MICROSAR CddDrm

Technical Reference

Version 7.0.1

Authors	visade, visygr, vsarcesta, vsarcmeba, vsarcmiem
Status	Released

Document Information

History

Author	Date	Version	Remarks
visade	2016-01-21	1.0.0	initial version
visygr	2018-09-14	2.1.0	**Added:** *2.15 Limitations* *4.2.4.19 CddDrm_GetFuncRequestResult()* **Modified:** *2.1 Features* *2.4 Request Transmission* *4.1 Type Definitions*
visygr	2019-06-18	3.0.0	**Modified:** *1 Component History*
vsarcesta, vsarcmeba	2020-05-07	3.1.0	**Modified:** *2.4 Request Transmission* *2.7 Timeout Supervision*
vsarcesta, vsarcmeba	2020-08-13	4.0.0	**Added:** *4.2.5.3 CddDrm_CancelReceive()* *4.4.4 CddDrm_StartOfReception()* *4.4.5 CddDrm_CopyRxData()* *4.4.6 CddDrm_CopyTxData()* *4.4.7 CddDrm_TpRxIndication()* *4.4.8 CddDrm_TpTxConfirmation()* **Modified:** *2.4 Request Transmission* *2.16.1 Development Error Reporting* *3.1.1 Static Files* *3.2 Include Structure* *3.4.1 Exclusive Area 0* *4.1 Type Definitions* *4.3 Services used by CddDrm* *5.4 Callback Functions* *5.3 Configuration to Diagnose ECUs Connected to the Network* *5.4 Configuration to Diagnose the own ECU* *6.2 Abbreviations*
vsarcmiem	2020-10-12	4.1.0	**Removed:** *1 Component History* **Modified:** *Reference Documents* *1 Introduction*

vsarcmiem	2021-06-14	5.0.0	**Added:** *2.6 Dynamic Connection* *5.6 Configuration for Dynamic Connection* *4.2.6 Interface CanTp* *4.2.6.1 CddDrm_CanTpRxMetaData()* *4.2.6.2 CddDrm_CanTpTxMetaData()* **Modified:** *2.15 Limitations* *2.15.1 Development Error Reporting* *2.16.1 Development Error Reporting* *4.2.4 CddDrm_SvcSend_<>()* *4.1 Type Definitions* *6.2 Abbreviations*
vsarcmiem	2021-09-01	5.0.3	**Modified:** *4.2.4.1 CddDrm_SvcSend()*
vsarcmiem	2021-09-07	5.0.4	**Modified:** *4.1 Type Definitions* *2.6 Dynamic Connection*
vsarcmiem	2021-11-26	5.0.5	**Modified:** *3.1.1* Static Files
vsarcmiem	2023-01-10	6.0.0	**Modified:** *4.1 Type Definitions*
vsarcmiem	2023-01-31	6.1.0	**Modified:** *2.4 Request Transmission* *4.4.4 CddDrm_StartOfReception()* *4.4.5 CddDrm_CopyRxData()* *4.4.6 CddDrm_CopyTxData()* *4.2.4.19 CddDrm_GetFuncRequestResult()*
vsarcmiem	2024-01-25	7.0.0	**Modified:** *2.4 Request Transmission* *2.8 Tester Present Message*
vsarcmiem	2024-02-21	7.0.1	**Modified:** *2.4 Request Transmission* *4.1 Type Definitions* *4.5.1.1 <ResponseNotification>()*

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	List of Basic Software Modules	R19-11
[2]	AUTOSAR	Specification of Default Error Tracer	R4.0.3

Scope of the Document

This technical reference describes the general use of the complex device driver CddDrm.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Introduction

This document describes the functionality, API and configuration of the MICROSAR complex device driver CddDrm.

Latest Analyzed AUTOSAR Release:	4	
AUTOSAR Schema Compatibility:	4.0.3	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CddDrm_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CddDrm_MODULE_ID	255 decimal (according to ref. [1])

The CddDrm sends diagnostic requests triggered by application and handles the responses. It can be used to implement a diagnostic on-board tester in the vehicle network.

1.1 Architecture Overview

The following figure shows where the CddDrm is located in the AUTOSAR architecture.

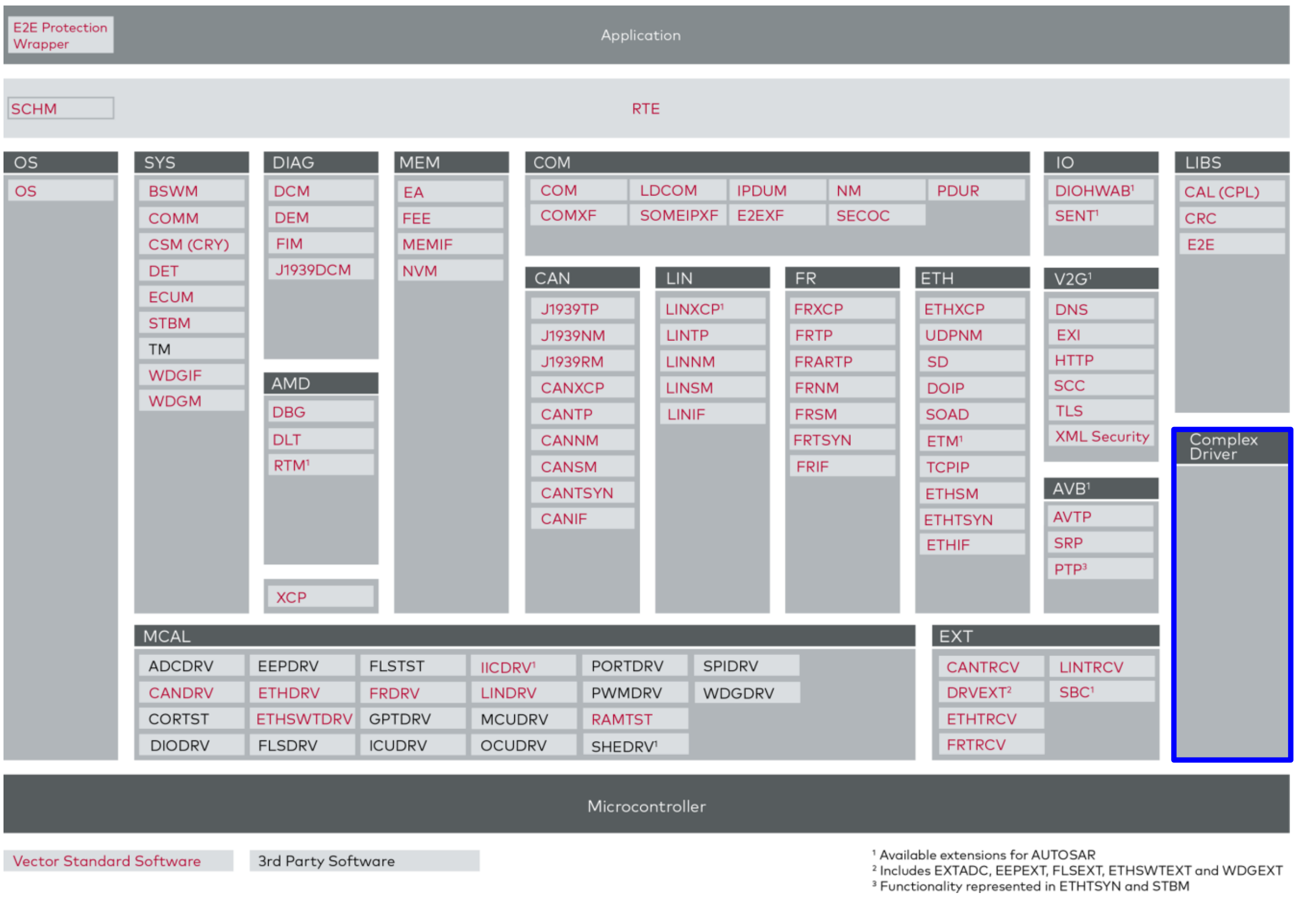

Figure 1-1 AUTOSAR 4.2 Architecture Overview

The next figure shows the interfaces to adjacent modules of the CddDrm. These interfaces are described in chapter 4.

