Elektrobit

EB tresos® AutoCore Generic 8 DCCM documentation

product release 8.8.7

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 DCCM documentation

Welcome to the EB tresos AutoCore Generic 8 DCCM (ACG8 DCCM) product documentation.

This document provides:

- Chapter 2, “Supported features“: list of features supported by ACG8 DCCM
- Chapter 3, “ACG8 DCCM release notes“: release notes for the ACG8 DCCM module
- Chapter 4, “ACG8 DCCM user guide“: background information and instructions
- Chapter 5, “ACG8 DCCM module references“: configuration parameters and the application programming interface

2. Supported features

- **Multiple parallel requests:** `Dccm` supports up to 255 parallel requests.
- **Generic send request:** Any UDS payload can be built and provided to the `Dccm` for communication.
- **Functional and physical communication:** `Dccm` supports both functional and physical addressing.
- **Periodic tester present:** Functional communication channels can be configured to send a periodic tester present message.
- **Buffer streaming:** `Dccm` can be configured for streaming.
- **Suppress positive response message indication bit:** `Dccm` supports the `suppressPosResponseMessageIndicationBit`.
- **Request correctly received response pending:** `Dccm` supports the negative response code RCRRP (requestCorrectlyReceivedResponsePending).
- **Communication interface for UDS services:** `Dccm` supports the following UDS services:
  - SID $10 - DiagnosticSessionControl
  - SID $11 - ECUReset
  - SID $27 - SecurityAccess
  - SID $28 - CommunicationControl
  - SID $3E - TesterPresent
  - SID $83 - AccessTimingParameter
  - SID $84 - SecuredDataTransmission
  - SID $85 - ControlDTCSetting
  - SID $87 - LinkControl
  - SID $22 - ReadDataByIdentifier
  - SID $23 - ReadMemoryByAddress
  - SID $24 - ReadScalingDataByIdentifier
  - SID $2C - DynamicallyDefineDataIdentifier
  - SID $2E - WriteDataByIdentifier
  - SID $3D - WriteMemoryByAddress
  - SID $14 - ClearDiagnosticInformation
  - SID $19 - ReadDTCInformation
  - SID $2F - InputOutputControlByIdentifier
  - SID $31 - RoutineControl

  - SID $34 - RequestDownload
  - SID $35 - RequestUpload
  - SID $36 - TransferData
  - SID $37 - RequestTransferExit
  - SID $38 - RequestFileTransfer
- **Configurable timing parameters:** `Dccm` supports the following parameters:
  - P2Client
  - P2*Client
  - P6Client
  - P6*Client
  - InternalTimeout
- **Validation of the response message:** In the context of UDS ISO 14229-1 (2013), the response message can be verified based on the request message. The validation is optional and is possible via the `Dccm_ValidateRespBasedOnRequest()` API.

3. ACG8 DCCM release notes

3.1. Overview

This chapter provides the ACG8 DCCM product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 29.2.0 b220916-0321

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 DCCM release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
No AUTOSAR modules available				

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
Dccm	2.1.3	Elektrobit Automotive GmbH

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

3.3. Module release notes

3.3.1. Dccm module release notes

- Module version: 2.1.3.B567464
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 2.1.3
2022-10-12

- Internal module improvement. This module version update does not affect module functionality.

Module version 2.1.2
2022-06-10

- Fixed non-compliant use of memory sections
- Improved request sending to prioritize functionally addressed messages over physically addressed messages

Module version 2.1.1
2022-02-18

[1]`$TRESOS_BASE` is the location at which you installed EB tresos Studio.

- Increased the possible number of entries in the lists of DccmPduIds in the XDM configuration file to max. 65535 elements

Module version 2.1.0
2021-10-08

- Improved sending the TesterPresent message. Removed the `Dccm_SetTesterPresentInterval()` API. Changed the signature for the `Dccm_EnableTesterPresent()` API. Added the parameter `Interval`.
- Updated the XDM configuration file. Renamed the configuration parameter `DccmMainfunctioCycle` to `DccmTaskTime`.
- ASCDCCM-248 Fixed known issue: Timeout tracking works only for the first diagnostic communication protocol instead of all

Module version 2.0.6
2021-06-25

- Added support for `P6Client` and `P6*Client` timing parameters

Module version 2.0.5
2021-03-05

- Internal module improvement. This module version update does not affect module functionality.

Module version 2.0.4
2020-10-23

- Added support for response validation. `Dccm_ValidateRespBasedOnRequest()` API is now available.
- Updated the configuration files. The configuration tables for functional and physical PDU IDs are in the same tab. Rx PDU ID and Tx PDU ID are easier to be allocated for a specific PDU ID.
- Changed the signature for the `Dccm_SendRequest()` API, removed the parameter `AddressingType`

Module version 2.0.3
2020-03-25

- Replaced the `Default Timeout` and `Default Negative Timeout` with `P2Client` and `P2*Client`

Module version 2.0.2
2020-01-24

- Changed the module name from UdsC to Dccm

Module version 2.0.1
2019-09-30

- Changed the signature for the `Dccm_AllocateDiagnosticProtocol()` API. The BufferStreaming-Callback parameter is now mandatory. If the BufferStreaming is not activated, this pointer should be null.

Module version 2.0.0
2019-04-10

- AUTOSAR 4.0.3 version

Module version 1.0.0
2018-03-14

- Initial AUTOSAR 4.0 version

3.3.1.2. New features

- Validation of the response message: In the context of UDS ISO 14229-1 (2013), the response message can be verified based on the request message. The validation is optional and is possible using the `Dccm_ValidateRespBasedOnRequest()` API.
- Support for two new timeout parameters: P6Client and P6*Client.
- Support for configuring individual timings for sending the TesterPresent message on each functional protocol. The automatic sending of the TesterPresent message is delayed by the last request that was sent on the same communication protocol.

3.3.1.3. Elektrobit-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.1.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- **A unique request for a single protocol**

  Description:

  When a protocol processes a request, it does not allow another request to be processed at the same time.

- **Unsupported UDS services**

  Description:

  The current version of Dccm does not support the following UDS services:

  - ResponseOnEvent
  - ReadDataByPeriodicIdentifier

- **Maximum number of parallel diagnostic protocols**

  Description:

  The maximum number of parallel diagnostic protocols can be configured but is limited to maximum 255.

- **Diagnostic protocols**

  Description:

  The application that is the client of Dccm cannot communicate with a server without first allocating a diagnostic protocol. The number of available diagnostic protocols is limited by:

  - the total number of protocols that is configured for the Dccm module
  - the number of protocols previously allocated by the client application
  - the number of protocols reserved for functional communication (from the configuration of Dccm)

- **Communication type**

  Description:

  From the total number of protocols, the client of the Dccm module can use any number of protocols for functional communication. The number of protocols available for physical communication is the difference between the total number of protocols and the number of protocols reserved for functional communication.

- **Limitation suppressPosRspMsgIndication**

  Description:

  If the suppressPosRspMsgIndication is set to TRUE, Dccm no longer listens to the server responses but sends to the client a notification with a response code. The Dccm protocol status is changed to DC-

CM_DIAGNOSTIC_PROTOCOL_STATUS_READY. If the server wants to send a negative response, is not possible because the StartOfReception() API can only be used if the Dccm protocol has the status DCCM_DIAGNOSTIC_PROTOCOL_STATUS_RECEIVE. This also applies to the NRC 0x78 (requestCorrectlyReceived-ResponsePending RCRRP).

- **ISO timers**

  Description:

  Dccm implements only P2Client, P2*Client, P6Client and P6*Client timers according to ISO14229-2 (2013). Any other timers mentioned in ISO14229-2 are not supported.

  For functional addressing, the handling for P2Client, P2*Client, P6Client and P6*Client is not different compared to physical communication (cf. chapter 8.2 Functional communication, ISO14229-2, 2013).

- **Recommendation for a generic client error handling**

  Description:

  Dccm does not handle errors as recommended in "Table 9 - Recommendation for a generic client error handling" (ISO14229-2, 2013). Dccm sends the error code to the client of the Dccm, and does not do any repeat of the request.

- **Maximum request length**

  Description:

  The maximum amount of data that can be sent using the Dccm_SendRequest() API is 65535 bytes.

- **Functional addressing limitations**

  Description:

  During functional addressing, Dccm sends the messages to a functional address. The system is responsible to broadcast the messages to the relevant servers. After sending a functional message, Dccm will wait or not for responses from the functional address, based on Dccm's configuration parameter Dccm_Functional_Communication_With_No_Response_From_Server. Dccm will not wait for responses from other servers, other than the functional address.

  (Cf. Table 5 - Functionally addressed request message with sub-function parameter and server response behaviour and Table 7 - Functionally addressed request message without sub-function parameter and server response behaviour, ISO14229-1, 2013)

- **Negative response code 0x21 busyRepeatRequest**

  Description:

  Dccm does not perform any special handling when the negative response code busyRepeatRequest is received. The response is forwarded to the client of the Dccm.

3.3.1.6. Open-source software

`Dccm` does not use open-source software.

4. ACG8 DCCM user guide

4.1. Overview

This document gives a short overview of the `Dccm` module. From this user guide you will learn about the basic functionality of the `Dccm`. You will also learn which related modules are necessary to configure the `Dccm` module. The `Dccm` module reference provides further information on configuring the `Dccm` itself.

Note that this user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the `Dccm`. The information provided here should help you to integrate the `Dccm` in your AUTOSAR project.

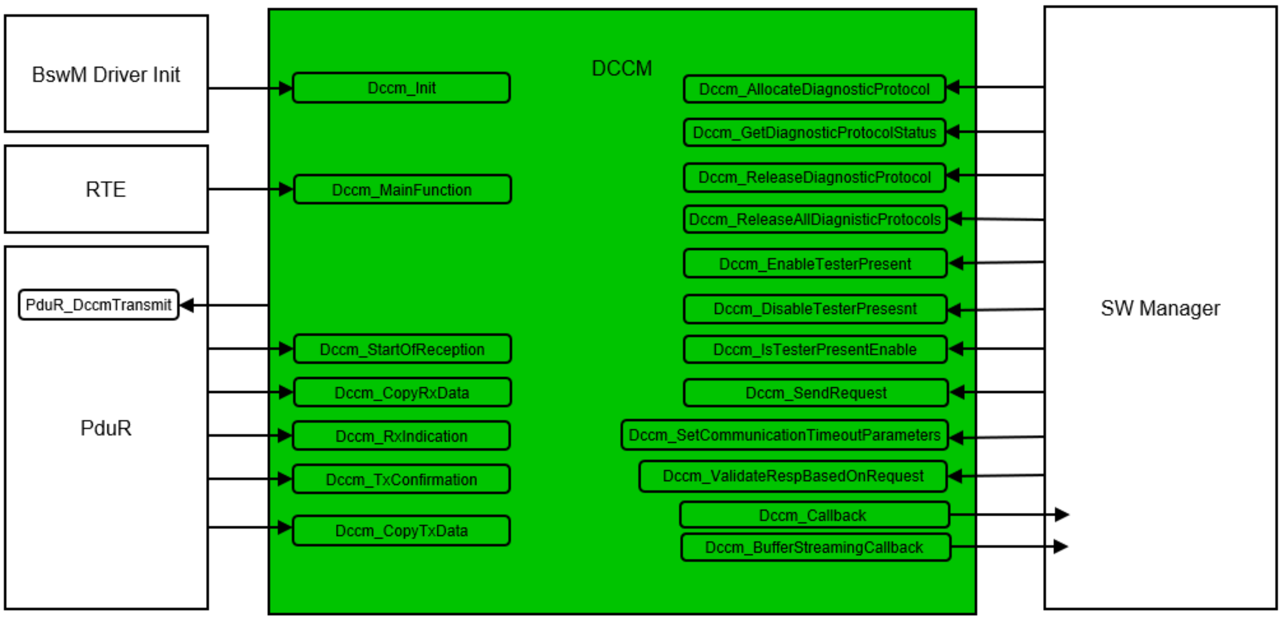
4.2. Background

The Unified Diagnostic Services (UDS) are standardized as part of ISO 14229-1 [4]. With UDS, a tester (client) has the ability to control diagnostic functions in an on-vehicle Electronic Control Unit (server).

