Elektrobit

EB tresos® AutoCore Generic 8 DLT documentation

product release 8.8.4

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

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 DLT documentation

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

This document provides:

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

2. Supported features

The ACG8 DLT product implements the AUTOSAR module `Dlt` based on AUTOSAR 4.2.1, with AUTOSAR 4.3.1 features and EB-specific enhancements that are compatible with the AUTOSAR standard.

2.1. Supported Dlt (Base) features

The following features are part of the basic feature set and are available when ACG8 DLT (Base) is licensed:

- **Dlt core functions:**
  - Generic creation of log and trace messages
  - Generic reception of log and trace messages
  - Buffering and filtering management
  - Message filtering based on log level and trace status
  - Generic handling of control requests
  - Parallel transmission of multiple frames within one PDU
  - Handling of log channels
- **Dlt service interface according to AUTOSAR 4.2.1 and AUTOSAR 4.3.1:**
  - Registering software components and contexts using Dlt_RegisterContext()
  - Un-registering software components and contexts using Dlt_UnregisterContext() (only for AUTOSAR 4.3.1)
  - Sending log and trace messages from SWCs using Dlt_SendLogMessage() and Dlt_SendTraceMessage()
  - Notifying SWCs about log level and trace status changes
  - Control behavior of Dlt at run-time from SWCs (only for AUTOSAR 4.3.1)
- **Virtual function bus tracing:**
  - Interface for VFB trace
  - Support for generating and implementing RTE hook functions
- **Persistent storage of configuration:**
  - Storing the run-time configuration in non-volatile (NV) memory
- **Dlt assistant (tooling):**
  - Support for generating Dlt ports according to service needs
- **Message transmission:**

- Transmitting communication data via a specific communication channel of a specific network (PDU-based, e.g. CAN, FlexRay) using underlying communication layers
- Adapting/segmenting a log or trace message according to the needs of the communication channel
- Support for delaying the transmission of Dlt messages after start-up for a configurable time

- **Verbose mode:**
  - Transmitting messages in verbose mode
  - Support of a configuration switch to enable/disable verbose mode
  - Building Dlt messages with extended headers [SWS_Dlt_00457]
- **Control messages:**
  - Controlling the Dlt via one communication channel by means of control messages, i.e. messages received on the bus from an external client
- **Support for synchronized time base:**
  - Support of synchronized time stamps/chronological order of log and trace data

2.2. Supported Dlt (feature package 1) features

The following features provide run-time optimizations for multi-core architectures and must be licensed in addition to the ACG8 DLT (Base) product with the ACG8 DLT (Feature Package 1) product:

- **Support for BSW distribution:**

  The BSW distribution support for Dlt permits the distribution of selected components of the Dlt in order to provide enhanced run-time performance. For multi-core architectures, the following Dlt operations/APIs are distributed to satellite cores:

  - C/S interface DltService operation `SendLogMessage`
  - C/S interface DltService operation `SendTraceMessage`
  - VFB tracing via internal representation of `Dlt_SendTraceMessage`

  With BSW distribution for Dlt, the operations and functions above are partially executed on the calling core via a satellite implementation. In case of `SendLogMessage`, the message filtering functionality is executed completely on the calling core. This reduces the inter-core communication and provides enhanced run-time performance.

  The BSW distribution for Dlt requires an EB run-time environment with BSW distribution support.

- **Traffic shaper:**
  - Support of bandwidth management

- Configurable bandwidth limits per communication channel
- Delaying of message transmission if bandwidth is exhausted

3. ACG8 DLT release notes

3.1. Overview

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

3.2. Scope of the release

3.2.1. Configuration tool

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

- EB tresos Studio: 28.2.0 b211016-0103

3.2.2. AUTOSAR modules

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

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
Dlt	4.2.1 []	4.2.1 [0000]	1.8.7	Elektrobit Automotive GmbH

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

3.2.3. EB (Elektrobit) modules

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

Module name	Module version	Supplier
No EB modules available		

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

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

3.3. Module release notes

3.3.1. Dlt module release notes

- AUTOSAR R4.2 Rev 1
- AUTOSAR SWS document version: 4.2.1
- Module version: 1.8.7.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 1.8.7
2021-10-27

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

Module version 1.8.6
2021-06-25

- ASCDLT-887 Fixed known issue: Dlt might refuse log/trace messages if AUTOSAR 4.3.1 is used and Message Options bitfield is defined according to the AUTOSAR 4.3.1 format
- ASCDLT-894 Fixed known issue: Dlt might have unexpected behavior when control messages are used and nested critical sections are not considered

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

Module version 1.8.5
2021-03-05

- ASCDLT-848 Fixed known issue: Enabling DltEnableBswDistribution and DltImplementVfbTrace can lead to compiler warnings and/or errors
- ASCDLT-858 Fixed known issue: Dlt returns a positive response for a GetLogInfo control message request with an invalid tuple
- ASCDLT-876 Fixed known issue: Dlt might send log/trace messages on wrong log channel(s) when pre-compile time configuration of log channel assignments is used

