# MICROSAR CddDrm

## Technical Reference

Version 1.0.0


| | |
|---|---|
| Authors | Alexander Ditte |
| Status | Released |

# Document Information

## History

| Author | Date | Version | Remarks |
|---|---|---|---|
| Alexander Ditte | 2016-01-21 | 1.0.0 | initial version |

## Reference Documents

| No. | Source | Title | Version |
|---|---|---|---|
| [1] | AUTOSAR | AUTOSAR_BasicSoftwareModules.pdf | V1.0.0 |
| [2] | AUTOSAR | AUTOSAR_SES_DET.pdf | V3.0.0 |

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 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

| Component Version | New Features |
|---|---|
| 1.00.00 | Initial Version |

Table 1-1 Component history

# 2 Introduction

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

| | | |
|---|---|---|
| **Supported AUTOSAR Release*:** | 4 | |
| **Supported Configuration Variants:** | pre-compile | |
| **Vendor ID:** | CddDrm_VENDOR_ID | 30 decimal<br>(= Vector-Informatik, according to HIS) |
| **Module ID:** | CddDrm_MODULE_ID | 255 decimal<br>(according to ref. [1]) |

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

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.

## 2.1 Architecture Overview

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

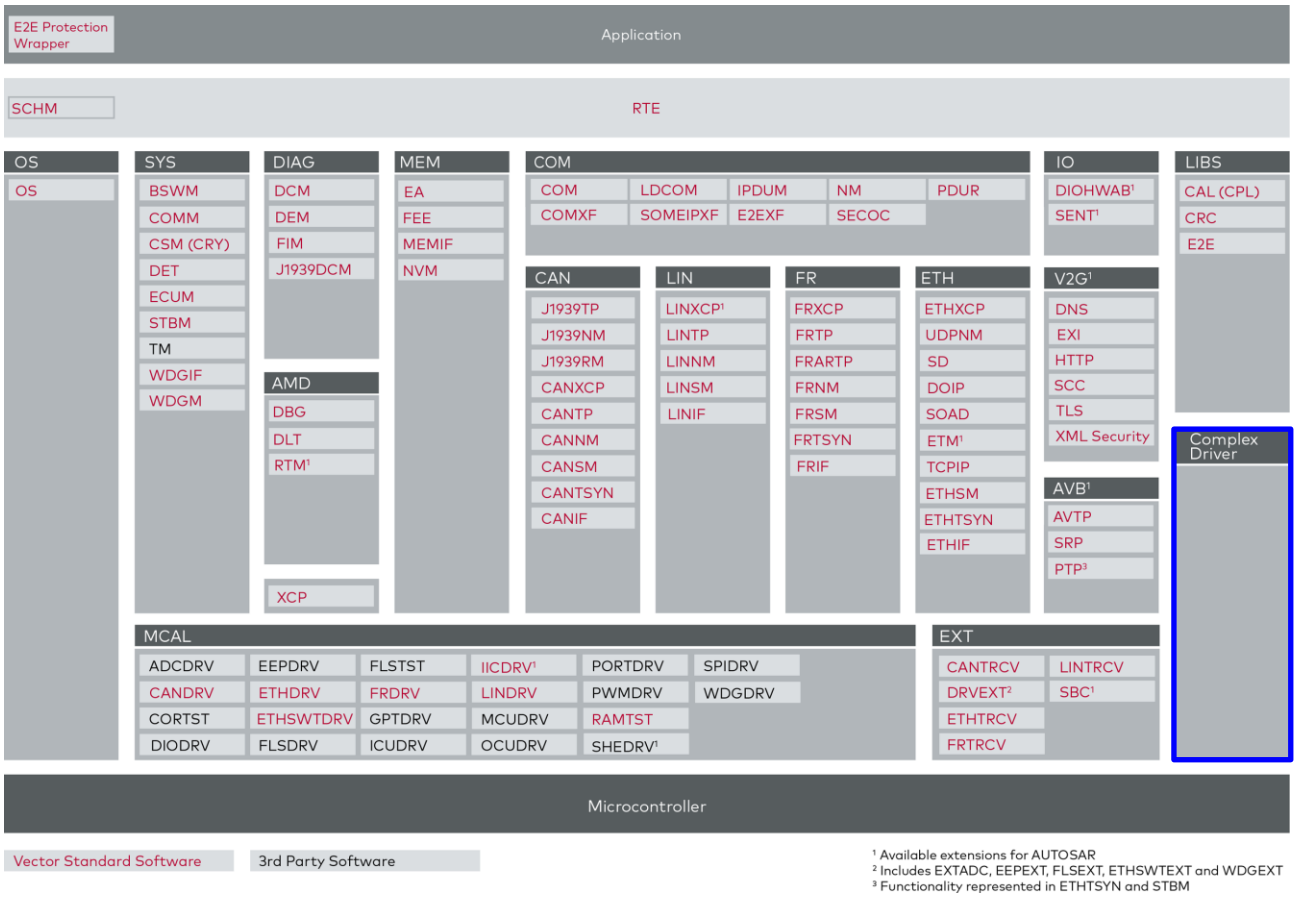


Figure 2-1 AUTOSAR 4.2 Architecture Overview

The next figure shows the interfaces to adjacent modules of the CddDrm. These interfaces are described in chapter 5.

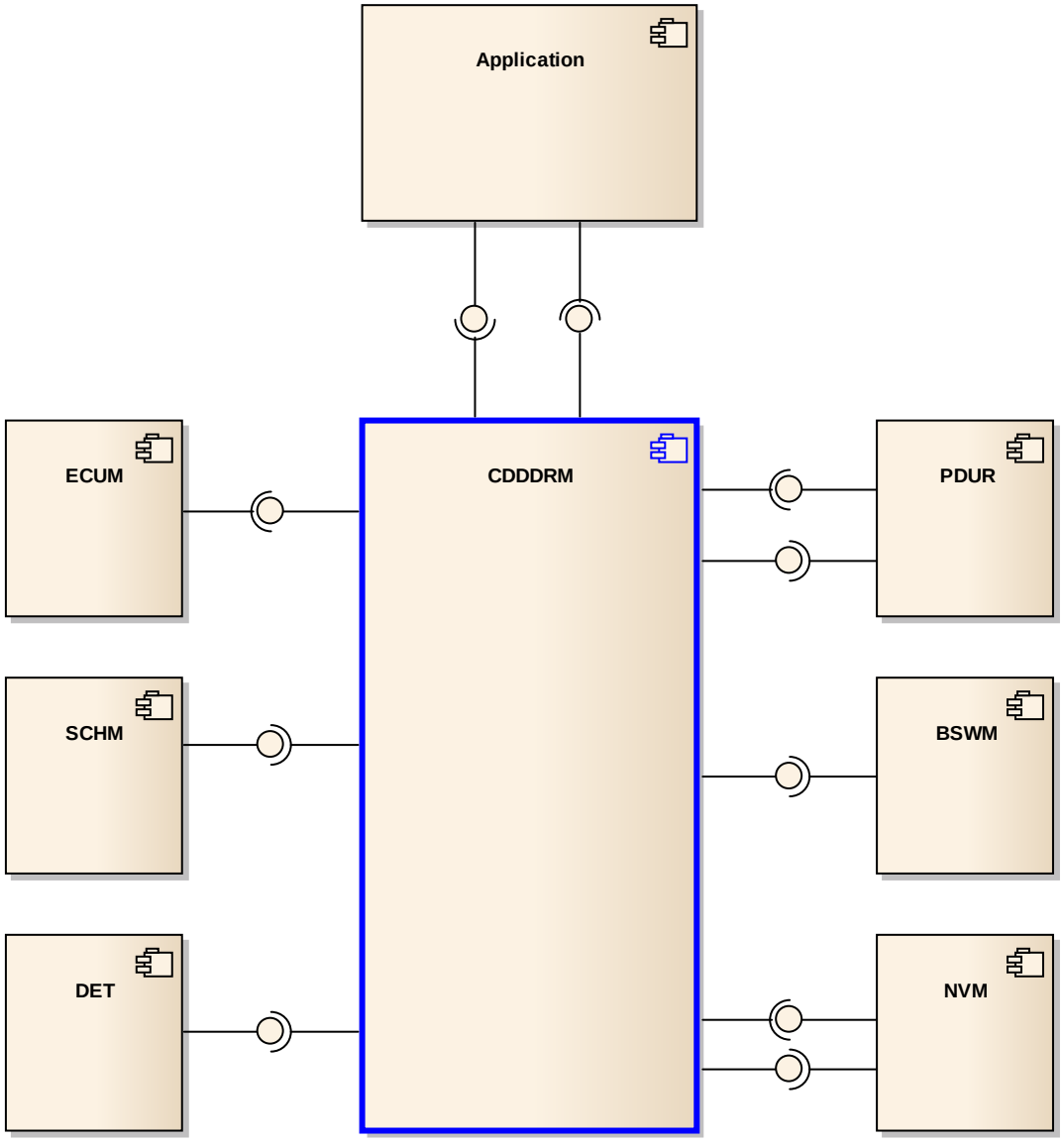


Figure 2-2 Interfaces to adjacent modules of the CddDrm

# 3 Functional Description

## 3.1 Features

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

> Table 3-1 Supported CddDrm features

> Table 3-2 Not supported CddDrm features

The following features are supported:

