9.2.6 SecuredDataTransmission (84 hex) service

There are neither additional requirements nor restrictions defined for this service for its implementation on CAN.

9.2.7 ControlDTCSetting (85 hex) service

Table 35 defines the subfunction parameters applicable for the implementation of this service on CAN.

Table 35 — Subfunction parameter definition

Hex (bit 6-0)	Description	Cvt	Mnemonic
01	on	M	ON
02	off	M	OFF

9.2.8 ResponseOnEvent (86 hex) service

The following requirements shall apply for this service when implemented on CAN.

a) Multiple ResponseOnEvent services may run concurrently with different requirements (different EventTypes, serviceToRespondTo-Records, ...) to start and stop diagnostic services.

b) While the ResponseOnEvent service is active, the server shall be able to process concurrent diagnostic request and response messages accordingly. This should be accomplished with a (different) pair of serviceToRespondTo-request/response CAN identifiers. See Figure 16. If the same diagnostic request/response CAN identifiers are used for diagnostic communication and the serviceToRespondTo-responses, the following restrictions shall apply.

   1) The server shall ignore an incoming diagnostic request after an event has occurred and the serviceToRespondTo-response is in progress, until the serviceToRespondTo-response is completed.

   2) After the client receives any response after sending a diagnostic request, the response shall be classified according to the possible serviceToRespondTo-responses and the expected diagnostic responses that have been sent.

   3) If the response is a serviceToRespondTo-response (one of the possible responses set up with ResponseOnEvent-service), the client shall repeat the request after the serviceToRespondTo-response has been received completely.

   4) If the response is ambiguous (i.e. the response could originate from the serviceToRespondTo initiated by an event or from the response to a diagnostic request), the client shall present the response both as a serviceToRespondTo-response and as the response to the diagnostic request. The client shall not repeat the request with the exception of NegativeResponseCode busyRepeatRequest (21 hex). (See the negative response code definitions in ISO 14229-1.)

c) The ResponseOnEvent service shall only be allowed to use those diagnostic services available in the active diagnostic session.

d) While the ResponseOnEvent service is active, any change in a diagnostic session shall terminate the current ResponseOnEvent service(s). For instance, if a ResponseOnEvent service has been set up during extendedDiagnosticSession, it shall terminate when the server switches to the defaultSession.

e) If a ResponseOnEvent (86 hex) service has been set up during defaultSession, then the following shall apply:

1) If Bit 6 of the eventType subfunction parameter is set to 0 (do not store event), then the event shall terminate when the server powers down The server shall not continue a ResponseOnEvent diagnostic service after a reset or power on (i.e. the ResponseOnEvent service is terminated).

2) If Bit 6 of the eventType subfunction parameter is set to 1 (store event), it shall resume sending serviceToRespondTo-responses according to the ResponseOnEvent-set up after a power cycle of the server.