Module version 1.8.4
2020-10-23

- ASCDLT-820 Fixed known issue: The Dlt has an increased memory usage when RteVfbTraceEnabled is enabled and trace functions are configured
- ASCDLT-846 Fixed known issue: Dlt fails to compile when DltImplementNVRamStorage is enabled but DltDevErrorDetect and DltImplementFilterMessages are disabled on AUTOSAR 4.2
- ASCDLT-849 Fixed known issue: Different ReadRamBlockFromNvmBlock function prototypes can lead to compiler warnings and/or errors

Module version 1.8.3
2020-06-19

- ASCDLT-796 Fixed known issue: Structure of the Register/UnregisterContext notification is wrong
- Implemented support for retrieving EcuId for DLT at runtime.
- ASCDLT-815 Fixed known issue: Dlt generation fails for DltVfbTracePayloadMaxSize when DltImplementVfbTrace is disabled

Module version 1.8.2
2020-02-21

- Added support for Dlt VFB tracing with arguments of the hook functions.
- ASCDLT-773 Fixed known issue: Dlt_SendTraceMessage() returns a wrong return value if Det is disabled.
- ASCDLT-780 Fixed known issue: Dlt behaves unexpectedly if message transmission is interrupted by a send log/trace request
- Added checks for valid SessionId in Dlt_SendLogMessage() and Dlt_SendTraceMessage() even if message filtering is OFF

- Changed the byte order handling of log channel names to be based on DltTupleEndianness' set value

Module version 1.8.1
2019-10-11

- ASCDLT-669 Fixed known issue: The Dlt does not take into consideration the platform's endianness when receiving AppId, ContextId and LogChannelName as control message payload
- ASCDLT-679 Fixed known issue: The GetVerboseModeStatus control message wrongly extracts the AppId/ContextId from the extended header
- ASCDLT-685 Fixed known issue: Dlt does not compile if DltGeneralStartUpDelayTimer is enabled and DltEnableTrafficShaper is disabled
- ASCDLT-686 Fixed known issue: The configured VFB traces use the wrong AppId when DltServiceAPI is set to AUTOSAR_431
- Implemented configurable behavior in case PduR_DltTransmit returns E_NOT_OK
- ASCDLT-699 Fixed known issue: GetLogInfo's LEN field is incorrect when the message is sent from the register/unregister context notification
- ASCDLT-698 Fixed known issue: Dlt could use wrong API identifiers when reporting errors to the Det.
- ASCDLT-702 Fixed known issue: When you unregister an invalid tuple, a critical section is closed before it opens
- Updated handling of AplicationId-ContextId tuples that are identical for multiple instances of the same SW-C
- ASCDLT-708 Fixed known issue: A ResetToFactoryDefault call does not erase the Dlt native block due to incorrect writing priority
- ASCDLT-704 Fixed known issue: Dlt can send unexpected log messages and/or trace messages due to wrong filtering of unregistered tuples
- ASCDLT-710 Fixed known issue: Dlt log and/or trace messages requested on satellite cores are sent on the default log channel regardless of configuration
- Implemented the GetSoftwareVersion control message.
- ASCDLT-726 Fixed known issue: Dlt uses wrong identifier for the StbM timebase when fetching the synchronized timebase

Module version 1.8.0
2019-06-14

- Implemented Logchannel concept from SWS 4.3.1

- Implemented AUTOSAR 4.3.1 APIs, control messages and operations.
- ASCDLT-569 Fixed known issue: Dlt_Rte_hook.c does not compile when trace functions are configured and DltEnableBswDistribution is enabled
- ASCDLT-560 Fixed known issue: VFB traces are not sent when there is no explicit setting of trace status via the Dlt command SetTraceStatus()
- ASCDLT-559 Fixed known issue: Trace/log messages sent by satellite cores are not filtered correctly when the log/trace settings are changed via Dlt commands using wildcards
- ASCDLT-584 Fixed known issue: Control messages that receive ApplicationId and ContextId parameters, wrongly respond with DLT_CTRL_ERROR and fail, if header endianness is LSB
- Implemented startup delay timer as specified by AUTOSAR 4.3.1.
- ASCDLT-568 Fixed known issue: SWCs are not notified when log level/trace status/verbose mode settings are changed via Dlt commands using wildcards
- ASCDLT-585 Fixed known issue: DltMaxCountContextIds is not allowed to have a value smaller than the product of DltMaxCountAppIds and DltMaxCountContextIdsPerAppId
- ASCDLT-587 Fixed known issue: Message filtering uses only the first found ContextId as input, when the same ContextId exists for multiple ApplicationIDs
- ASCDLT-615 Fixed known issue: Reception of a DLT message can fail when the message is received in multiple packages
- Updated types used for keeping information regarding the message length