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

Table 3-1 Supported CddDrm features

The following features are not supported:

| Not Supported Features |
|---|
| Functional Request Transmission |

Table 3-2 Not supported CddDrm features

## 3.2 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.

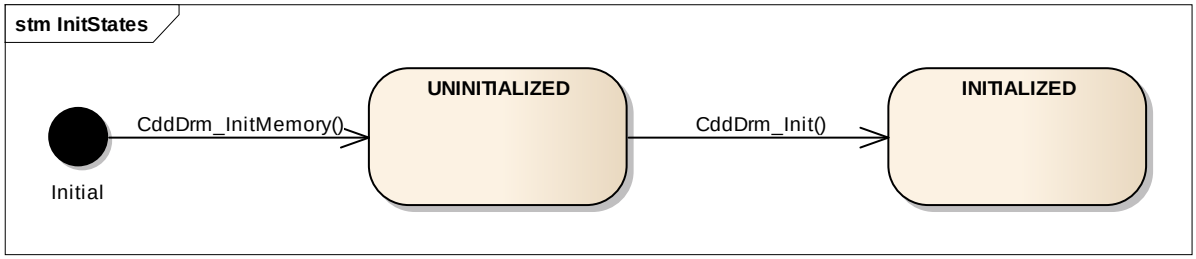


Figure 3-1 Initialization states of the CddDrm

## 3.3 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.

## 3.4 Request Transmission

One of the main tasks of the CddDrm is to send UDS requests to other ECUs. To trigger the transmission of a diagnostic request, the CddDrm provides service specific APIs to the application (refer to chapter 5.2.4).

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 5.5.1) 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.

If the SPRMIB bit is set, the DRM 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.

## 3.5 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 not notified immediately to the application but is handled internally. If the maximum number of accepted RCRRPs is reached, the application is notified with an NRC.

## 3.6 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 extended the timeout with the $P2^*_{Client}$ time.

## 3.7 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 with physical addressing.

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.

## 3.8 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.

## 3.9 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.

## 3.10 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.

## 3.11 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).

## 3.12 Service ID Firewall

The firewall mechanism prevent 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.

## 3.13 Service Cancelation

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.

## 3.14 Error Handling

### 3.14.1 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 5.2. 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 |
| 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 |
| 0x41 | CddDrm_ChangeParameter |
| 0x42 | CddDrm_CancelReceive |

Table 3-3 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 |

| Error Code | | Description |
|---|---|---|
| 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 |
| 0x0A | CDDDRM_ E_INVALID_BUFFER_LENGTH | Service was called with too small buffer size |
| 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 3-4 Errors reported to DET

### 3.14.2 Production Code Error Reporting

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

# 4 Integration

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

## 4.1 Scope of Delivery

The delivery of the CddDrm contains the files which are described in the chapters 4.1.1 and 4.1.2:

### 4.1.1 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_Types.h | This header file contains all CddDrm data types. Do not include this file directly, but include CddDrm.h instead. |
| CddDrm_Cbk.h | This header file contains callback functions intended for the NvM module. Include this in the NvM configuration for the declarations of the initialization and notification functions. |