The Diagnostic Client Communication Manager (`Dccm`) module provides a UDS communication library that can speed up the development of a UDS client.

4.2.1. Integration interfaces

Before you can use the `Dccm`, you must integrate it into the software environment. The following picture provides an overview of the integration interfaces:


BswM Driver Init
RTE
PduR_DccmTransmit
PduR
DCCM
Dccm_Init
Dccm_MainFunction
Dccm_StartOfReception
Dccm_CopyRxData
Dccm_RxIndication
Dccm_TxConfirmation
Dccm_CopyTxData
Dccm_AllocateDiagnosticProtocol
Dccm_GetDiagnosticProtocolStatus
Dccm_ReleaseDiagnosticProtocol
Dccm_ReleaseAllDiagnisticProtocols
Dccm_EnableTesterPresent
Dccm_DisableTesterPresesnt
Dccm_IsTesterPresentEnable
Dccm_SendRequest
Dccm_SetCommunicationTimeoutParameters
Dccm_ValidateRespBasedOnRequest
Dccm_Callback
Dccm_BufferStreamingCallback
SW Manager

The green box represents the `Dccm`. All functions of the `Dccm` that have to be called are shown.

The white boxes represent the modules of the standard software. The arrows show which functions of those modules are called by the `Dccm` and which functions of the `Dccm` have to be called by the modules. For a description of the interactions, see Section 4.2.4, “External modules”.

4.2.2. Files of the Dccm

The `Dccm` consists of the following source files, which have to be compiled to obtain the full functionality. The `Dccm` configuration files are generated by EB tresos Studio.

- Core files:
  - `Dccm.h`
  - `Dccm_Cbk.h`
  - `Dccm_Internal.h`
  - `Dccm_MainFunction.c`
  - `Dccm_Cbk.c`
  - `Dccm_Service.c`
  - `Dccm_Validation.c`
- Configuration files:
  - `Dccm_Cfg.h`
  - `Dccm_Cfg.c`

4.2.3. External identifiers

The `Dccm` uses certain external identifiers that have to be provided by the software environment. The `Dccm` uses only external identifiers that would be provided by a complete AUTOSAR environment.

4.2.3.1. Platform types

The `Dccm` uses platform types as described in [1]. To obtain those types, it includes the file `Std_Types.h`. The following subset of types is used:

- `uint8`
- `uint16`
- `uint32`
- `boolean`
- `Std_ReturnType`

The following subset of macros is used:

- `TRUE`
- `FALSE`
- `E_OK`
- `E_NOT_OK`

4.2.3.2. Compiler abstraction

The `Dccm` uses compiler abstraction macros as described in [2]. To obtain those types, it includes the file `Std_-Types.h`. The following subset of macros is used:

- `FUNC`
- `P2VAR`
- `P2FUNC`
- `CONST`
- `VAR`
- `STATIC`
- `AUTOMATIC`
- `STD_ON`

- `STD_OFF`

The `Dccm` uses the following `Dccm`-specific macros that also have to be defined:

- `DCCM_VAR`
- `DCCM_CODE`
- `DCCM_APPL_DATA`

4.2.3.3. Memory mapping

The `Dccm` uses memory mapping as described in [3]. For this, it includes the file `Dccm_MemMap.h`. The following macros are used:

- `DCCM_START_SEC_CODE/DCCM_STOP_SEC_CODE`
- `DCCM_START_SEC_CONST_UNSPECIFIED/DCCM_STOP_SEC_CONST_UNSPECIFIED`
- `DCCM_START_SEC_VAR_CLEARED_UNSPECIFIED/DCCM_STOP_SEC_VAR_CLEARED_UNSPECIFIED`
- `DCCM_START_SEC_VAR_INIT_8/DCCM_STOP_SEC_VAR_INIT_8`
- `DCCM_START_SEC_VAR_INIT_UNSPECIFIED/DCCM_STOP_SEC_VAR_INIT_UNSPECIFIED`

4.2.3.4. ComStack types

The `Dccm` interacts with the `PduR` module. For this, it includes the file `ComStack_Types.h`. The following macros are used:

- `NTFRSLT_OK`
- `NTFRSLT_E_TIMEOUT_A`
- `NTFRSLT_E_TIMEOUT_BS`
- `NTFRSLT_E_TIMEOUT_CR`
- `TP_DATACONF`
- `TP_DATARETRY`
- `TP_CONFPENDING`
- `BUFREQ_OK`
- `BUFREQ_E_NOT_OK`
- `BUFREQ_E_BUSY`

The following types are used:

- `NotifResultType`

- `PduIdType`
- `PduInfoType`
- `PduLengthType`
- `RetryInfoType`
- `TpDataStateType`
- `BufReq_ReturnType`

4.2.4. External modules

4.2.4.1. PduR

The `Dccm` has to send and receive data. To do this, it uses the `PduR` module. The header files `PduR.h` and `PduR_Dccm.h` are included and the following function is used: `PduR_DccmTransmit()`.

The `PduR` itself has to be configured to work with the `Dccm`. It must include the header file `Dccm_PduR.h` and use the following callback functions of the `Dccm`:

- `Dccm_CopyRxData()`
- `Dccm_CopyTxData()`
- `Dccm_RxIndication()`
- `Dccm_StartOfReception()`
- `Dccm_TxConfirmation()`

4.2.4.2. Rte

The `Dccm` has to be triggered cyclically. To do this, it uses the `Rte` module. The `Rte` uses the following function: `Dccm_MainFunction()`.

The `Rte` itself has to be configured to work with the `Dccm`. It must include the header `SchM_Dccm.h`.

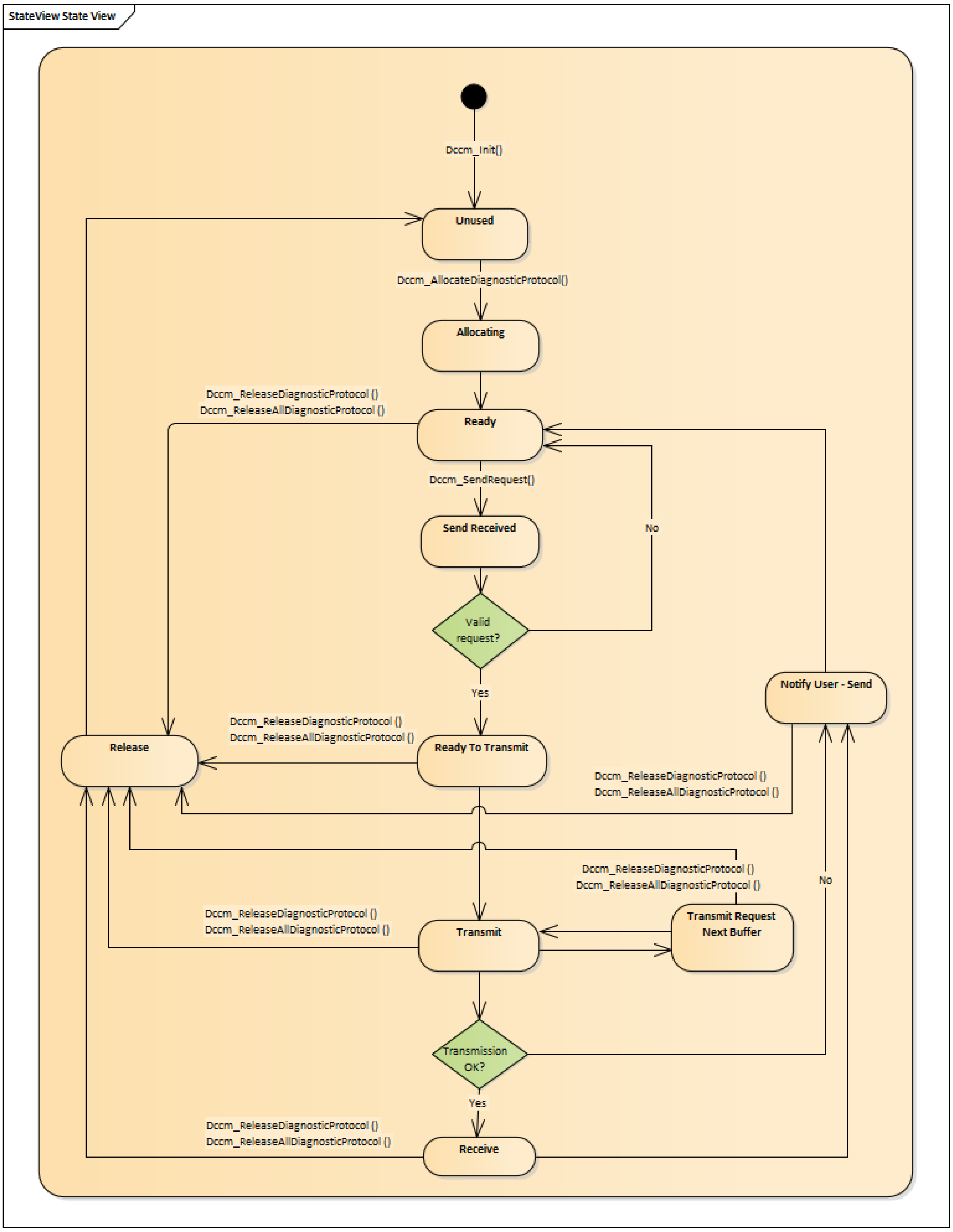
4.2.5. General functions

The main function of the `Dccm` has to be called cyclically. To be able to use this function, the header `Dccm.h` has to be included in the application.

The `Dccm` uses a state machine. This means that most of its tasks are requested asynchronously and are actually performed in `Dccm_MainFunction()`. The integrator has to make sure that the `Dccm_MainFunction()` is called cyclically by the software environment and that the execution time is defined as Periodic task time.

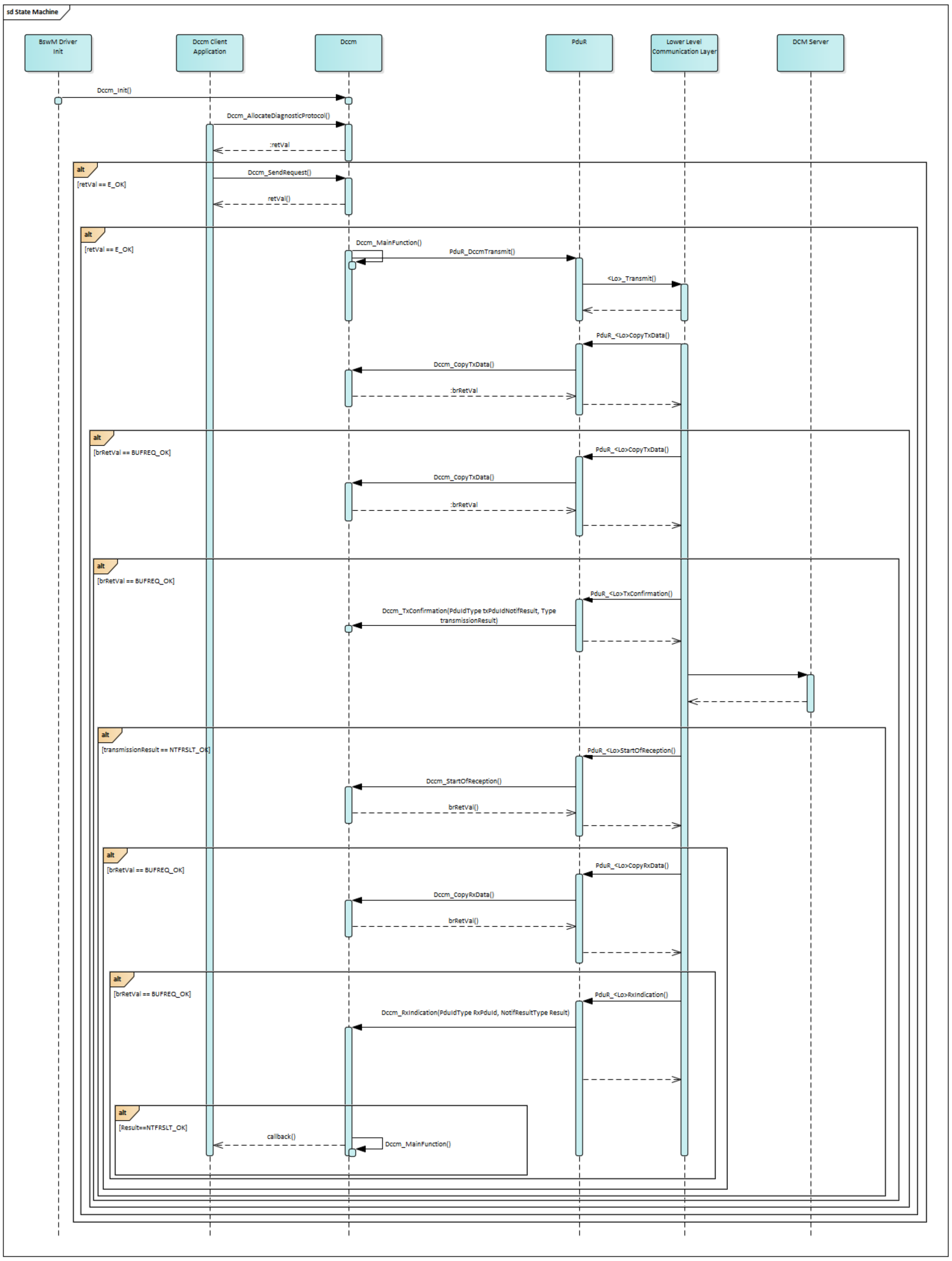
4.2.6. State machine of a diagnostic protocol

The following diagram describes all possible states of a diagnostic protocol. It also shows how transitions take place.