Module version 1.7.1
2019-02-15

- ASCDLT-503 Fixed known issue: A store request interrupted by another store request corrupts the Dlt internal buffer

Module version 1.7.0
2018-10-26

- ASCDLT-447 Fixed known issue: Data related to context registration and log level could be out of sync on satellite cores in case of multi-core setup
- Implemented support for timestamps provided by the StbM module
- ASCDLT-452 Fixed known issue: Wrong check in control message validation can lead to out-of-bounds access
- ASCDLT-501 Fixed known issue: Dlt generates incorrectly the function pointers to RTE calls for the application notification callbacks

Module version 1.6.0
2018-06-22

- ASCDLT-351 Fixed known issue: Missing memory sections in the Dlt software component description cause uncompilable code
- Implemented verbose mode. Note: NvM layout has changed, NvM block size of NVM_BLOCK_DLT_-DATASET was increased by 1.
- Changed the data types Dlt_MessageLogInfoType and Dlt_MessageTraceInfoType to comply with the SWS. Note: External APIs (Dlt_SendLogMessage, Dlt_SendTraceMessage) are affected.
- The Dlt_RegisterContext, Dlt_SendLogMessage and Dlt_SendTraceMessage interfaces are now exposed in the new Dlt_BSW.h header file.
- ASCDLT-393 Fixed known issue: The error code assigned to DLT_E_NOT_IN_VERBOSE_MODE is wrong in Dlt_swc_interface.arxml
- Memory sections have been updated; deprecated sections have been removed.
- ASCDLT-423 Fixed known issue: Wrong Dlt instance ID is reported to Det when using multicore configuration
- ASCDLT-424 Fixed known issue: Certain control API functions that manipulate the Dlt packet header are not working if their corresponding configuration parameter is initialized to FALSE
- Implemented reception of Control Messages.

Module version 1.5.0
2018-02-23

- ASCDLT-331 Fixed known issue: Registration information is not loaded from NvM for SWCs
- Implemented compliance to MISRA-C:2012
- ASCDLT-329 Fixed known issue: Compile error occurs when Dlt is used with a non-EB NvM module
- Update according to the latest Tresos Studio

Module version 1.4.2
2017-10-09

- ASCDLT-285 Fixed known issue: Initialization process for both slave cores and master core is not correctly done
- ASCDLT-289 Fixed known issue: Buffer overflow can occur if messages from SWC are sent with appId/contextId registered from a BSWC
- ASCDLT-288 Fixed known issue: DLT reports unnecessary DET error when all datasets have been processed

- ASCDLT-316 Fixed known issue: Corruption of Dlt table data during NvM restore defaults functionality
- ASCDLT-318 Fixed known issue: Incorrect behavior for NULL values of parameters Application_ID and Context_ID in Dlt_SetLogLevel and Dlt_SetTraceStatus APIs
- Improved the usage of NvM: NvMInit callbacks made optional, NvM retry mechanism was removed
- Implemented new configuration option for configurable queue size of server ports created by the DLT assistant

Module version 1.4.1
2017-03-21

- ASCDLT-268 Fixed known issue: Restoring of the runtime data from NvM fails if a SWC already called Dlt_RegisterContext
- ASCDLT-273 Fixed known issue: Out of bounds array access happens if DltMaxCountAppIds is less than DltMaxCountContextIdsPerAppId

Module version 1.4.0
2016-12-19

- ASCDLT-215 Fixed known issue: DLT message transmission stops
- ASCDLT-223 Fixed known issue: Dlt_SendLog/TraceMessage() is not reentrant
- ASCDLT-235 Fixed known issue: Out of bound access in Dlt_AppToContextIdTable table when Dlt_RegisterContext() is called more than DltMaxCountContextIdsPerAppId times
- ASCDLT-236 Fixed known issue: DLT stops sending log Messages after restore from NVM

Module version 1.3.0
2016-11-04

- ASCDLT-206 Fixed known issue: The Message Counter (MCNT) is wrongfully reset
- ASCDLT-207 Fixed known issue: Messages in transmission might be corrupted
- ASCDLT-214 Fixed known issue: Dlt runtime variables reset to configuration values after restore from NvRAM is successful
- ASCDLT-208 Fixed known issue: Incompatible design between Dlt persistent storage functionality and NvM

Module version 1.2.2
2016-10-07

- Improved validation of the Dlt Assistant

Module version 1.2.1
2016-09-09

- ASCDLT-149 Fixed known issue: Dlt buffer indexes point to incorrect location in message buffer
- ASCDLT-144 Fixed known issue: Transmission of DLT Log messages stops after some time
- ASCDLT-156 Fixed known issue: Incompatibile code is generated due to the definition of implementation data types
- ASCDLT-150 Fixed known issue: The implementation for Rte VFB tracing functions is always generated
- ASCDLT-164 Fixed known issue: Warnings during importer run after enabling DLT BSW dsitribution
- ASCDLT-189 Fixed known issue: Det.h is always included by Dlt_Int.h