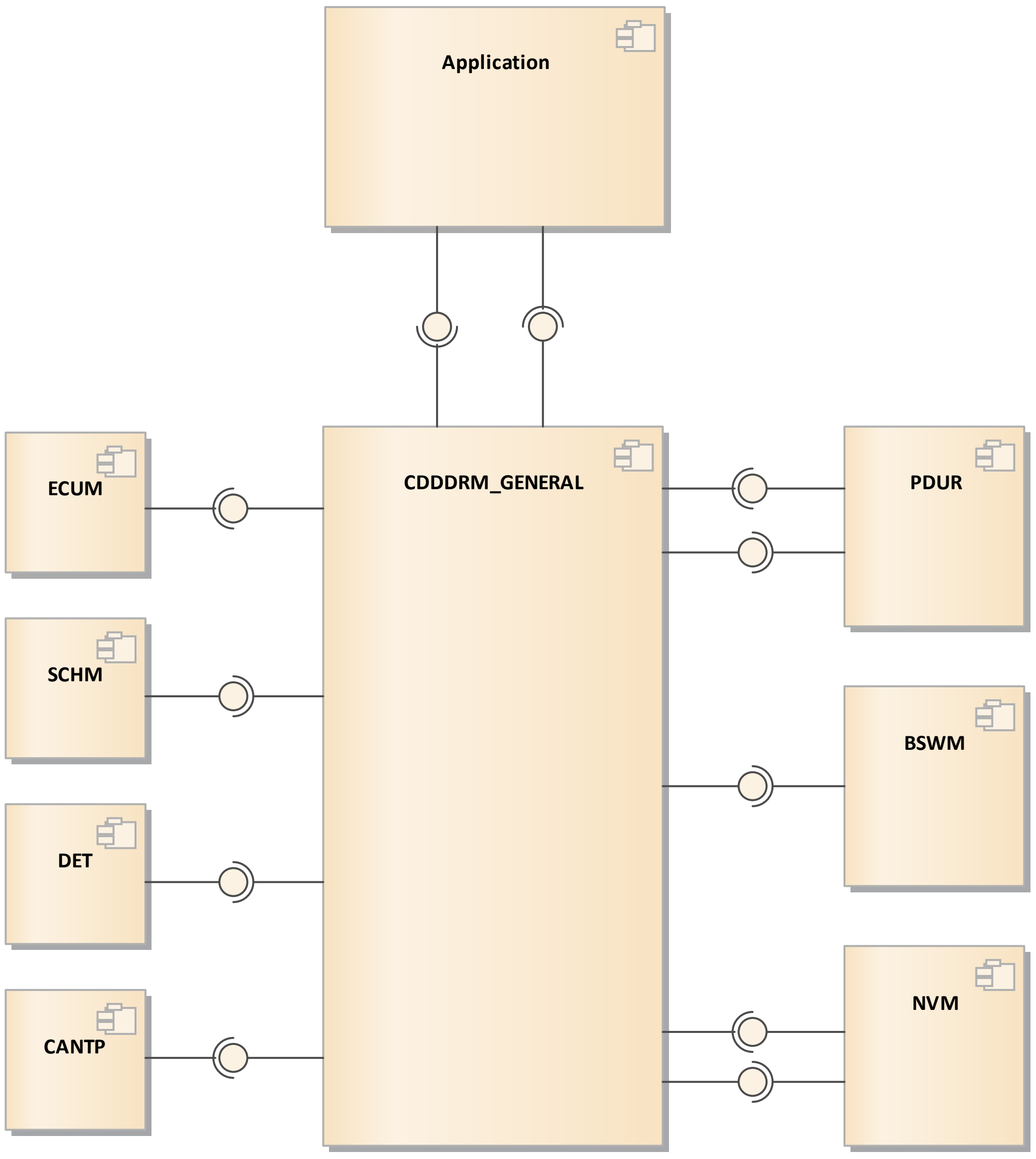

Figure 1-2 Interfaces to adjacent modules of the CddDrm

2 Functional Description

2.1 Features

The features listed in the following tables cover the complete functionality specified for the CddDrm.

> Table 2-1 Supported CddDrm features

The following features are supported:

Supported Features
Physical Request Transmission
Functional Request Transmission
Response Evaluation
Dynamic Connection
Timeout Supervision
Tester Present Message
Parallel Connections
Self-Diagnosis
Other Tester Active Mode
ECU Detection
Service ID Firewall
Service Cancelation

Table 2-1 Supported CddDrm features

2.2 Initialization

The CddDrm module is initialized via the API CddDrm_Init().

After (re)start of the ECU the CddDrm is in state “UNINITIALIZED”. In this state the CddDrm is not operable. API calls in this state will cause a DET error report.

CddDrm_Init() will change the state to “INITIALIZED”. In this state the CddDrm is fully operable and all API services can now be requested.

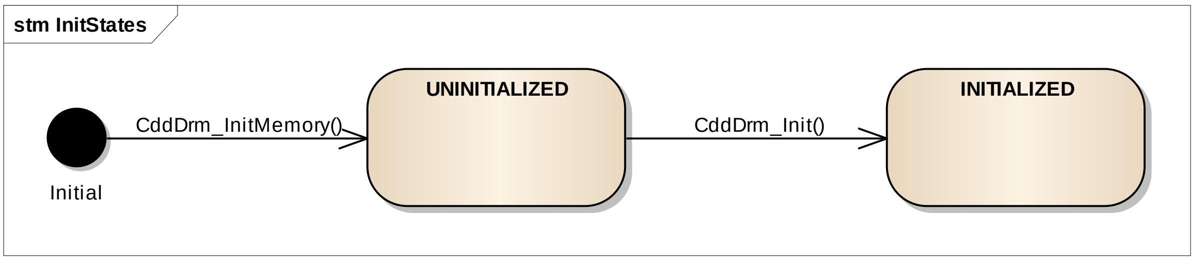

Figure 2-1 Initialization states of the CddDrm

2.3 Main Function

The CddDrm main function CddDrm_MainFunction() has to be called periodically in the configured time period. The main function processes all internal timer and state tasks.

2.4 Request Transmission

One of the main tasks of the CddDrm is to send UDS requests to other ECUs. CddDrm can handle physical and functional request transmission and reception. To trigger the transmission of a diagnostic request, the CddDrm provides service specific APIs to the application (refer to chapter 4.2.4).

Additional to those service specific interfaces the CddDrm supports a generic interface to be more flexible. One use-case can be the support of OEM or supplier specific diagnostic services.

The CddDrm supports a configurable interface (refer to chapter 4.5.1) to notify the application for a successful data transmission as soon as all request data was transferred to the PduR.

To reduce the bus load the CddDrm supports two timer values. The delay time is a global timeout value between two requests send by the CddDrm, whereas the separation time is handled connection specific.

The CddDrm supports P3 functional request delay that can be configured globally and will be applied for each functional connection.

If the SPRMIB bit is set, the CddDrm does not expect a positive response within P2 time. Therefore, the application will be notified for a positive response with the response length set to zero.

Note
For functional request processing, the application needs to collect the connection specific result/response (refer to chapter 4.2.4.19). It can be done within the <ResponseNotification>() callback for the connection response during the functional request. The results are not provided directly in the request/response buffer of the application like it is done for physical request processing.

Note
For processing of a functional TesterPresent request(0x3E) without the SPRMIB bit set, the application will be notified with a timeout response and the response message length set to 0, after the P2 timer has elapsed. However, for a functional TesterPresent request(0x3E) with SPRMIB bit set, CddDrm will notify the application as soon as the request processing is finished and not wait for the P2 timer to elapse. This is to allow for other requests to get processed.

The functional TesterPresent request(0x3E) with SPRMIB bit set will disregard the P3 functional request delay so the request will always be processed despite the configured delay.

Note
To send requests with a length larger than 65kB be aware that the length type of the Send API depends on the size of the maximal configured Pdu size.

2.5 Response Evaluation

After reception of a request, the diagnostic server returns a response. The CddDrm waits for these responses, checks for consistency and notifies the application, depending on the kind of the response:

- **positive response:** check for requested SID + 0x40 offset
- **negative response:** check for response length equals 3 byte and requested SID is available
  - **RCRRP:** the reception of a response pending is notified to the application if forwarding functionality is enabled, either way the counter is decremented for each RCRRP reception. If the maximum number of accepted RCRRPs is reached, the application is notified with an NRC.

2.6 Dynamic Connection

Dynamic connection can be used to re-use the CddDrm configuration in different vehicle variants and reduce manual effort as the dynamic connections can be redirected to the wanted CAN ID during runtime.

Dynamic connection allows the application to send diagnostic requests to the wanted CAN ID by providing the request and response CAN IDs to CddDrm within the request to send call. Response CAN ID is used to map the following response to the correct request. The request CAN ID shall be placed in the requestMetaDataBufferPtr element in bufferInfo argument when sending request through either the generic service API or the service specific APIs. The response metadata shall be placed in the responseMetaDataBufferPtr element in bufferInfo.

The responseBufferSize in bufferInfo argument for both the generic service API and the service specific APIs will have to be extended with the supported metadata length as the response CAN ID will be included in the response buffer. The response CAN ID received from the response will be placed at the start of the buffer in responseBufferDataPtr.

When using functional request together with physical dynamic connections, an offset shall be placed in the responseMetaDataBufferPtr element in bufferInfo instead of a CAN ID so that when receiving the response on the physical dynamic connections, the request CAN ID for the FC-frame is calculated using the response CAN ID and the offset. The offset can only be a positive number up to 255 but it has to be smaller than the response CAN ID.

The feature is realized using metadata to send the CAN ID between the layers. More details regarding configuration of the feature are found in chapter 5.6

Note
Dynamic Connection is only supported for CAN bus and standard addressing format(11-bit addressing).

Note
Dynamic Connection cannot be configured with ECU detection.

2.7 Timeout Supervision

Supervision of the P2$_{Client}$-timeout is supported to avoid blocking of the CddDrm after a diagnostic request if no response is received as expected. It is also taken into account, that the server may send a response pending (RCRRP) to extend the timeout with the P2*$_{Client}$ time.

Note
If a functional request of a Tester present message (0x3E) with SPRMIB bit set is sent, it will not wait for P2 timer to get elapsed for processing of another request.

2.8 Tester Present Message

In order to keep the diagnostic kernel in the target ECU active, a periodic keep alive message can be sent. The CddDrm provides the API CddDrm_SvcSend_3E() for the application to trigger transmission of such a tester present request.

Note
CddDrm is able to process functional tester present request during any active physical or functional request if there is an available channel that can be used.

Note
The application has to trigger the transmission of a tester present message whenever the S3$_{Client}$ time expires, i.e. the CddDrm will not send a tester present by itself.