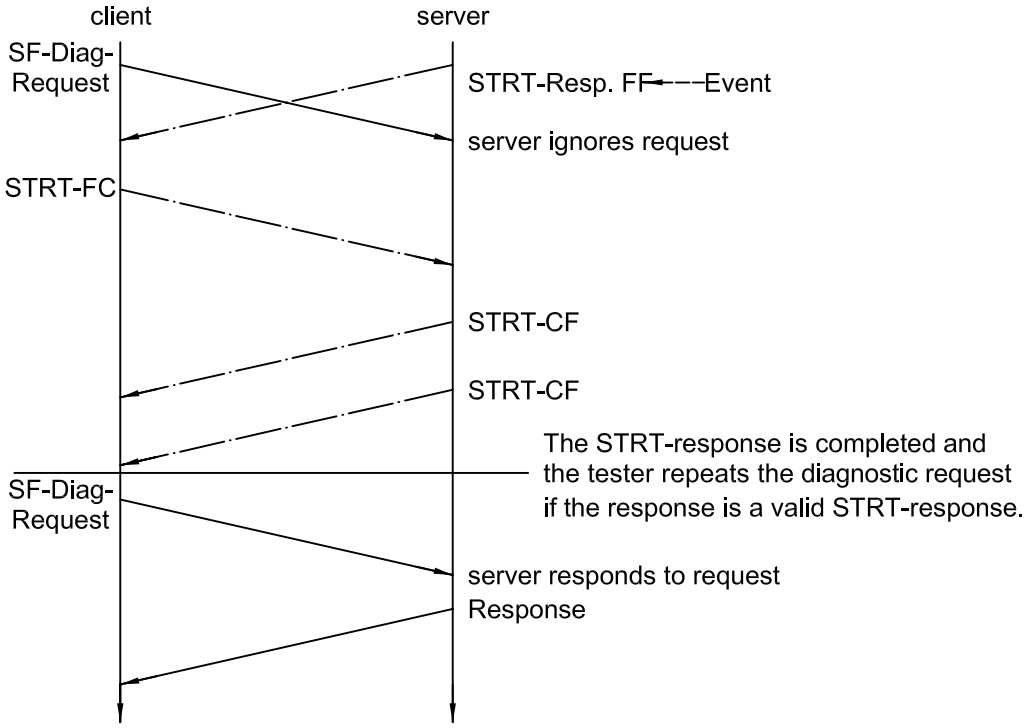


Figure 16 — Concurrent request when the event occurs

f) The subfunction parameter value responseRequired = “no” should only be used for the eventType = stopResponseOnEvent, startResponseOnEvent or clearResponseOnEvent The server shall always return a response to the event-triggered response when the specified event is detected.

g) The server shall return a final positive response to indicate the ResponseOnEvent (86 hex) service has reached the end of the finite event window, unless one of the following conditions apply:

1) if eventTypes do not setup ResponseOnEvent, such as stopResponseOnEvent, startResponseOnEvent, clearResponseOnEvent or reportActivatedEvents;

2) if the infinite event window was established

— if the Service has been deactivated before the event window was closed,

— Bit 6 of the eventType subfunction parameter is set to 0 (do not store) and the server powers down or resets.

h) When the specified event is detected, the server shall respond immediately with the appropriate serviceToRespondTo-response message. The immediate serviceToRespondTo-response message shall not disrupt any other diagnostic request or response transmission already in progress (i.e. the

serviceToRespondTo-response shall be delayed until the current message transmission has been completed — see Figure 17).