Module version 1.2.0
2016-05-25

- Implement Dlt_MainFunction()
- ASCDLT-96 Fixed known issue: Dlt_ComCopyTxData returns a wrong type if SDU length is 0
- ASCDLT-99 Fixed known issue: Dlt does not unlock the transmit buffer after it has received a negative Tx confirmation
- BSW Distribution - implementation
- Added possibility to transmit multiple Dlt frames in one Pdu

Module version 1.1.0
2015-11-25

- Implement DLT Persistency Configuration
- Implement RTE / VFB Tracing
- Implement SW-C Informer
- Implement Dlt Assistant

Module version 1.0.0
2015-07-08

- Initial version

3.3.1.2. New features

- No new features have been added since the last release.

3.3.1.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- New traffic shaper configuration option

  Description:

  The configuration parameter `DltEnableTrafficShaper` has been introduced in order to enable/disable the traffic shaper functionality. An EB_DLT_MC_OPT license must be available. Otherwise this parameter is not visible.

  The following configuration parameters are editable if `DltEnableTrafficShaper` is enabled:

  - `DltBandwidthForComModule`
  - `DltTimePeriodTrafficShaping`
- New PDAV configuration option

  Description:

  The configuration parameter `DltCSPortsQueueLength` has been introduced in order to provide a customizable queue length for the server ports created by the Dlt.

  The value of `DltCSPortsQueueLength` controls the following behavior:

Value of `DltCSPortsQueueLength`	**Description**
0	No SERVER-COM-SPECs are created. The queue length is calculated by Rte based on the number of connected ports.
Any other value	Defines the queue size of the SERVER-COM-SPECs created by the DLT to be the configured value.

Table 3.3. Values of `DltCSPortsQueueLength`

- The Dlt_MessageLogInfoType and Dlt_MessageTraceInfoType types are now AUTOSAR compliant.
- Published interfaces for BSW usage

  Description:

The Dlt_RegisterContext, Dlt_SendLogMessage and Dlt_SendTraceMessage interfaces are now published for BSW usage, and are located in the new Dlt_BSW.h header file, if the DltRteUsage configuration parameter is set to false.

- Different Dlt_ReturnType values based on the AUTOSAR SWS version

  Description:

  Dlt_ReturnType is defined differently in the 4.2.1 and 4.2.2 AUTOSAR SWS. Based on the new configuration parameter, DltDefaultAutosarReturnValues, the customer can select with what AUTOSAR SWS (4.2.1/4.2.2) should the Dlt module comply, regarding Dlt_ReturnType's definition.

- Implemented control message service requests are also available as C-APIs

  Description:

Control service request	**C-API**
Set_LogLevel	Dlt_SetLogLevel
Set_DefaultLogLevel	Dlt_SetDefaultLogLevel
Set_TraceStatus	Dlt_SetTraceStatus
Set_DefaultTraceStatus	Dlt_SetDefaultTraceStatus
Get_LogInfo, option 4	Dlt_GetLogLevel, used for one specific tuple at a time
Get_LogInfo, option 5	Dlt_GetTraceStatus, used for one specific tuple at a time
Get_DefaultLogLevel	Dlt_GetDefaultLogLevel
Get_DefaultTraceStatus	Dlt_GetDefaultTraceStatus
Store_Config	Dlt_StorePersistent
SetComInterfaceMaxBandwidth	Dlt_SetComInterfaceMaxBandwidth
GetComInterfaceMaxBandwidth	Dlt_GetComInterfaceMaxBandwidth
SetVerboseMode	Dlt_SetVerboseMode
GetVerboseModeStatus	Dlt_GetVerboseModeStatus
SetMessageFilterering	Dlt_SetMessageFiltering
GetMessageFiltereringStatus	Dlt_GetMessageFilteringStatus
SetUseECUID	Dlt_SetUseECUID
GetUseECUID	Dlt_GetUseECUID
SetUseSessionID	Dlt_SetUseSessionID
GetUseSessionID	Dlt_GetUseSessionID

Control service request	C-API
UseTimestamp	Dlt_SetUseTimestamp
GetUseTimestamp	Dlt_GetUseTimestamp
UseExtendedHeader	Dlt_SetUseExtendedHeader
GetUseExtendedHeader	Dlt_GetUseExtendedHeader

Table 3.4. Control service requests and C-APIs

- Additional control APIs for global logging

  Description:

API	Description
Dlt_SetGlobalLogging	When message logging is disabled, all messages are discarded by the Dlt, regardless of log level or trace status settings
Dlt_GetGlobalLogging	Retrieves the current message logging status

Table 3.5. Additional control APIs for global logging

- The transmission of sync data between master and slave cores is done using a one to one sender-receiver channel with a configurable queue size.

3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- No support for Dem interface

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  Interface to Dem is currently not supported.

  Rationale:

  Not planned for the current release.