StateView State View
Dccm_Init()
Unused
Dccm_AllocateDiagnosticProtocol()
Allocating
Dccm_ReleaseDiagnosticProtocol ()
Dccm_ReleaseAllDiagnosticProtocol ()
Ready
Dccm_SendRequest()
Send Received
No
Valid request?
Yes
Notify User - Send
Dccm_ReleaseDiagnosticProtocol ()
Dccm_ReleaseAllDiagnosticProtocol ()
Release
Ready To Transmit
Dccm_ReleaseDiagnosticProtocol ()
Dccm_ReleaseAllDiagnosticProtocol ()
Dccm_ReleaseDiagnosticProtocol ()
Dccm_ReleaseAllDiagnosticProtocol ()
No
Dccm_ReleaseDiagnosticProtocol ()
Dccm_ReleaseAllDiagnosticProtocol ()
Transmit
Transmit Request Next Buffer
Transmission OK?
Yes
Dccm_ReleaseDiagnosticProtocol ()
Dccm_ReleaseAllDiagnosticProtocol ()
Receive

4.2.7. Sequence diagram of a request

The following diagram describes the entire scenario when a request is sent. It shows the interaction between the `Dccm` module and the other modules involved. The request shown is a request on a physical protocol for any of the supported services.


sd State Machine
BswM Driver Init
Dccm Client Application
Dccm
PduR
Lower Level Communication Layer
DCM Server
Dccm_Init()
Dccm_AllocateDiagnosticProtocol()
:retVal
alt
[retVal == E_OK]
Dccm_SendRequest()
retVal()
alt
[retVal == E_OK]
Dccm_MainFunction()
PduR_DccmTransmit()
<Lo>_Transmit()
PduR_<Lo>CopyTxData()
Dccm_CopyTxData()
:brRetVal
alt
[brRetVal == BUFREQ_OK]
PduR_<Lo>CopyTxData()
Dccm_CopyTxData()
:brRetVal
alt
[brRetVal == BUFREQ_OK]
PduR_<Lo>TxConfirmation()
Dccm_TxConfirmation(PduIdType txPduIdNotifResult, Type transmissionResult)
alt
[transmissionResult == NTFRSLT_OK]
PduR_<Lo>StartOfReception()
Dccm_StartOfReception()
brRetVal()
alt
[brRetVal == BUFREQ_OK]
PduR_<Lo>CopyRxData()
Dccm_CopyRxData()
brRetVal()
alt
[brRetVal == BUFREQ_OK]
PduR_<Lo>RxIndication()
Dccm_RxIndication(PduIdType RxPduId, NotifResultType Result)
alt
[Result==NTFRSLT_OK]
callback()
Dccm_MainFunction()

4.2.8. Functional communication

The `Dccm` sends the functionally addressed messages (including TesterPresent) with higher priority than the physically addressed messages. This reduces the risk of a timeout of diagnostic sessions, security settings, and authentication states.

4.2.9. TesterPresent message

Sending a message to the TesterPresent service can be enabled only for protocols that use functional addressing. The message indicates to the server that the tester is still present and that it is necessary to keep the session active.

Specific APIs are:

- `Dccm_EnableTesterPresent()`
- `Dccm_DisableTesterPresent()`
- `Dccm_IsTesterPresentEnabled()`

The timing for sending the TesterPresent message must be configured individually for each functional protocol using the API `Dccm_EnableTesterPresent()`.

The automatic sending of the TesterPresent message is delayed by the last request that was sent on the same communication protocol, in order to reduce the network traffic.

4.2.10. Input/output buffer

The `Dccm` client application is responsible to provide an input/output buffer and to maintain the integrity of the buffer for the period of the UDS service request. The `Dccm` client application should estimate the expected amount of return data and provide a buffer of the expected size. The `Dccm` client application should be aware that the buffer data is overwritten by the `Dccm` module during the operation.

4.2.11. Timing parameters

The `Dccm` provides the following timing parameters that are implemented according to [5]:

- DccmTimeoutP2Client
- DccmTimeoutP2StarClient
- DccmTimeoutP6Client
- DccmTimeoutP6StarClient

In addition, you can configure an Internal timeout.

For information on how to configure the timing parameters, see Section 4.3.2, “Configuring Dccm timing parameters”.

4.2.11.1. DccmTimeoutP2Client

The P2Client represents the maximum amount of time in milliseconds between a successfully transmitted request and the start of the corresponding response. The successful transmission of the request message is indicated with `Dccm_TxConfirmation()`. The start of the response message is indicated with `Dccm_StartOfReception()`.

4.2.11.2. DccmTimeoutP2StarClient

The P2*Client represents the maximum amount of time in milliseconds between a response that contains the negative response code 0x78 "requestCorrectlyReceived-ResponsePending" and the start of the next response. The reception of a negative response is indicated with `Dccm_RxIndication()`. The start of an incoming response message is indicated with `Dccm_StartOfReception()`.

4.2.11.3. DccmTimeoutP6Client

The P6Client represents the maximum amount of time in milliseconds between a successfully transmitted request and the complete reception of the corresponding response. The successful transmission of the request message is indicated via `Dccm_TxConfirmation()`. The complete reception of the response message is indicated via `Dccm_RxIndication()`.

4.2.11.4. DccmTimeoutP6StarClient

The P6*Client represents the maximum amount of time in milliseconds between a response containing the negative response code 0x78 "requestCorrectlyReceived-ResponsePending" and the complete reception of the response. The reception of a negative response is indicated via `Dccm_RxIndication()`. The complete reception of the response message is also indicated via `Dccm_RxIndication()`.

4.2.11.5. Internal timeout

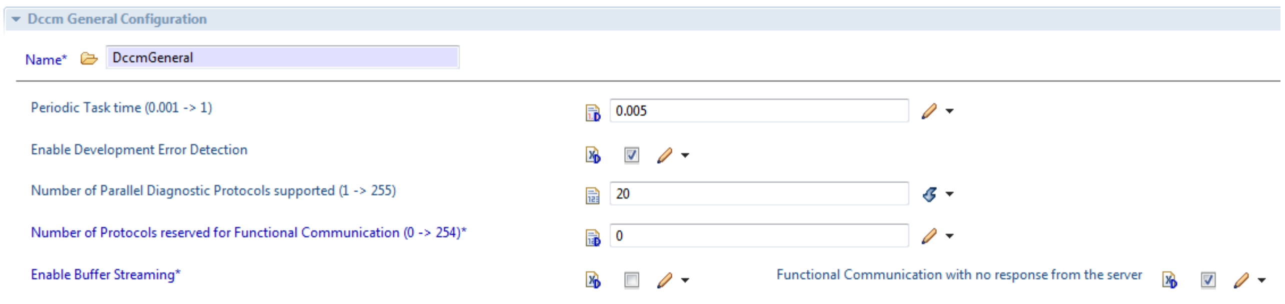
This timer starts twice during the processing of a request:

1. between the start of transmission and the confirmation of transmission for a request message
   The start of transmission is triggered by the `Dccm` call of `PduR_DccmTransmit()`. The successful transmission of the request message is indicated with `Dccm_TxConfirmation()`.

2. between the start and the end of reception for the response message
   The start of the response message is indicated with `Dccm_StartOfReception()`. The end of reception is indicated with `Dccm_RxIndication()`.

4.3. Configuring the Dccm module

4.3.1. Configuring general values



Dccm General Configuration

Name*	DccmGeneral
Periodic Task time (0.001 -> 1)	0.005
Enable Development Error Detection	✓
Number of Parallel Diagnostic Protocols supported (1 -> 255)	20
Number of Protocols reserved for Functional Communication (0 -> 254)*	0
Enable Buffer Streaming*	☐
Functional Communication with no response from the server	✓

4.3.1.1. Periodic task time

With the `Periodic task time` parameter, you configure the scheduling time for the periodic task in seconds.

`DccmTaskTime` affects the scheduling of `Dccm_MainFunction()`. The `Dccm_MainFunction()` is executed after every `DccmTaskTime`.

4.3.1.2. Enable development error detection

This parameter enables the error reporting to the Development Error Tracer (`Det`) module.

- TRUE: Development error detection mechanism is enabled, i.e. switched on.
- FALSE: Development error detection mechanism is disabled, i.e. switched off.

4.3.1.3. Number of parallel diagnostic protocols supported

This parameter sets the number of parallel diagnostic protocols supported.

4.3.1.4. Number of protocols reserved for functional communication

This parameter sets the number of diagnostic protocols used for functional communication.

The value must be smaller than the number of parallel diagnostic protocols.

4.3.1.5. Enable buffer streaming

With this parameter, you can enable the buffer streaming.

When a `Dccm` request needs to transmit bigger quantity of data and not enough memory is available on the ECU, the buffer streaming can be enabled. This allows the client to provide a smaller buffer when calling the `Dccm_SendRequest()` function. After the buffer data is provided to the `PduR`, the `Dccm` requests the next chunk of data from the client.

- TRUE: Buffer streaming is enabled, i.e. switched on.
- FALSE: Buffer streaming is disabled, i.e. switched off.

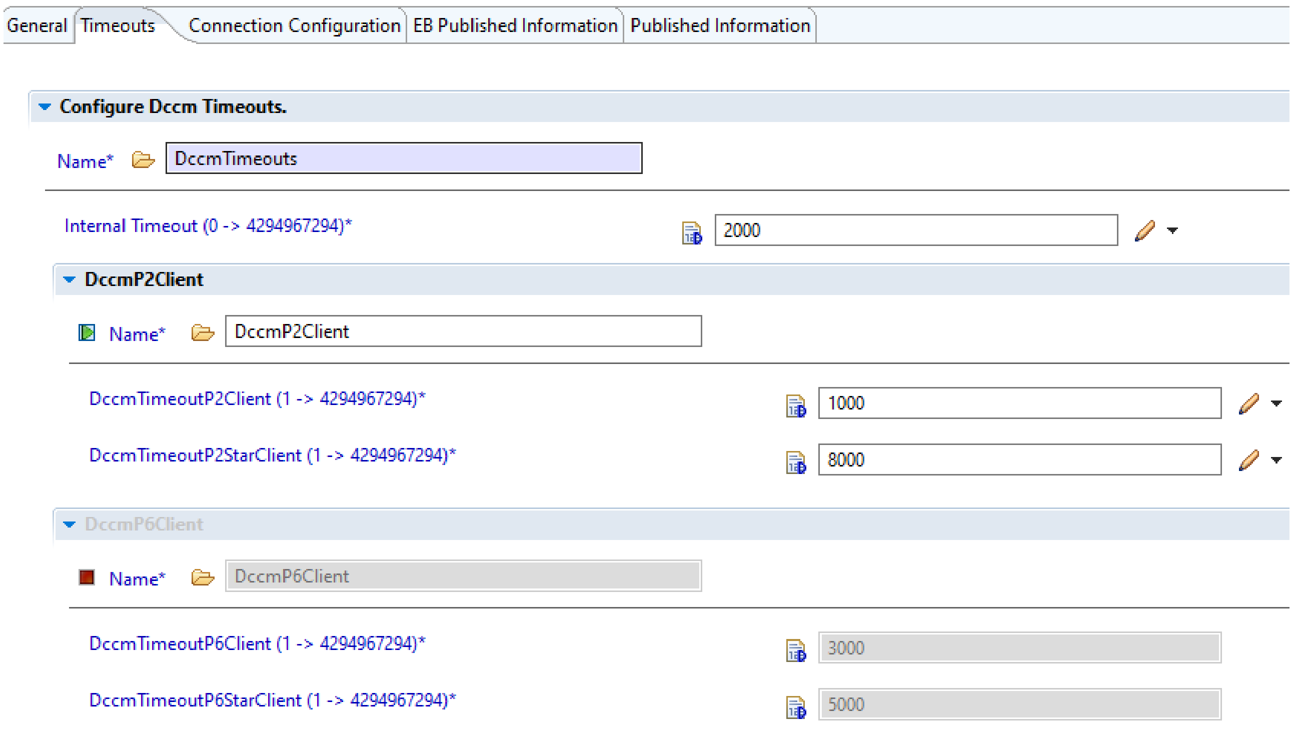
4.3.1.6. Functional communication with no response from the server

The server does not send any response for the requests that use functional communication.

