7.4.4 Functional description

7.4.4.1 Overview

The DSL submodule provides the following functionality:

Request Handling

- Forward requests from the PduR module to the DSD submodule.
- Concurrent TesterPresent (”keep alive logic”).

Response Handling

- Forward responses from the DSD submodule to the PduR module.
- Guarantee response timing to tester.
- Support of periodic transmission.
- Support of ResponseOnEvent (ROE) transmission.
- Support of segmented response.
- Support of ResponsePending response triggered by the application.

Security Level Handling

- Manage security level.

Session State Handling

- Manage session state.
- Keep track of active non-default sessions.
- Allows modifying timings.

Diagnostic Protocol Handling

- Handling of different diagnostic protocols.
- Manage resources.

Communication Mode Handling

- Handling of communication requirements (Full- / Silent- / No Communication).
- Indicating of active / inactive diagnostic.
- Enabling / disabling all kinds of diagnostic transmissions.

7.4.4.2 Forward requests from the PduR module to the DSD submodule

The PduR module indicates the Dcm module whenever a reception of new diagnostic request content is started on a DcmDslProtocolRxPduId, which is assigned to the Dcm module. This is done by calling Dcm_StartOfReception, which inform the Dcm module of the data size to be received and provides the data of the first frame or single frame, and allows the Dcm to reject the reception if the data size overflows its buffer size, or if the requested service is not available. The further call to Dcm_CopyRxData request the Dcm module to copy the data from the provided buffer to the Dcm buffer. If the reception of a diagnostic request is finished (when Dcm_StartOfReception succeeded) the PduR module will call Dcm_TpRxIndication to give a receive indication to the Dcm module. The Dcm shall be able to use generic connections, where the addressing information is provided to Dcm by Dcm_StartOfReception via the MetaData of the DcmRxPdu. This addressing information must be stored and used for the response and for detection of requests from the same tester. see section 7.4.4.5 Generic Connection Handling for further details.

[SWS_Dcm_00111] ⌈The DSL submodule shall forward received data to the DSD submodule only after a call of Dcm_TpRxIndication with parameter Result = E_OK (see [SWS_Dcm_00093]).⌋*(RS_Diag_04249)*

[SWS_Dcm_00241] ⌈As soon as a request message is received (after a call of Dcm_TpRxIndication with parameter Result = E_OK (see [SWS_Dcm_00093]) and until a call to Dcm_TpTxConfirmation (see [SWS_Dcm_00351]) for the associated TxDcmPduId), the DSL submodule shall block the corresponding DcmPduId. During the processing of this request, no other request of the same DcmDslConnection (e.g. an enhanced session can be ended by a OBD session) can be received, until the corresponding response message is sent and the DcmPduId is released again (except for Concurrent TesterPresent requests).⌋*()*

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of PduR module [14].

It is allowed to have different DcmPduIds for different diagnostic communication applications. For example:

- OBD DcmDslProtocolRxPduId: for reception of OBD requests,
- OBD DcmTxPduId: for transmission of OBD responses,
- UDS phys DcmDslProtocolRxPduId: for reception of UDS physically addressed requests,
- UDS func DcmDslProtocolRxPduId: for reception of UDS functionally addressed requests,
- UDS DcmTxPduId: for transmission of UDS responses.

Address type (physical/functional addressing) is configured per DcmDslProtocolRxPduId. A configuration per DcmDslProtocolRxPduId is possible because there will always be different DcmDslProtocolRxPduId values for functional and physi-

cal receptions, independent of the addressing format of the Transport Layer (extended addressing, normal addressing).

7.4.4.2.1 Dcm_StartOfReception

[SWS_Dcm_00444] ⌈If the requested size is large than the buffer available in the DCM, the function Dcm_StartOfReception shall return BUFREQ_E_OVFL (see [SWS_Dcm_00094]).⌋*()*

[SWS_Dcm_00642] ⌈When the API Dcm_StartOfReception is invoked with TpSduLength equal to 0, the value BUFREQ_E_NOT_OK shall be returned.⌋*(RS_Diag_04147)*

[SWS_Dcm_01671] Message reception according to ISO 14229-2 ⌈The Dcm shall receive and handle diagnostic messages according to the session layer services of ISO14229-2 [13].⌋*(RS_Diag_04196)*

[SWS_Dcm_01672] Mapping of T_DataSOM.ind ⌈The Dcm shall map a call of Dcm_StartOfReception to the T_DataSOM.ind from ISO14229-2 [13].⌋*(RS_Diag_04196)*

[SWS_Dcm_01673] Mapping of T_Data.ind ⌈The Dcm shall map a call of Dcm_TpRxIndicationto T_Data.ind from ISO14229-2 [13].⌋*(RS_Diag_04196)*

[SWS_Dcm_01674] Multi client handling ⌈The Dcm shall handle requests from multiple clients according to ISO14229-1 [1] Annex J.⌋*(RS_Diag_04209)*

”Figure J.2 - Multiple client handling flow” of ISO14229-1 [1] gives a comprehensive overview of parallel client request handling. A major aspect is the support of additional protocol resources. The Dcm follows this flow chart and provides protocol resources per configured DcmDslProtocolRow. In default session this allows the Dcm to process requests in parallel.

[SWS_Dcm_01675] Protocol resource availability ⌈The Dcm shall assign additional protocol resource to each configured DcmDslProtocolRow with a own DcmDslBuffer.⌋*(RS_Diag_04209)*

The configuration parameter DcmDslDiagRespOnSecondDeclinedRequest defines the behavior of the decision box ”Is NRC 0x21 handling supported?” of ”Figure J.2 - Multiple client handling flow” of ISO14229-1 [1].

[SWS_Dcm_01676] Decline a second request with NRC 0x21 ⌈If the Dcm is rejecting a second received request according to ISO14229-1 [1] Annex J and DcmDslDiagRespOnSecondDeclinedRequest is set to TRUE, the Dcm shall return a NRC 0x21 as response for the rejected second received request.⌋*(RS_Diag_04209)*

[SWS_Dcm_01677] Ignoring a second request ⌈If the Dcm is rejecting a second received request according to ISO14229-1 [1] Annex J and DcmDslDiagRespOnSec-

ondDeclinedRequest is set to FALSE, the Dcm shall ignore the new request. .⌋ *(RS_Diag_04209)*

ISO14229-1 [1] Annex J misses a definition of the server in case a second diagnostic request is received from the same connection, while the server is still processing a request on that connection. The Dcm fills this gap by not accepting further diagnostic requests for processing. In that case no response is send, even no NRC 0x21.

[SWS_Dcm_01678] Mulitple requests on the same connection ⌈If the Dcm receives a further diagnostic request for processing and the Dcm is still processing an ongoing request on the same connection, the Dcm shall decline the new received request.⌋ *(RS_Diag_04209)*

7.4.4.2.2 Dcm_CopyRxData

[SWS_Dcm_00443] ⌈If Dcm_StartOfReception returns BUFREQ_OK, the further call to Dcm_CopyRxData shall copy the data from the buffer provided in info parameter) to the Dcm buffer and update the bufferSizePtr parameter with remaining free place in Dcm receive buffer after completion of this call.⌋ *()*

[SWS_Dcm_00996] ⌈When the API Dcm_CopyRxData is invoked with SduLength from info equal to 0, the value BUFREQ_OK shall be returned and bufferSizePtr shall be filled with the remaining size of the Rx buffer.⌋ *()*

Note: The size of the Rx buffer is based on the buffer length, which is returned in the parameter RxBufferSizePtr of API Dcm_StartOfReception.