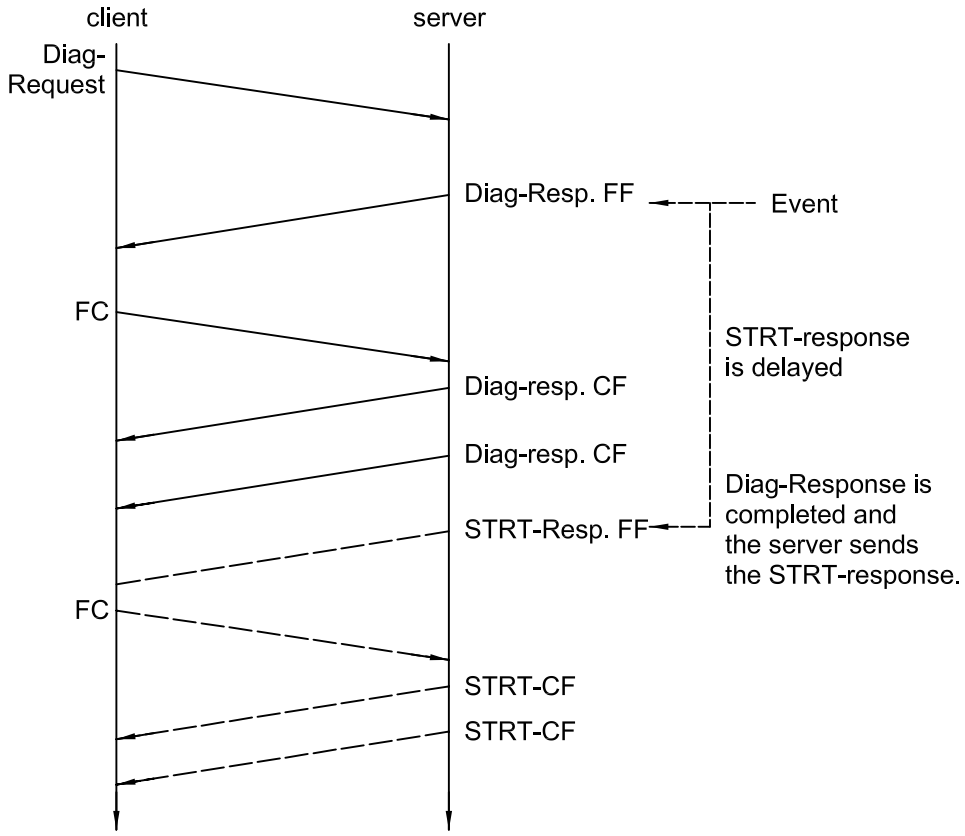


Figure 17 — Event occurrence during a message in progress

i) The ResponseOnEvent service shall only apply to transient events and conditions. The server shall return a response once per event occurrence. For a condition that is continuously sustained over a period of time, the response service shall be executed only one time at the initial occurrence. In case the eventType is defined so that serviceToRespondTo-responses could occur at a high frequency, then appropriate measures have to be taken in order to prevent back to back serviceToRespondTo-responses. A minimum separation time between serviceToRespondTo-responses could be part of the eventTypeRecord (vehicle-manufacturer-specific).

Tables 36 and 37 define the subfunction parameters applicable for the implementation of this service on CAN.

Table 38 defines the data parameters applicable for the implementation of this service on CAN.

Table 36 — eventType subfunction bit 6 definition — StorageState

Bit 6 value	Description	Cvt	Mnemonic
0	doNotStoreEvent	M	DNSE
1	storeEvent	U	SE

Table 37 — Subfunction parameter definition

Hex (bit 5-0)	Description	Cvt	Mnemonic
00	stopResponseOnEvent	U	STPROE
01	onDTCStatusChange	U	ONDTCS
02	onTimerInterrupt	U	OTI
03	onChangeOfDataIdentifier	U	OCOCID
04	reportActivatedEvents	U	RAE
05	startResponseOnEvent	U	STRTROE
06	clearResponseOnEvent	U	CLRROE
07	onComparisonOfValues	M	OCOV

Table 38 — Data parameter definition — serviceToRespondToRecord.serviceId

Recommended services (ServiceToRespondTo)	RequestService Identifier (SId)
ReadDataByIdentifier	22 hex
ReadDTCInformation	19 hex
RoutineControl	31 hex
InputOutputControlByIdentifier	2F hex

9.2.9 LinkControl (87 hex) service

Table 39 defines the subfunction parameters applicable for the implementation of this service on CAN.

Table 39 — Subfunction parameter definition

Hex (bit 6-0)	Description	Cvt	Mnemonic
01	verifyBaudrateTransitionWithFixedBaudrate	U	VBTWFBR
02	verifyBaudrateTransitionWithSpecificBaudrate	U	VBTWSBR
03	transitionBaudrate	U	TB

9.3 Data transmission functional unit

9.3.1 ReadDataByIdentifier (22 hex) service

There are neither additional requirements nor restrictions defined for this service for its implementation on CAN.

9.3.2 ReadMemoryByAddress (23 hex) service

There are neither additional requirements nor restrictions defined for this service for its implementation on CAN.

9.3.3 ReadScalingDataByIdentifier(24 hex) service

There are neither additional requirements nor restrictions defined for this service for its implementation on CAN.

9.3.4 ReadDataByPeriodicIdentifier (2A hex) service

The two types of response messages as defined for this service in ISO 14229-1 are mapped onto CAN as follows.