- TRUE: In the case of functional communication, `Dccm` takes into account that the server does not send any response. For the messages that are sent to the server, `Dccm` overwrites the value of the bit `suppress-PosRspMsgIndicationBit` with TRUE.
- FALSE: `Dccm` takes into account that the server sends a response for functional requests. The `Dccm` does not modify the messages that are sent to the server.

4.3.2. Configuring Dccm timing parameters

For background information on the `Dccm` timing parameters, see Section 4.2.11, “Timing parameters”.



Configure Dccm Timeouts.

Name* DccmTimeouts

Internal Timeout (0 -> 4294967294)* 2000

DccmP2Client

Name* DccmP2Client

DccmTimeoutP2Client (1 -> 4294967294)* 1000

DccmTimeoutP2StarClient (1 -> 4294967294)* 8000

DccmP6Client

Name* DccmP6Client

DccmTimeoutP6Client (1 -> 4294967294)* 3000

DccmTimeoutP6StarClient (1 -> 4294967294)* 5000

Figure 4.1. Timeout parameters

Configuring a timeout

Step 1
Go to the **Timeouts** tab.

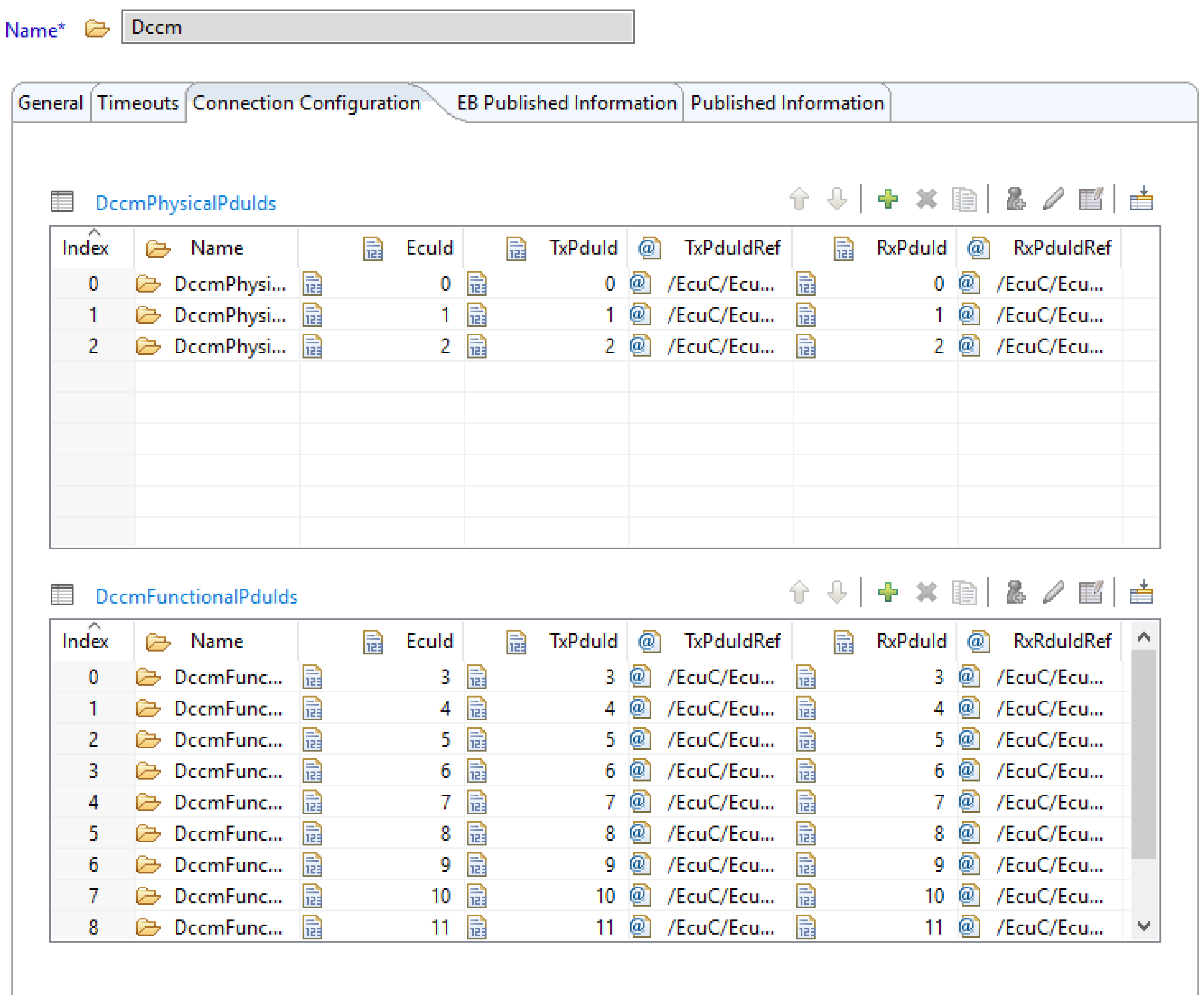
Step 2
Enable the container with the desired client timer:

- To configure P2Client or P2StarClient timeouts, enable the **DccmP2Client** container.
- To configure P6Client or P6StarClient timeouts, enable the **DccmP6Client** container.

Step 3
In the desired timeout parameter (see Figure 4.1, “Timeout parameters”), enter the value in milliseconds.

Step 4
To configure an internal timeout, enter the value in milliseconds in the `Internal Timeout` parameter. If you set the timer value to zero, the timeout is disabled.

4.3.3. Configuring a UDS connection


Name* Dccm

General | Timeouts | Connection Configuration | EB Published Information | Published Information

DccmPhysicalPduIds

Index	Name	EcuId	TxPduId	TxPduIdRef	RxPduId	RxPduIdRef
0	DccmPhysi...	0	0	/EcuC/Ecu...	0	/EcuC/Ecu...
1	DccmPhysi...	1	1	/EcuC/Ecu...	1	/EcuC/Ecu...
2	DccmPhysi...	2	2	/EcuC/Ecu...	2	/EcuC/Ecu...

DccmFunctionalPduIds

Index	Name	EcuId	TxPduId	TxPduIdRef	RxPduId	RxRduIdRef
0	DccmFunc...	3	3	/EcuC/Ecu...	3	/EcuC/Ecu...
1	DccmFunc...	4	4	/EcuC/Ecu...	4	/EcuC/Ecu...
2	DccmFunc...	5	5	/EcuC/Ecu...	5	/EcuC/Ecu...
3	DccmFunc...	6	6	/EcuC/Ecu...	6	/EcuC/Ecu...
4	DccmFunc...	7	7	/EcuC/Ecu...	7	/EcuC/Ecu...
5	DccmFunc...	8	8	/EcuC/Ecu...	8	/EcuC/Ecu...
6	DccmFunc...	9	9	/EcuC/Ecu...	9	/EcuC/Ecu...
7	DccmFunc...	10	10	/EcuC/Ecu...	10	/EcuC/Ecu...
8	DccmFunc...	11	11	/EcuC/Ecu...	11	/EcuC/Ecu...

4.3.3.1. UDS physical connection

4.3.3.1.1. DccmPhysicalPduIds

In the **DccmPhysicalPduIds** container, you configure the `Dccm` transmission and reception channels for physical communication.

4.3.3.2. UDS functional connection

4.3.3.2.1. DccmFunctionalPduIds

In the **DccmFunctionalPduIds** container, you configure the `Dccm` transmission and reception channels for functional communication.

5. ACG8 DCCM module references

5.1. Overview

This chapter provides module references for the ACG8 DCCM product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 DCCM user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

5.2. Dccm

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
DccmGeneral	1..1	**Label:** Dccm General Configuration This container contains the configuration parameters and sub containers of the Dccm module supporting multiple configuration sets. This container and its sub-containers exist once per configuration set.
DccmTimeouts	1..1	**Label:** Configure Dccm Timeouts. This container contains the Dccm timeout configuration.
DccmPhysicalPduIds	0..65535	**Label:** DccmPhysicalPduIds Configuration of the communication with ECUs for physical addressing.
DccmFunctionalPduIds	0..65535	**Label:** DccmFunctionalPduIds Configuration of the communication with ECUs for functional addressing.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

5.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1

Parameters included	
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	255
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.2. DccmGeneral

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
DccmTaskTime	1..1
Dccm_Dev_Error_Detect	1..1
Dccm_Num_Of_Parallel_Diagnostic_Protocols	1..1
Dccm_Num_Of_Functional_Diagnostic_Protocols	1..1
Dccm_Buffer_Streaming	1..1
Dccm_Functional_Communication_With_No_Response_From_Server	1..1

Parameter Name	DccmTaskTime
Label	Periodic Task time
Description	Defines the scheduling time for the periodic task in seconds. Dccm_MainFunction is executed after every DccmTaskTime.
Multiplicity	1..1
Type	FLOAT
Default value	0.005
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	Dccm_Dev_Error_Detect
Label	Enable Development Error Detection
Description	Enables the error-reporting to the Development Error Tracer (DET). ▶ `TRUE`: Development Error Detection mechanism is enabled (switched on). ▶ `FALSE`: Development Error Detection mechanism is disabled (switched off). **Optimization Effect:** ▶ **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ▶ **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	FALSE

Parameter Name	Dccm_Num_Of_Parallel_Diagnostic_Protocols

Label	Number of Parallel Diagnostic Protocols supported
Description	Sets the number of parallel diagnostic protocols supported. ▶ `1`: Only one diagnostic protocol is supported. ▶ `2 .. 255`: Number of parallel diagnostic protocols.
Multiplicity	1..1
Type	INTEGER
Default value	8

Parameter Name	**Dccm_Num_Of_Functional_Diagnostic_Protocols**
Label	Number of Protocols reserved for Functional Communication
Description	Sets the number of diagnostic protocols used for functional communication. Must be smaller than the number of parallel diagnostic protocols.
Multiplicity	1..1
Type	INTEGER
Default value	0

Parameter Name	**Dccm_Buffer_Streaming**
Label	Enable Buffer Streaming
Description	Enables the buffer streaming. When a Dccm request needs to transmit bigger quantity of data and not enough memory is available on the ECU, the buffer streaming can be enabled. This allows the client to provide a smaller buffer when calling the Dccm_SendRequest() function. After the buffer data is provided to PduR, the Dccm requests the next chunk of data from the client. ▶ `TRUE`: Buffer streaming is enabled (switched on). ▶ `FALSE`: Buffer streaming is disabled (switched off).
Multiplicity	1..1
Type	BOOLEAN
Default value	FALSE

Parameter Name	**Dccm_Functional_Communication_With_No_Response_From_Server**
Label	Functional Communication with no response from the server
Description	The server does not send any response for the requests that use functional communication. ▶ `TRUE`: In the case of functional communication, Dccm considers that the server does not send any response. For the messages that are sent to the

	server, Dccm overwrites the value of the bit suppressPosRspMsgIndicationBit to TRUE. ▶ `FALSE`: Dccm considers that the server sends a response for functional requests. The messages that are sent to the server are not modified by Dccm.
Multiplicity	1..1
Type	BOOLEAN
Default value	FALSE

5.2.1.3. DccmTimeouts

Containers included		
Container name	**Multiplicity**	**Description**
DccmP2Client	0..1	Defines the configuration for the timeout P2Client. This container can be enabled only if the container for P6Client is disabled.
DccmP6Client	0..1	Defines the configuration for the timeout P6Client. This container can be enabled only if the container for P2Client is disabled.