Table 4-1 Static files

### 4.1.2 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 4-2 Generated files

## 4.2 Include Structure

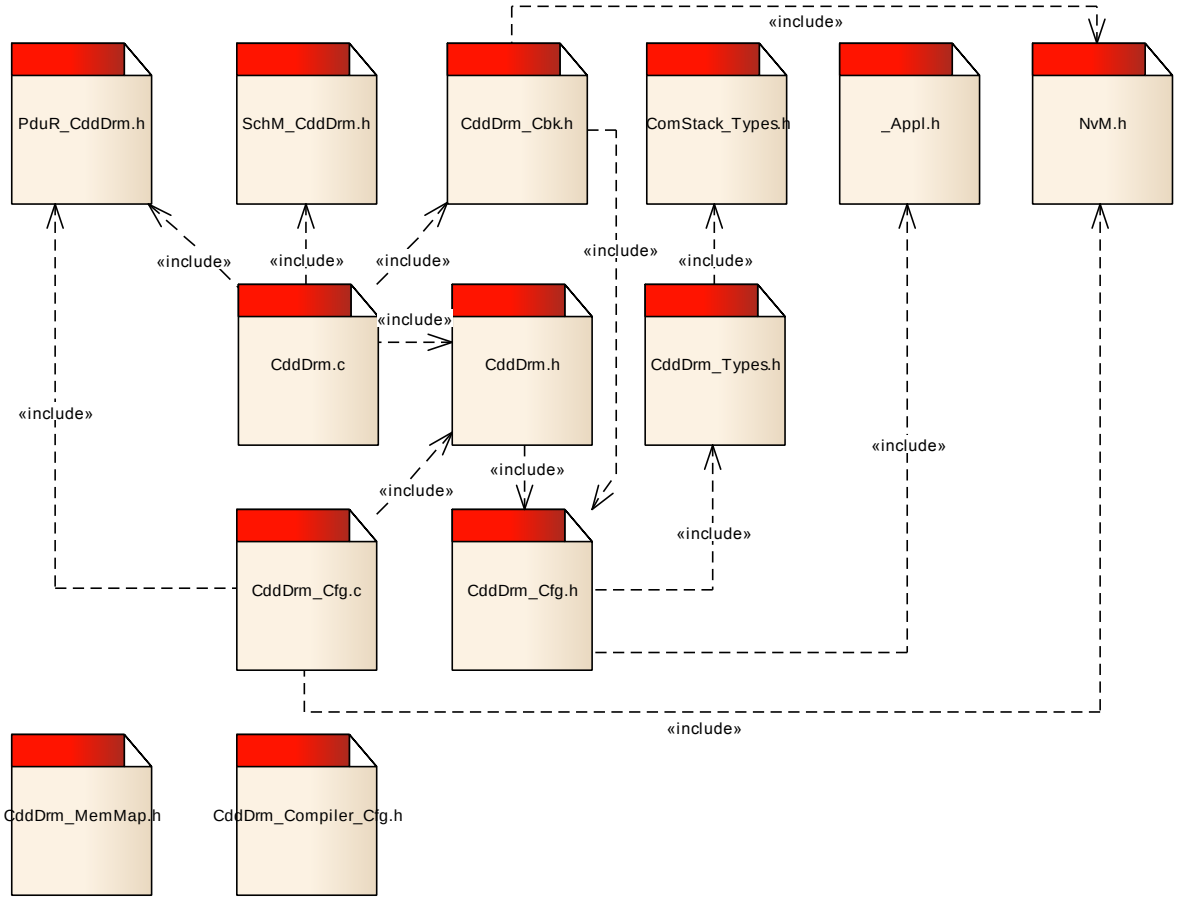


Figure 4-1 Include Structure

## 4.3 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.

| Memory Mapping Sections \ Compiler Abstraction Definitions | CDDDRM_CODE | CDDDRM_CONST |
|---|---|---|
| CDDDRM_START_SEC_CODE<br>CDDDRM_STOP_SEC_CODE | ■ | |
| CDDDRM_START_SEC_CONST_<size><br>CDDDRM_STOP_SEC_CONST_<size> | | ■ |

Table 4-3 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><br>CDDDRM_STOP_SEC_VAR_INIT_<size> | ■ | | |
| CDDDRM_START_SEC_VAR_NOINIT_<size><br>CDDDRM_STOP_SEC_VAR_NOINIT_<size> | | ■ | |
| CDDDRM_START_SEC_VAR_ZERO_INIT_<size><br>CDDDRM_STOP_SEC_VAR_ZERO_INIT_<size> | ■ | | |
| CDDDRM_START_SEC_VAR_SAVED_ZONE0_<size><br>CDDDRM_STOP_SEC_VAR_SAVED_ZONE0_<size> | | | ■ |

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

## 4.4 Critical Sections

The CddDrm uses the Critical Section implementation of the SchM.

### 4.4.1 Exclusive Area 0

**Channel Manager**

**Purpose:**
Ensures data consistency of the channel allocation in case of preempted execution of send service interfaces.