— Response message type #1 (including the service identifier, the echo of the periodicDataIdentifier and the data of the periodicDataIdentifier): This type of response message is mapped onto a USDT[3)] message, using the same response CAN identifier as used for any other USDT response message. The USDT message for a single periodicDataIdentifier shall not exceed the size of a single CAN frame, which means that the complete USDT response message shall fit into a SingleFrame N_PDU.

— Response message type #2 (including the periodicDataIdentifier and the data of the periodicDataIdentifier): This type of response message is mapped onto a UUDT[4)] message, using a different CAN identifier as used for the USDT response message. The UUDT message for a single periodicDataIdentifier shall not exceed the size of a single CAN frame.

The mapping of the two response types lead to certain client and server requirements as listed in Tables 40 and 41.

Table 40 — Periodic transmission — Requirements for the response type #1 message mapping

Message type	Client request requirements	Server response requirements	Further server restrictions
USDT uses the same CAN identifier for diagnostic communication and periodic transmission	No restrictions	Only single-frame responses for periodic transmission Multi-frame responses to new (non-periodic-transmission) requests possible	Any other new incoming request shall be prioritized and the periodic transmission may be delayed.
			The periodic response is processed in the server as a regular USDT message (with protocol control information (PCI), service identifier (SId) and periodicDataIdentifier) and is processed by the server network layer. This means that a maximum of 5 data bytes are available for the data of a periodicDataIdentifier when using normal addressing and 4 data bytes when using extended addressing for the response message.
			For an incoming multi-frame request message, any scheduled periodic transmission shall be delayed in the server immediately after the N_USDataFF.ind of a multi-frame request or the N_USData.ind of a SingleFrame request is processed by the application. Once the complete service is processed (including the final response message), the transmission of the periodic messages shall be continued.

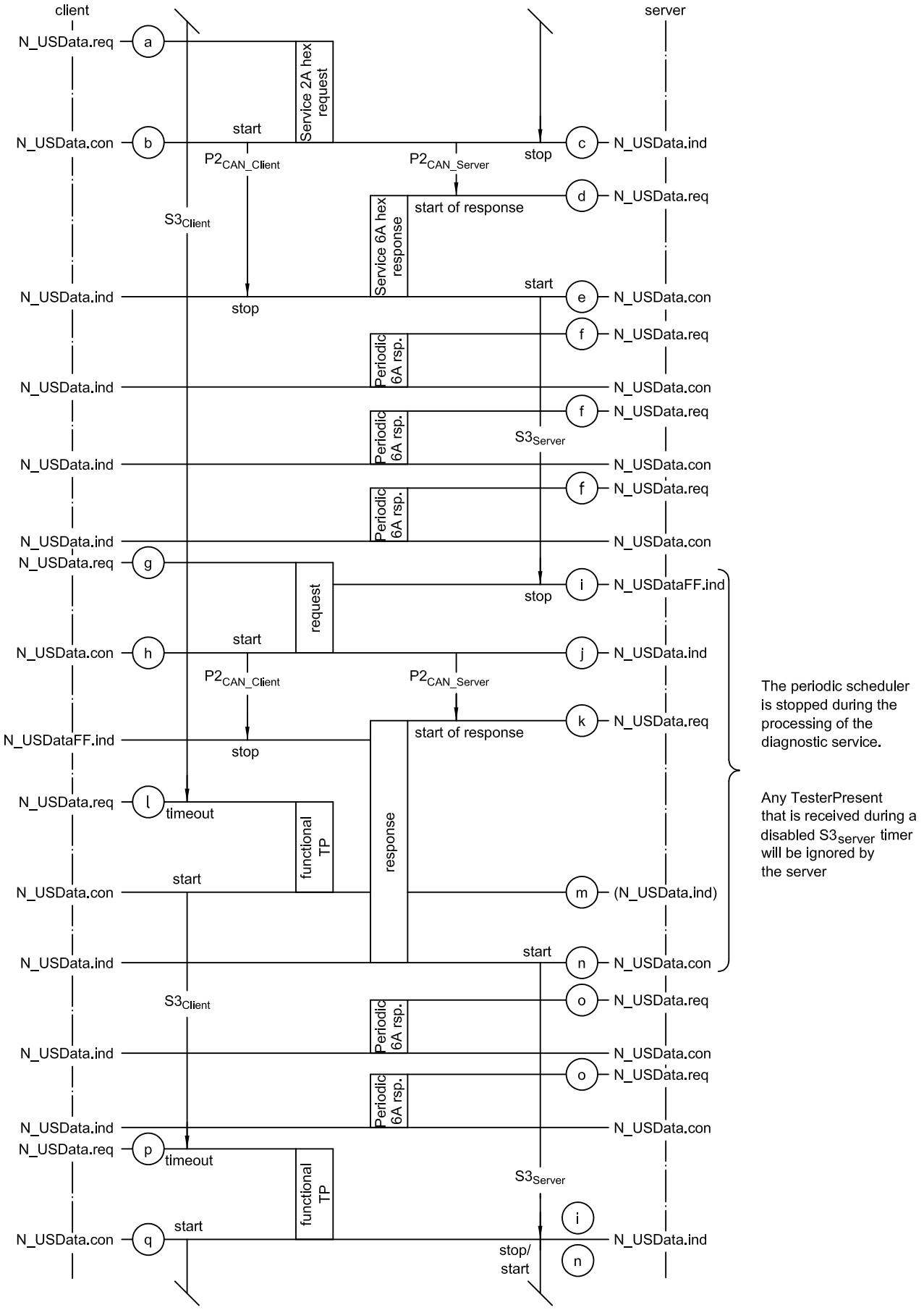
3) USDT Unacknowledged Segmented Data Transfer, ISO 15765-2 network layer, includes protocol control information for segmented data transmission.