2.9 Parallel Connections

In some cases, the CddDrm will have to maintain connections to multiple ECUs at the same time. Therefore, the CddDrm must be able to maintain these connections in parallel and independent of each other.

These parallel connections are called channels which are dynamically assigned during runtime by the CddDrm to a requested connection.

2.10 Diagnostic Requests within the Own ECU

Requests generated by the CddDrm will not only have to be transmitted to other ECUs, but will also have to be forwarded to the diagnostic module of the ECU where the CddDrm itself resides.

This can’t be achieved by provisions in the CddDrm itself, but by an appropriate PduR configuration. For more info of configurations see section 5.

2.11 Other Tester Active Mode

Usually the diagnostic server in an ECU only accepts one client at a time. It is therefore important that all CddDrm related diagnostic communication is stopped, if another tester (on-board or external) becomes active.

The CddDrm provides the interface CddDrm_ExternalTesterConnected() to trigger the transition into the “other tester active mode”. As long as this mode is active, the CddDrm does not accept any service request.

2.12 ECU Detection

In a model line, not all ECUs may be available in each model or will be installed at a later point in time. To avoid unnecessary requests and bus traffic, the CddDrm supports the detection of the actually available ECUs.

For this, connections to a superset of all possible ECUs have to be configured. A request is sent to each ECU and dependent on the response the following assumptions will be made:

ECU available

- positive response (valid or SID + 0x40 not fulfilled)
- negative response (valid or invalid length)
- any response but response buffer too small

ECU not available

- connection timeout (P2 or P2*)

ECU not scanned yet

- request transmission cancelled (by CddDrm or application)
- any request or response error
- detection interrupted due to an external tester was detected

The request to be used by the module is configurable, e.g. $10 01 (default diagnostic session).

2.13 Service ID Firewall

The firewall mechanism prevents that services which are harmless in most other ECUs, can be sent to a specific ECU where they may trigger potentially dangerous actions (e.g. activating end of life deployment of the airbag).

Note
A service id configured in the firewall is also blocked for the ECU detection.

2.14 Service Cancellation

To allow the application to cancel an ongoing service processing, the CddDrm provides the API CddDrm_CancelRequest() to cancel an ongoing request or the respective response.

Note
When issuing a Cancel Request to a functional connection id CddDrm will cancel all active channels as long as it is not cancelling a functional 3E request with SPRMIB, in which case it will only cancel the functional connection id.

2.15 Limitations

- ECU detection and functional request processing cannot run in parallel.
- During functional request processing, no other request processing is possible.
- During physical request processing, no functional request processing is possible. Except of functional requests for service 0x3E with SPRMIB==TRUE, which is possible during active physical request processing, in which case no responses (not positive nor negative) for that request are expected or handled by CddDrm.
- Dynamic Connection cannot be configured with ECU detection.
- Dynamic Connection is only supported for CAN bus and standard addressing format(11-bit addressing).

2.16 Error Handling

2.16.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `CDDDRM_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported CddDrm ID is 255.

The reported service IDs identify the services which are described in 0. The following table presents the service IDs and the related services:

Service ID	Service
0x00	CddDrm_Init
0x02	CddDrm_GetVersionInfo
0x03	CddDrm_InitMemory
0x04	CddDrm_MainFunction
0x10	CddDrm_Transmit
0x11	CddDrm_CancelRequest
0x12	CddDrm_ExternalTesterConnected
0x13	CddDrm_StartEcuDetection
0x14	CddDrm_StopEcuDetection
0x15	CddDrm_GetEcuDetectionResult
0x16	CddDrm_NvM_InitEcuDetectionData
0x17	CddDrm_GetFuncRequestResult
0x18	CddDrm_StartOfReception
0x19	CddDrm_CopyRxData
0x1A	CddDrm_TpRxIndication
0x1B	CddDrm_CopyTxData
0x1C	CddDrm_TpTxConfirmation
0x1D	CddDrm_CanTpRxMetaData
0x1E	CddDrm_CanTpTxMetaData
0x20	CddDrm_SvcSend
0x21	CddDrm_SvcSend_10
0x22	CddDrm_SvcSend_11
0x23	CddDrm_SvcSend_1902
0x24	CddDrm_SvcSend_1904
0x25	CddDrm_SvcSend_22
0x26	CddDrm_SvcSend_27
0x27	CddDrm_SvcSend_28
0x28	CddDrm_SvcSend_31
0x29	CddDrm_SvcSend_34
0x2A	CddDrm_SvcSend_36
0x2B	CddDrm_SvcSend_37
0x2C	CddDrm_SvcSend_3E
0x2D	CddDrm_SvcSend_85
0x40	CddDrm_CancelTransmit
0x42	CddDrm_CancelReceive

Table 2-2 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x03	CDDDRM_ E_PARAM_POINTER	Service was called with a NULL pointer argument
0x04	CDDDRM_ E_PARAM_VALUE	Service used with invalid parameter value
0x05	CDDDRM_ E_UNINIT	Service was called before the CddDrm module has been initialized
0x06	CDDDRM_ E_ALREADY_INITIALIZED	Service was called after the CddDrm module has already been initialized
0x07	CDDDRM_ E_INVALID_CONNECTION	Service was called with an invalid connection id
0x08	CDDDRM_ E_INVALID_STATE	Service used in invalid state
0x0A	CDDDRM_ E_INVALID_BUFFER_LENGTH	Service was called with too small buffer size
0x0B	CDDDRM_ E_PDU_ID_RX_OUT_OF_RANGE	Service was called with invalid Rx Pdu-Id
0x0C	CDDDRM_ E_PDU_ID_TX_OUT_OF_RANGE	Service was called with invalid Tx Pdu-Id
0x0D	CDDDRM_ E_API_ERROR	Unexpected call of API

Table 2-3 Errors reported to DET

2.16.2 Production Code Error Reporting

The CddDrm does not report any production errors to the Dem.

3 Integration

This chapter gives necessary information for the integration of the MICROSAR CddDrm into an application environment of an ECU.

3.1 Scope of Delivery

The delivery of the CddDrm contains the files which are described in the chapters 3.1.1 and 3.1.2:

3.1.1 Static Files

File Name	Description
CddDrm.c	This is the source file of the CddDrm. It contains the main functionality of the CddDrm.
CddDrm.h	This header file provides the CddDrm API functions for the application. This file is supposed to be included by client modules.
CddDrm_LowerLayer.c	This is the source file for the LL of CddDrm. It contains the main functionality of the LL of the CddDrm.
CddDrm_Types.h	This header file contains all CddDrm data types. It is recommended to not include this file directly, but include CddDrm.h instead.
CddDrm_Cbk.h	This header file contains callback functions. Include this in the NvM configuration for the declarations of the initialization and notification functions.

Table 3-1 Static files

3.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator Pro.

File Name	Description
CddDrm_Cfg.c	This source file contains configuration values and tables of the CddDrm.
CddDrm_Cfg.h	This header file contains the configuration switches and provides access functions to the configuration values and tables for the CddDrm.