**Interfaces:**

- `SchM_Enter_CddDrm_CDDDRM_EXCLUSIVE_AREA_0`
- `SchM_Exit_CddDrm_CDDDRM_EXCLUSIVE_AREA_0`

**Runtime:**
Short runtime; The runtime will increase the more channels are configured.

**Dependency:**

- `CddDrm_SvcSend()`
- `CddDrm_SvcSend_<SID>()`
- `CddDrm_Transmit()`

**Recommendation:**

| Channel Manager |
| --- |
| No recommendation. |

Table 4-5 Exclusive Area 0

## 4.5 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.

### 4.5.1 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 4-6 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 4-6 NVRAM Blocks

### 4.5.2 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 4-7 NVRAM initialization

#### 4.5.2.1 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.

#### 4.5.2.2 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 5.4).

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.

# 5 API Description

For an interfaces overview please see Figure 2-2.

## 5.1 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<br>uint16 | Handle for the ECU that shall be diagnosed | 0 – 255<br>0 – 65535 |
| CddDrm_ LengthType | uint16 | Length of data record | 0 – 65535 |
| CddDrm_ BufferLengthType | uint16 | Length of buffer | 0 – 65535 |
| CddDrm_ RequestReturnType | uint8 | The result / status of the job | CDDDRM_REQUEST_OK<br>Request accepted |
| | | | CDDDRM_REQUEST_NO_CHANNEL<br>Request not accepted due to no channel could be allocated |
| | | | CDDDRM_REQUEST_TESTER_ACTIVE<br>Request not accepted due to an external tester is currently active |
| | | | CDDDRM_REQUEST_FIREWALL_BLOCKED<br>Request not accepted due to service is blocked by firewall |
| | | | CDDDRM_REQUEST_CONNECTION_BUSY<br>Request not accepted due to given connection is currently in use |
| CddDrm_EcudStateType | uint8 | connection specific ECU detection state | CDDDRM_ECUD_CONNECTION_NOT_DISOVERED<br>ECU/ connection not discovered yet |
| | | | CDDDRM_ECUD_CONNECTION_NOT_AVAILABLE<br>ECU/connection not available |
| | | | CDDDRM_ECUD_CONNECTION_AVAILABLE<br>ECU/connection available |

Table 5-1 Type definitions

### CddDrm_BufferStructType

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| requestBufferDataPtr | uint8* | pointer to the request buffer | - |
| responseBufferSize | uint16 | Size of the response buffer | 0 – 65535 |
| responseBufferDataPtr | uint8* | pointer to the response buffer | - |

Table 5-2 CddDrm_BufferStructType

### CddDrm_RespInfoStructType

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| responseLength | uint16 | length of the received data | 0 – 65535 |
| responseCode | uint8 | result of the response | CDDDRM_RESPONSE_POSITIVE<br>Positive response |
| | | | CDDDRM_RESPONSE_CONNECTION_TIMEOUT<br>No response received within P2/P2* time |
| | | | CDDDRM_RESPONSE_INVALID_NRC_LENGTH<br>NRC message does not equal 3 byte |
| | | | CDDDRM_RESPONSE_RCRRP_LIMIT_REACHED<br>Configured RCRRP limit reached |
| | | | CDDDRM_RESPONSE_BUFFER_TOO_SMALL<br>Receive message does not fit into the given buffer |
| | | | CDDDRM_RESPONSE_WRONG_SERVICE<br>Received service ID does not conform to SID + 0x40 |
| | | | CDDDRM_RESPONSE_FORCE_CANCEL<br>Request/response processing canceled from application |
| | | | CDDDRM_RESPONSE_TESTER_DETECTED<br>External tester detected |
| | | | CDDDRM_RESPONSE_PDUR_RX_ERROR<br>Receive of data from PduR has failed |

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| | | | CDDDRM_RESPONSE_PDUR_TX_ERROR<br>Data transmission to PduR has failed |
| | | | CDDDRM_RESPONSE_PDUR_RX_CANCELED<br>Data reception canceled from PduR |
| connectionId | uint8<br>uint16 | Handle for the ECU that was diagnosed | 0 – 255<br>0 – 65535 |

Table 5-3 CddDrm_RespInfoStructType

## 5.2 Services provided by CddDrm

### 5.2.1 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.<br>> This function is synchronous. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-4 CddDrm_GetVersionInfo()