Parameters included	
Parameter name	**Multiplicity**
DccmTimeoutInternal	1..1

Parameter Name	**DccmTimeoutInternal**
Label	Internal Timeout
Description	This timer starts twice during the processing of a request: 1. between start of transmission and confirmation of transmission for a request message. The start of transmission is triggered by the Dccm call of PduR_-DccmTransmit(). The successful transmission of the request message is indicated via Dccm_TxConfirmation(). 2. between the start and the end of reception for the response message. The start of the response message is indicated via Dccm_StartOfReception(). The end of reception is indicated via Dccm_RxIndication(). Configurable time in milliseconds. Setting zero as value disables this timeout.
Multiplicity	1..1

Type	INTEGER
Default value	2000

5.2.1.4. DccmP2Client

Parameters included	
Parameter name	**Multiplicity**
DccmTimeoutP2Client	1..1
DccmTimeoutP2StarClient	1..1

Parameter Name	**DccmTimeoutP2Client**	
Label	DccmTimeoutP2Client	
Description	The P2Client represents the maximum amount of time in milliseconds between a successfully transmitted request and the start of the corresponding response. The successful transmission of the request message is indicated via Dccm_TxConfirmation(). The start of the response message is indicated via Dccm_StartOfReception().	
Multiplicity	1..1	
Type	INTEGER	
Default value	1000	
Range	<=4294967294	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DccmTimeoutP2StarClient**
Label	DccmTimeoutP2StarClient
Description	The P2*Client represents the maximum amount of time in milliseconds between a response containing the negative response code 0x78 "requestCorrectlyReceived-ResponsePending" and the start of the next response. The reception of a negative response is indicated via Dccm_RxIndication(). The start of incoming response messages is indicated via Dccm_StartOfReception().
Multiplicity	1..1
Type	INTEGER
Default value	8000
Range	<=4294967294

	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.5. DccmP6Client

Parameters included	
Parameter name	**Multiplicity**
DccmTimeoutP6Client	1..1
DccmTimeoutP6StarClient	1..1

Parameter Name	**DccmTimeoutP6Client**	
Label	DccmTimeoutP6Client	
Description	The P6Client represents the maximum amount of time in milliseconds between a successfully transmitted request and the complete reception of the corresponding response. The successful transmission of the request message is indicated via Dccm_TxConfirmation(). The complete reception of the response message is indicated via Dccm_RxIndication().	
Multiplicity	1..1	
Type	INTEGER	
Default value	3000	
Range	<=4294967294	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DccmTimeoutP6StarClient**
Label	DccmTimeoutP6StarClient
Description	The P6*Client represents the maximum amount of time in milliseconds between a response containing the negative response code 0x78 "requestCorrectlyReceived-ResponsePending" and the complete reception of the response. The reception of a negative response is indicated via Dccm_RxIndication(). The complete reception of the response message is also indicated via Dccm_RxIndication().
Multiplicity	1..1
Type	INTEGER
Default value	5000

Range	<=4294967294	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.6. DccmPhysicalPduIds

Parameters included	
Parameter name	**Multiplicity**
DccmPhysicalEcuId	1..1
DccmPhysicalTxPduId	1..1
DccmPhysicalTxPduIdRef	1..1
DccmPhysicalRxPduId	1..1
DccmPhysicalRxPduIdRef	1..1

Parameter Name	**DccmPhysicalEcuId**
Label	EcuId
Description	ECU ID of the target ECU that should be addressed.
Multiplicity	1..1
Type	INTEGER

Parameter Name	**DccmPhysicalTxPduId**
Label	TxPduId
Description	Handle ID for the PDU used for physical transmission.
Multiplicity	1..1
Type	INTEGER

Parameter Name	**DccmPhysicalTxPduIdRef**
Label	TxPduIdRef
Multiplicity	1..1
Type	REFERENCE
Origin	EB

Parameter Name	**DccmPhysicalRxPduId**
Label	RxPduId

Description	Handle ID for the PDU used for physical transmission.
Multiplicity	1..1
Type	INTEGER

Parameter Name	**DccmPhysicalRxPduIdRef**
Label	RxPduIdRef
Multiplicity	1..1
Type	REFERENCE
Origin	EB

5.2.1.7. DccmFunctionalPduIds

Parameters included	
Parameter name	**Multiplicity**
DccmFunctionalEcuId	1..1
DccmFunctionalTxPduId	1..1
DccmFunctionalTxPduIdRef	1..1
DccmFunctionalRxPduId	1..1
DccmFunctionalRxPduIdRef	1..1

Parameter Name	**DccmFunctionalEcuId**
Label	EcuId
Description	ECU ID of the target ECU that should be addressed.
Multiplicity	1..1
Type	INTEGER

Parameter Name	**DccmFunctionalTxPduId**
Label	TxPduId
Description	Handle ID for the PDU used for functional addressing.
Multiplicity	1..1
Type	INTEGER

Parameter Name	**DccmFunctionalTxPduIdRef**
Label	TxPduIdRef

Multiplicity	1..1
Type	REFERENCE
Origin	EB

Parameter Name	**DccmFunctionalRxPduId**
Label	RxPduId
Description	Handle ID for the PDU used for functional addressing.
Multiplicity	1..1
Type	INTEGER

Parameter Name	**DccmFunctionalRxPduIdRef**
Label	RxRduIdRef
Multiplicity	1..1
Type	REFERENCE
Origin	EB

5.2.1.8. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the Dccm can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.2. Application programming interface (API)

5.2.2.1. Type definitions

5.2.2.1.1. Dccm_BufferStreamingCallbackType

Purpose	Diagnostic Protocol Callback type to request next chunk of data in case Buffer Streaming is enabled.
Type	`Std_ReturnType(*)(Dccm_ProtocolIdType ProtocolId, uint8 *Buffer, uint16 StartLocation, PduLengthType *AvailableDataPtr, uint8 RetryInformation)`

5.2.2.1.2. Dccm_CallbackType

Purpose	Diagnostic Protocol Callback type to notify SW-Manager.
Type	`void(*)(Dccm_ProtocolIdType ProtocolId, Dccm_DiagProtocolRe-sponseCodeType ResponseCode)`

5.2.2.1.3. Dccm_DiagProtocolResponseCodeType

Purpose	This type contains all Dccm Diagnostic Protocol result values, which can be reported via the callback method.
Type	`uint8`

5.2.2.1.4. Dccm_DiagnosticProtocolStatusType

Purpose	Status of a diagnostic protocol.
Type	`uint8`

5.2.2.1.5. Dccm_ProtocolIdType

Purpose	This type is used to identify the diagnostic protocol,.
Type	`uint8`

5.2.2.1.6. Dccm_TimeoutType

Purpose	Type for timeout counter.
Type	`uint32`

5.2.2.2. Macro constants

5.2.2.2.1. BITS3210_BIT_MASK

Purpose	Mask used to extract the low nibble from a specific parameter.
Value	0xFU

5.2.2.2.2. BITS_TO_SHIFT_4

Purpose	Mask used to shift a parameter with 4 bits, used especially to extract the high nibble.
Value	4U

5.2.2.2.3. BITS_TO_SHIFT_8

Purpose	Mask used to shift a parameter with 8 bits.
Value	8U

5.2.2.2.4. DCCM_BIT_MAPPED_REPORTED_WITH_OUT_MASK

Purpose	Macro for ReadScalingDataByIdentifier service which represents the bitMappe-dReportedWithOutMask encoding from scalingByte (High Nibble) parameter.
Value	0x2U

5.2.2.2.5. DCCM_BUFFER_STREAMING

Purpose	

Value	STD_ON

5.2.2.2.6. DCCM_CLEAR_DYNAMICALLY_DEFINED_DATA_IDENTIFIER

Purpose	Macro used for DynamicallyDefineDataIdentifier service which represents the sub-function parameter when equal to 0x03 (clearDynamicallyDefinedDataIdentifier).
Value	0x03U

5.2.2.2.7. DCCM_DEFINE_BY_IDENTIFIER

Purpose	Macro used for DynamicallyDefineDataIdentifier service which represents the sub-function parameter when equal to 0x01 (defineByIdentifier).
Value	0x01U

5.2.2.2.8. DCCM_DEFINE_BY_MEMORY_ADDRESS

Purpose	Macro used for DynamicallyDefineDataIdentifier service which represents the sub-function parameter when equal to 0x02 (defineByMemoryAddress).
Value	0x02U

5.2.2.2.9. DCCM_DEV_ERROR_DETECT

Purpose	
Value	STD_ON

5.2.2.2.10. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_ALLOCATING

Purpose	The status of protocol that is currently going through the allocation process.
Value	0x01U

5.2.2.2.11. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_READY

Purpose	The status of protocol that is ready to start communication.

Value	0x02U

5.2.2.2.12. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_READY_TO_TRANSMIT

Purpose	The status of a protocol that has finished processing a request and is ready to forward it.
Value	0x04U

5.2.2.2.13. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_RECEIVE

Purpose	The status of a protocol that is in the process of receiving the response.
Value	0x06U

5.2.2.2.14. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_RELEASE

Purpose	The status of protocol that is currently going through the release process.
Value	0x08U

5.2.2.2.15. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_SEND_NOTIFY

Purpose	The status of a protocol that has just finished receiving the response and is in the process of transmitting the callback to the Dccm client application.
Value	0x07U

5.2.2.2.16. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_SEND_RECEIVED

Purpose	The status of protocol that is currently going through processing a request.
Value	0x03U

5.2.2.2.17. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_TRANSMIT

Purpose	The transmission was triggered and the protocol is in the process of forwarding the message.

Value	0x05U

5.2.2.2.18. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_TRANSMIT_REQUEST_NEXT_BUFFER

Purpose	The status of protocol when BufferStreaming is enabled and the transmission of the next data packet is requested.
Value	0x09U

5.2.2.2.19. DCCM_DIAGNOSTIC_PROTOCOL_STATUS_UNUSED

Purpose	The status of an unallocated protocol.
Value	0x00U

5.2.2.2.20. DCCM_DTCFORMAT_2

Purpose	Macro used for ReadDTCInformation service which represents the DTCFormatIdentifier parameter when equal to 0x2 (SAE_J1939-73_DTCFormat).
Value	0x2U

5.2.2.2.21. DCCM_DTCFORMAT_4

Purpose	Macro used for ReadDTCInformation service which represents the DTCFormatIdentifier parameter when equal to 0x4 (SAE_J2012-DA_DTCFormat_04).
Value	0x4U

5.2.2.2.22. DCCM_EXE_INTERVAL

Purpose	
Value	[!"num:i($UdsExeInterval)"!]U

5.2.2.2.23. DCCM_E_INVALID_RESPONSE_FORMAT

Purpose	Response code when the response format is wrong.

