final response message (positive or negative response message including a response code other than 78 hex), it is deleted from the list of pending response messages. Where no further response messages are pending, the client re-uses the default reload value for its $P2_{CAN_Client}$ timer. For simplicity, the figure shows only a single negative response message including response code 78 hex from server #1.
- f As long as there is at least one server stored as pending in the client, any start of a further response message from any server that was addressed by the request will cause a restart of the $P2_{CAN_Client}$ timer using the enhanced reload value $P2^*_{CAN_Client}$. (In Figure 9, this is shown when the client receives the start of the response message of the second server.)
- g As for physical communication, the server that requested enhanced response timing is required to start with its response message within the enhanced $P2_{CAN_Server}$ ($P2^*_{CAN_Server}$). Once the server can provide the requested information, it starts with its final response message by issuing a $N_USData.req$ to its network layer. When the server can still not provide the requested information within the enhanced $P2^*_{CAN_Server}$, then a further negative response message including response code 78 hex can be sent. This will cause the client to restart its $P2_{CAN_Client}$ timer again, using the enhanced reload value $P2^*_{CAN_Client}$. A negative response message including response code 78 hex from a server that is already stored in the list of pending response messages has no affect to the client internal list of pending response message.
- h As described in 6.3.5.2.1, in the case of a multi-frame response message the reception of the FirstFrame from any server is indicated in the client via the $N_USDataFF.ind$ of the network layer. A single-frame response message is indicated via $N_USData.ind$. When receiving the FirstFrame/SingleFrame indication of an incoming response message, the client either stops its $P2_{CAN_Client}$ in the case where it knows the servers to be expected to respond and all servers have responded, or restarts its $P2_{CAN_Client}$ timer in the case where not all expected servers have yet responded or the client does not know the servers to be expected to respond (client awaits the start of further response messages).
- i The network layer will generate a final $N_USData.ind$ in case any multi-frame response message is completely received or an error occurred during the reception. This will not have any influence on the $P2_{CAN_Client}$ timer. Furthermore, the handling of the list of pending response messages as described above applies.
- j The completion of the transmission will also be indicated in the servers via $N_USData.con$.

Figure 8 — Functional communication during default session – enhanced response timing

6.3.5.2.3 Functional communication during non-default session

Figure 9 graphically depicts the timing handling in the client and two (2) servers for a functionally addressed request message during the non-default session (e.g. programmingSession), where one server requests an enhanced response timing via a negative response message including response code 78 hex.



- ^a The diagnostic application of the client starts the transmission of the functionally addressed DiagnosticSessionControl (10 hex) request message by issuing a N_USData.req to its network layer. The network layer transmits the request message to the servers. The request message is a single-frame message.
- ^b The completion of the request message is indicated in the client via N_USData.con. Now the response timing as described in 6.3.5.2.1 and 6.3.5.2.2 applies. In addition, the generated N_USData.con in the client causes the start of the S3_{Client} timer (session timer).
- ^c The completion of the request message is indicated in the servers via N_USData.ind. Now the response timing as described in 6.3.5.2.1 and 6.3.5.2.2 applies.
- ^d For the figure as given, it is assumed that the client requires a response from the servers. The servers have to transmit the DiagnosticSessionControl (10 hex) positive response messages.
- ^e The completion of the transmission of the positive response message is indicated in the servers via N_USData.con. The servers start their S3_{Server} timers, which keeps the activated non-default session active as long as S3_{Server} does not time out. It is the client's responsibility to ensure that the S3_{Server} timer is reset prior to its timeout, in order to keep the servers in the non-default session.
- ^f Once the S3_{Client} timer is started in the client, this causes the transmission of a functionally addressed TesterPresent (3E hex) request message, which does not require a response message each time the S3_{Client} timer times out.
- ^g Upon the indication of the completed transmission of the TesterPresent (3E hex) request message via N_USData.con of its network layer, the client once again starts its S3_{Client} timer. This means that the functionally addressed TesterPresent (3E hex) request message is sent on a periodic basis every time S3_{Client} times out.
- ^h Any time a server is in the process of handling any diagnostic service, it stops its S3_{Server} timer.
- ⁱ When the diagnostic service is completely processed, then the server restarts its S3_{Server} timer. A diagnostic service is meant to be in progress any time between the start of the reception of the request message (N_USDataFF.ind or N_USData.ind receive) and the completion of the transmission of the final response message, where a response message is required, or the completion of any action that is caused by the request, where no response message is required (point in time reached that would cause the start of the response message).
- ^j Any TesterPresent (3E hex) request message that is received during processing of another request message can be ignored by the server, because it has stopped its S3_{Server} timer and will restart it once the other service is processed completely.

Figure 9 — Functional communication during non-default session

The handling of the P2_{CAN_Client} and P2_{CAN_Server} timing is identical to the handling as described in 6.3.5.2.1 and 6.3.5.2.2, the only exception being that the reload values on the client side and the resulting time the server shall send its final response time might differ. This is based on the transition into a session other than the default session where different P2_{CAN_Client} timing parameters might apply (see DiagnosticSessionControl (10 hex) service in 9.2.1 for details on how the timing parameters are reported to the client).