  Requirements:

  AUTOSAR 4.2.1: SWS_Dlt_00474, SWS_Dlt_00475, SWS_Dlt_00476, SWS_Dlt_00377, SWS_Dlt_-00477, SWS_Dlt_00478, SWS_Dlt_00479, SWS_Dlt_00470 AUTOSAR 4.3.1: SWS_Dlt_00474, SWS_-

Dlt_00475, SWS_Dlt_00377, SWS_Dlt_00477, SWS_Dlt_00478, SWS_Dlt_00270, SWS_Dlt_00479, SWS_Dlt_00274, SWS_Dlt_00031, SWS_Dlt_00781

- No support for Det interface

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  Interface to Det is currently not supported.

  Rationale:

  Not planned for the current release.

  Requirements:

  AUTOSAR 4.2.1: SWS_Dlt_00430, SWS_Dlt_00431, SWS_Dlt_00376, SWS_Dlt_00480, SWS_Dlt_-00432 AUTOSAR 4.3.1: SWS_Dlt_00430, SWS_Dlt_00376, SWS_Dlt_00031, SWS_Dlt_00432

- No support for response on event

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  Response on event is currently not supported.

  Rationale:

  Not planned for the current release.

  Requirements:

  AUTOSAR 4.2.1: SWS_Dlt_00339, SWS_Dlt_00037, SWS_Dlt_00340, SWS_Dlt_00039

- No support for ApplicationID and ContextID description

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  ApplicationID and ContextID description are not stored in NvM.

  Rationale:

ApplicationID and ContextID description are not stored when calling Dlt_RegisterContext

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00064, ECUC_Dlt_00815

- No support for SWC injection

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

Injecting data to SWC is currently not supported.

Rationale:

Not planned for the current release.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00017, SWS_Dlt_00018, SWS_Dlt_00019, SWS_Dlt_00259, SWS_Dlt_-00260, SWS_Dlt_00498, ECUC_Dlt_00819 AUTOSAR 4.3.1: SWS_Dlt_00259, SWS_Dlt_00498, SWS_-Dlt_00778, ECUC_Dlt_00847

- No support for timing messages

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

Timing messages are currently not supported.

Rationale:

Not planned for the current release.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00221, SWS_Dlt_00222

- No support DLT communication module

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The DLT module does not provide a separate communication module. The mechanism of the PduR is used instead.

Rationale:

Data transfer via UDP is provided by the AUTOSAR communication stack.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00040, SWS_Dlt_00042, SWS_Dlt_00043, SWS_Dlt_00461, SWS_Dlt_-00463, SWS_Dlt_00263, SWS_Dlt_00264, SWS_Dlt_00485, SWS_Dlt_00265

- No support for DltVfbTraceLogLevel configuration parameter

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The DltVfbTraceLogLevel configuration parameter is not supported.

Rationale:

The parameter is of no use. VFB messages are "trace" messages, not "log" messages, thus there is no "log level" which can be assigned to them.

Requirements:

AUTOSAR 4.2.1: ECUC_Dlt_00839

- DltSwcApplicationId configuration parameter is not of type string

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

DltSwcApplicationId configuration parameter is of type integer instead of type string.

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00858

- DltSwcContextId configuration parameter is not of type string

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

DltSwcContextId configuration parameter is of type integer instead of type string.

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00859

- DltLogChannelMaxNumOfRetries not used

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Implementation for DltLogChannelMaxNumOfRetries is not supported

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00884, SWS_Dlt_00761 Dlt.ASR431.Ref_SWS_Dlt_00697_NegativeTx-Confirmation_LogChannelMaxNumOfRetries

- DltLogChannelTransmitCycle not used

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Implementation for DltLogChannelTransmitCycle is not supported

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00885

- DltITxPduUsesTp not used

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Implementation for DltITxPduUsesTp is not supported

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00913

- API Gpt_GetTimeElapsed() for timestamp is not used

  Affected AUTOSAR releases:

  - R4.3 rev 1

  Description:

  No support from API Gpt_GetTimeElapsed() in getting message timestamp .

  Requirements:

  AUTOSAR 4.3.1: SWS_Dlt_00654

- The message buffer size must not be zero

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  The message buffer size must always be at least as large as DltMaxMessageLength. It is not possible to configure a message buffer size of length zero in order to force an immediate transmission of messages.

  Rationale:

  There must always be at least a small buffer in which the message and its header fields can be assembled. The DLT uses the ring buffer for this. A configuration check assures that the ring buffer is large enough for the largest possible message. When the message has been assembled, the PduR is triggered. Thus, if a "connection less interface" is connected to the PduR, the data is sent immediately and this requirement is thus implicitly fulfilled.

  Requirements:

  AUTOSAR 4.2.1: SWS_Dlt_00493

- Scheduled function

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  The DLT module provides a scheduled function Dlt_MainFunction() which must be called periodically by the BSW scheduler.