Value	0x03U

5.2.2.2.24. DCCM_E_INVALID_RESPONSE_LENGTH

Purpose	Response code when the response length is wrong.
Value	0x02U

5.2.2.2.25. DCCM_E_RESPONSE_PENDING

Purpose	Error code returned from server.
Value	0x78U

5.2.2.2.26. DCCM_FUNCTIONAL_COMM_NO_RESPONSE_EXPECTED

Purpose	
Value	STD_ON

5.2.2.2.27. DCCM_INVALID_PROTOCOL_ID

Purpose	A Protocol ID that is considered as invalid value.
Value	0xFFU

5.2.2.2.28. DCCM_LENGTH_0

Purpose	Macro representing the length of 0.
Value	0U

5.2.2.2.29. DCCM_LENGTH_1

Purpose	Macro representing the length of 1.
Value	1U

5.2.2.2.30. DCCM_LENGTH_2

Purpose	Macro representing the length of 2.
Value	2U

5.2.2.2.31. DCCM_LENGTH_3

Purpose	Macro representing the length of 3.
Value	3U

5.2.2.2.32. DCCM_LENGTH_4

Purpose	Macro representing the length of 4.
Value	4U

5.2.2.2.33. DCCM_LENGTH_5

Purpose	Macro representing the length of 5.
Value	5U

5.2.2.2.34. DCCM_LENGTH_6

Purpose	Macro representing the length of 6.
Value	6U

5.2.2.2.35. DCCM_LENGTH_7

Purpose	Macro representing the length of 7.
Value	7U

5.2.2.2.36. DCCM_LENGTH_8

Purpose	Macro representing the length of 8.
Value	8U

5.2.2.2.37. DCCM_MAX_DIAGNOSTIC_PROTOCOLS

Purpose	
Value	[!"num:integer(DccmGeneral/Dccm_Num_Of_Parallel_Diagnostic_Protocols)"!]U

5.2.2.2.38. DCCM_MAX_DTC_EXT_DATA_RECORD_NR_16

Purpose	Macro used for ReadDTCInformation service which represents the maximum value for DTCExtDataRecordNumber parameter for 0x16 subfunction.
Value	0xEFU

5.2.2.2.39. DCCM_MAX_DTC_EXT_DATA_RECORD_NR_19

Purpose	Macro used for ReadDTCInformation service which represents the maximum value for DTCExtDataRecordNumber parameter for 0x19 subfunction.
Value	0xFEU

5.2.2.2.40. DCCM_MAX_DTC_EXT_DATA_RECORD_NR_6_10

Purpose	Macro used for ReadDTCInformation service which represents the maximum value for DTCExtDataRecordNumber parameter for 0x06 and 0x10 subfunctions.
Value	0xFDU

5.2.2.2.41. DCCM_MAX_PHYSICAL_DIAGNOSTIC_PROTOCOLS

Purpose	
Value	[!"num:integer(DccmGeneral/Dccm_Num_Of_Parallel_Diagnostic_Protocols - Dccm-General/Dccm_Num_Of_Functional_Diagnostic_Protocols)"!]U

5.2.2.2.42. DCCM_MAX_SERVERS_FUNCTIONAL_ADDRESSING

Purpose	
Value	[!WS!][!"$Udsserversf"!]U

5.2.2.2.43. DCCM_MAX_SERVERS_PHYSICAL_ADDRESSING

Purpose	
Value	[!WS!][!"$Udsservers"!]U

5.2.2.2.44. DCCM_MODE_OF_OPERATION_DELETE_FILE

Purpose	Macro used for FileTransfer service which represents the modeOfOperation parameter when equal to 0x02 (DeleteFile).
Value	0x02U

5.2.2.2.45. DCCM_MODE_OF_OPERATION_READ_DIR

Purpose	Macro used for FileTransfer service which represents the modeOfOperation parameter when equal to 0x05 (ReadDir).
Value	0x05U

5.2.2.2.46. DCCM_P2CLIENT_ENABLED

Purpose	
Value	STD_ON

5.2.2.2.47. DCCM_P6CLIENT_ENABLED

Purpose	
Value	STD_ON

5.2.2.2.48. DCCM_READ_CURRENTLY_ACTIVE_TIMING_PARAMETERS

Purpose	Macro used for AccessTimingParameter service which represents the sub-function parameter when equal to 0x03 (readCurrentlyActiveTimingParameters).
Value	0x03U

5.2.2.2.49. DCCM_READ_EXTEND_TIMING_PARAMETER_SET

Purpose	Macro used for AccessTimingParameter service which represents the sub-function parameter when equal to 0x01 (readExtendedTimingParameterSet).
Value	0x01U

5.2.2.2.50. DCCM_RETRY_INFO_NULL

Purpose	Macro used to mark that the RetryInfoPtr parameter of the Dccm_CopyTxData() function is null.
Value	0x0FU

5.2.2.2.51. DCCM_RSP_INVALID_RESPONSE_PENDING_FORMAT

Purpose	Response to indicate that a ResponsePending message was received for another service, not the one for which the request was made.
Value	0x04U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.52. DCCM_RSP_OK

Purpose	Requested service executed without error.
Value	0x00U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.53. DCCM_RSP_RX_FAILED

Purpose	Data receiving failed.
Value	0x03U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.54. DCCM_RSP_TIMEOUT

Purpose	Response code received from the (bottom module) PduR.

Value	0x07U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.55. DCCM_RSP_TIMEOUT_INTERNAL

Purpose	No response from server during the internal timer.
Value	0x06U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.56. DCCM_RSP_TIMEOUT_P2CLIENT

Purpose	No response from server during the P2Client or P2*Client timeout.
Value	0x05U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.57. DCCM_RSP_TIMEOUT_P6CLIENT

Purpose	No response from server during the P6Client or P6*Client timeout.
Value	0x09U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.58. DCCM_RSP_TX_FAILED

Purpose	Transmitting of data failed.
Value	0x01U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.59. DCCM_RSP_TX_TRIG_FAILED

Purpose	Triggering of data transmit failed.
Value	0x02U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.60. DCCM_RSP_WRONG_BUFFER_SIZE

Purpose	Receive buffer size is wrong.
Value	0x08U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.61. DCCM_SCALING_BYTE_FORMULA

Purpose	Macro for ReadScalingDataByIdentifier service which represents the formula encoding from scalingByte (High Nibble) parameter.
Value	0x9U

5.2.2.2.62. DCCM_SCALING_BYTE_UNIT_FORMAT

Purpose	Macro for ReadScalingDataByIdentifier service which represents the unit/format encoding from scalingByte (High Nibble) parameter.
Value	0xAU

5.2.2.2.63. DCCM_SERVICE_ECU_RESET_SUBFUNCTION_ENABLE_RAPID_POWER_SHUTDOWN

Purpose	Macro used for EcuReset service which represents the sub-function parameter when equal to 0x04 (enableRapidPowerShutDown).
Value	0x04U

5.2.2.2.64. DCCM_STD_E_BUSY

Purpose	Dccm is busy at the moment: all available diagnostic protocols are in use.
Value	0x02U
Description	Dccm-specific Std_ReturnType value used by Dccm APIs

5.2.2.2.65. DCCM_SUBFUNCTION_MASK

Purpose	Mask used for subfunction to ignore the suppressPosRspMsgIndicationBit.
Value	0x7FU

5.2.2.2.66. DCCM_SUPPRESSBIT_MASK

Purpose	Mask used to extract only the suppressPosRspMsgIndicationBit.
Value	0x80U

5.2.2.2.67. DCCM_TIMEOUT_INTERNAL

Purpose	
Value	0xFFFFFFFFUL

5.2.2.2.68. DCCM_TIMEOUT_P2CLIENT

Purpose	
Value	[!"num:i($DccmTimeoutP2Client)"!]UL

5.2.2.2.69. DCCM_TIMEOUT_P2STARCLIENT

Purpose	
Value	[!"num:i($DccmTimeoutP2StarClient)"!]UL

5.2.2.2.70. DCCM_TIMEOUT_P6CLIENT

Purpose	
Value	[!"num:i($DccmTimeoutP6Client)"!]UL

5.2.2.2.71. DCCM_TIMEOUT_P6STARCLIENT

Purpose	
Value	[!"num:i($DccmTimeoutP6StarClient)"!]UL

5.2.2.2.72. DCCM_TX_CONFIRMATION_OK

Purpose	Response when Tx confirmation was OK and suppress bit is set.

Value	0x11U
Description	Dccm::Dccm_ResponseCode value provided to the upper module in the callback

5.2.2.2.73. DCCM_ZERO_SUBFUNCTION

Purpose	Macro which represents the sub-function parameter when equal to 0x00.
Value	0x00U

5.2.2.2.74. Dccm_ProvideRxBuffer

Purpose	Added for backward compatibility with old PduR implementations.
Value	Dccm_CopyRxData

5.2.2.2.75. Dccm_ProvideTxBuffer

Purpose	Added for backward compatibility with old PduR implementations.
Value	Dccm_CopyTxData

5.2.2.2.76. FUNCTIONAL_ADDRESSING

Purpose	Macro used to define AddressingType of a diagnostic protocol.
Value	1U

5.2.2.2.77. MAX_NO_OF_SERVICES

Purpose	Maximum number of services.
Value	26U

5.2.2.2.78. NEGATIVE_RESPONSE_LENGTH

Purpose	The length of a negative response.
Value	3U

5.2.2.2.79. NEGATIVE_RESPONSE_SID

Purpose	Service ID for a negative response message.
Value	0x7FU

5.2.2.2.80. PHYSICAL_ADDRESSING

Purpose	Macro used to define AddressingType of a diagnostic protocol.
Value	0U

5.2.2.2.81. SID_ACCESS_TIMING_PARAMETER

Purpose	Service ID for access timing parameter request.
Value	0x83U

5.2.2.2.82. SID_ACCESS_TIMING_PARAMETER_RSP

Purpose	Service ID for access timing response.
Value	0xC3U

5.2.2.2.83. SID_CLEAR_DIAGNOSTIC_INFORMATION

Purpose	Service ID for clear diagnostic information request.
Value	0x14U

5.2.2.2.84. SID_CLEAR_DIAGNOSTIC_INFORMATION_RSP

Purpose	Service ID for clear diagnostic information response.
Value	0x54U

5.2.2.2.85. SID_COMMUNICATION_CONTROL

Purpose	Service ID for communication control request.

Value	0x28U

5.2.2.2.86. SID_COMMUNICATION_CONTROL_RSP

Purpose	Service ID for communication control response.
Value	0x68U

5.2.2.2.87. SID_CONTROL_DTC_SETTING

Purpose	Service ID for control DTC setting request.
Value	0x85U

5.2.2.2.88. SID_CONTROL_DTC_SETTING_RSP

Purpose	Service ID for control DTC setting response.
Value	0xC5U

5.2.2.2.89. SID_DIAGNOSTIC_SESSION_CONTROL

Purpose	Service ID for diagnostic session control request.
Value	0x10U

5.2.2.2.90. SID_DIAGNOSTIC_SESSION_CONTROL_RSP

Purpose	Service ID for diagnostic session response.
Value	0x50U

5.2.2.2.91. SID_DYNAMICALLY_DEFINE_DATA_IDENTIFIER

Purpose	Service ID for dynamically define data identifier request.
Value	0x2CU

5.2.2.2.92. SID_DYNAMICALLY_DEFINE_DATA_IDENTIFIER_RSP

Purpose	Service ID for dynamically define data identifier response.
Value	0x6CU

5.2.2.2.93. SID_ECU_RESET

Purpose	Service ID for ECU reset request.
Value	0x11U

5.2.2.2.94. SID_ECU_RESET_RSP

Purpose	Service ID for ECU reset response.
Value	0x51U

5.2.2.2.95. SID_FILE_TRANSFER

Purpose	Service ID for file transfer request.
Value	0x38U

5.2.2.2.96. SID_FILE_TRANSFER_RSP

Purpose	Service ID for file transfer response.
Value	0x78U

5.2.2.2.97. SID_INPUT_OUTPUT_CONTROL_BY_IDENTIFIER

Purpose	Service ID for input output control by identifier request.
Value	0x2FU

5.2.2.2.98. SID_INPUT_OUTPUT_CONTROL_BY_IDENTIFIER_RSP

Purpose	Service ID for input output control by identifier response.