6.3.5.3 Minimum time between client request messages

The minimum time between request messages transmitted by the client is required in order to allow for a polling driven service data interpretation in the server, for example. Based on its normal functionality, a server might process diagnostic request messages with a certain scheduling rate (e.g. 10 ms). The time for the diagnostic service data interpretation scheduler shall be smaller than the performance requirement P2_{CAN_Server} in order to meet the server requirements of 6.3.5 and 6.3.5.1.3.2.

The timing parameter for the minimum time between request message is divided into the following two timing parameters.

- P3_{CAN_Functional}: this timing parameter applies to any functionally addressed request message, because it can be the case that a server is not required to respond to a functionally addressed request message if it does not support the requested data.
- P3_{CAN_Physical}: this timing parameter applies to any physically addressed request message where there is no response required to be transmitted by the server (suppressPosRspMsgIndicationBit = TRUE).

In the case of physical communication where a response is required by the server, the client can transmit the next request immediately after the complete reception of the last response message, because the server has responded completely to the request — which means that the request is completely handled by the server.

Figure 10 graphically depicts an example of a problem that can occur during functional communication, when the client transmits the next request immediately after it has determined that all expected servers responded to a previous request message.

This scenario not only applies to functionally addressed requests but also to physically addressed requests where the client does not want to receive any response message (`suppressPosRspMsgIndicationBit = TRUE`).

In order to handle the described scenarios, the minimum times $P3_{CAN_Physical}$ and $P3_{CAN_Functional}$ between the end of a physically or functionally addressed request message and the start of a new physically or functionally addressed request message, are defined for the client.

- a) The value of $P3_{CAN_Physical}$ will be identical to $P2_{CAN_Server_max}$ for the physically addressed server. The timing applies to any physically addressed request message in any diagnostic session (default and non-default session) and in case no response is required by the server.

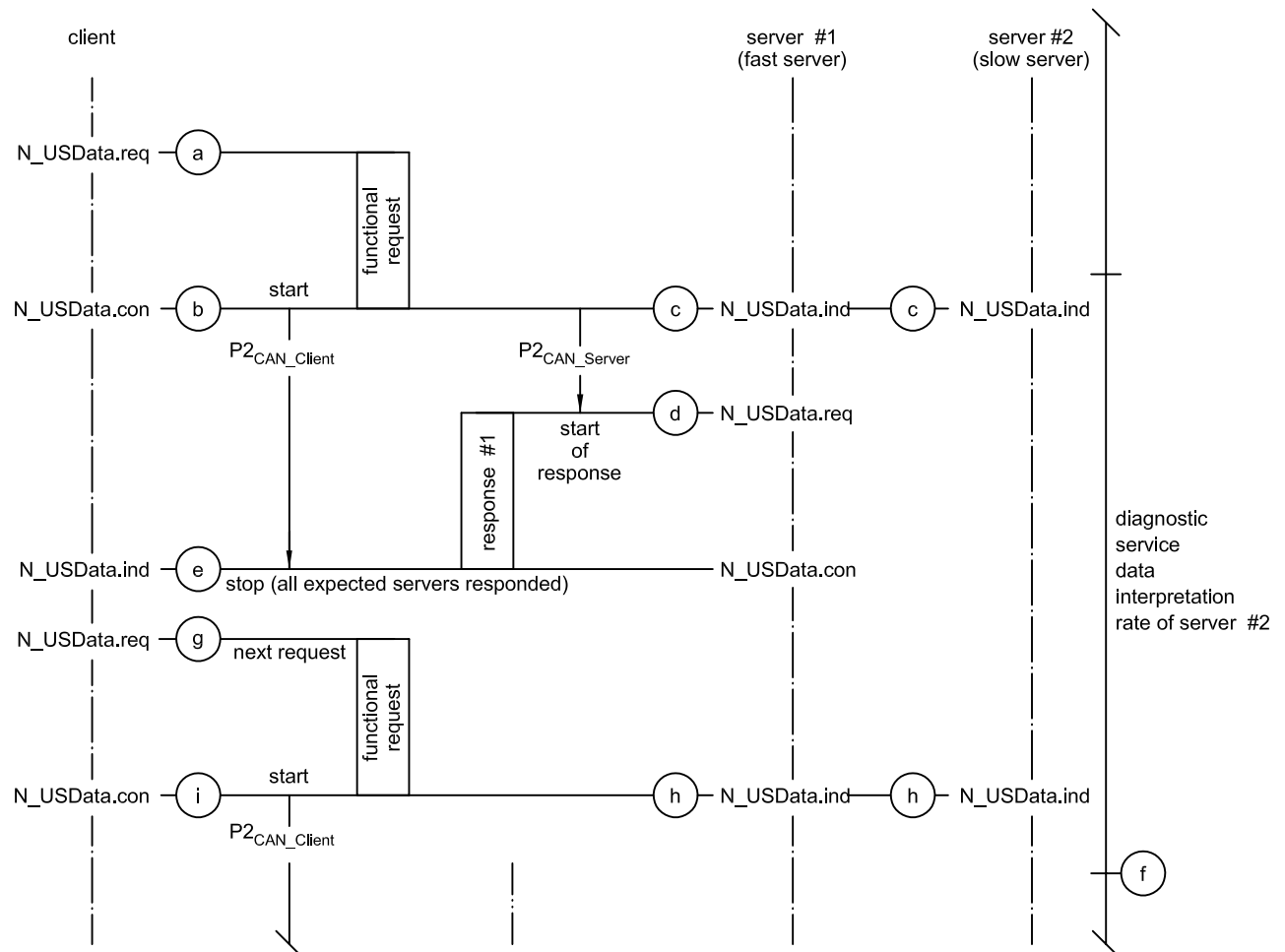
The $P3_{CAN_Physical}$ timer is started in the client each time a physically addressed request message with no response required is successfully transmitted onto the bus, which is indicated via `N_USData.con` in the client. When the client wants to transmit a new physically addressed request message following a previous request that was completely handled, then this is only allowed in case the $P3_{CAN_Physical}$ timer is no longer active at the time the client wants to transmit the physically addressed request message. In case $P3_{CAN_Physical}$ would still be active at the point in time the client would like to transmit a new physically addressed request message, then the transmission shall be postponed until $P3_{CAN_Physical}$ is timed out.