  Rationale:

This is required for a proper implementation of the traffic shaper feature.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00468

- Different time base

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The Dlt module does not use the Gpt module for the timestamp functionality. Instead, the timestamp is provided by:

- the Os, through the usage of the Os_GetTimeStamp API (if configured to be used)
- the StbM, through the usage of the StbM_GetCurrentTime API (if configured to be used)

Rationale:

- Less configuration effort necessary
- Higher performance
- The OS provides proper functions for calculations involving the counter values, which automatically handle counter overflows

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00481 AUTOSAR 4.2.1: ECUC_Dlt_00905

- Traffic shaping

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The DLT module does not have the functionality for the diagnostic module DCM.

Requirements:

Dlt.TrafficShaping.DiagInterfaces, Dlt.TrafficShaping.DiagChannel

- Store persistent

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The Dlt module shall let the user configure the restore value for the trace status belonging to a tuple of AppId and ContextId via the callout Dlt_ApplGetConfigFactoryDefault().

Rationale:

- The information provided by the AUTOSAR SWS 2.1. regarding this parameter is not specific enough.

Requirements:

SWS_Dlt_00288, SWS_Dlt_00348

- VFB Trace message type and message info

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

In VFB tracing messages, the Message Type (MSTP) header field is set to DLT_TYPE_APP_TRACE. The Message Trace Info (MSIN) header field is set to DLT_TRACE_VFB.

Rationale:

- Requirements SWS_Dlt_00120, SWS_Dlt_00484 and SWS_00123 contradict each other. Requirements SWS_Dlt_00120 and SWS_Dlt_00123 provide the better fitting values.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00484

- Dlt_MessageTypeType

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The Dlt_MessageTypeType enum values start from 0, not from 1.

Rationale:

- Requirements SWS_Dlt_00120 and SWS_00224 contradict each other. The values mentioned in SWS_Dlt_00120 have been chosen, since enums naturally start with 0 in C.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00224

- VFB Trace Network Trace Info Type

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The Dlt_MessageNetworkTraceInfoType provides an additional DLT_NW_TRACE_MOST element.

Rationale:

- Requirements SWS_Dlt_00125 and SWS_00233 contradict each other. The additional DLT_NW_-TRACE_MOST value specified by SWS_Dlt_00125 is provided to assure compatibility and avoid compilation errors.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00233

- No support for some control messages.

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The following Control messages to DLT are currently not supported: - 0x03: Get_LogInfo, Options: 3 - 0x07: SetComInterfaceStatus - 0x0B: SetTimingPackets - 0x0C: GetLocalTime - 0x14: MessageBuffer-Overflow - 0x16: GetComInterfaceStatus - 0x17: GetComInterfaceNames - 0xFFF ... 0xFFFFFFFF: Cal-ISW-CInjection

Rationale:

Not planned for the current release.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00465, SWS_Dlt_00290, SWS_Dlt_00048, SWS_Dlt_00049, SWS_Dlt_-00050, SWS_Dlt_00051, SWS_Dlt_00201, SWS_Dlt_00501, SWS_Dlt_00197, SWS_Dlt_00502, SWS_-Dlt_00489, SWS_Dlt_00207, SWS_Dlt_00208, SWS_Dlt_00217, SWS_Dlt_00218, SWS_Dlt_00219, SWS_Dlt_00220, SWS_Dlt_00487, SWS_Dlt_00488, SWS_Dlt_00247, SWS_Dlt_00248, SWS_Dlt_-00249, SWS_Dlt_00428, Dlt.GetLogInfo.Option3.NoLogLevelNoTraceStatus, Dlt.GetLogInfo.ComInterface

- Prioritization of Dlt Control Messages responses

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  Dlt does not prioritize Control Messages responses over normal log or trace messages.

  Rationale:

  Not planned for the current release.

  Requirements:

  AUTOSAR 4.2.1: SWS_Dlt_00490
- Accept new control messages

  Affected AUTOSAR releases:

  - R4.2 rev 1

  Description:

  Dlt does not reject new Control Messages while an old one is not finished.

  Rationale:

  Adapted due to a popular Dlt external viewer that sends Control Messages without waiting for a response for each one before sending the new requests.

  Requirements:

  AUTOSAR 4.2.1: SWS_Dlt_00417
- Several imported types are not in use

  Affected AUTOSAR releases:

  - R4.2 rev 1
  - R4.3 rev 1

  Description:

  The Dlt module does not use the following imported types:

  - Dem_DTCFormatType, Dem_EventIdType, Dem_UdsStatusByteType
  - Gpt_ChannelType, Gpt_ValueType

- NvM_BlockIdType
- StbM_SynchronizedTimeBaseType, StbM_TimeStampExtendedType

Rationale:

- The Dem types are not used because the Dem interface is not supported
- The Gpt types are not used because the Dlt uses the Os/StbM instead, for timestamp purposes
- The NvM type is not used because the Dlt does not require it
- The StbM types are not used because the Dlt does not require them