[SWS_Dcm_00342] ⌈After starting to copy the received data (see [SWS_Dcm_00443]), the Dcm module shall not access the receive buffer until it is notified by the service Dcm_TpRxIndication about the successful completion or unsuccessful termination of the reception.⌋ *()*

Note: Dcm_TpRxIndication is only expected when Dcm_StartOfReception succeeded

7.4.4.2.3 Dcm_TpRxIndication

[SWS_Dcm_00344] ⌈If Dcm_TpRxIndication is called with parameter Result different from E_OK, then the Dcm module shall not evaluate the buffer assigned to the I-PDU, which is referenced in parameter DcmRxPduId.⌋ *()*

Rationale for [SWS_Dcm_00344]: It is undefined which part of the buffer contains valid data in this case

7.4.4.3 Concurrent TesterPresent ("keep alive logic")

The Concurrent TesterPresent is defined by ISO14229-1 [1] and also called "bypass logic" or "keep alive logic". The purpose is to keep the non-default session active and reset the S3 timer defined by ISO14229-2 [13]. Concurrent TesterPresent are only received by functional addressing and are processed independent from any activity on a physical request and service processing.

[SWS_Dcm_01666] Concurrent tester present support ⌈The Dcm shall handle and process the Concurrent TesterPresent according to ISO14229-1 [1].⌋*(RS_Diag_04249)*

In addition to ISO14229-1 [1] the Dcm limits the processing for received Concurrent TesterPresent messages to the connection from where the request to enter a non-default session was received. This avoids that a tester that did not request the non-default session can influence the protocol and timing behavior of another tester.

[SWS_Dcm_01667] Concurrent tester present from different connections ⌈The Dcm shall only process and handle Concurrent TesterPresent messages received on the same DcmDslConnection that requested the Dcm to enter the non-default session.⌋*(RS_Diag_04249)*

7.4.4.3.1 Dcm_CopyTxData

If the copied data is smaller than the length requested to transmit within the service PduR_DcmTransmit() the Dcm module will be requested by the service Dcm_CopyTxData to provide another data when the current copied data have been transmitted.

[SWS_Dcm_00346] ⌈If the function Dcm_CopyTxData is called and the Dcm module successfully copied the data in the buffer provided in info parameter, then the function shall return BUFREQ_OK.⌋*()*

[SWS_Dcm_00350] ⌈Caveats of Dcm_CopyTxData:

- The value of parameter availableDataPtr of function Dcm_CopyTxData shall not exceed the number of Bytes still to be sent.
- If this service returns BUFREQ_E_NOT_OK the transmit requests issued by calling the service PduR_DcmTransmit() is still not finished. A final confirmation (indicating an error with call of service Dcm_TpTxConfirmation) is required to finish this service and to be able to start another transmission (call to PduR_DcmTransmit()). So it is up to the transport protocol to confirm the abort of transmission.

⌋*()*

7.4.4.3.2 Dcm_TpTxConfirmation

[SWS_Dcm_00352] ⌈If the function Dcm_TpTxConfirmation is called, then the Dcm module shall unlock the transmit buffer.⌋*()*

[SWS_Dcm_00353] ⌈If the function Dcm_TpTxConfirmation is called, then the Dcm module shall stop error handling (Page buffer timeout, P2ServerMax/P2*ServerMax timeout).⌋*()*

For transmission via FlexRay the following restriction has to be considered: Since the FlexRay Specification does not mandate the existence of a transmit interrupt, the exact meaning of this confirmation (i.e. "transfer into the FlexRay controller's send buffer" OR "transmission onto the FlexRay network") depends on the capabilities of the FlexRay communication controller and the configuration of the FlexRay Interface.

7.4.4.4 Forward responses from the DSD submodule to the PduR module

[SWS_Dcm_00114] ⌈The DSD submodule shall request the DSL submodule for transmission of responses.⌋*(RS_Diag_04249)*

[SWS_Dcm_00115] ⌈When the diagnostic response of a DcmDslMainConnection is ready, the DSL submodule shall trigger the transmission of the diagnostic response to the PduR module by calling PduR_DcmTransmit() using the corresponding DcmDslProtocolTxPduRef parameter as PduId.⌋*(RS_Diag_04249)*

[SWS_Dcm_01072] ⌈In case of PeriodicTransmission, the Dcm shall provide in the call to PduR_DcmTransmit() the full payload data and expect no call to Dcm_CopyTxData.⌋*()*

[SWS_Dcm_01073] ⌈In case of PeriodicTransmission, the Dcm will be called for periodic transmission with Dcm_TxConfirmation to indicate the transmission result.⌋*(RS_Diag_04249)*

Responses are sent with the DcmTxPduId, which is linked in the Dcm module configuration to the DcmDslProtocolRxPduId, i.e. the ID the request was received with (see configuration parameter DcmDslProtocolTx) Within PduR_DcmTransmit() only the length information and, for generic connections, the addressing information, is given to the PduR module. After the Dcm module has called successfully PduR_DcmTransmit(), the PduR module will call Dcm_CopyTxData to request the Dcm module to provide the data to be transmitted and will call Dcm_TpTxConfirmation after the complete PDU has successfully been transmitted or an error occurred. see section 7.4.4.5 "Generic Connection Handling for further details on address information handling within generic connections".

[SWS_Dcm_00117] ⌈If the DSL submodule receives a confirmation after the complete Dcm PDU has successfully been transmitted or an error occurred by a call of Dcm_TpTxConfirmation, then the DSL submodule shall forward this confirmation to the DSD submodule.⌋*(RS_Diag_04249)*

[SWS_Dcm_00118] ⌈In case of a failed transmission (failed PduR_DcmTransmit() request) or error confirmation (Dcm_TpTxConfirmation with error), the DSD submodule shall not repeat the diagnostic response transmission.⌋*()*

Note: Dcm_TpTxConfirmation is only expected when PduR_DcmTransmit succeeded.

[SWS_Dcm_01166] ⌈If the Multiplicity of DcmDslProtocolTx is set to "0" the Dcm shall process the received diagnostic request without sending a response.⌋*()*

More descriptions of the APIs (prototype, input/output parameter) can be found in the interface description of the PduR module [14].

7.4.4.5 Generic Connection Handling

The Dcm shall be able to handle generic connections, identified by DcmPdus with MetaDataItems of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16. These connections carry the actual tester address at run time. Generic connections are supported for diagnostics over IP and FlexRay diagnostics, and CAN diagnostics using normal fixed or mixed 29 bit addressing formats according to ISO15765-2 [15]. Depending on the actual layout of the CAN IDs, generic connections could also be used for extended or normal and mixed 11 bit addressing formats. The Dcm is not aware of the actual addressing format used by CanTp. Several connections may reference the same DcmPdus.