- b) The value of $P3_{CAN_Functional}$ will be the maximum (worst-case) value of all functionally addressed server's $P2_{CAN_Server_max}$ for any functionally addressed request message in any diagnostic session (default and non-default session).

The $P3_{CAN_Functional}$ timer is started in the client each time a functionally addressed request message with response required or with no response required is successfully transmitted onto the bus, which is indicated via `N_USData.con` in the client. When the client wants to transmit a new functionally addressed request message following a previous request that was completely handled, then this is only allowed in case the $P3_{CAN_Functional}$ timer is no longer active at the time the client wants to transmit the functionally addressed request message. In case $P3_{CAN_Functional}$ would still be active at the point in time the client would like to transmit a new functionally addressed request message, then the transmission shall be postponed until $P3_{CAN_Functional}$ is timed out.

NOTE “Completely handled” means that either no response is received in case no response is required, all expected responses to a functionally addressed request are received in case the responding servers are known and responses are required or a $P2_{CAN_Client}$ timeout occurred in case the responding servers are not known and responses are required.

The requirement for the server is that it shall start with its response message within $P2_{CAN_Server}$ (see 7.3). This means that the diagnostic data interpretation rate of the server shall be less than $P2_{CAN_Server}$.



- a The diagnostic application of the client starts the transmission of a functionally addressed request message by issuing a N_USData.req to its network layer. The network layer transmits the request to the servers.
- b The completion of the request message is indicated in the client via N_USData.con. The client starts its P2_{CAN_Client} timer, using the default reload value P2_{CAN_Client}.
- c The completion of the request message is indicated in the server via N_USData.ind. The server starts its P2_{CAN_Server} timer, using the default reload value P2_{CAN_Server}.
- d For the request message, it is assumed that only server #1 supports the requested information, which means that there will be no response message from server #2. Server #1 is a fast server and can immediately process the received request message and transmits its response within P2_{CAN_Server}.
- e The client receives the response message. This is indicated via N_USData.ind. The client only expected a response message from server #1, therefore it stops its timer P2_{CAN_Client}.
- f Server #2 is a slow server and interprets received requests on a periodic basis (diagnostic service data interpretation rate). In the worst-case, the last check for incoming request a message is prior to the network layer reception of the functionally addressed request message. This would mean that the request would be stored in a buffer and processed at the earliest the next time the scheduler checks for an incoming request. When server #2 processes the request, then it determines that it does not need to answer, because it does not support the requested information. As shown in the figure, this would be after the completion of the response message of server #1, and even after the completion of the next request message transmitted by the client.
- g The client would send the next request right after the completion of all expected response messages.
- h The completion of the request message is indicated in the servers via N_USData.ind, but only processed by the fast server #1, because server #2 did not yet handle the last request.
- i The completion of the new request is indicated in the client via N_USData.con.

Figure 10 — Example of critical issue when transmitting next request too early

Figure 11 graphically depicts the $P3_{CAN_Functional}$ timing handling for the client (based on the communication scenario illustrated by Figure 10). In addition, Figure 11 shows the handling of a functionally addressed TesterPresent (3E hex) request message in the client in the case in which the $P3_{CAN_Functional}$ timer is still active when $S3_{Client}$ times out (request will be postponed until $P3_{CAN_Functional}$ times out).