Table 3-2 Generated files

3.2 Include Structure

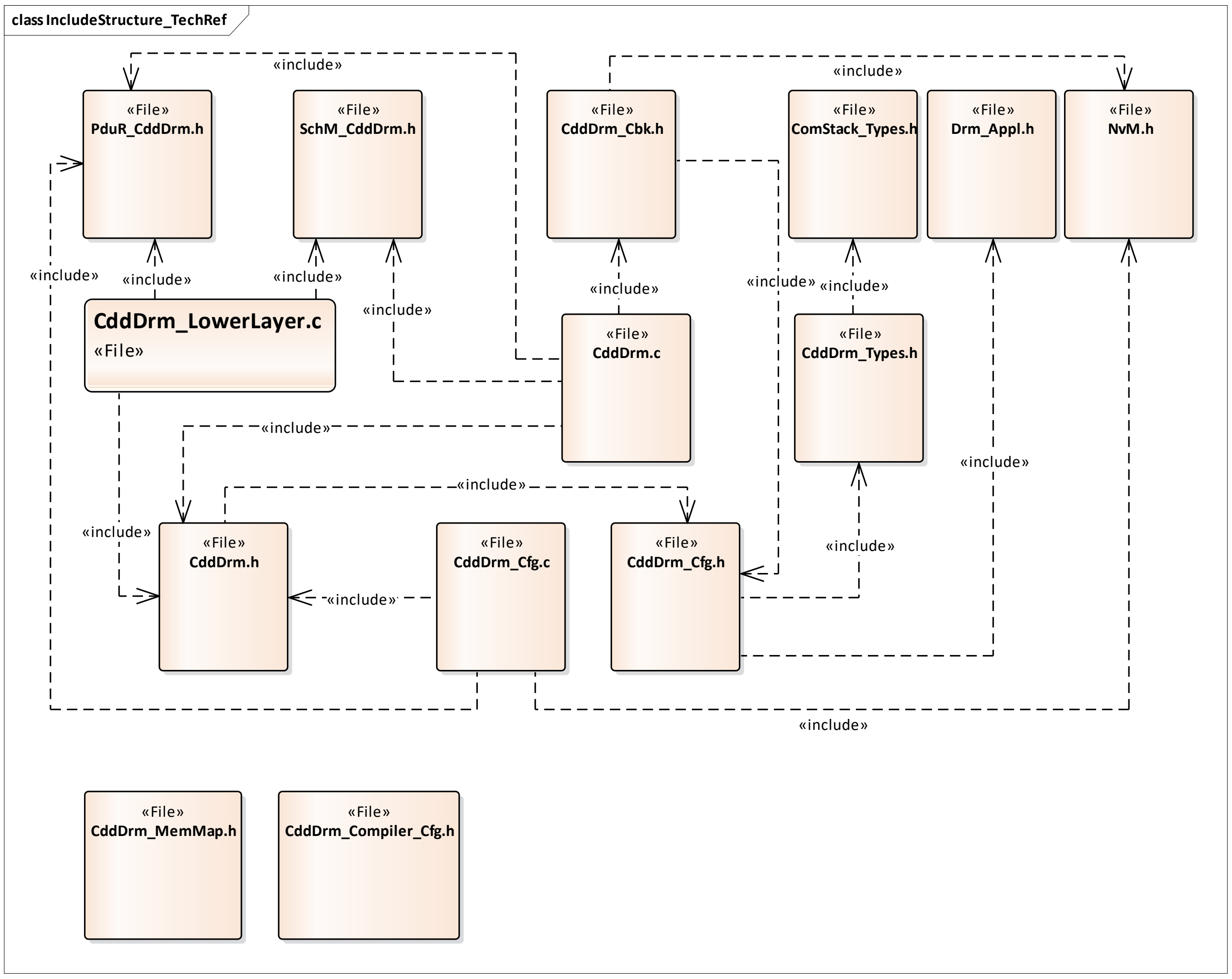

Figure 3-1 Include Structure

3.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions of the CddDrm and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	CDDDRM_CODE	CDDDRM_CONST
CDDDRM_START_SEC_CODE CDDDRM_STOP_SEC_CODE	■	

Memory Mapping Sections \ Compiler Abstraction Definitions	CDDDRM_CODE	CDDDRM_CONST
CDDDRM_START_SEC_CONST_<size> CDDDRM_STOP_SEC_CONST_<size>		■

Table 3-3 Compiler abstraction and memory mapping, constant sections

Memory Mapping Sections \ Compiler Abstraction Definitions	CDDDRM_VAR_INIT	CDDDRM_VAR_NOINIT	CDDDRM_NVM_DATA_NOINIT
CDDDRM_START_SEC_VAR_INIT_<size> CDDDRM_STOP_SEC_VAR_INIT_<size>	■		
CDDDRM_START_SEC_VAR_NOINIT_<size> CDDDRM_STOP_SEC_VAR_NOINIT_<size>		■	
CDDDRM_START_SEC_VAR_ZERO_INIT_<size> CDDDRM_STOP_SEC_VAR_ZERO_INIT_<size>	■		
CDDDRM_START_SEC_VAR_SAVED_ZONE0_<size> CDDDRM_STOP_SEC_VAR_SAVED_ZONE0_<size>			■

Table 3-4 Compiler abstraction and memory mapping, variable sections

3.4 Critical Sections

The CddDrm uses the Critical Section implementation of the SchM.

3.4.1 Exclusive Area 0

Channel Manager
Purpose: Ensures data consistency of the channel allocation in case of concurrent execution API calls and main function task. **Interfaces:** > `SchM_Enter_CddDrm_CDDDRM_EXCLUSIVE_AREA_0`

Channel Manager
> `SchM_Exit_CddDrm_CDDDRM_EXCLUSIVE_AREA_0` **Runtime:** Medium runtime; The runtime will increase the more channels are configured. **Dependency:** > `CddDrm_MainFunction()` > `CddDrm_SvcSend*()` > `CddDrm_CancelRequest()` > `CddDrm_CopyTxData()` > `CddDrm_TpTxConfirmation()` > `CddDrm_StartOfReception()` > `CddDrm_CopyRxData()` > `CddDrm_TpRxIndication()` **Recommendation:** No recommendation.

Table 3-5 Exclusive Area 0

3.5 NVM Integration

In general, the CddDrm module is designed to work with an Autosar NvM to provide non-volatile data storage.

It is expected that all NVRAM blocks used by the CddDrm are configured with the parameters detailed in the following chapters:

> RAM buffer

> Initialization method: initialization function

> Single block job end notification

> Enabled ReadAll

When using a non-Autosar NVRAM manager, please also refer to the Autosar SWS of the NvM module for more details on the expected behavior.

3.5.1 NVRAM Demand

The non-volatile data blocks used by the CddDrm must be configured to match the size of the underlying type. Since the actual size depends on compiler settings and platform properties, this size cannot be calculated by the configuration tool.

To find the correct data structure sizes, you can use temporary code to perform a ‘sizeof’ operation on the data types involved, or check your linker map file if it contains this kind of data.

The MICROSAR NvM implementation supports a feature to verify the correct configuration of block sizes. It is strongly recommended to enable this feature; it also provides a very easy way to find out the correct block sizes.

Table 3-6 lists the types used by the different data elements.

NVRAM Item	RAM	Type	Comment
ECU Detection Data	CddDrm_Cfg_EcuDetectionData	CddDrm_Cfg_EcuDetectionDataType	only if ECU detection is enabled

Table 3-6 NVRAM Blocks

3.5.2 NVRAM Initialization

The NvM provides a means to initialize RAM buffers, if the backing storage cannot restore a preserved copy – e.g. because none has ever been stored yet.

For this, the CddDrm provides initialization functions. The Init functions are declared in CddDrm.h.

NVRAM Item	Initialization
ECU Detection Data	Call `CddDrm_NvM_InitEcuDetectionData()`

Table 3-7 NVRAM initialization

3.5.2.1 Controlled Re-initialization

Some use-cases require the total reset of all stored data. A simple way for that is to change the CddDrm configuration id (CddGeneral/CddCompiledConfigId) in the configuration tool.

This is especially useful during development, when a different software configuration is loaded while the NVRAM contents still remain from an older software version. Please be aware that changing the CddDrm configuration is likely to require resetting the NVRAM data.

If a different configuration id is detected during CddDrm_Init(), the CddDrm will completely re-initialize all data. This can be helpful if you do not want to use the similar feature provided by NvM.

3.5.2.2 Manual Re-Initialization

If you need to reset the CddDrm's data manually, you can do so by calling all NvM initialization callbacks (CddDrm_NvM_Init*, see chapter 4.4).

Please be aware that this will not cause the NvM to actually persist the changes into NVRAM. You also need to mark the corresponding NvBlockIds as changed – refer to your configuration to find out the correct handles.

Caution
Do not modify the CddDrm NV data blocks while the CddDrm is active. This will cause inconsistent data.

4 API Description

For an interfaces overview please see Figure 1-2.

4.1 Type Definitions

The types defined by the CddDrm are described in this chapter.

Type Name	C-Type	Description	Value Range
CddDrm_ ConfigPtrType	uint8	Pointer to the configuration that shall be used	-

CddDrm_ ConnectionIdType	uint8
	uint16
		Handle for the ECU that shall be diagnosed	`0 - 255`
			`0 - 65535`

CddDrm_ LengthType	PduLen gthType	Length of data record	`0 - 65535/4294967295`
CddDrm_ BufferLengthType	PduLen gthType	Length of buffer	`0 - 65535/4294967295`
CddDrm_ ResponseCodeType	uint8	Response code	`CDDDRM_RESPONSE_POSITIVE` Positive response
			`CDDDRM_RESPONSE_TIMEOUT` No response received within P2/P2* time
			`CDDDRM_RESPONSE_INVALID _NRC_LENGTH` NRC message is not equal 3 bytes
			`CDDDRM_RESPONSE_RCRRP_LIMIT _REACHED` Configured RCRRP limit reached
			`CDDDRM_RESPONSE_BUFFER_TOO _SMALL` Receive message does not fit into the given buffer
			`CDDDRM_RESPONSE_WRONG _SERVICE` Received service ID does not conform to SID + 0x40
			`CDDDRM_RESPONSE_CHANNELS _CLOSED` Request/response processing canceled from application
			`CDDDRM_RESPONSE_TESTER _DETECTED` External tester detected
			`CDDDRM_RESPONSE_PDUR_RX _ERROR` Receive of data from PduR has failed

Type Name	C-Type	Description	Value Range
			CDDDRM_RESPONSE_PDUR_TX _ERROR Data transmission to PduR has failed
			CDDDRM_RESPONSE_FORWARDED_RC RRP RCRRP message forwarded to application.
CddDrm_ RequestReturnType	uint8	The result / status of the job	CDDDRM_REQUEST_OK Request accepted
			CDDDRM_REQUEST_NO_CHANNEL Request not accepted due to no channel could be allocated
			CDDDRM_REQUEST_TESTER _ACTIVE Request not accepted due to an external tester is currently active
			CDDDRM_REQUEST_FIREWALL _BLOCKED Request not accepted due to service is blocked by firewall
			CDDDRM_REQUEST_CONNECTION _BUSY Request not accepted due to given connection is currently in use
			CDDDRM_REQUEST_FUNCTIONAL _ACTIVE Request not accepted due to an active functional request
			CDDDRM_REQUEST_ECUD_ACTIVE Request not accepted due to an active ECU detection
CddDrm_ EcudStateType	uint8	Connection specific ECU detection state	CDDDRM_ECUD_CONNECTION_NOT _DISOVERED ECU connection not discovered yet
			CDDDRM_ECUD_CONNECTION_NOT _AVAILABLE ECU connection not available
			CDDDRM_ECUD_CONNECTION _AVAILABLE ECU connection available
CddDrm_FuncReqResultStateType	uint8	Connection specific result state of a functional request	CDDDRM_FUNC_REQ_RESULT_OK Result provided
			CDDDRM_FUNC_REQ_RESULT_NOT _OK Operation failed