[SWS_Dcm_CONSTR_06044] ⌈Generic connections shall be consistent. This means that the MetaDataItems and the PduLength of all referenced PDUs of a DcmDslConnection (DcmDslProtocolRxPduRef, DcmDslProtocolTxPduRef, DcmDslPeriodicTxPduRef)are identical.⌋*()*

[SWS_Dcm_00848] ⌈The source address of diagnostic requests received via a generic connection must be stored. It is provided in the MetaDataItem SOURCE_ADDRESS_16 provided via Dcm_StartOfReception.⌋*()*

[SWS_Dcm_00849] Target address for generic connection transmission ⌈If the Dcm is about to send a response, response on event, or periodic message for a generic connection request, the Dcm shall set TARGET_ADDRESS_16 to the value of the stored source address in the MetaDataPtr in the PduR_DcmTransmit().⌋*(RS_Diag_04153)*

[SWS_Dcm_01429] ⌈The source address of diagnostic requests received via a generic connection shall be provided in the parameter TesterSourceAddress to the application [SWS_Dcm_01339], [SWS_Dcm_01340], [SWS_Dcm_01341], [SWS_Dcm_01342], [SWS_Dcm_00692], [SWS_Dcm_00694], [SWS_Dcm_00698]).⌋*()*

[SWS_Dcm_01347] ⌈The target address of diagnostic requests received via a generic connection can be provided in the MetaDataItem TARGET_ADDRESS_16 received via Dcm_StartOfReception(). In this case, the Dcm shall ignore physical requests where

the target address is not equal to the configured ECU address DcmDspProtocolEcuAddr.⌋(*RS_Diag_04153*)

[SWS_Dcm_01348] ⌈The source address of the response transmitted via generic connections can be read from the configuration parameter DcmDspProtocolEcuAddr. It shall be provided to PduR_DcmTransmit() in the MetaDataItem SOURCE_ADDRESS_16, if that is configured for the transmit PDU.⌋(*RS_Diag_04153*)

Note: If different source addresses are required for certain transmitted diagnostic messages of the same DcmDslProtocolRow, the MetaDataItem SOURCE_ADDRESS_16 can be omitted from the PDUs, and the address can then be configured in the lower layers. The same is possible for physical requests, where the TARGET_ADDRESS_16 can be omitted from the PDUs.

7.4.4.6 Guarantee timing to tester by sending busy responses

[SWS_Dcm_00024] ⌈If the Application (or the DSP submodule) is able to perform a requested diagnostic task, but needs additional time to finish the task and prepare the response, then the DSL submodule shall send a negative response with NRC 0x78 (Response pending) when reaching the response time (DcmDspSessionP2ServerMax - DcmTimStrP2ServerAdjust respectively DcmDspSessionP2StarServerMax - DcmTimStrP2StarServerAdjust).⌋(*RS_Diag_04016, RS_Diag_04249*)

Rationale for [SWS_Dcm_00024]: The DSL submodule guarantees the response timing to tester.

[SWS_Dcm_00119] ⌈The DSL submodule shall send negative responses as required in [SWS_Dcm_00024] from a separate buffer.⌋()

Rationale for [SWS_Dcm_00119]: This is needed in order to avoid overwriting the ongoing processing of requests, e.g. the application already prepared response contents in the diagnostic buffer. The number of negative responses with NRC 0x78 (response pending) for one diagnostic request can be limited by the configuration parameter DcmDslDiagRespMaxNumRespPend to avoid endless NRC 0x78 transmission in case of an application deadlock.

[SWS_Dcm_01567] ⌈The maximum number of negative responses with NRC 0x78 can be configured using the optional configuration parameter DcmDslDiagRespMaxNumRespPend (see ECUC_Dcm_00693). If this parameter is not configured, the default amount of negative responses with NRC 0x78 is infinite.⌋()

7.4.4.7 Support of periodic transmission

The UDS service ReadDataByPeriodicIdentifier (0x2A) allows the tester to request the periodic transmission of data record values from the ECU identified by one or more periodicDataIdentifiers.

[SWS_Dcm_00122] ⌈The Dcm module shall send responses for periodic transmissions using a separate protocol and a separate buffer of configurable size.⌋*()*

The DcmDslPeriodicTransmissionConRef configuration parameter allows linking the protocol used to receive the periodic transmission request / transmit the periodic transmission response to the protocol used for the transmission of the periodic transmission messages. Note that multiple DcmTxPduIds can be assigned to the periodic transmission protocol. The Dcm module respects several restrictions according to the communication mode:

[SWS_Dcm_00123] ⌈Periodic transmission communication shall only take place in Full Communication Mode.⌋*()*

Periodic transmission events can occur when not in Full Communication Mode. So the following requirement exists:

[SWS_Dcm_00125] ⌈The Dcm module shall discard periodic transmission events beside Full Communication Mode and shall not queue it for transmission.⌋*()*

[SWS_Dcm_00126] ⌈Periodic transmission events shall not activate the Full Communication Mode.⌋*()*

7.4.4.8 Support of segmented response (paged-buffer)

[SWS_Dcm_00028] ⌈If enabled (DcmPagedBufferEnabled=TRUE), the Dcm module shall provide a mechanism to send responses larger than the configured and allocated diagnostic buffer.⌋*()*

[SWS_Dcm_CONSTR_06055] Dependency for DcmDslProtocolMaximumResponseSize ⌈DcmDslProtocolMaximumResponseSize shall be only present if DcmPagedBufferEnabled is set to TRUE.⌋*()*

[SWS_Dcm_01058] ⌈If DcmPagedBufferEnabled == TRUE and the generated Response for a Request is longer than DcmDslProtocolMaximumResponseSize, the Dcm shall respond with NRC 0x14 (DCM_E_RESPONSETOOLONG).⌋*()*

[SWS_Dcm_01059] ⌈If DcmPagedBufferEnabled == FALSE and the generated Response for a Request is longer than Dcm_MsgContextType structure element resMaxDataLen, the Dcm shall respond with NRC 0x14 (DCM_E_RESPONSETOOLONG) .⌋*()*

With paged-buffer handling the ECU is not forced to provide a buffer, which is as large as the maximum length of response. Please note:

- paged-buffer handling is for transmit only - no support for reception.
- paged-buffer handling is not available for the Application (DCM-internal use only).

[SWS_Dcm_01186] ⌈The Dcm shall provide the correct amount of Data requested by the TP or return BUFREQ_E_BUSY in case the requested amount of data is not available.⌋*(RS_Diag_04147)*

Note: In case the requested amount of data is not available, the Dcm should fill up the paged buffer immediately.

7.4.4.9 Support of ResponsePending response triggered by the Application

In some cases, e.g. in case of routine execution, the Application needs to request an immediate NRC 0x78 (Response pending), which shall be sent immediately and not just before reaching the response time (P2ServerMax respectively P2*ServerMax).

When the Dcm module calls an operation and gets an error status DCM_E_FORCE_RCRRP, the DSL submodule will trigger the transmission of a negative response with NRC 0x78 (Response pending). This response needs to be sent from a separate buffer, in order to avoid overwriting the ongoing processing of the request.

7.4.4.10 Manage security level

[SWS_Dcm_00020] ⌈The DSL submodule shall save the level of the current active security level.⌋*(RS_Diag_04005)*

For accessing this level, the DSL submodule provides interfaces to:

- get the current active security level: Dcm_GetSecurityLevel
- set a new security level: DslInternal_SetSecurityLevel()

[SWS_Dcm_00033] ⌈During Dcm initialization the security level is set to the value 0x00 (DCM_SEC_LEV_LOCKED).⌋*(SRS_BSW_00101, RS_Diag_04005)*

[SWS_Dcm_00139] ⌈The DSL shall reset the security level to the value 0x00 (i.e. the security is enabled) under one of the following conditions: - if a transition from any diagnostic session other than the defaultSession to another session other than the defaultSession (including the currently active diagnostic session) is performed or - if a transition from any diagnostic session other than the defaultSession to the defaultSession (DslInternal_SetSecurityLevel()) (initiated by UDS Service DiagnosticSessionControl (0x10) or S3Server timeout) is performed.⌋*()*

Only one security level can be active at a time.