4) UUDT Unacknowledged Unsegmented Data Transfer, single CAN frames, do not include protocol control information, which results in max. 7/8 data bytes for normal/extended addressing.

Table 41 — Periodic transmission — Requirements for response type #2 message mapping

Message type	Client request requirements	Server response requirements	Further server restrictions
UUDT uses a different CAN identifier for periodic transmission	No restrictions	Only single-frame responses for periodic transmission Multi-frame responses to new (non-periodic-transmission) requests are possible	The request for periodic transmission is processed as a regular diagnostic request and the response is sent via the network layer (as a USDT message with service identifier 6A hex).
			On receiving the N_USData.con that indicates the completion of the transmission of the positive response, the application starts an independent scheduler, which handles the periodic transmission.
			The scheduler in the server processes the periodic transmission as a single frame UUDT-message in a by-pass (i.e. writes the UUDT message directly to the CAN-controller/data link layer driver without using the network-layer).
			For an UUDT-message there is no need to include protocol control information (PCI) and service identifier (SId), only the periodic identifier is included, so a maximum of 7 data bytes can be used for the data of a periodicDataIdentifier for normal addressing and 6 data bytes for extended addressing.

Figures 18 and 19 graphically depict the two types of periodic response messages, as the server should handle them. Furthermore, the figures show that the periodically transmitted response messages do not have any influence on the $S3_{Server}$ timer of the server. For both figures it is assumed that a non-defaultSession has been activated prior to the configuration of the periodic scheduler (the ReadDataByPeriodicIdentifier service requires a non-defaultSession in order to be executed).



The periodic scheduler is stopped during the processing of the diagnostic service.

Any TesterPresent that is received during a disabled $S3_{server}$ timer will be ignored by the server

[a] The diagnostic application of the client starts the transmission of the ReadDataByPeriodicIdentifier (2A hex) request message by issuing a N_USData.req to its network layer. The network layer transmits the ReadDataByPeriodicIdentifier (2A hex) request message to the server. The request message can either be a single-frame message or a multi-frame message (depends on the number of periodicDataIdentifier contained in the request message). For the example given, it is assumed that the request message is a SingleFrame message.

[b] The completion of the request message is indicated in the client via N_USData.con. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.

[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. Furthermore, the server stops its $S3_{Server}$ timer.