Type Name	C-Type	Description	Value Range
			CDDDRM_FUNC_REQ_RESULT _BUFFER_TOO_SMALL Provided buffer too small
			CDDDRM_FUNC_REQ_RESULT _PENDING Functional request in progress

Table 4-1 Type definitions

CddDrm_BufferStructType

Struct Element Name	C-Type	Description	Value Range
requestBufferDataPtr	uint8*	pointer to the request buffer	–
responseBufferSize	PduLengthType	Size of the response buffer	0 – 65535/4294967295
responseBufferDataPtr	uint8*	pointer to the response buffer	–
requestMetaDataBufferPtr	uint8*	pointer to the request metadata buffer for dynamic connection (only present when dynamic connection is supported)	–
responseMetaDataBufferPtr	uint8*	pointer to the response metadata buffer for dynamic connection (only present when dynamic connection is supported)	–

Table 4-2 CddDrm_BufferStructType

CddDrm_RespInfoStructType

Struct Element Name	C-Type	Description	Value Range
responseLength	PduLengthType	length of the received data	0 – 65535/4294967295
responseCode	CddDrm_ ResponseCodeType	Result of the response	Refer to definition of CddDrm_ ResponseCodeType (see *Table 4-1*)
connectionId	uint8 uint16	Handle for the ECU that was diagnosed	0 – 255 0 – 65535
serviceId	uint8	serviceId of the request	0 – 255

Table 4-3 CddDrm_RespInfoStructType

CddDrm_FuncReqRespInfoStructType

Struct Element Name	C-Type	Description	Value Range
responseBufferDataPtr	uint8*	Pointer to response buffer	–
responseBufferSize	PduLengthType	IN: Response buffer size. OUT: Number of bytes copied in response buffer	`0 – 65535/4294967295`
responseCode	CddDrm_ ResponseCodeType	Result of the response	Refer to definition of CddDrm_ ResponseCodeType (see *Table 4-1*)

Table 4-4 CddDrm_FuncReqRespInfoStructType

4.2 Services provided by CddDrm

4.2.1 CddDrm_GetVersionInfo()

Prototype	
`void` **`CddDrm_GetVersionInfo`** `( Std_VersionInfoType *versionInfo )`	
Parameter	
versionInfo	Pointer to where to store the version information. Parameter must not be NULL.
Return code	
void	none
Functional Description	
Returns the version information.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function can be called from any context.	

Table 4-5 CddDrm_GetVersionInfo()

4.2.2 CddDrm_MainFunction()

Prototype	
`void CddDrm_MainFunction ( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Handles all internal used timers and state machines.	
Particularities and Limitations	
> This function is not reentrant. > The function is synchronous.	
Call Context	
> This function can be called from any context.	

Table 4-6 CddDrm_MainFunction()

4.2.3 Interface EcuM

4.2.3.1 CddDrm_Init()

Prototype	
`void CddDrm_Init ( const CddDrm_ConfigPtrType *configPtr )`	
Parameter	
configPtr	Configuration structure for initializing the module
Return code	
void	none
Functional Description	
Initialization function.	
Particularities and Limitations	
> This function is not reentrant. > The function is synchronous. > ConfigPtr is not used, so NULL_PTR can be set.	
Call Context	
> This function may not interrupt any other CddDrm function.	

Table 4-7 CddDrm_Init()

4.2.3.2 CddDrm_InitMemory()

Prototype	
`void` **`CddDrm_InitMemory`** `( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
– Extension to Autosar – Use this function to initialize static RAM variables in case the start-up code is not used to initialize RAM.	
Particularities and Limitations	
> The function is not reentrant. > The function is synchronous.	
Call Context	
> This function may not interrupt any other CddDrm function.	

Table 4-8 CddDrm_InitMemory()

4.2.4 Interface Application

4.2.4.1 CddDrm_SvcSend()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend`** `( CddDrm_ConnectionIdType connectionId, boolean sprmib, CddDrm_BufferStructType *bufferInfo, CddDrm_LengthType requestLength )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
sprmib	provides the info to CddDrm if suppress positive response message indication bit is set and thus CddDrm does not need to receive a positive response within P2 time.
bufferInfo	contains information about request/response buffer and metadata buffer
requestLength	request data length (excluding the metadata length)
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Generic interface to send a diagnostic request to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-9 CddDrm_SvcSend()

4.2.4.2 CddDrm_SvcSend_10()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_10`** `( CddDrm_ConnectionIdType connectionId, uint8 sessionType, boolean sprmib, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
sessionType	diagnostic session that shall be requested
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $10 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-10 CddDrm_SvcSend_10()

4.2.4.3 CddDrm_SvcSend_11()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_11`** `( CddDrm_ConnectionIdType connectionId, uint8 resetType, boolean sprmib, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
resetType	ECU reset type
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $11 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-11 CddDrm_SvcSend_11()

4.2.4.4 CddDrm_SvcSend_1902()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_1902`** `( CddDrm_ConnectionIdType connectionId, uint8 dtcStatusMask, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
dtcStatusMask	DTC status bit mask
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $1902 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-12 CddDrm_SvcSend_1902()

4.2.4.5 CddDrm_SvcSend_1904()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_1904`** `( CddDrm_ConnectionIdType connectionId, uint32 dtc, uint8 recordNumber, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
dtc	The DTC that shall be requested
recordNumber	DTC record number
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $1904 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-13 CddDrm_SvcSend_1904()

4.2.4.6 CddDrm_SvcSend_22()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_22`** `( CddDrm_ConnectionIdType connectionId, uint16 DID, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
DID	the data identifier that shall be requested
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $22 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-14 CddDrm_SvcSend_22()

4.2.4.7 CddDrm_SvcSend_27()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_27`** `( CddDrm_ConnectionIdType connectionId, uint8 subFunction, CddDrm_LengthType dataLength, boolean sprmib, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
subFunction	select request seed / send key
dataLength	length of the given seed/key
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $27 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-15 CddDrm_SvcSend_27()

4.2.4.8 CddDrm_SvcSend_28()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_28`** `( CddDrm_ConnectionIdType connectionId, uint8 subFunction, uint8 communicationType, uint16 nodeIdNumber, boolean sprmib, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
subFunction	control type
communicationType	is bit coded to control multiple communication types
nodeIdNumber	node identification number (only required if sub-function 0x04 or 0x05)
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $28 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-16 CddDrm_SvcSend_28()

4.2.4.9 CddDrm_SvcSend_31()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_31`** `( CddDrm_ConnectionIdType connectionId, uint8 subFunction, uint16 routineId, CddDrm_LengthType routineOptionLength, boolean sprmib, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
subFunction	routine control sub-function
routineId	routine identifier
routineOptionLength	length of routine options
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $31 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-17 CddDrm_SvcSend_31()

4.2.4.10 CddDrm_SvcSend_34()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_34`** `( CddDrm_ConnectionIdType connectionId, uint8 dataFormatId, uint8 addressAndLength, CddDrm_LengthType dataLength, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
dataFormatId	data format identifier
addressAndLength	memory address and memory size
dataLength	request data length of memory address and memory size
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $34 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-18 CddDrm_SvcSend_34()

4.2.4.11 CddDrm_SvcSend_36()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_36`** `( CddDrm_ConnectionIdType connectionId, uint8 blockSeqCounter, CddDrm_LengthType transferDataLength, const CddDrm_BufferStructType *bufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
blockSeqCounter	block sequence number
transferDataLength	length of the data that will be transferred
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $36 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-19 CddDrm_SvcSend_36()

4.2.4.12 CddDrm_SvcSend_37()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_37`** `( CddDrm_ConnectionIdType ConnectionId, CddDrm_LengthType TransferDataLength, const CddDrm_BufferStructType *BufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
transferDataLength	length of the data that will be transferred
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $37 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-20 CddDrm_SvcSend_37()

4.2.4.13 CddDrm_SvcSend_3E()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_3E`** `( CddDrm_ConnectionIdType ConnectionId, boolean SPRMIB, const CddDrm_BufferStructType *BufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $3E to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-21 CddDrm_SvcSend_3E()

4.2.4.14 CddDrm_SvcSend_85()

Prototype	
`CddDrm_RequestReturnType` **`CddDrm_SvcSend_85`** `( CddDrm_ConnectionIdType ConnectionId, uint8 SubFunction, CddDrm_LengthType RecordLength, boolean SPRMIB, const CddDrm_BufferStructType *BufferInfo )`	
Parameter	
connectionId	connection related to the ECU to which the request shall be send
subFunction	sub-function
recordLength	length of optional record data
sprmib	TRUE: set suppresses positive response message indication bit FALSE: do not set suppresses positive response message indication bit
bufferInfo	contains information about request/response buffer and metadata buffer
Return code	
CddDrm_RequestReturnType	CDDDRM_REQUEST_OK: Service request accepted CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, CddDrm is in external tester active mode CDDDRM_REQUEST_FIREWALL_BLOCKED: Service request not accepted, service is not allowed for the connection
Functional Description	
Service specific interface to send a diagnostic request $85 to the given connection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-22 CddDrm_SvcSend_85()

4.2.4.15 CddDrm_CancelRequest()

Prototype	
`Std_ReturnType` **`CddDrm_CancelRequest`** `( CddDrm_ConnectionIdType connectionId )`	
Parameter	
connectionId	Connection that shall to be closed.
Return code	
Std_ReturnType	E_OK: cancel request accepted E_NOT_OK: cancel request not accepted
Functional Description	
Cancels the communication with the particular connection	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-23 CddDrm_CancelRequest()