### 5.2.2 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 5-5 CddDrm_MainFunction()

### 5.2.3 Interface EcuM

#### 5.2.3.1 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 5-6 CddDrm_Init()

#### 5.2.3.2 CddDrm_InitMemory()

| Prototype | |
|---|---|
| `void CddDrm_InitMemory ( void )` | |
| **Parameter** | |
| void | none |
| **Return code** | |
| void | none |
| **Functional Description** | |
| – Extension to Autosar –<br>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 5-7 CddDrm_InitMemory()

### 5.2.4 Interface Applikation

#### 5.2.4.1 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 the DRM if the suppress positive response message indication bit is set and thus the DRM does not need to receive a positive response within P2 time. |
| BufferInfo | contains information about the request/response buffer |
| RequestLength | request data length |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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 5-8 CddDrm_SvcSend()

### 5.2.4.2 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-9 CddDrm_SvcSend_10()

### 5.2.4.3 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |

| Return code | |
|---|---|
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-10 CddDrm_SvcSend_11()

### 5.2.4.4 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 |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |

| |
|---|
| > This function can be called from any context. |

Table 5-11 CddDrm_SvcSend_1902()

### 5.2.4.5 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 |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-12 CddDrm_SvcSend_1904()

### 5.2.4.6 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 |

| Return code | |
|---|---|
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-13 CddDrm_SvcSend_22()

### 5.2.4.7 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. |
| Call Context |
| > This function can be called from any context. |

Table 5-14 CddDrm_SvcSend_27()

### 5.2.4.8 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-15 CddDrm_SvcSend_28()

### 5.2.4.9 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-16 CddDrm_SvcSend_31()

### 5.2.4.10 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 |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-17 CddDrm_SvcSend_34()

### 5.2.4.11 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 |
| **Return code** |
| CddDrm_RequestReturnType: CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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 5-18 CddDrm_SvcSend_36()

### 5.2.4.12 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 |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-19 CddDrm_SvcSend_37()

### 5.2.4.13 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-20 CddDrm_SvcSend_3E()

### 5.2.4.14 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<br>FALSE: do not set suppresses positive response message indication bit |
| BufferInfo | contains information about request/response buffer |
| **Return code** | |
| CddDrm_RequestReturnType | CDDDRM_REQUEST_OK: Service request accepted<br>CDDDRM_REQUEST_NO_CHANNEL: Service request not accepted due to no channels available<br>CDDDRM_REQUEST_CONNECTION_BUSY: Service request not accepted, connection already in use<br>CDDDRM_REQUEST_TESTER_ACTIVE: Service request not accepted, DRM is in external tester active mode<br>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.<br>> This function is reentrant. |
| Call Context |
| > This function can be called from any context. |

Table 5-21 CddDrm_SvcSend_85()

### 5.2.4.15 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<br>E_NOT_OK: cancel request not accepted |
| **Functional Description** | |
| Cancels the communication with the particular connection | |
| **Particularities and Limitations** | |
| > The function is synchronous.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-22 CddDrm_CancelRequest()

### 5.2.4.16 CddDrm_StartEcuDetection()

| Prototype | |
|---|---|
| `Std_ReturnType` **`CddDrm_StartEcuDetection`** `( void )` | |
| **Parameter** | |
| void | none |
| **Return code** | |
| Std_ReturnType | E_OK: ECU Detection has been started<br>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.<br>> This function is reentrant. |
| Call Context |
| > This function can be called from any context. |

Table 5-23 CddDrm_StartEcuDetection()

### 5.2.4.17 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.<br>> This function is reentrant. | |
| Call Context | |
| > This function can be called from any context. | |

Table 5-24 CddDrm_StopEcuDetection()

### 5.2.4.18 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<br>CDDDRM_ECUD_CONNECTION_NOT_AVAILABLE: The ECU is not available<br>CDDDRM_ECUD_CONNECTION_NOT_DISCOVERED: The ECU Detection has not been processed yet on the requested Connection |
| **Functional Description** | |
| Tries to start new service request to a connection. | |

| Particularities and Limitations |
|---|
| > The function is synchronous.<br>> This function is reentrant. |
| Call Context |
| > This function can be called from any context. |

Table 5-25 CddDrm_GetEcuDetectionResult()

### 5.2.5 Interface PduR

#### 5.2.5.1 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.<br>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.<br>> The function is synchronous. | |
| Call Context | |
| > This function can be called from any context | |

Table 5-26 CddDrm_Transmit()