[d] For the figure given, it is assumed that the client requires a response from the server. The server shall transmit the ReadDataByPeriodicIdentifier positive response message to indicate that the request has been processed and that the transmission of the periodic messages will start afterwards.

[e] The completion of the transmission of the ReadDataByPeriodicIdentifier response message is indicated in the server via N_USData.con. Now the server restarts its $S3_{Server}$ timer, which keeps the activated non-default session active as long as it does not time out.

[f] The server starts to transmit the periodic response messages (SingleFrame message). Each periodic message utilizes the network layer protocol and uses the response CAN identifier that is also used for any other response message. Therefore, the server issues a N_USData.req to the network layer each time a periodic message is transmitted and no other service is currently in the process of being handled by the server. For the example given, it is assumed that the server is able to transmit three (3) periodic messages prior to the next request message that is issued by the client. The transmission of the periodic response messages has no influence on the $S3_{Server}$ timer (see 6.3.5.4).

[g] The diagnostic application of the client starts the transmission of the next 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. For the example given, it is assumed that the request message is a multi-frame message.

[h] The completion of the request message is indicated in the client via N_USData.con. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.

[i] Once the start of a request message is indicated in the server via N_USDataFF.ind (or N_USData.ind for SingleFrame request messages) while a periodic scheduler is active, the server shall temporarily stop the periodic scheduler for the duration of processing the received request message. Furthermore, any time the server is in the process of handling any diagnostic service it stops its $S3_{Server}$ timer.

[j] The completion of the multi-frame 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. The scheduler for the transmission of the periodic messages remains disabled.

[k] For the figure given, it is assumed that the client requires a response from the server. The server shall transmit the positive (or negative) response message via issuing N_USData.req to its network layer. For the example, it is assumed that the response is a multi-frame message.

[l] When the $S3_{Client}$ timer times out in the client, then the client transmits a functionally addressed TesterPresent (3E hex) request message to reset the $S3_{Server}$ timer in the server.

[m] The server is in the process of transmitting the multi-frame response of the previous request. Therefore, the server shall not act on the received TesterPresent (3E hex) request message, because its $S3_{Server}$ timer is not yet re-activated.