4.2.4.16 CddDrm_StartEcuDetection()

Prototype	
`Std_ReturnType` **`CddDrm_StartEcuDetection`** `( void )`	
Parameter	
void	none
Return code	
Std_ReturnType	E_OK: ECU Detection has been started E_NOT_OK: ECU Detection is already running or external tester active
Functional Description	
Starts the background ECU detection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-24 CddDrm_StartEcuDetection()

4.2.4.17 CddDrm_StopEcuDetection()

Prototype	
`Std_ReturnType` **`CddDrm_StopEcuDetection`** `( void )`	
Parameter	
void	none
Return code	
Std_ReturnType	E_OK: is always returned
Functional Description	
Stops an ongoing ECU detection.	
Particularities and Limitations	
> The function is synchronous. > This function is reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-25 CddDrm_StopEcuDetection()

4.2.4.18 CddDrm_GetEcuDetectionResult()

Prototype	
`CddDrm_EcudStateType` **`CddDrm_GetEcuDetectionResult`** `( CddDrm_ConnectionIdType connectionId )`	
Parameter	
connectionId	The connection id for which the detection result shall be fetched.
Return code	
CddDrm_EcudStateType	CDDDRM_ECUD_CONNECTION_AVAILABLE: The ECU is available CDDDRM_ECUD_CONNECTION_NOT_AVAILABLE: The ECU is not available CDDDRM_ECUD_CONNECTION_NOT_DISCOVERED: The ECU Detection has not been processed yet on the requested Connection
Functional Description	
Returns the ECU detection result of a connection.	
Particularities and Limitations	
> The function is synchronous. > This function is not reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-26 CddDrm_GetEcuDetectionResult()

4.2.4.19 CddDrm_GetFuncRequestResult()

Prototype	
`CddDrm_FuncReqResultStateType` **`CddDrm_GetFuncRequestResult`** `( CddDrm_ConnectionIdType connectionId, CddDrm_FuncReqRespInfoStructType *respInfoStruct )`	
Parameter	
connectionId	The connection id for which the functional request result shall be fetched.
respInfoStruct	IN: Specifies the pointer to the buffer and the available size. OUT: Returns the response code and the size of the provided response message.
Return code	
CddDrm_ FuncReqResultStateType	CDDDRM_FUNC_REQ_RESULT_OK: The request result has been provided. CDDDRM_FUNC_REQ_RESULT_NOT_OK: The operation failed. CDDDRM_FUNC_REQ_RESULT_BUFFER_TOO_SMALL: The provided buffer size is too small for the request result message.
Functional Description	
Returns the connection specific result to a functional request.	
Particularities and Limitations	
> The function is asynchronous. > This function is not reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-27 CddDrm_GetFuncRequestResult()

4.2.5 Interface PduR

4.2.5.1 CddDrm_Transmit()

Prototype	
`Std_ReturnType **CddDrm_Transmit** ( PduIdType txPduId, const PduInfoType *pduInfoPtr )`	
Parameter	
txPduId	id of the CddPduRLowerLayerTxPdu.
pduInfoPtr	a PduInfoType pointing to the transmit buffer.
Return code	
Std_ReturnType	E_OK: the transmission request has been accepted. E_NOT_OK: the transmission request has NOT been accepted.
Functional Description	
Initiate a request for transmission of a TX I-PDU.	
Particularities and Limitations	
> The function is reentrant. > The function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-28 CddDrm_Transmit()

4.2.5.2 CddDrm_CancelTransmit()

Prototype	
`Std_ReturnType **CddDrm_CancelTransmit** ( PduIdType txPduId)`	
Parameter	
txPduId	id of the CddPduRLowerLayerTxPdu.
Return code	
Std_ReturnType	E_OK: the transmission request was canceled. E_NOT_OK: the transmission was not canceled.
Functional Description	
Request cancellation of an ongoing transmission.	
Particularities and Limitations	
> The function is reentrant. > The function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-29 CddDrm_CancelTransmit()

4.2.5.3 CddDrm_CancelReceive()

Prototype	
`Std_ReturnType` **`CddDrm_CancelReceive`** `( PduIdType rxPduId)`	
Parameter	
rxPduId	id of the CddPduRLowerLayerRxPdu.
Return code	
Std_ReturnType	E_OK: the reception request was canceled. E_NOT_OK: the reception was not canceled.
Functional Description	
Request cancellation of an ongoing reception.	
Particularities and Limitations	
> The function is reentrant. > The function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-30 CddDrm_CancelReceive()

4.2.6 Interface CanTp

4.2.6.1 CddDrm_CanTpRxMetaData()

Prototype	
`Std_ReturnType CddDrm_CanTpRxMetaData(const PduInfoType* NPduPayLoad, const PduInfoType* NPduMetaData, PduInfoType* MetaDataOfNSduOutPtr, uint8* MetaDataOfFcNPduOutPtr, PduInfoType* PayloadOfFcNPduInOutPtr)`	
Parameter	
NPduPayLoad	Payload which CanTp has retrieved from CanIf. Note: Not used by CddDrm
NPduMetaData	MetaData which CanTp has retrieved from CanIf.
MetaDataOfNSduOutPtr	MetaData which shall be forwarded to the N-Sdu. The SduLength of this struct has the configured MetaDataLength value from the N-SDU.
MetaDataOfFcNPduOutPtr	MetaData used for sending the FC-frame. The same metadata length for incoming NPdu and outgoing FC-NPdu.
PayloadOfFcNPduInOutPtr	Contains the max length of Payload in CanTp memory and buffer, used for sending the FC-frame. Note: Not used by CddDrm.
Return code	
`Std_ReturnType`	E_OK : CanTp will continue with reception/transmission of TP-data in this transfer. E_NOT_OK: There was a problem in parsing address-information. The transfer will be stopped.
Functional Description	
This function is used by CanTp to interpret the MetaData and return the needed information to CanTp for further transmission/reception handling.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by CanTp.	

Table 4-40 CddDrm_CanTpRxMetaData()

4.2.6.2 CddDrm_CanTpTxMetaData()

Prototype	
`Std_ReturnType` **`CddDrm_CanTpTxMetaData`**`(const PduInfoType* MetaDataOfNSduPtr, PduInfoType * MetaDataOfDataNPduOutPtr, uint8* MetaDataOfFcNPduOutPtr, PduInfoType* PayloadOfDataNPduInOutPtr, uint8* PayloadOfFcNPduInOutPtr)`	
Parameter	
MetaDataOfNSduPtr	MetaData which CanTp has retrieved from UL.
MetaDataOfDataNPduOutPtr	MetaData which shall be forwarded to the N-Pdu and used for all data frames (SF, FF, CF). The SduLength of this struct has the configured MetaDataLength value from the N-PDU.
MetaDataOfFcNPduOutPtr	MetaData used for matching received FC-frame. The metadata length of FC-NPdu shall be identical to the length of NPdu.
PayloadOfDataNPduInOutPtr	Contains the max length of Payload in CanTp memory and buffer, used for all data frames (SF, FF, CF). Note: Not used by CddDrm.
PayloadOfFcNPduInOutPtr	Contains the max length of Payload in CanTp memory and buffer, used for matching received FC-frame. Note: Not used by CddDrm.
Return code	
`Std_ReturnType`	E_OK : CanTp will continue with reception/transmission of TP-data in this transfer. E_NOT_OK: There was a problem in parsing address-information. The transfer will be stopped.
Functional Description	
This function is used by CanTp to interpret the MetaData and return the needed information to CanTp for further transmission/reception handling.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by CanTp.	

Table 4-41 CddDrm_CanTpTxMetaData()

4.3 Services used by CddDrm

In the following table services provided by other components, which are used by the CddDrm are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
Det	optional Det_ReportErrorStatus
NvM	optional NvM_WriteBlock
PduR	PduR_CddDrmTransmit PduR_CddDrmCopyTxData PduR_CddDrmTxConfirmation PduR_CddDrmStartOfReception PduR_CddDrmCopyRxData PduR_CddDrmRxIndication Optional: PduR_CddDrmCancelTransmit PduR_CddDrmCancelReceive
SchM	optional SchM_Enter_CddDrm_<ExclusiveArea> optional SchM_Exit_CddDrm_<ExclusiveArea>

Table 4-31 Services used by the CddDrm

4.4 Callback Functions

This chapter describes the callback functions that are implemented by the CddDrm and can be invoked by other modules. The prototypes of the callback functions are provided in the header file CddDrm_Cbk.h by CddDrm.

4.4.1 CddDrm_NvM_JobFinished()

Prototype	
`Std_ReturnType` **`CddDrm_NvM_JobFinished`** `( uint8 serviceId, NvM_RequestResultType jobResult )`	
Parameter	
serviceId	The ServiceId indicates which one of the asynchronous services triggered via the operations of Interface NVM Service (Read/Write) the notification belongs to.
jobResult	Provides the result of the asynchronous job. NVM_REQ_OK: last asynchronous request has been finished successfully NVM_REQ_NOT_OK: not used in this context NVM_REQ_PENDING: not used in this context NVM_REQ_INTEGRITY_FAILED: not used in this context NVM_REQ_BLOCK_SKIPPED: not used in this context NVM_REQ_NV_INVALIDATED: not used in this context
Return code	
`Std_ReturnType`	E_OK: is always returned
Functional Description	
Is triggered from NVM to notify that the requested job which is processed asynchronous has been finished.	
Particularities and Limitations	
> This function is reentrant. > This function is asynchronous. > Must be configured for every CddDrm related NVRAM block	
Expected Caller Context	

> This function can be called from any context.