[SWS_Dcm_01329] ⌈On every security level change the Dcm shall update the ModeDeclarationGroup DcmSecurityAccess with the new security level.⌋*()*

[SWS_Dcm_CONSTR_06083] Dependency on DcmDspSecurityAttemptCounterEnabled ⌈If DcmDspSecurityNumAttDelay is not configured, the DcmDspSecurityAttemptCounterEnabled on the same DcmDspSecurityRow shall be set to FALSE.⌋*(RS_Diag_04005)*

[SWS_Dcm_CONSTR_06101] ⌈DcmDspSecurityResetAttemptCounterOnTimeout shall be present only if the DcmDspSecurityAttemptCounterEnabled for DcmDspSecurityRow is set to TRUE.⌋*()*

7.4.4.10.1 Initialization sequence

[SWS_Dcm_01154] ⌈At initialization, for each DcmDspSecurityRow entry for which the DcmDspSecurityAttemptCounterEnabled configuration parameter is set to TRUE, the corresponding Xxx_GetSecurityAttemptCounter shall be called in order to get the value of the AttemptCounter for each of these DcmDspSecurityRow entries.⌋*()*

[SWS_Dcm_01156] ⌈If Xxx_GetSecurityAttemptCounter has returned E_NOT_OK the attempt counter shall be set to the value configured in DcmDspSecurityNumAttDelay of the according SecurityLevel.⌋*()*

[SWS_Dcm_01351] ⌈If any Xxx_GetSecurityAttemptCounter operation returns a DCM_E_PENDING value, the Dcm shall interrupt calling the Xxx_GetSecurityAttemptCounter() in order to resume this chain of calls within the next Dcm_MainFunction() cycle.⌋*()*

Note: this may be the case when these values are stored within some specific non-volatile memory.

[SWS_Dcm_CONSTR_06076] Dependency for DcmDspSecurityGetAttemptCounterFnc ⌈DcmDspSecurityGetAttemptCounterFnc shall be present only if DcmDspSecurityUsePort is set to USE_ASYNCH_FNC and DcmDspSecurityAttemptCounterEnabled is set to TRUE.⌋*()*

[SWS_Dcm_01352] ⌈If the delay after the first call of the Dcm_MainFunction() which is configured in DcmDspSecurityMaxAttemptCounterReadoutTime has been reached and all the Xxx_GetSecurityAttemptCounter have not been called yet (i.e. one operation has returned a DCM_E_PENDING status in the previous Dcm_MainFunction() cycle), the pending operation shall be cancelled by a call with the OpStatus set to DCM_CANCEL.⌋*()*

[SWS_Dcm_01353] ⌈In the conditions of [SWS_Dcm_01352], the AttemptCounters of remaining security levels (which have not been obtained via the calls to their Xxx_GetSecurityAttemptCounter) shall be initialized with the value configured in DcmDspSecurityNumAttDelay of the according SecurityLevel.⌋*()*

[SWS_Dcm_01354] ⌈While not all Xxx_GetSecurityAttemptCounter operations have returned a final status and the operation chain has not been cancelled, the conditionsNotCorrect (0x22) NRC shall be returned to any SecurityAccess (0x27) requestSeed subfunction request.⌋*()*

[SWS_Dcm_01355] ⌈Once all the AttemptCounter values have been successfully or unsuccessfully retrieved (all the Xxx_GetSecurityAttemptCounter() operations have been executed and have returned a final, non-PENDING error value or the operation chain has been cancelled), if at least one of the restored AttemptCounter values is greater than or equal to the DcmDspSecurityNumAttDelay configured for its corresponding DcmDspSecurityRow, the Dcm shall start the SecurityDelayTimer with the higher value of DcmDspSecurityDelayTimeOnBoot / DcmDspSecurityDelayTime of the according DcmDspSecurityRow.⌋*()*

[SWS_Dcm_01356] ⌈A timer (DcmDspSecurityDelayTime, DcmDspSecurityMaxAttemptCounterReadoutTime) which is configured with 0 shall be considered to have timed out instantaneously when it is started, i.e. shall have no delay effect.⌋*()*

[SWS_Dcm_CONSTR_06074] Dependency for DcmDspSecurityMaxAttemptCounterReadoutTime ⌈DcmDspSecurityMaxAttemptCounterReadoutTime shall be a multiple and at minimum equal to DcmTaskTime.⌋*()*

7.4.4.10.2 AttemptCounter update

[SWS_Dcm_01357] ⌈A successful sendKey subfunction request shall reset that security level's specific AttemptCounter.⌋*()*

[SWS_Dcm_01599] ⌈If DcmDspSecurityResetAttemptCounterOnTimeout is set to TRUE and SecurityDelayTimer expires, the Dcm shall reset that security level's specific AttemptCounter.⌋*()*

[SWS_Dcm_01155] ⌈The Dcm shall call Xxx_SetSecurityAttemptCounter() (in case the configuration parameter DcmDspSecurityAttemptCounterEnabled for the according DcmDspSecurityRow is set to TRUE) when the Dcm has changed the attempt counter to inform the application about the counter change.⌋*()*

[SWS_Dcm_CONSTR_06078] Dependency for DcmDspSecuritySetAttemptCounterFnc ⌈DcmDspSecuritySetAttemptCounterFnc shall be present only if DcmDspSecurityUsePort is set to USE_ASYNCH_FNC and the DcmDspSecurityAttemptCounterEnabled set to TRUE.⌋*()*

7.4.4.11 Manage session state

[SWS_Dcm_00022] ⌈The DSL submodule shall save the state of the current active session.⌋*(RS_Diag_04006)*

For accessing this variable, the DSL submodule provides interfaces to:

- get the current active session: `Dcm_GetSesCtrlType`
- set a new session: DslInternal_SetSesCtrlType()

[SWS_Dcm_00034] ⌈During `Dcm` initialization, the session state is set to the value 0x01 ("DefaultSession").⌋*(SRS_BSW_00101)*

[SWS_Dcm_01062] ⌈The call to `Dcm_ResetToDefaultSession` allows the application to reset the current session to Default session and invokes the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl(RTE_MODE_DcmDiagnostic SessionControl_DCM_DEFAULT_SESSION).⌋*()*

Example: Automatic termination of an extended diagnostic session upon exceeding of a speed limit.

7.4.4.12 Manage authentication state

The Dcm provides means for authenticated diagnostics. The DSL sub-module provides an authentication state per diagnostic connection. It initializes this state upon startup and takes care about fallback into non-authenticated states if the connection is idle for some time.

[SWS_Dcm_01477] Authentication state per connection ⌈The Dcm shall provide an authentication state per configured DcmDslConnection.⌋*(RS_Diag_04230)*

[SWS_Dcm_01478] Mode declaration group per authentication state ⌈The Dcm shall provide the state of each authentication state via the ModeDeclarationGroupPrototype DcmAuthentication_<ConnectionName>.⌋*(RS_Diag_04230)*

The Dcm maintains an authentication state and mirrors this state to the mode declaration group DcmAuthentication_<ConnectionName>. This mode declaration group is intended to be changed only by the Dcm, however applications changing this state have no influence on the Dcm authentication state.