Value	0x6FU

5.2.2.2.99. SID_LINK_CONTROL

Purpose	Service ID for link control request.
Value	0x87U

5.2.2.2.100. SID_LINK_CONTROL_RSP

Purpose	Service ID for link control response.
Value	0xC7U

5.2.2.2.101. SID_READ_DATA_BY_IDENTIFIER

Purpose	Service ID for read data by identifier request.
Value	0x22U

5.2.2.2.102. SID_READ_DATA_BY_IDENTIFIER_RSP

Purpose	Service ID for read data by identifier response.
Value	0x62U

5.2.2.2.103. SID_READ_DATA_BY_PERIODIC_IDENTIFIER

Purpose	Service ID for read data by periodic identifier request.
Value	0x2AU

5.2.2.2.104. SID_READ_DATA_BY_PERIODIC_IDENTIFIER_RSP

Purpose	Service ID for read data by periodic identifier response.
Value	0x6AU

5.2.2.2.105. SID_READ_DTC_INFORMATION

Purpose	Service ID for read DTC information request.
Value	0x19U

5.2.2.2.106. SID_READ_DTC_INFORMATION_RSP

Purpose	Service ID for read DTC information response.
Value	0x59U

5.2.2.2.107. SID_READ_MEMORY_BY_ADDRESS

Purpose	Service ID for read memory by address request.
Value	0x23U

5.2.2.2.108. SID_READ_MEMORY_BY_ADDRESS_RSP

Purpose	Service ID for read memory by address response.
Value	0x63U

5.2.2.2.109. SID_READ_SCALING_DATA_BY_IDENTIFIER

Purpose	Service ID for read scaling data by identifier request.
Value	0x24U

5.2.2.2.110. SID_READ_SCALING_DATA_BY_IDENTIFIER_RSP

Purpose	Service ID for read scaling data by identifier response.
Value	0x64U

5.2.2.2.111. SID_REQUEST_DOWNLOAD

Purpose	Service ID for request download request.
Value	0x34U

5.2.2.2.112. SID_REQUEST_DOWNLOAD_RSP

Purpose	Service ID for request download response.
Value	0x74U

5.2.2.2.113. SID_REQUEST_TRANSFER_EXIT

Purpose	Service ID for transfer exit request.
Value	0x37U

5.2.2.2.114. SID_REQUEST_TRANSFER_EXIT_RSP

Purpose	Service ID for request transfer exit response.
Value	0x77U

5.2.2.2.115. SID_REQUEST_UPLOAD

Purpose	Service ID for request upload request.
Value	0x35U

5.2.2.2.116. SID_REQUEST_UPLOAD_RSP

Purpose	Service ID for request upload response.
Value	0x75U

5.2.2.2.117. SID_RESPONSE_ON_EVENT

Purpose	Service ID for response on event request.
Value	0x86U

5.2.2.2.118. SID_RESPONSE_ON_EVENT_RSP

Purpose	Service ID for response on event response.
Value	0xC6U

5.2.2.2.119. SID_ROUTINE_CONTROL

Purpose	Service ID for routine control request.
Value	0x31U

5.2.2.2.120. SID_ROUTINE_CONTROL_RSP

Purpose	Service ID for routine control response.
Value	0x71U

5.2.2.2.121. SID_SECURED_DATA_TRANSMISSION

Purpose	Service ID for secured data transmission request.
Value	0x84U

5.2.2.2.122. SID_SECURED_DATA_TRANSMISSION_RSP

Purpose	Service ID for secured data transmission response.
Value	0xC4U

5.2.2.2.123. SID_SECURITY_ACCESS

Purpose	Service ID for security access request.
Value	0x27U

5.2.2.2.124. SID_SECURITY_ACCESS_RSP

Purpose	Service ID for security access response.
Value	0x67U

5.2.2.2.125. SID_TESTER_PRESENT

Purpose	Service ID for tester present request.
Value	0x3EU

5.2.2.2.126. SID_TESTER_PRESENT_RSP

Purpose	Service ID for tester present response.
Value	0x7EU

5.2.2.2.127. SID_TRANSFER_DATA

Purpose	Service ID for transfer data request.
Value	0x36U

5.2.2.2.128. SID_TRANSFER_DATA_RSP

Purpose	Service ID for transfer data response.
Value	0x76U

5.2.2.2.129. SID_WRITE_DATA_BY_IDENTIFIER

Purpose	Service ID for write data by identifier request.
Value	0x2EU

5.2.2.2.130. SID_WRITE_DATA_BY_IDENTIFIER_RSP

Purpose	Service ID for write data by identifier response.
Value	0x6EU

5.2.2.2.131. SID_WRITE_MEMORY_BY_ADDRESS

Purpose	Service ID for write memory by address request.
Value	0x3DU

5.2.2.2.132. SID_WRITE_MEMORY_BY_ADDRESS_RSP

Purpose	Service ID for write memory by address response.
Value	0x7DU

5.2.2.3. Functions

5.2.2.3.1. Dccm_AllocateDiagnosticProtocol

Purpose	An interface to allocate a diagnostic protocol.	
Synopsis	`Std_ReturnType` **`Dccm_AllocateDiagnosticProtocol`** `( uint16 TxPduId , uint16 RxPduId , Dccm_ProtocolIdType * ProtocolId , uint8 AddressingType , Dccm_CallbackType Callback , Dccm_BufferStreamingCallbackType BufferStreamingCallback );`	
Service ID	Dccm_AllocateDiagnosticProtocol	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`TxPduId`	The PduId that will be used for sending data.
	`RxPduId`	The PduId that will be used for receiving data.
	`AddressingType`	The protocol is allocated for physical or functional communication. Expected values: PHYSICAL_ADDRESSING or FUNCTIONAL_ADDRESSING.
	`Callback`	The callback function that will be used by the Dccm to inform the caller of a service about the result.
	`BufferStreamingCallback`	The callback function that will be used by the Dccm to ask the next data packet from the client application in the use-case with buffer streaming. If BufferStreaming is not activated this parameter should be null pointer.
Parameters (out)	`ProtocolId`	The ID of the protocol will be returned to the caller of the function.
Return Value	Std_ReturnType	
	`E_OK`	The protocol was allocated
	`E_NOT_OK`	There was an error related with the parameters provided to the function. The ProtocolId OUT parameter contains a value that is not valid (DCCM_INVALID_PROTOCOL_ID). The maximum number of di-

		agnostic protocols used for physical communication has been reached.
	`DCCM_STD_E_BUSY`	There are no available protocols. After a protocol will be released by the client application, it can be allocated again.
Description	This function is used to allocate a diagnostic protocol. It shall be called before sending an Dccm Request. If BufferStreaming is not activated the parameter BufferStreamingCallback should be null pointer. The application that is the client of Dccm can not communicate with a sever without first allocating a diagnostic protocol. The number of available diagnostic protocols is limited by: ▶ the total number of protocols that is configured for the Dccm module, ▶ the number of protocols previously allocated by the client application, and ▶ the number of protocols reserved for functional communication (from the configuration of Dccm). From the total number of protocols, the client of the Dccm module can use any number of protocols for functional communication, but the number of protocols available for physical communication is just the difference between the total number of protocols and the number of protocols reserved for functional communication. A specific TxPduId can be used only once, for a single Dccm communication protocol. A specific RxPduId can be used only once, for a single Dccm communication protocol. Configuration: The maximum number of parallel diagnostic protocols can be configured but is limited to maximum 255.	

5.2.2.3.2. Dccm_CheckBufferSuppressBit

Purpose	Check the Buffer for Suppress Bit.	
Synopsis	`boolean` **`Dccm_CheckBufferSuppressBit`** `( uint16 DataLength , uint8 * Buffer );`	
Service ID	Dccm_CheckBufferSuppressBit	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Return Value	boolean	
	`TRUE`	bit is set

	`FALSE`	bit is not set
Description	This function checks if the suppress positive response message indication bit is set in the input buffer. Configuration: No configuration is needed for this function	

5.2.2.3.3. Dccm_CheckResponsePending

Purpose	Check the Buffer for Response Pending message.	
Synopsis	`boolean` **`Dccm_CheckResponsePending`** `( uint16 DataLength , uint8 * Buffer );`	
Service ID	Dccm_CheckResponsePending	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Return Value	Std_ReturnType	
	`TRUE`	if the response pending is set
	`FALSE`	the buffer is not big enough or if the response pending is not set
Description	This function is used to check if the Response Pending message has been set in the input buffer. Configuration: No configuration is needed for this function	

5.2.2.3.4. Dccm_CopyRxData

Purpose	API to copy data from receive buffer.	
Synopsis	`BufReq_ReturnType` **`Dccm_CopyRxData`** `( PduIdType RxPduId , PduInfoType * PduInfoPtr , PduLengthType * RxBufferSizePtr );`	
Service ID	Dccm_CopyRxData	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`RxPduId`	- Dccm handle ID to be used for Dccm APIs to be called from PduR.
	`PduInfoPtr`	- Pointer providing received data and data length.

Parameters (out)	`RxBufferSizePtr`	- The number of bytes that are still unused in the receive buffer, and that can be used to store the next data packages that will be received.
Return Value	BufReq_ReturnType	
	`BUFREQ_OK`	- Data is copied.
	`BUFREQ_E_NOT_OK`	- Request failed.
Description	This function copies the TpRx data to the Dccm receive buffer.	

5.2.2.3.5. Dccm_CopyTxData

Purpose	API to request data to transmit.	
Synopsis	`BufReq_ReturnType` **`Dccm_CopyTxData`** `( PduIdType TxPduId , PduInfoType * PduInfoPtr , RetryInfoType * RetryInfoPtr , PduLengthType * AvailableDataPtr );`	
Service ID	Dccm_CopyTxData	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`TxPduId`	- Dccm handle ID to be used for Dccm APIs to be called from PduR.
	`PduInfoPtr`	- Pointer providing a buffer and length to copy the Tx data.
	`RetryInfoPtr`	- This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. Please see the document Autosar SWS for PduRouter for details.
Parameters (out)	`AvailableDataPtr`	- Pointer which returns remaining number of bytes to be copied. Eg.: from a total of 10 bytes, only 3 were transmitted during the current call of Dccm_CopyTxData(), so AvailableDataPtr will show that there are 7 bytes that wait to be transmitted, with further calls to the same function Dccm_CopyTxData().
Return Value	BufReq_ReturnType	

	`BUFREQ_OK`	- Data is copied.
	`BUFREQ_E_BUSY`	- The number of bytes that still wait to be transmitted, after the call to this function ends. Eg.: from a total of 10 bytes, only 3 were transmitted during the current call of Dccm_CopyTxData(), so AvailableDataPtr will show that there are 7 bytes that wait to be transmitted, with further calls to the same function Dccm_CopyTxData().
	`BUFREQ_E_NOT_OK`	- Request failed.
Description	This function copies the Dccm transmit data to the CanTp transmit buffer.	

5.2.2.3.6. Dccm_DisableTesterPresent

Purpose	Disable the periodic sending of tester present.	
Synopsis	`Std_ReturnType` **`Dccm_DisableTesterPresent`** `( Dccm_ProtocolIdType ProtocolId );`	
Service ID	Dccm_DisableTesterPresent	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	The ID of the Dccm protocol.
Return Value	Std_ReturnType	
	`E_OK`	TesterPresent functional message was disabled for the Dccm protocol ProtocolId.
	`E_NOT_OK`	There was an error: ▶ the sending of the message TesterPresent is not enabled for the ProtocolId provided as input parameter, or ▶ the ProtocolId is not valid, or ▶ the module was not properly initialized.
Description	This function is used to disable the periodic sending of tester present for the ProtocolID provided. Use-case 1: if the sending of the TesterPresent message is not ongoing, TesterPresent will be disabled during the call of this function. Use-case 2: if the sending of the TesterPresent message was already triggered by Dccm, Dccm will disable the periodic sending of TesterPresent message after the invocation of Dccm_Tx-	

	Confirmation(), in the next call of Dccm_MainFunction(). Until then, Dccm_IsTesterPresentEnabled() will return false.

5.2.2.3.7. Dccm_EnableTesterPresent

Purpose	Enable the periodic sending of the TesterPresent message.	
Synopsis	`Std_ReturnType` **`Dccm_EnableTesterPresent`** `( Dccm_ProtocolIdType ProtocolId , uint8 * Buffer , uint16 BufferLength , uint16 * DataLengthPtr , Dccm_TimeoutType Interval );`	
Service ID	Dccm_EnableTesterPresent	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	The ID of the Dccm protocol.
	`BufferLength`	- should be at least 3 bytes length, to have enough room for a negative response from the server
	`Interval`	The time interval between initiating two consecutive TesterPresent messages. The value should be long enough to permit the proper sending of the messages: ▶ Dccm_MainFunction() must be called a number of times to assure that the protocols switch through a number of states; ▶ PduR must have enough time to call the call-back functions (provided by Dccm) involved in the sending of the messages.
Parameters (out)	`Buffer`	- the content of buffer is ignored
	`DataLengthPtr`	- will contain the answer from the server (if it is received)
Return Value	Std_ReturnType	
	`E_OK`	Status was retrieved and returned to the user
	`E_NOT_OK`	ProtocolId not correct, buffer is too short, Buffer is null, DataLengthPtr is null, no

		functional address is set in the configuration of Dccm, BufferLength is smaller than 3, or the protocol identified with ProtocolId was not allocated for functional communication.
Description	This function is used to enable the periodic sending of the TesterPresent message on a specific Dccm protocol. In Dccm, the TesterPresent functionality can be used only for functional communication. Because of this limitation, the Dccm protocol must be allocated specifically for functional communication.	