Table 4-32 CddDrm_NvM_JobFinished()

4.4.2 CddDrm_NvM_InitEcuDetectionData()

Prototype	
`Std_ReturnType` **`CddDrm_NvM_InitEcuDetectionData`** `( void )`	
Parameter	
void	none
Return code	
Std_ReturnType	E_OK: is always returned
Functional Description	
Initializes NVRAM block for ECU detection data. This function is supposed to be called by the NVM in order to (re)initialize the data in case the non-volatile memory has never been stored, or was corrupted (see NvMBlockDescriptor/NvMInitBlockCallback). This API is intended as callback function the NvM module. It will not mark the initialized block ‘changed’.	
Particularities and Limitations	
> The function is synchronous. > This function is not reentrant.	
Call Context	
> This function can be called from any context.	

Table 4-33 CddDrm_NvM_InitEcuDetectionData()

4.4.3 CddDrm_ExternalTesterConnected()

Prototype	
`Std_ReturnType` **`CddDrm_ExternalTesterConnected`** `( bool present )`	
Parameter	
present	TRUE: An external tester is currently present. FALSE: The external tester was disconnected.
Return code	
Std_ReturnType	E_OK: is always returned
Functional Description	
After the calling this API the CddDrm will cancel all pending requests and prevent that a new service request can be sent.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function can be called from any context.	

Table 4-34 CddDrm_ExternalTesterConnected()

4.4.4 CddDrm_StartOfReception()

Prototype	
`BufReq_ReturnType` **`CddDrm_StartOfReception`**`(PduIdType id, PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr)`	
Parameter	
id	Identification of the I-PDU.
info	Not used.
TpSduLength	Total length of the N-SDU to be received.
bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the BlockSize (BS) in the transport protocol module.
Return code	
`BufReq_ReturnType`	BUFREQ_OK: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. BUFREQ_E_OVFL: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Functional Description	
This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF).	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-35 CddDrm_StartOfReception()

4.4.5 CddDrm_CopyRxData()

Prototype	
`BufReq_ReturnType` **`CddDrm_CopyRxData`**`(PduIdType id, PduInfoType* info, PduLengthType* bufferSizePtr)`	
Parameter	
id	Identification of the received I-PDU.
info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the UL module. In this case, the SduDataPtr may be a NULL_PTR.
bufferSizePtr	Available receive buffer after data has been copied.
Return code	
`BufReq_ReturnType`	BUFREQ_OK: Data copied successfully. BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Functional Description	
This function is called to provide the received data of an I-PDU segment (N-PDU) to the UL. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-36 CddDrm_CopyRxData()

4.4.6 CddDrm_CopyTxData()

Prototype	
`BufReq_ReturnType` **`CddDrm_CopyTxData`**`(PduIdType id, PduInfoType* info, RetryInfoType* retry, PduLengthType* availableDataPtr)`	
Parameter	
id	Identification of the transmitted I-PDU.
info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the UL module. In this case, the SduDataPtr may be a NULL_PTR.
retry	CddDrm does not support retransmission of data. If the retry parameter is not a NULL_PTR and TpDataState is TP_DATARETRY, BUFREQ_E_NOT_OK is returned alongwih a DET Error, otherwise the parameter is ignored.
availableDataPtr	Indicates the remaining number of bytes that are available in the UL module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return code	
`BufReq_ReturnType`	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Functional Description	
This function is called to acquire the transmit data of an I-PDU segment (N-PDU).	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-37 CddDrm_CopyTxData()

4.4.7 CddDrm_TpRxIndication()

Prototype	
`void CddDrm_TpRxIndication(PduIdType id, Std_ReturnType result)`	
Parameter	
id	Identification of the received I-PDU.
result	Result of the reception.
Return code	
`Void`	None
Functional Description	
Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-38 CddDrm_TpRxIndication()

4.4.8 CddDrm_TpTxConfirmation()

Prototype	
`void CddDrm_TpTxConfirmation(PduIdType id, Std_ReturnType result)`	
Parameter	
id	Identification of the transmitted I-PDU.
result	Result of the transmission of the I-PDU.
Return code	
`Void`	None
Functional Description	
This function is called after the I-PDU has been transmitted via the TP API, the result indicates whether the transmission was successful or not.	
Particularities and Limitations	
> This function is reentrant. > This function is synchronous.	
Call Context	
> This function is called by PduR.	

Table 4-39 CddDrm_TpTxConfirmation()

4.5 Configurable Interfaces

4.5.1 Notifications

At its configurable interfaces the CddDrm defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the CddDrm but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

4.5.1.1 <ResponseNotification>()

Prototype	
`void <ResponseNotification> ( const CddDrm_RespInfoStructType *response )`	
Parameter	
response	Information about the response (response length, response code, service id and connection id)
Return code	
void	none
Functional Description	
This function notifies the application about the response, e.g. positive response with the corresponding length or negative response.	
Particularities and Limitations	
> This function shall be synchronous. > This function shall be reentrant.	
Call Context	
> This function is called from task context.	

Table 4-40 <ResponseNotification>()

4.5.1.2 <TxConfirmation>()

Prototype	
`void <TxConfirmation> ( CddDrm_ConnectionIdType connectionId )`	
Parameter	
connectionId	Connection to which the request has been transmitted.
Return code	
void	none
Functional Description	
This function notifies the application that the data has been sent on the bus or to local Dcm.	
Particularities and Limitations	
> This function shall be synchronous. > This function shall be reentrant.	
Call Context	
> This function is called from task context.	

Table 4-41 <TxConfirmation>()

4.5.1.3 <EcuDetectionFinished>()

Prototype	
`void <EcuDetectionFinished> ( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
This function notifies the application that the ECU Detection is finished.	
Particularities and Limitations	
> This function shall be synchronous. > This function shall be reentrant.	
Call Context	
> This function is called from task context.	

Table 4-42 <EcuDetectionFinished>()

4.5.1.4 <FirewallUserCallback>()

Prototype	
`boolean <`**`FirewallUserCallback>`**` ( CddDrm_ConnectionIdType connectionId, uint8 serviceId )`	
Parameter	
connectionId	Connection for which the firewall shall be checked from application.
serviceId	Service ID that shall be requested.
Return code	
boolean	TRUE: Service request allowed FALSE: Service request denied
Functional Description	
This function is a callout to the application to handle user defined services.	
Particularities and Limitations	
> This function shall be synchronous. > This function shall be reentrant.	
Call Context	
> This function is called from task context.	

Table 4-43 <FirewallUserCallback>()

5 Configuration

The CddDrm module is configured with the help of the configuration tool DaVinci Configurator Pro.

5.1 Configuration Variants

The CddDrm supports the configuration variants

> `VARIANT-PRE-COMPILE`

The configuration classes of the CddDrm parameters depend on the supported configuration variants. For their definitions please see the CddDrm_bswmd.arxml file.

5.2 Configurable Attributes

The description of each configurable option is described within the CddDrm_bswmd.arxml file. You can use the online help of DaVinci Configurator Pro to access these parameter descriptions comfortably.

5.3 Configuration to Diagnose ECUs Connected to the Network

To send diagnostic messages to other ECUs and to receive their responses the CddDrm, acts as an UL to PduR so CddDrm has to be configured accordingly, see Figure 5-1.

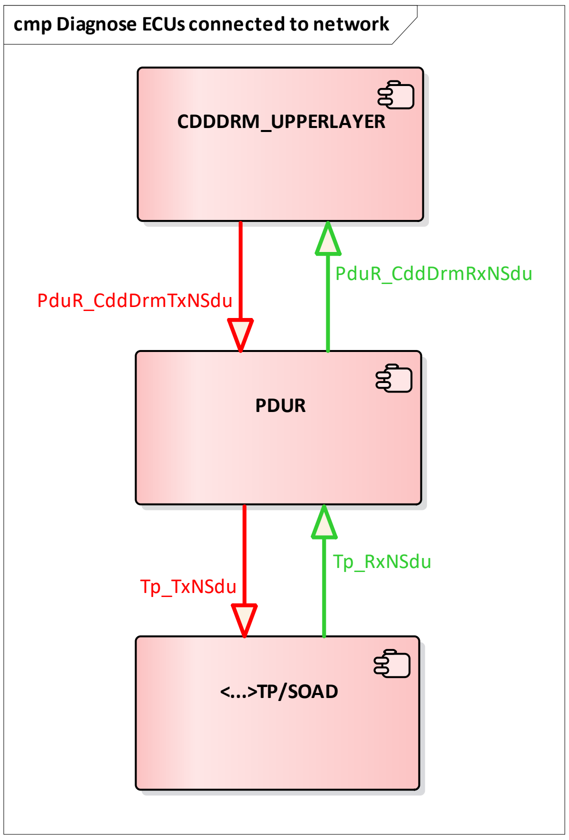

Figure 5-1 Communication of CddDrm as an UL Component

As an example, for configuration follow these steps:

1. Add CddPduRUpperLayerContribution under CddComStackContribution.
2. Add one CddPduRUpperLayerRxPdus and one CddPduRUpperLayerTxPdus per connection that is to be used.
3. Add the connection under CddConfig->CddConnections and add the references to the UL contributions.