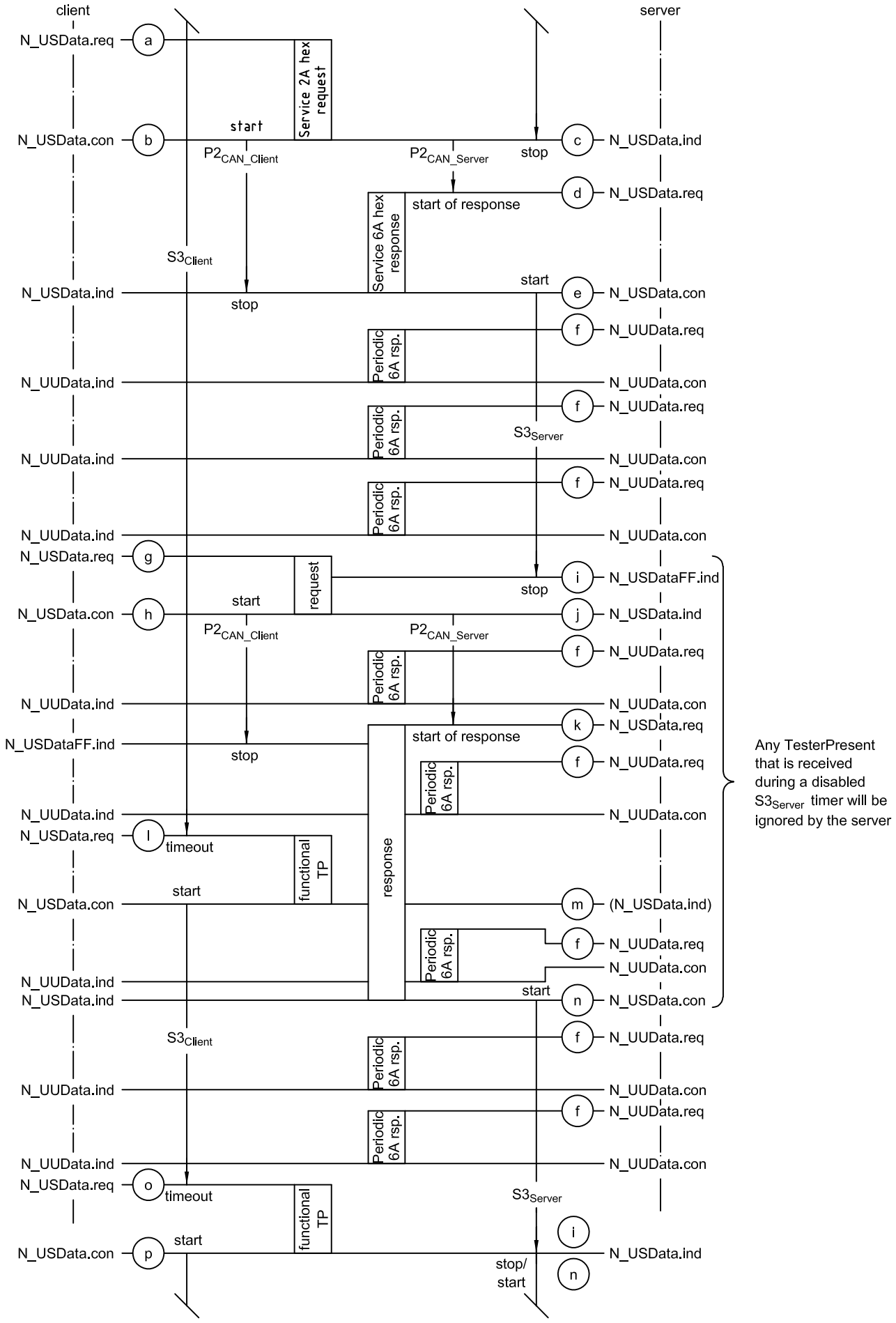
[n] When the diagnostic service is completely processed, then the server restarts its $S3_{Server}$ timer. This means that any diagnostic service, including TesterPresent (3E hex), resets the $S3_{Server}$ timer. A diagnostic service is meant to be in progress any time between the start of the reception of the request message (N_USDataFF.ind or N_USData.ind receive) and the completion of the transmission of the 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). This includes negative response messages including response code 78 hex. The server re-enables the periodic scheduler when the service is completely processed (final response message completely transmitted).

[o] The server restarts the transmission of the periodic response messages (SingleFrame message). Each periodic message utilizes the network layer protocol and uses the response CAN identifier that is also used for any other response message. Therefore, the server issues a N_USData.req to the network layer each time a periodic message is transmitted and no other service is currently in the process of being handled by the server. The transmission of the periodic response messages has no influence on the $S3_{Server}$ timer (see 6.3.5.4).

[p] Once the $S3_{Client}$ timer is started in the client (non-defaultSession active), 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.

[q] 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.

Figure 18 — Response message type #1 handling



[a] The diagnostic application of the client starts the transmission of the ReadDataByPeriodicIdentifier (2A hex) request message by issuing a N_USData.req to its network layer. The network layer transmits the ReadDataByPeriodicIdentifier (2A hex) request message to the server. The request message can either be a single-frame or multi-frame message (depends on the number of periodicDataIdentifier contained in the request message). For the example given, it is assumed that the request message is a SingleFrame message.

[b] The completion of the request message is indicated in the client via N_USData.con. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.

[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. Furthermore, the server stops its $S3_{Server}$ timer.

[d] It is assumed that the client requires a response from the server. The server shall transmit the ReadDataByPeriodicIdentifier positive response message to indicate that the request has been processed and that the transmission of the periodic messages will start afterwards.

[e] The completion of the transmission of the ReadDataByPeriodicIdentifier response message is indicated in the server via N_USData.con. Now the server restarts its $S3_{Server}$ timer, which keeps the activated non-default session active as long as it does not time out.