[SWS_Dcm_01479] Authentication states ⌈The Dcm shall support per connection the two authentication states:⌋*(RS_Diag_04230)*

- deauthenticated
- authenticated

Upon startup, the Dcm is in deauthenticated state or restores the persisted state. A transition to authenticated state can only be done after the client successfully executed the authentication sequence. In some use cases as in production, a frequent power-on/power off sequence is performed. To keep the achieved authentication state over the power off, there is a dedicated mode rule requesting the Dcm to persist the authenticated state.

[SWS_Dcm_01480] Initialization of authentication state ⌈If DcmDspAuthenticationPersistStateModeRuleRef is not configured or the mode rule referenced by DcmDspAuthenticationPersistStateModeRuleRef is evaluated to false, the Dcm shall initialize within Dcm_Init all authentication states to deauthenticated state.⌋ *(RS_Diag_04230)*

[SWS_Dcm_01481] Initialization of persisted authentication states ⌈If the mode rule referenced by DcmDspAuthenticationPersistStateModeRuleRef is evaluated to true, the Dcm shall initialize the persisted authentication state including role and white list on each connection.⌋ *(RS_Diag_04230)*

Transitions between authenticated states are controlled by both DSL and DSP sub-modules. The DSL sub-module is in charge for fallback of authenticated state into deauthenticated state. The DSP sub-module is in charge for transition changes triggered from a client by diagnostic services.

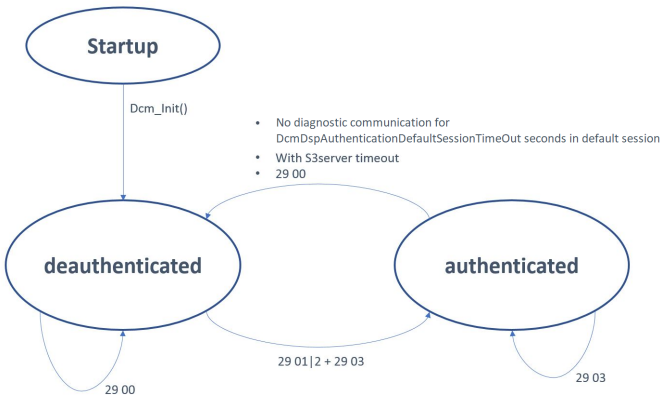

Figure 7.7: Authenticated state transitions without persistent states

[SWS_Dcm_01482] Fallback to deauthenticated session on idle connection ⌈The Dcm shall make a transition from authenticated into deauthenticated state for a configured connection if the following conditions apply:

- The Dcm was in default session when the last diagnostic response was send on that connection and
- DcmDspAuthenticationDefaultSessionTimeOut is configured and no valid diagnostic request was received on that connection for DcmDspAuthenticationDefaultSessionTimeOut seconds after the last Dcm_TpTxConfirmation on that connection.

⌋ *(RS_Diag_04230)*

[SWS_Dcm_01483] Fallback to deauthenticated session on S3server timeout ⌈If the Dcm is in a non-default session and a S3server timeout occurs, the Dcm shall perform a transition from authenticated into deauthenticated state on the authentication state assigned to that connection which was in a non-default session.⌋ *(RS_Diag_04230)*

[SWS_Dcm_01484] Clearing persisted authentication state ⌈If the authentication state of a connection performs a transition to deauthenticated state, the Dcm shall clear all persisted authentication information on that connection.⌋*(RS_Diag_04230)*

[SWS_Dcm_01485] Reaction of fallback into deauthenticated state ⌈Upon a transition from authenticated into deauthenticated state, the Dcm shall discard the current role, white list and use the configured deauthentication role from `DcmDspAuthenticationDeauthenticatedRoleRef`.⌋*(RS_Diag_04230)*

In some use cases, it is desirable that the application set the role instead of using a diagnostic service with its potentially time-consuming certificate parsing. The Dcm provides the API `Dcm_SetDeauthenticatedRole` to overwrite the configured deauthentication role. The overwritten role is only valid in deauthenticated state will not be persisted and is overwritten by a role provided by certificates via service 0x29.

[SWS_Dcm_01486] Default authentication role set from SWC ⌈If a connection is in deauthenticated state and the API `Dcm_SetDeauthenticatedRole` is called, the Dcm shall use the provided deauthenticatedRole as new role per deauthenticated state for this connection.⌋*(RS_Diag_04230)*

[SWS_Dcm_01487] Setting deauthenticated role by SWC only in deauthenticated state ⌈The Dcm shall process a call of `Dcm_SetDeauthenticatedRole` only if the connection is in deauthenticated state.⌋*(RS_Diag_04230)*

[SWS_Dcm_01488] Lifetime of deauthenticated role by SWC ⌈A deauthenticated role set by `Dcm_SetDeauthenticatedRole` is discarded when that connection performs a transition authenticated state.⌋*(RS_Diag_04230)*

[SWS_Dcm_01489] No persistency for deauthenticated roles by SWC ⌈At startup the ECU shall always use the deauthentication state configured in `DcmDspAuthenticationDeauthenticatedRoleRef`.⌋*(RS_Diag_04230)*

7.4.4.13 Non-default sessions

The `Dcm` supports the ISO14229-1 [1] defined sessions. There are two kind of sessions, the default session and the non-default sessions. In a non-default session the S3 timer is used to detect inactive testers and if the timer elapses the default session is entered.

[SWS_Dcm_01668] S3Server session timer support ⌈The `Dcm` shall support the S3Server and start and stop this timer according to ISO14229-2 [13].⌋*(RS_Diag_04249)*

[SWS_Dcm_01669] Sessions and connections ⌈The `Dcm` shall start or stop the S3Server according to ISO14229-2 [13] but only for the client on the `DcmDslConnection` that requested the non-default session.⌋*(RS_Diag_04249)*

[SWS_Dcm_01670] Change to default session upon S3Server elapses ⌈If the S3Server elapses, the `Dcm` shall switch back to default session.⌋*(RS_Diag_04249)*

7.4.4.14 Allow to modify timings

[SWS_Dcm_00027] ⌈The Dcm module shall handle the following protocol timing parameters in compliance with ISO14229-2 [13]: P2ServerMin, P2ServerMax, P2*ServerMin, P2*ServerMax, S3Server⌋*(RS_Diag_04015, RS_Diag_04249)*

[SWS_Dcm_00143] ⌈P2min / P2*min shall be set to defined values: P2min = 0ms, P2*min = 0ms.⌋*(RS_Diag_04015, RS_Diag_04249)*

[SWS_Dcm_01679] Default S3 timeout value ⌈If the parameter DcmS3ServerTimeoutOverwrite is not configured, Dcm shall take the value of 5000ms as the S3Server timeout.⌋*(RS_Diag_04015, RS_Diag_04249)*

[SWS_Dcm_01680] Overwritten S3 timeout value ⌈If the parameter DcmS3ServerTimeoutOverwrite is configured, the Dcm shall take the configured value as the S3Server timeout.⌋*()*

These protocol timing parameters have influence on the session layer timing (no influence on Transport Layer timing). Some of these timing parameters can be modified while protocol is active with the following means:

- UDS Service DiagnosticSessionControl (0x10)
- UDS Service AccessTimingParameter (0x83)

The DSL submodule provides the following functionalities to modify the timing parameters:

- Provide the active timing parameters,
- Set the new timing parameters. Activation of new timing values is only allowed after sending the response.