5.4 Configuration to Diagnose the own ECU

To send diagnostic requests to the internal diagnostic server on the same ECU as CddDrm and to receive the corresponding responses, CddDrm has to be configured as a LL module as well as an UL module.

The request sent from CddDrm UL is transmitted to the LL through PduR which is then forwarded to DCM through PduR.

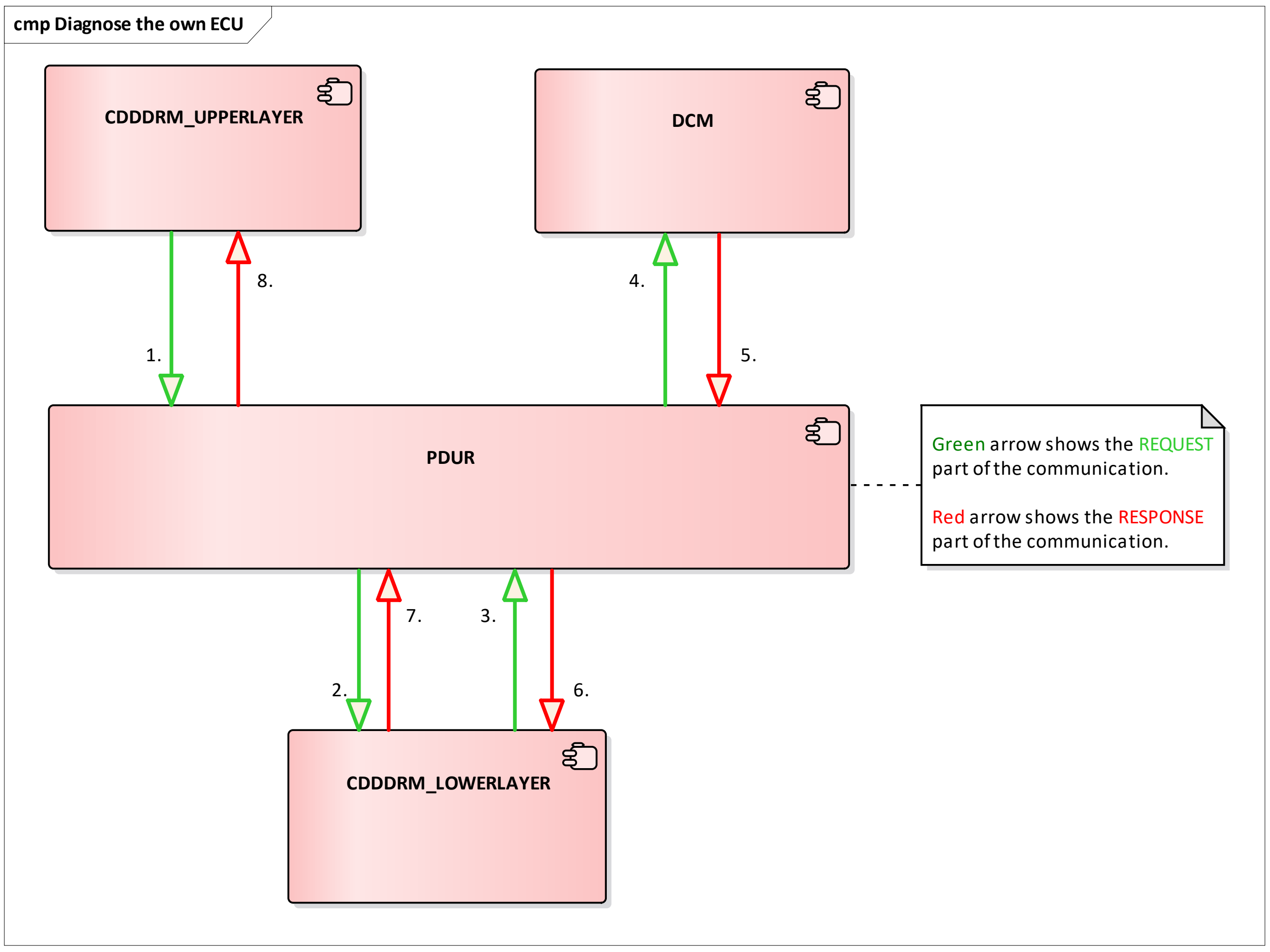
cmp Diagnose the own ECU
CDDDRM_UPPERLAYER
DCM
8.
4.
1.
5.
PDUR
Green arrow shows the REQUEST part of the communication.
Red arrow shows the RESPONSE part of the communication.
7.
3.
2.
6.
CDDDRM_LOWERLAYER

Figure 5-2 Communication of CddDrm as a LL Component

As an example, for configuration follow these steps:

1. Add CddPduRUpperLayerContribution and CddPduRLowerLayerContribution under CddComStackContribution.

2. In CddPduRUpperLayerContribution add one CddPduRUpperLayerTxPdus and one CddPduRUpperLayerTxPdus for the internal connection that is to be used.
3. In CddPduRLowerLayerContribution
    a. Under CddPduRLowerLayerContribution add one Rx Pdu per UL component that the LL is communicating with, i.e. one for CddDrm and one for Dcm.
    b. Under CddPduRLowerLayerContribution add one Tx Pdu per UL component that the LL is communicating with, i.e. one for CddDrm and one for Dcm.
4. Add the internal connection under CddConfig->CddConnections and add the references to the UL contributions. These references are used by the UL, hence only requiring the UL contributions.
5. Under CddInternalEcuRoutings create a CddInternalEcuRouting, then create a subcontainer for it. In the subcontainer add the references to the UL and LL contributions created in step 3.
    a. Request Rx Pdu ref refers to the Rx Pdu of Dcm.
    b. Request Tx Pdu ref refers to the Tx Pdu of UL CddDrm.
    c. Response Rx Pdu ref refers to the Rx Pdu of UL CddDrm.
    d. Response Tx Pdu ref refers to the Tx Pdu of Dcm.

Note
The DcmDslConnection connected to the CddDrm shall reference in its parameter DcmDslProtocolRxComMChannelRef a ComM channel of type COMM_BUS_TYPE_INTERNAL. Otherwise an external bus will be kept awake during diagnostic requests triggered CddDrm.

5.5 Configuration for External Tester Detection

The PduR can be configured to trigger the BswM for the reception of a specific TP message (e.g. on reception of the RxNPdu related to CANID 0x7DF). Hereby the BswM can be configured to call the API CddDrm_ExternalTesterConnected() to notify the CddDrm about a connected external tester.

5.6 Configuration for Dynamic Connection

5.6.1 CddDrm

The Dynamic connections are configured in CddDrm and can be configured for each physical connection. This is achieved by setting the Dynamic Connection parameter in the CddPhysical container. The connecting Tx/Rx Pdu of the connection must be configured with a Meta Data Length of two.

5.6.2 CanTp

The Tx/Rx N-SDUs of the dynamic connections in CddDrm shall be mapped to one single Tx/Rx N-PDU in CanTp which will be configured to a specific CAN network.

The CanTp Tx/Rx channels will have to be configured with CANTP_CUSTOM as CanTpRxAddressingFormat.

5.6.3 CanIf

For the response L-PDU in CanIf, either CanIfRxPduCanIdRange or CanIfRxPduCanIdMask will have to be configured to cover all response CAN IDs which are expected from the network. For the request L-PDU in CanIf, CanIfTxPduCanIdMask will have to be set to 0 for dynamic CAN ID. See CanIf documentation for more info about configuring dynamic CAN IDs.

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
$P2_{Client}$	Timeout for the client to wait after the successful transmission of a request message for the start of incoming response messages. The timer is started at the time of a call of CddDrm_TpTxConfirmation() with result E_OK and stopped with a call of CddDrm_StartOfReception().
$P2^{*}_{Client}$	Enhanced timeout for the client to wait after the reception of a negative response message with NRC 0x78 for the start of incoming response messages. Timer is extended with the call of CddDrm_TpRxIndication() and stopped with a call of CddDrm_StartOfReception().
$S3_{Client}$	Time between TesterPresent (0x3E) request messages transmitted by the client to keep a diagnostic session other than the defaultSession active in server.

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
BswM	Basic Software Mode Manager
BSWMD	Basic Software Module Description
Cdd	Complex Device Driver
Dcm	Diagnostic Communication Manager
Dem	Diagnostic Event Manager
Det	Development Error Tracer
DID	Data Identifier
Drm	Diagnostics Request Manager
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EcuM	ECU State Manager
LL	Lower Layer
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
NRC	Negative Response Code
NvM	NVRAM Manager
NVRAM	Non-Volatile Random-Access Memory
OBT	On-Board Tester
OEM	Original Equipment Manufacturer
PDU	Protocol Data Unit

PduR	PDU Router
RCRRP	Response Correctly Received, Response Pending
Rte	Runtime Environment
SchM	Schedule Manager
SID	Service Identifier
SoAd	Socket Adapter
SWC	Software Component
Tp	Transport Protocol
UL	Upper Layer
UDS	Unified Diagnostic Services
CAN ID	Can Identifier
N-PDU	Network Layer PDU. Used by transport protocol modules to fragment an I-PDU
N-SDU	In layered systems, a SDU refers to a set of data that is sent by a user of the services of a given layer, and is transmitted to a peer service user, whilst remaining semantically unchanged. In the AUTOSAR architecture, it is a set of data coming from the PDU Router.
I-PDU	Interaction Layer PDU. An I-PDU consists of data (buffer), length and I-PDU ID.

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

> News

> Products

> Demo software

> Support

> Training data

> Addresses

www.vector.com