[f] The server starts to transmit the periodic response messages (single-frame message). Each periodic message is a UUDT message and uses a different CAN identifier as used for any other response message (USDT CAN identifier). Therefore, the server issues a N_UUData.req each time a periodic message is transmitted independent of any other service currently processed by the server. This means that the transmission of the periodic response messages continues even when the server is in the process of handling another diagnostic service request. The transmission of the periodic response messages has no influence on the $S3_{Server}$ timer (see 6.3.5.4).

[g] The diagnostic application of the client starts the transmission of the next 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. For the example given, it is assumed that the request message is a multi-frame message.

[h] The completion of the request message is indicated in the client via N_USData.con. Now the response timing as described in 6.3.5.1.1 and 6.3.5.1.2 applies.

[i] The start of a request message is indicated in the server via N_USDataFF.ind (or N_USData.ind for SingleFrame request messages) while a periodic scheduler is active. The server does not stop the periodic scheduler for the duration of processing the received request message. This means that the server transmits further periodic messages for the duration of processing the diagnostic service. The client shall be aware of receiving these periodic response messages. Furthermore, any time the server is in the process of handling any diagnostic service it stops its $S3_{Server}$ timer.

[j] The completion of the multi-frame 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.

[k] For the figure given, it is assumed that the client requires a response from the server. The server shall transmit the positive (or negative) response message via issuing N_USData.req to its network layer. For the example, it is assumed that the response is a multi-frame message. While the multi-frame response message is transmitted by the network layer, the periodic scheduler continues to transmit the periodic response messages.

[l] When the $S3_{Client}$ timer times out in the client, then the client transmits a functionally addressed TesterPresent (3E hex) request message to reset the $S3_{Server}$ timer in the server.

[m] The server is in the process of transmitting the multi-frame response of the previous request. Therefore, the server shall not act on the received TesterPresent (3E hex) request message, because its $S3_{Server}$ timer is not yet re-activated.

[n] 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 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). This includes negative response messages including response code 78 hex.

[o] Once the $S3_{Client}$ timer is started in the client (non-defaultSession active), 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.

[p] 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.

Figure 19 — Response message type #2 handling

Table 42 defines the data parameters applicable for the implementation of this service on CAN.

Table 42 — Data parameter definition — TransmissionMode

Hex	Description	Cvt	Mnemonic
01	sendAtSlowRate	U	SASR
02	sendAtMediumRate	U	SAMR
03	sendAtFastRate	U	SAFR
04	stopSending	U	SS

9.3.5 DynamicallyDefineDataIdentifier (2C hex) service

When the client dynamically defines a periodicDataIdentifier and the total length of the dynamically defined periodicDataIdentifier exceeds the maximum length that fits into a single frame periodic response message, then the request shall be rejected with a negative response message including negative response code 31 hex (requestOutOfRange). See ReadDataByPeriodicIdentifier (9.3.4) for details regarding the periodic response message format.

When multiple DynamicallyDefineDataIdentifier request messages are used to configure a single periodicDataIdentifier and the server detects the overrun of the maximum number of bytes during a subsequent request for this periodicDataIdentifier, then the server shall leave the definition of the periodicDataIdentifier as it was prior to the request that lead to the overrun.

Table 43 defines the subfunction parameters applicable for the implementation of this service on CAN.

Table 43 — Subfunction parameter definition

Hex (bit 6-0)	Description	Cvt	Mnemonic
01	defineByIdentifier	U	DBID
02	defineByMemoryAddress	U	DBMA
03	clearDynamicallyDefinedDataIdentifier	U	CDDDI

9.3.6 WriteDataByIdentifier (2E hex) service

There are neither additional requirements nor restrictions defined for this service for its implementation on CAN.

9.3.7 WriteMemoryByAddress (3D hex) service

There are neither additional requirements nor restrictions defined for this service for its implementation on CAN.

9.4 Stored data transmission functional unit

9.4.1 ReadDTCInformation (19 hex) service

Table 44 defines the subfunction parameters applicable for the implementation of this service on CAN.