7.4.4.14.1 Different service tables

For the different protocols a different set of allowed diagnostic services is valid (e.g. the UDS commands for the enhanced diagnosis, the OBD mode services for the OBD protocol). It is possible to create different service tables and link them to the diagnostic protocol.

[SWS_Dcm_00035] ⌈With every protocol initialization, the DSL submodule sets a link to the corresponding service table (see configuration parameter DcmDslProtocol-SIDTable).⌋*(SRS_BSW_00101)*

The DSD submodule uses this link for further processing of diagnostic requests.

7.4.4.14.2 Prioritization of protocol

The configuration parameter DcmDslProtocolPriority makes it possible to give each protocol its own relative priority. Possible use case: There are ECUs, communicating with a vehicle-internal diagnostic tester (running on enhanced diagnosis) and a vehicle-external OBD-II/WWH-OBD tester. The OBD-II/WWH-OBD communication must have a higher priority than the enhanced diagnosis.

[SWS_Dcm_00015] ⌈A protocol with higher priority is allowed to preempt the already running protocol.⌋*(RS_Diag_04021)*

Differentiation of diagnostic protocols is possible, because of different DcmDslProtocolRxPduId values (configured per protocol, see configuration parameter DcmDslProtocolRxPduRef) referenced in the protocol configuration.

7.4.4.14.3 Preemption of protocol

[SWS_Dcm_00459] Callback notification for preempted protocols ⌈If a running diagnostic request is preempted by a higher priority request, the DSL submodule shall call all configured Xxx_StopProtocol() functions on the preempted protocol.⌋*(RS_Diag_04021)*

XXX_StopProtocol functions are configured via the configuration parameter DcmDslCallbackDCMRequestService. Protocol preemption can't be activated with a Concurrent TesterPresent of a higher priority protocol.

[SWS_Dcm_00079] ⌈If a protocol is preempted and this protocol has a running pending response transmission, the Dcm shall call PduR_DcmCancelTransmit () for this transmission with the following parameters: PduId: the id of the Pdu to be canceled⌋*(RS_Diag_04021)*

[SWS_Dcm_00460] ⌈When PduR_DcmCancelTransmit() returns E_NOT_OK, the Dcm module shall stop the current protocol.⌋*(RS_Diag_04021)*

[SWS_Dcm_01046] ⌈If a running diagnostic request is preempted by a higher priority request, the Dcm shall cancel all external pending operations on the preempted protocol with Dcm_OpStatus set to DCM_CANCEL.⌋*(RS_Diag_04021)*

[SWS_Dcm_01047] ⌈In case an operation to the Dem is pending and the new request also requires an interaction with the Dem, the Dcm shall accept the new request and call the corresponding Dem API with the parameters from the new request.⌋*()*

[SWS_Dcm_00575] ⌈If the Dcm is preempting a protocol with a pending reception, the Dcm module shall call cancel that reception with PduR_DcmCancelReceive().⌋*(RS_Diag_04021)*

[SWS_Dcm_00576] ⌈If PduR_DcmCancelReceive () returns E_NOT_OK, the Dcm shall stop the current protocol.⌋*(RS_Diag_04021)*

[SWS_Dcm_00625] ⌈A Low-priority or same-priority request can preempt a higher priority protocol if this higher priority protocol is in default session and no active request is in execution phase. In this case the DSL submodule shall call all configured Xxx_StopProtocol() functions (see configuration parameter DcmDslCallbackDCMRequestService).⌋*()*

[SWS_Dcm_00728] ⌈The handling of protocols with equal priority shall be possible.⌋*()*

[SWS_Dcm_00727] ⌈If a diagnostic request cannot be processed due to a higher priority protocol and DcmDslDiagRespOnSecondDeclinedRequest is set to True, the Dcm shall send NRC 0x21 (BusyRepeatRequest) for the not processed request.⌋*(RS_Diag_04021)*

[SWS_Dcm_01605] ⌈If a diagnostic cannot be processed due to a higher priority protocol and DcmDslDiagRespOnSecondDeclinedRequest is set to False, the Dcm shall ignore the request. In this case no response message at all is generated.⌋*(RS_Diag_04021)*

[SWS_Dcm_00729] ⌈In case of multiple clients with different PduIDs which are requesting the same protocol, as all the connections of the same protocol are having the same priority, a second request (with the different RxPduId) will not be processed. If the configuration parameter DcmDslDiagRespOnSecondDeclinedRequest is TRUE, a negative response with NRC 0x21 (BusyRepeatRequest) shall be issued for the second request. If the configuration parameter is FALSE, no response shall be issued.⌋*()*

Note:

- A multitude of RxPduIDs may be configured per DcmDslProtocol
- These RxPduIDs may be themselves connected to different Testers via the PduR configuration
- This means that many Testers may be configured for the same Protocol
- And this represents a non-UDS extension/use case. In order to have a UDS-compliant flow, there should be one DcmDslProtocol instance per Tester.

[SWS_Dcm_01050] ⌈In case of diagnostic parallel requests, with same / lower priority than the active request then the ComM APIs (ComM_DCM_ActiveDiagnostic, ComM_DCM_InactiveDiagnostic) shall not be called.⌋*()*

7.4.4.14.4 Parallel diagnostic protocol processing

Multiple testers are a common scenario in today's vehicles. In order to reduce the interference between concurrent tester requests to a minimum the Dcm supports parallel diagnostic service processing. This behavior is according to recommended practice of ISO 14229-1 Appendix J. There are certain restrictions, that in non-default session only diagnostic communication from one tester is allowed. In default session and for OBD-II communication it is possible to process diagnostic requests in parallel. Parallel

OBD and UDS communication is particularly important if vehicles are equipped with so called 'OBD dongles' or with electronical logging devices. These devices are installed by the vehicle owner and do diagnostic communication over standardized OBD services. The presence of such devices shall interfere as little as possible with vehicle internal UDS communication. Therefore, whenever it is possible, the Dcm supports parallel processing.

[SWS_Dcm_01602] Processing of parallel requests in default session ⌈If the Dcm receives a request and no further protocol with a higher priority is currently in a non-default session, the Dcm shall accept the new incoming request and process it.⌋ *(RS_Diag_04021)*

[SWS_Dcm_01603] No parallel processing in non-default session ⌈If the Dcm receives a request and a further protocol with a higher priority is currently in a non-default session, the Dcm shall decline the new received request according to [SWS_Dcm_00727].⌋*(RS_Diag_04021)*

Some Dcm interfaces provide access to different diagnostic services, e.g. interface RoutineService for subfunctions Start, Stop and Request Result of a RoutineControl (0x31) or the interface DataServices for the Read and Write operations. On these interfaces only a single client shall access to the data at any point in time.

[SWS_Dcm_01604] Delay parallel processing on the same interface ⌈If the Dcm receives a request and the service processing of this request requires a call to the same interface that is currently processing another request, the Dcm shall delay the call to the interface until the running operation on that interface has finished.⌋*(RS_Diag_04021)*

If the Dcm delays the service processing due to [SWS_Dcm_01604] the standard timing behavior with P2 and NRC 0x78 apply. From an outside perspective, the delayed call to the application looks like that the application itself is taking more time for execution.

[SWS_Dcm_01367] ⌈The Dcm shall process incoming OBD-II requests in parallel to a running UDS request. In this case the protocol priority check according to [SWS_Dcm_00015] is skipped and no protocol pre-emption is done.⌋*(RS_Diag_04163)*

With the container DcmDslProtocolRow, the Dcm configuration supports multiple protocols. Each protocol has a configured DcmDemClientRef defining the Dem client interacting with the Dem. This client Id allows the Dem to distinguish between concurrent calls of the Dcm of the same function or set of functions to process a certain request.