#### 5.2.5.2 CddDrm_CancelTransmit()

| Prototype | |
|---|---|
| `Std_ReturnType CddDrm_CancelTransmit ( PduIdType TxPduId)` | |
| **Parameter** | |
| TxPduId | id of the CddPduRLowerLayerTxPdu. |
| **Return code** | |
| Std_ReturnType | E_OK: the transmission (reception of a diagnostic response from Drm point of view) request was canceled.<br>E_NOT_OK: the transmission was not canceled. |
| **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 can be called from any context |

Table 5-27 CddDrm_CancelTransmit()

### 5.2.6 Interface BswM

#### 5.2.6.1 CddDrm_ExternalTesterConnected()

| Prototype | |
|---|---|
| `Std_ReturnType CddDrm_ExternalTesterConnected ( bool Present )` | |
| **Parameter** | |
| Present | TRUE: An external tester is currently present.<br>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 5-28 CddDrm_ExternalTesterConnected()

## 5.3 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_CddDrmCopyTxData<br>PduR_CddDrmTxConfirmation<br>PduR_CddDrmStartOfReception<br>PduR_CddDrmCopyRxData<br>PduR_CddDrmRxIndication |

| Component | API |
|---|---|
| SchM | optional SchM_Enter_CddDrm_<ExclusiveArea><br>optional SchM_Exit_CddDrm_<ExclusiveArea> |

Table 5-29 Services used by the CddDrm

## 5.4 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 the CddDrm.

### 5.4.1 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.<br>NVM_REQ_OK: last asynchronous request has been finished successfully<br>NVM_REQ_NOT_OK: not used in this context<br>NVM_REQ_PENDING: not used in this context<br>NVM_REQ_INTEGRITY_FAILED: not used in this context<br>NVM_REQ_BLOCK_SKIPPED: not used in this context<br>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.<br>> This function is asynchronous.<br>> Must be configured for every CddDrm related NVRAM block | |
| Expected Caller Context | |
| > This function can be called from any context. | |

Table 5-30 CddDrm_NvM_JobFinished()

### 5.4.2 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.<br>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 5-31 CddDrm_NvM_InitEcuDetectionData()

## 5.5 Configurable Interfaces

### 5.5.1 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.

#### 5.5.1.1 <ResponseNotification>()

| Prototype | |
|---|---|
| `void <ResponseNotification> ( const CddDrm_RespInfoStructType *Response )` | |
| **Parameter** | |
| Response | Information about the response (response length, response code 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 5-32 <ResponseNotification>()

#### 5.5.1.2 <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 request data was successful copied to PduR. | |
| **Particularities and Limitations** | |
| > This function shall be synchronous.<br>> This function shall be reentrant. | |
| Call Context | |
| > This function is called from task context. | |

Table 5-33 <TxConfirmation>()

### 5.5.1.3 <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.<br>> This function shall be reentrant. | |
| Call Context | |
| > This function is called from task context. | |

Table 5-34 <EcuDetectionFinished>()

### 5.5.1.4 <FirewallUserCallback>()

| Prototype | |
|---|---|
| `boolean <FirewallUserCallback> ( CddDrm_ConnectionIdType ConnectionId, uint8 ServiceId )` | |
| **Parameter** | |
| ConnectionId | Connection for which the firewall shall be checked from application. |
| Uint8 | Service ID that shall be requested. |
| **Return code** | |
| boolean | TRUE: Service request allowed<br>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.<br>> This function shall be reentrant. |
| Call Context |
| > This function is called from task context. |

Table 5-35 <FirewallUserCallback>()

# 6 Configuration

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

## 6.1 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.

## 6.2 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.

## 6.3 Configuration to Diagnose ECUs Connected to the Network

To send diagnostic messages to other ECUs and to receive their responses the CddDrm, PduR and the underlying bus specific communication stack (e.g. CanTp) has to be configured accordingly. Here the PduR has to be configured to support gateway functionality. Therefore the CddDrm related N-SDUs and the TP related N-SDUs and N-PDUs have to be configured as shown in Figure 6-1.