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00729

- Several optional interfaces are not in use

Affected AUTOSAR releases:

- R4.2 rev 1
- R4.3 rev 1

Description:

The Dlt module does not use the following optional interfaces:

- Dem_DltGetAllExtendedDataRecords, Dem_DltGetMostRecentFreezeFrameRecordData, Dem_GetDTCOfEvent
- Gpt_EnableNotification, Gpt_StartTimer
- StbM_GetCurrentTimeExtended

Rationale:

- The Dem optional interfaces are not used because the Dem interface is not supported
- The Gpt optional interfaces are not used because the Dlt uses the Os/StbM instead, for timestamp purposes
- The StbM optional interface is not used because the standard timestamp format is sufficient for the Dlt

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00763

- The timestamp resolution is platform-dependent

Affected AUTOSAR releases:

- R4.2 rev 1

- R4.3 rev 1

Description:

The Dlt module cannot always provide 0.1 milliseconds resolution for the timestamps.

Rationale:

The requirement implies the usage of the Gpt module for the timestamp functionality. The Gpt can provide resolutions down to 1 microseconds, hence the requirement specifies a resolution that can be provided by the Gpt. The currently implemented timestamp providers (Os/StbM) are highly platform-dependent and offer different resolutions, as a consequence.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00309 AUTOSAR 4.3.1: PRS_Dlt_00309

- Information about enabled/disabled interfaces is not stored persistently

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

When the persistent storage mechanism is enabled, the Dlt module does not persistently store any information related to the enabled or disabled state of its interfaces.

Rationale:

The functionality to enable/disable interfaces is not implemented.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00074

- Dlt command "BufferOverflowNotification" is not available

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The AUTOSAR Dlt module does not support the following Dlt Command identified by the following Services ID: ---------- | Service ID | Dlt Command Name | Description | ---------- | 0x23 | BufferOverflowNotification | Indication of a buffer overflow within the DLT module | ----------

Rationale:

The mentioned command is not currently implemented.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00643, SWS_Dlt_00670, SWS_Dlt_00776, SWS_Dlt_00777, SWS_Dlt_00760

- Dlt_GetLogInfo's "logInfo" is now a pointer to uint8

Affected AUTOSAR releases:

- R4.2 rev 1
- R4.3 rev 1

Description:

The Dlt_LogInfoType structure type used for the "logInfo" parameter of the Dlt_GetLogInfo() API is not fully compatible with the function's intended purpose.

Rationale:

The structure type only supports the storage of one AppId/ContextId tuple, which is not fully compatible with the API's intended use: the "appId" and "contextId" parameters can be given as "0", which triggers the API to write multiple entries in the structure type. Due to this issue, the "logInfo" parameter has been changed to a pointer to uint8. This can fully support multiple tuples but comes at a disadvantage because its size must be known beforehand.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00197 AUTOSAR 4.3.1: SWS_Dlt_00732

- Dlt does not respond with "DLT_NOT_SUPPORTED" when receiving deprecated commands

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

When the Dlt module receives any of the following deprecated commands, it does not respond with "DLT_NOT_SUPPORTED": * 0x07 SetComInterfaceStatus * 0x08 SetComInterfaceMaxBandwidth * 0x09 SetVerboseMode * 0x0A SetMessageFiltering * 0x0C GetLocalTime * 0x0D SetUseECUID * 0x0E SetUseSessionID * 0x0F SetUseTimestamp * 0x10 SetUseExtendedHeader * 0x14 MessageBufferOverflow * 0x16 GetComInterfacelStatus * 0x18 GetComInterfaceMaxBandwidth * 0x19 GetVerboseModeStatus * 0x1A GetMessageFilteringStatus * 0x1B GetIseECUID * 0x1C GetUseSessionID * 0x1D GetUseTimestamp * 0x1E GetUseExtendedHeader

Rationale:

The functionality is currently not implemented.

Requirements:

AUTOSAR 4.3.1: PRS_Dlt_00644

- DltGeneralRxDataPathSupport is not used

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The DltGeneralRxDataPathSupport configuration parameter is not used by the Dlt module to accept/reject incoming control messages. Instead, the DltRxPduId parameter is used for this purpose. If it exists and is configured correctly, the Dlt uses a macro to accept/reject incoming control messages.

Rationale:

The functionality is currently not implemented.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00698, SWS_Dlt_00699, ECUC_Dlt_00848

- DltLogChannelBufferOverflowTimer is not used

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

DltLogChannelBufferOverflowTimer's functionality is not supported.

Rationale:

The configuration parameter's functionality is not currently implemented.

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00886, SWS_Dlt_00760

- DltLogChannelTrafficShapingBandwidth is not used

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

DltLogChannelTrafficShapingBandwidth's functionality is not supported.

Rationale:

The configuration parameter's functionality is not currently implemented.