[SWS_Dcm_01369] ⌈While processing a diagnostic request received from a given protocol, the Dcm shall determine the DcmDemClientRef of the DcmDslProtocolRow of the processed protocol. The Dcm shall use this value in all Dem API calls that have a ClientId as parameter.⌋*(RS_Diag_04162)*

[SWS_Dcm_01370] Serialization of multiple calls to the same interface ⌈The Dcm shall internally serialize all asynchronous C/S interface or C function calls to the same

port interface or C function during parallel diagnostic services processing and return a pending to the re-entrant caller.⌋*(RS_Diag_04162)*

[SWS_Dcm_01371] ⌈If the Dcm receives a request on a higher priority protocol than the currently processed request and a diagnostic service in a non-default session is currently processed, the Dcm shall cancel the running diagnostic request, make a transition into default session and process the new received request.⌋*(RS_Diag_04162)*

Integrators will assign OBD protocols the highest priority to meet the legislated response and timing requirements. Therefore, all definitions of 'higher priority protocols' apply to the use case where OBD is used.

[SWS_Dcm_01372] ⌈If the Dcm processes a request from a high priority protocol in default session and the Dcm is receiving a diagnostic request to change in a non-default session, the Dcm shall delay the session change request until the high priority protocol service is finished according to [SWS_Dcm_01371] and make a transition into the requested non-default session.⌋*(RS_Diag_04162)*

[SWS_Dcm_CONSTR_06102] Limitation to one single OBD protocol ⌈The Dcm shall support only one DcmDslProtocolRow with a configured DcmDslProtocolType set to DCM_OBD_ON_<XYZ>.⌋*()*

[SWS_Dcm_CONSTR_06103] OBD procotol shall have highest priority ⌈The Dcm shall support a DcmDslProtocolRow with DcmDslProtocolType set to DCM_OBD_ON_<XYZ> as the highest priority.⌋*()*

7.4.4.14.5 Detection of protocol start

[SWS_Dcm_00036] ⌈With first request of a diagnostic protocol, the DSL submodule shall call all configured Xxx_StartProtocol() functions (see configuration parameter DcmDslCallbackDCMRequestService).⌋*(SRS_BSW_00101)*

Inside this function, the Application can examine the environment conditions and enable/disable further processing of the protocol.

[SWS_Dcm_00144] ⌈After all Xxx_StartProtocol() functions have returned E_OK (meaning all components have allowed the start of the protocol), the default timing parameters are loaded from the default session configuration (see configuration parameter DcmDspSessionRow).⌋*(RS_Diag_04015)*

[SWS_Dcm_CONSTR_06000] Harmonize the naming between interfaces and modes ⌈The shortname of DcmDspSessionRow shall match names of Dcm_SesCtrlType and of the mode declarations of DcmDiagnosticSessionControl. The "DCM_" prefix is mandatory for all shortnames.⌋*()*

[SWS_Dcm_CONSTR_06001] Provide standardized names for ISO standardized diagnostic sessions ⌈The following values of DcmDspSessionLevel which represent ISO defined diagnostic sessions shall be used for the shortname of DcmDspSessionRow:

1 DCM_DEFAULT_SESSION
2 DCM_PROGRAMMING_SESSION
3 DCM_EXTENDED_DIAGNOSTIC_SESSION
4 DCM_SAFETY_SYSTEM_DIAGNOSTIC_SESSION
⌋()

[SWS_Dcm_00145] ⌈After all Xxx_StartProtocol() functions have returned E_OK (meaning all components have allowed the start of the protocol), the service table is set (see configuration parameter DcmDslProtocolSIDTable).⌋()

[SWS_Dcm_00146] ⌈After all Xxx_StartProtocol() functions have returned E_OK (meaning all components have allowed the start of the protocol), the security state is reset.⌋()

[SWS_Dcm_00147] ⌈After all Xxx_StartProtocol() functions have returned E_OK (meaning all components have allowed the start of the protocol), the session state is reset to default session. Furthermore the Dcm module shall invoke the the mode switch of the ModeDeclarationGroupPrototype DcmDiagnosticSessionControl by calling SchM_Switch_<bsnp>_DcmDiagnosticSessionControl(RTE_MODE_DcmDiagnosticSessionControl_DEFAULT_SESSION) .⌋()

Note: <bsnp> is the BSW Scheduler Name Prefix

[SWS_Dcm_00674] ⌈If Xxx_StartProtocol() doesn't return E_OK, the Dcm shall return NRC 0x22.⌋()

7.4.4.14.6 Protocol stop

A protocol stop can appear only in case of protocol preemption (see chapter 7.4.4.14.3 Preemption of protocol).

[SWS_Dcm_00624] ⌈With the reception of Dcm_TpTxConfirmation connected to the response given by the DSL submodule, the Dcm shall not stop the current protocol (no call to xxx_StopProtocol).⌋()

Note: A protocol (e.g. OBD) will be active till reset or other protocol preempts.

[SWS_Dcm_01190] ⌈If Xxx_StopProtocol() doesn't return E_OK, the Dcm shall return NRC 0x22.⌋()

7.4.4.15 Manage resources

Due to limited resources, the following points should be considered as hints for the design:

- It is allowed to use and allocate only one diagnostic buffer in the Dcm module. This buffer is then used for processing the diagnostic requests and responses.

- Output of NRC 0x78 (Response pending) responses is done with a separate buffer.
- paged-buffer handling (see [SWS_Dcm_00028]).

7.4.4.16 Communication Mode Handling

Communication Mode Handling is an interface between Dcm and ComM. The ComM informs the Dcm about the current communication state of a channel. The Dcm is calling the ComM about active Diagnostic which shall prevent an Ecu shutdown/sleep.

The status ActiveDiagnostic shows if diagnostic requests shall keep the ECU awake (ActiveDiagnostic ==’DCM_COMM_ACTIVE’) or if diagnostic requests shall not prevent an Ecu shutdown/sleep (ActiveDiagnostic ==’DCM_COMM_NOT_ACTIVE’). Application can change the status ActiveDiagnostic regarding to system conditions.

[SWS_Dcm_CONSTR_06027] ⌈The application will inform the Dcm by calling Xxx_SetActiveDiagnostic() about the ActiveDiagnostic status.⌋*()*

[SWS_Dcm_01069] ⌈After Dcm_Init, the Dcm shall set ActiveDiagnostic to ’DCM_COMM_ACTIVE’.⌋*()*

[SWS_Dcm_01070] ⌈If Xxx_SetActiveDiagnostic() is called with ’false’ the Dcm set ActiveDiagnostic to ’DCM_COMM_NOT_ACTIVE’.⌋*()*

[SWS_Dcm_01071] ⌈If Xxx_SetActiveDiagnostic() is called with ’true’ the Dcm set ActiveDiagnostic to ’DCM_COMM_ACTIVE’.⌋*()*

[SWS_Dcm_01142] ⌈The Dcm shall wait the Full Communication mode indication from the ComM (call to Dcm_ComM_FullComModeEntered) before initiating the transmission of the diagnostic answer. The time to wait should be no longer than the P2ServerMax calculated from the moment the request was received.⌋*()*

[SWS_Dcm_01143] ⌈If the Dcm fails to confirm a response pending transmission (DCM_E_FORCE_RCRRP) due to [SWS_Dcm_01142], the Dcm shall trigger the Det error DCM_E_FORCE_RCRRP_IN_SILENT_COMM.⌋*()*

Note : On the reception side a silent communication mode can lead to the lost of the request in case of segmented transmission.