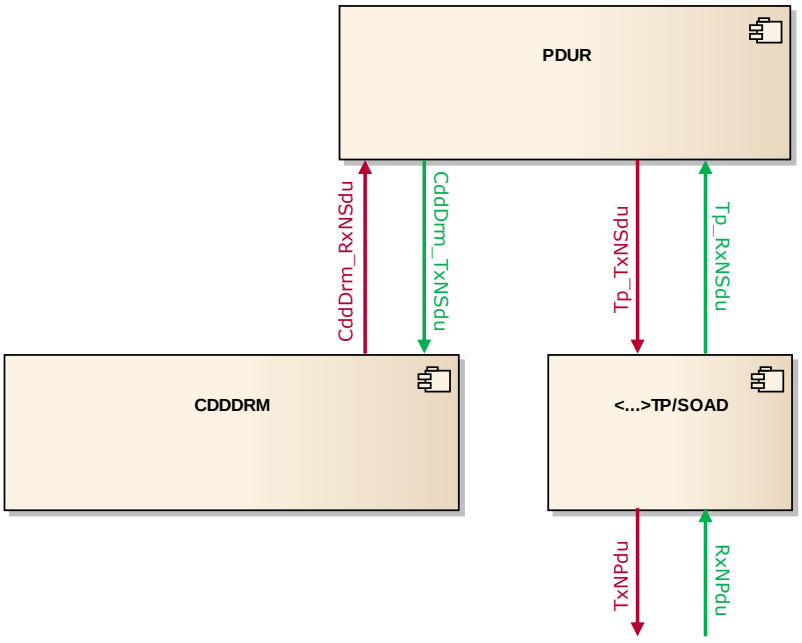


Figure 6-1 PduR gateway routing

## 6.4 Configuration to Diagnose the own ECU

To send diagnostic messages to the own ECU and to receive the corresponding responses the CddDrm, PduR and the Dcm has to be configured accordingly. Here the PduR has to support API forwarding functionality.

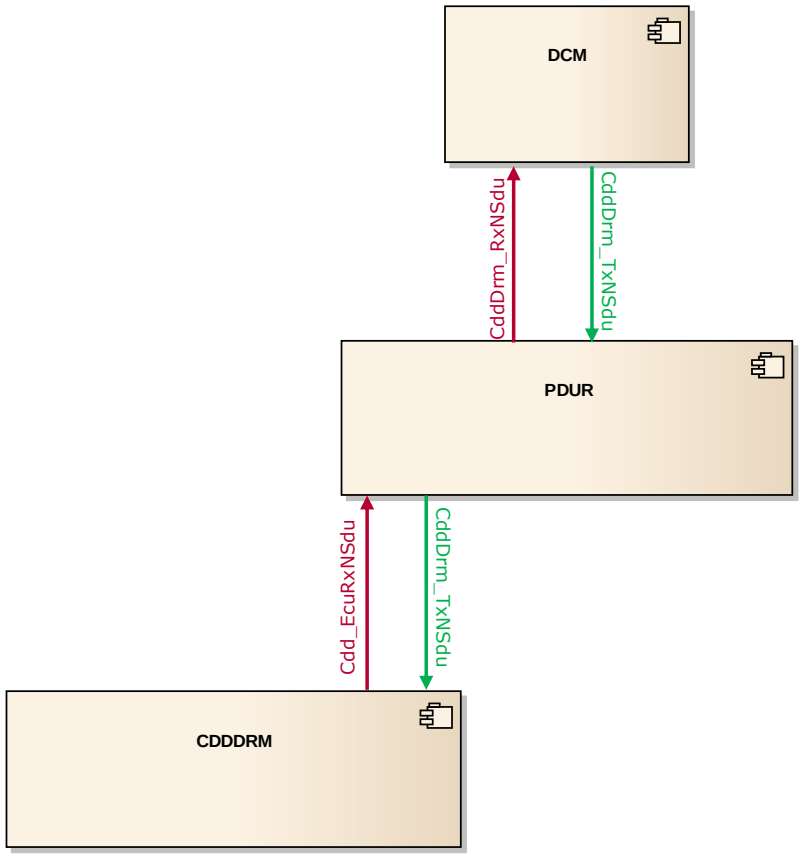


Figure 6-2 PduR API forwarding

**Note**
The DcmDslConnection connected to the Drm 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 the Drm.

## 6.5 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.

# 7 Glossary and Abbreviations

## 7.1 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. |
| $P2^*_{Client}$ | 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. |
| $S3_{Client}$ | Time between TesterPresent (0x3E) request messages transmitted by the client to keep a diagnostic session other than the defaultSession active in server. |

Table 7-1 Glossary

## 7.2 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 |
| 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 |
| UDS | Unified Diagnostic Services |
| 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 7-2 Abbreviations

# 8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com