5.2.2.3.8. Dccm_GetDiagnosticProtocolStatus

Purpose	Returns the status of a Diagnostic Protocol.	
Synopsis	`Std_ReturnType` **`Dccm_GetDiagnosticProtocolStatus`** `( Dccm_ProtocolIdType ProtocolId , Dccm_DiagnosticProtocolStatusType * Status );`	
Service ID	Dccm_GetDiagnosticProtocolStatus	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	The status of this protocol ID is queried.
Parameters (out)	`Status`	This is the status of the protocol that is returned to the user.
Return Value	Std_ReturnType	
	`E_OK`	Status was retrieved and returned to the user
	`E_NOT_OK`	ProtocolID is out of the pre-configured range of values, or the Status parameter is null.
Description	This function is used to query the status of a specific Diagnostic Protocol.	

5.2.2.3.9. Dccm_Init

Purpose	Initializes or reinitializes the Dccm module.
Synopsis	`void` **`Dccm_Init`** `( void );`
Service ID	Dccm_Init

Sync/Async	Synchronous
Reentrancy	Non reentrant
Description	This function resets all relevant variables to the default values. This function shall be used during the startup phase of the ECU after the NVRAM Manager has finished the restore of NVRAM data. SW-Components including Monitor Functions are initialized afterwards. Caveats: The Dccm is not functional until this function has been called.

5.2.2.3.10. Dccm_IsTesterPresentEnabled

Purpose	Check if the tester present is enabled or not.	
Synopsis	`Std_ReturnType` **`Dccm_IsTesterPresentEnabled`** `( Dccm_ProtocolId-Type ProtocolId , boolean * IsTesterPresentEnabled );`	
Service ID	Dccm_IsTesterPresentEnabled	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	The ID of the Dccm protocol.
Parameters (out)	`IsTesterPresentEnabled`	will be TRUE if TesterPresent notification is enabled for the protocol with the ID Pro-tocolId.
Return Value	Std_ReturnType	
	`E_OK`	Status was retrieved and returned to the user
	`E_NOT_OK`	the ProtocolId is not good, or IsTesterPre-sentEnabled is NULL.
Description	This function will return the status of tester present for the selected Dccm protocol.	

5.2.2.3.11. Dccm_MainFunction

Purpose	Processes the Dccm requests.
Synopsis	`void` **`Dccm_MainFunction`** `( void );`
Service ID	Dccm_MainFunction

Sync/Async	Synchronous
Reentrancy	Non reentrant
Description	This function is used to process the Dccm requests. It shall be called periodically as a cyclic task by the software system (e.g. by operating system). If a Main function of a un-initialized module is called from the BSW Scheduler, then it shall return immediately without performing any functionality and without raising any errors. Timing: fixed cyclic Configuration: The cyclic time for the main function has to be defined as an operating system task or runnable entity.

5.2.2.3.12. Dccm_ReleaseAllDiagnosticProtocols

Purpose	An interface to release all the Dccm Diagnostic Protocols.	
Synopsis	`Std_ReturnType` **`Dccm_ReleaseAllDiagnosticProtocols`** `( void );`	
Service ID	Dccm_ReleaseAllDiagnosticProtocols	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Return Value	Std_ReturnType	
	`E_OK`	All the diagnostic protocols were properly released. Non-releasable statuses: RELEASE or ALLOCATING.
	`E_NOT_OK`	No protocol has been released because at least one protocol is still in one of the following situations: ▶ the state is RELEASE or ALLOCATING; ▶ or the TesterPresent feature is enabled and the sending of the TesterPresent message is on-going.
Description	This function is used to release all allocated diagnostic protocols It shall be called when there is no need for communication. Configuration: The maximum number of parallel diagnostic protocols can be configured but is limited to maximum 255 (0x00 - 0xFE: 0U - 254U).	

5.2.2.3.13. Dccm_ReleaseDiagnosticProtocol

Purpose	An interface to release a Dccm Diagnostic Protocol.	
Synopsis	`Std_ReturnType` **`Dccm_ReleaseDiagnosticProtocol`** `( Dccm_ProtocolIdType ProtocolId );`	
Service ID	Dccm_ReleaseDiagnosticProtocol	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	Release the protocol with this ID.
Return Value	Std_ReturnType	
	`E_OK`	The protocol was released
	`E_NOT_OK`	The ProtocolID is outside of the pre-configured range of values, or the protocol identified with this ProtocolID is in one of the following states: UNUSED, RELEASE, ALLOCATING.
Description	This function is used to release a diagnostic protocol It shall be called when there is no need for communication over the specific PduId. If TesterPresent is enabled for the current protocol, Dccm will take care to disable the TesterPresent sending. Configuration: The maximum number of parallel diagnostic protocols can be configured but is limited to maximum 255 (0x00 - 0xFE: 0U - 254U).	

5.2.2.3.14. Dccm_RxIndication

Purpose	API to indicate that all receptions have finished.	
Synopsis	`void` **`Dccm_RxIndication`** `( PduIdType RxPduId , NotifResultType Result );`	
Service ID	Dccm_RxIndication	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`RxPduId`	- Dccm handle ID to be used for Dccm APIs to be called from PduR.
	`Result`	- Result of the finished reception.
Description	This function will be called if data has been received.	

5.2.2.3.15. Dccm_SendRequest

Purpose	Sends an UDS payload over a Diagnostic Protocol.	
Synopsis	`Std_ReturnType` **`Dccm_SendRequest`** `( Dccm_ProtocolIdType ProtocolId , uint8 * Buffer , uint16 BufferLength , uint16 * DataLengthPtr );`	
Service ID	Dccm_SendRequest	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	The Protocol ID that will be used in the communication with the server.
	`BufferLength`	Size of the buffer.
Parameters (in,out)	`Buffer`	A pointer to the start of the buffer where the data received from the server will be stored.
	`DataLengthPtr`	IN: The number of bytes that will be sent to the server. OUT: A pointer to return the number of bytes received from the server.
Return Value	Std_ReturnType	
	`E_OK`	Service accepted
	`E_NOT_OK`	ProtocolId not correct, buffer is too short, Buffer is null or DataLengthPtr is null.
	`DCCM_STD_E_BUSY`	A request is active.
Description	This function is used to initiate the sending of the UDS payload over a Diagnostic Protocol. The Diagnostic Protocol should be in the READY state otherwise the call will return with error. Configuration: The maximum number of parallel diagnostic protocols can be configured but is limited to maximum 255. The PduIds should be configured and the function will check if the provided PduId is not over the limit.	

5.2.2.3.16. Dccm_SetCommunicationTimeoutParameters

Purpose	Set the timeout parameters for a specific diagnostic protocol.
Synopsis	`Std_ReturnType` **`Dccm_SetCommunicationTimeoutParameters`** `( Dccm_ProtocolIdType ProtocolId , Dccm_TimeoutType P2ClientConfigurationValue , Dccm_TimeoutType InternalTimeout , Dccm_TimeoutType P2StarClientConfigurationValue );`

Service ID	Dccm_SetCommunicationTimeoutParameters	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`ProtocolId`	The ID of the Dccm protocol.
	`P2ClientConfigurationValue`	The value to be used for the start of P2Client timeout.
	`InternalTimeout`	The value to be used for the start of internal timeout.
	`P2StarClientConfigurationValue`	The value to be used for the start of P2StarClient timeout.
Return Value	Std_ReturnType	
	`E_OK`	Diagnostic Protocol communication parameters were successfully updated
	`E_NOT_OK`	the ProtocolId is not valid or not in the DCCM_DIAGNOSTIC_PROTOCOL_STATUS_READY
Description	This function is used to set the timeout parameters for a specific diagnostic protocol. Depending on the configuration, the input parameters may be P2ClientConfigurationValue and P2StarClientConfigurationValue or P6ClientConfigurationValue and P6StarClientConfigurationValue.	

5.2.2.3.17. Dccm_StartOfReception

Purpose	API to start a reception.	
Synopsis	`BufReq_ReturnType` **`Dccm_StartOfReception`** `( PduIdType RxPduId , PduLengthType TpTotalLength , PduLengthType * RxBufferSizePtr );`	
Service ID	Dccm_StartOfReception	
Sync/Async	Synchronous	
Reentrancy	::Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`RxPduId`	- Dccm handle ID to be used for Dccm APIs to be called from PduR.
	`TpTotalLength`	- Message length.
Parameters (out)	`RxBufferSizePtr`	Available Rx buffer in the Dccm module.
Return Value	BufReq_ReturnType	

	`BUFREQ_OK`	- Reception request has been accepted. RxBufferSizePtr indicates the available receive buffer.
	`BUFREQ_E_NOT_OK`	- Reception request has been rejected. RxBufferSizePtr remains unchanged.
Description	This function is called once by PduR if a connection has been established.	

5.2.2.3.18. Dccm_TxConfirmation

Purpose	API to confirm a TCP transmission.	
Synopsis	`void` **`Dccm_TxConfirmation`** `( PduIdType TxPduId , NotifResultType Result );`	
Service ID	Dccm_TxConfirmation	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	`TxPduId`	- Dccm handle ID to be used for Dccm APIs to be called from PduR.
	`Result`	- Parameter indicates the result of the transmission.
Description	This function indicates if the transmission was successful.	

5.2.2.3.19. Dccm_ValidateRespBasedOnRequest

Purpose	This function validates a response based on request.	
Synopsis	`Std_ReturnType` **`Dccm_ValidateRespBasedOnRequest`** `( uint8 * RequestBuffer , uint32 RequestDataLength , uint8 * ResponseBuffer , uint32 ResponseDataLength );`	
Parameters (in)	`RequestBuffer`	The request buffer used for validation.
	`RequestDataLength`	The length of the request.
	`ResponseBuffer`	The response buffer to be validated.
	`ResponseDataLength`	The length of the response.
Return Value	Std_ReturnType	
	`E_OK`	The response buffer is correct.

	`E_NOT_OK`	The parameters that the function was called up are invalid or the minimum length of request required to perform the checks is not met.
	`DCCM_E_INVALID_RESPONSE_LENGTH`	The positive response may have a fixed length or may be a changeable length. If the length varies the function will only check the minimum length. For negative response length should be 3 bytes. If the length does not meet the requirements stated above, this error will be returned.
	`DCCM_E_INVALID_RESPONSE_FORMAT`	The SID from the positive response does not match the SID that should follow the request; the 2nd byte of the negative response is not the SID in the request; if the service has a DID, sub-function or a byte that must be echo, those that come in response do not match those in the request.
Description	The function is used to validate a response based on the request in terms of length and format.	

5.2.3. Integration notes

5.2.3.1. Exclusive areas

This section describes the exclusive areas used by the `Dccm` module.

5.2.3.2. Production errors

The module does not report any production errors.

5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CONST_UNSPECIFIED
VAR_CLEARED_UNSPECIFIED
VAR_INIT_8
VAR_INIT_UNSPECIFIED

5.2.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Dccm module.

5.2.3.5. Platform integration

Search for `Platforms Setting` in the source code of the Dccm plugin or in this document to find all locations where a platform specific setting is required.

Bibliography

[1] *Specification of Platform Types, AUTOSAR 4.0.3*

[2] *Specification of Compiler Abstraction, AUTOSAR 4.0.3*

[3] *Specification of Memory Mapping, AUTOSAR 4.0.3*

[4] *Road vehicles - Unified diagnostic services (UDS) ISO14229-1, 2013*

[5] *Road vehicles - Unified diagnostic services (UDS) ISO14229-2, 2013*