7.4.4.16.1 No Communication

The ComM module will indicate the No Communication Mode to the Dcm module by calling Dcm_ComM_NoComModeEntered. In response, the Dcm will immediately disable all transmissions (see the definition of Dcm_ComM_NoComModeEntered for details).

[SWS_Dcm_00148] ⌈Dcm_ComM_NoComModeEntered shall disable all kinds of transmissions (receive and transmit) of communication. This means that the message reception and also the message transmission shall be off.⌋*()*

[SWS_Dcm_00149] ⌈Dcm_ComM_NoComModeEntered shall disable the ResponseOnEvent transmissions.⌋*()*

[SWS_Dcm_00150] ⌈Dcm_ComM_NoComModeEntered shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier).⌋*()*

[SWS_Dcm_00151] ⌈Dcm_ComM_NoComModeEntered shall disable normal transmissions.⌋*()*

[SWS_Dcm_00152] ⌈After Dcm_ComM_NoComModeEntered has been called, the Dcm module shall not call the function PduR_DcmTransmit().⌋*()*

[SWS_Dcm_01324] ⌈In case Dcm_ComM_NoComModeEntered is called with a NetworkId for a ComM channel not referenced within the Dcm (see configuration parameter DcmDslProtocolComMChannelRef), the Dcm shall return without performing any further action.⌋*()*

7.4.4.16.2 Silent Communication

The ComM module will indicate the Silent Communication Mode to the Dcm module by calling Dcm_ComM_SilentComModeEntered. In response, the Dcm will immediately disable all transmissions (see the definition of Dcm_ComM_SilentComModeEntered for details).

[SWS_Dcm_00153] ⌈Dcm_ComM_SilentComModeEntered shall disable all transmission. This means that the message transmission shall be off.⌋*()*

[SWS_Dcm_00154] ⌈Dcm_ComM_SilentComModeEntered shall disable the ResponseOnEvent transmissions.⌋*()*

[SWS_Dcm_00155] ⌈Dcm_ComM_SilentComModeEntered shall disable the periodicId transmissions (ReadDataByPeriodicIdentifier) shall be disabled.⌋*()*

[SWS_Dcm_00156] ⌈Dcm_ComM_SilentComModeEntered shall disable the normal transmissions.⌋*()*

[SWS_Dcm_01325] ⌈In case Dcm_ComM_SilentComModeEntered is called with a NetworkId for a ComM channel not referenced within the Dcm (see configuration parameter DcmDslProtocolComMChannelRef), the Dcm shall return without performing any further action.⌋*()*

7.4.4.16.3 Full Communication

The ComM module will indicate the Full Communication Mode to the Dcm module by calling Dcm_ComM_FullComModeEntered. In response, the Dcm will enable all transmissions (see the definition of Dcm_ComM_FullComModeEntered for details).

[SWS_Dcm_00157] ⌈Dcm_ComM_FullComModeEntered shall enable all kind of communication. This means that the message reception and also the message transmission shall be on.⌋*()*

[SWS_Dcm_00159] ⌈Dcm_ComM_FullComModeEntered shall enable the ResponseOnEvent transmissions.⌋*()*

[SWS_Dcm_00160] ⌈Dcm_ComM_FullComModeEntered shall enable the periodicId transmissions (ReadDataByPeriodicIdentifier).⌋*()*

[SWS_Dcm_00161] ⌈Dcm_ComM_FullComModeEntered shall enable the normal transmissions.⌋*()*

[SWS_Dcm_00162] ⌈After Dcm_ComM_FullComModeEntered has been called, the Dcm shall handle the functions DslInternal_ResponseOnOneDataByPeriodicId() or DslInternal_ResponseOnOneEvent() without restrictions.⌋*()*

[SWS_Dcm_01326] ⌈In case Dcm_ComM_FullComModeEntered is called with a NetworkId for a ComM channel not referenced within the Dcm (see configuration parameter DcmDslProtocolComMChannelRef), the Dcm shall return without performing any further action.⌋*()*

7.4.4.16.4 Diagnostic Activation State

The Dcm notifies the ComM module about the internal diagnostic state for all networks. There are two options for the diagnostic state on a network. In 'active' diagnostic state, the Dcm is processing one or more diagnostic requests from this network or the Dcm is in a non-default session. In 'inactive' diagnostic state, the Dcm is in default session and is not processing a diagnostic request on that network.

When a network has no communication in progress, the Dcm will set the diagnostic activation state to 'inactive'. When there is a diagnostic communication on a network the Dcm sets the diagnostic state to 'active'. In any non-default session, the diagnostic state remains in state 'active'. The communication state can also be controlled by the API Xxx_SetActiveDiagnostic according to [SWS_Dcm_01070] and [SWS_Dcm_01071].

[SWS_Dcm_01373] ⌈The Dcm shall go into 'active' diagnostic state on a network, if a diagnostic request is received on a network or the diagnostic session is changed to any non-default session.⌋*(RS_Diag_04006)*

[SWS_Dcm_01374] ⌈The Dcm shall go into 'inactive' diagnostic state on a network when the current diagnostic request processing is finished and the Dcm is not processing a diagnostic request of another protocol on this network and if the Dcm is in default session.⌋*(RS_Diag_04006)*

[SWS_Dcm_01375] ⌈The Dcm shall go into 'inactive' diagnostic state on all networks if a S3Server timeout occurs and the Dcm makes a transition into default session.⌋*(RS_Diag_04006)*

[SWS_Dcm_01376] ⌈If ActiveDiagnostic is 'DCM_COMM_ACTIVE' and the Dcm is doing a transition into 'active' diagnostic state of a diagnostic protocol, the Dcm shall call ComM_DCM_ActiveDiagnostic(NetworkId), with the networkId associated to the received Pdu (see DcmDslProtocolComMChannelRef), with every request, to inform the ComM module about the need to stay in Full Communication Mode.⌋*(RS_Diag_04006)*

[SWS_Dcm_01377] ⌈Upon a diagnostic state transition into 'inactive', the Dcm shall notify the ComM module about an inactive diagnostic state on a network by calling ComM_DCM_InactiveDiagnostic(NetworkId), with the networkId associated to the received Pdu (see DcmDslProtocolComMChannelRef).⌋*(RS_Diag_04006)*

[SWS_Dcm_01378] ⌈The definition of a finished diagnostic request according to [SWS_Dcm_01374], shall be as follows:

- the Dcm has sent a positive or negative response unequal to NRC 0x78 by receiving the Dcm_TpTxConfirmation connected to the response given by the DSL submodule
- the Dcm has processed the service with SPRMIB=true and the positive response was suppressed
- in case of functional addressing, the Dcm has processed the service and the negative response was suppressed.

⌋*(RS_Diag_04006)*

7.5 Diagnostic Service Dispatcher (DSD)

7.5.1 Introduction

The DSD submodule is responsible to check the validity of an incoming diagnostic request (Verification of Diagnostic Session/Security Access levels/Application permission) and keeps track of the progress of a service request execution.

[SWS_Dcm_00178] ⌈The DSD submodule shall only process valid requests and shall reject invalid ones.⌋*()*