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00883, SWS_Dlt_00758, ECUC_Dlt_00849

- No support for multiple SWC instances on R4.2

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

Dlt module does not support handling of multiple SWC instances on R4.2.

Rationale:

Support for handling multiple SWC instances implies the possibility to register the same AppId/ContextId tuple for different SessionIds. This is not possible when `DltServiceAPI` configuration parameter is set to other value than AUTOSAR_431.

Requirements:

Dlt_Chap7.1.6_Implicit1

- Set/GetDefaultTraceStatus are global and not log channel specific

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The Dlt_Set/GetDefaultTraceStatus() APIs, as well as the Set/GetDefaultTraceStatus control commands/ interface operations do not change the log channel specific default trace status, but change the global default trace status instead.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00744, SWS_Dlt_00745, SWS_Dlt_00747, SWS_Dlt_00748, SWS_Dlt_-00743, SWS_Dlt_00746, SWS_Dlt_00772

- GetLogChannelNames's parameter "logChannelNames" is now of type "Dlt_LogChannelNameArrayType"

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The logChannelNames parameter of the Dlt_GetLogChannelNames() API and of the GetLogChannelNames control command and interface operation is now of type Dlt_LogChannelNameArrayType instead of Dlt_LogChannelNameType. This new array type has its size equal to the number of configured log channels.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00749, SWS_Dlt_00772

- VFB Trace Messages use 0x10000000U as Message ID

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

VFB Trace Messages currently use 0x10000000U as Message ID for the Dlt_SendTraceMessage() calls. However, SWS_Dlt_00031 states that they should use 0x00000003U for Message IDs. This functionality is not planned for the current release.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00031

- Unused DET error codes

Affected AUTOSAR releases:

- R4.2 rev 1
- R4.3 rev 1

Description:

AUTOSAR 4.2.1: Type or error | Relevance | Related error code | Value [hex] ---------- API was unable to | Development | DLT_E_INIT_FAILED | 0x07 initialize the service. | | | ---------- AUTOSAR 4.3.1: Type of error | Related error code | Value [hex] ---------- API service called with wrong parameter | DLT_E_PARAM | 0x01 Initialization failed | DLT_E_INIT_FAILED | 0x03 Registration failed | DLT_E_REGISTRATION | 0x04 ----------
The listed DET error codes are currently not used.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00447 AUTOSAR 4.3.1: SWS_Dlt_00727

- No support for runtime error reporting

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Support for runtime error code reporting is not planned for the current release.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00728

- Dlt_TriggerTransmit() is not a public API

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Dlt_TriggerTransmit() is a static function called only from Dlt_MainFunction() at the moment. Publishing of this API is not planned for the current release.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00754

- Dlt_TxFunction() is not implemented

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The Dlt_TxFunction() API is not implemented and not planned for the current release.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_91005, SWS_Dlt_00760, SWS_Dlt_00761, SWS_Dlt_00673

- DltEcuIdCalloutChoice is not configurable and DltEcuId is not used.

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The possibility to retrieve EcuIds via call-outs using DltEcuIdCalloutChoice is implemented using the current user callout approach, meaning that the name of the callout is not configurable. The Dlt_AppGetEcuIdAddress() user callout is used for EcuId retrieval.

Additionally the container DltEcuId is not used, DltEcuIdValueChoice and DltEcuIdCalloutChoice are placed in the container DltProtocol and named DltEcuId and DltEcuIdCallout respectively.

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00860, ECUC_Dlt_00902, ECUC_Dlt_00862

- RxPdu configuration parameters are not implemented

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

DltIRxPduHandleId, DltIRxPduUsesTp and DltRxPduIdRef are not implemented and not planned for the current release.

Requirements:

AUTOSAR 4.3.1: ECUC_Dlt_00899, ECUC_Dlt_00900, ECUC_Dlt_00912, ECUC_Dlt_00898

- Getter functions not supported under multiple instances of SwCs

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Getter functions for log level and trace status do not currently support multiple instances of SwCs, as AR specifications do not provide argument for session IDs, in order to distinguish identical tuples under different sessions.

Requirements:

Dlt.ASR431.Chap7.1.2_Implicit1

- Several GetSoftwareVersion requirements are not implemented

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The requirements listed below are for the external client and cannot be implemented by the Dlt module.

Rationale:

The Dlt module cannot detect when a conection is established. The Dlt is initialized and then waits until the startup delay timer has elapsed. The messages are sent afterwards until the buffers are empty. The external client is responsible for making sure that the first control message sent is GetSoftwareVersion.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00394, SWS_Dlt_00395, SWS_Dlt_00492

- No support for post-build selectable behavior

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The Dlt module does not support post-build behavior for any parameter.

Requirements:

ECUC_Dlt_00800, ECUC_Dlt_00831, ECUC_Dlt_00802, ECUC_Dlt_00803, ECUC_Dlt_00835, ECUC_Dlt_00805, ECUC_Dlt_00843, ECUC_Dlt_00807, ECUC_Dlt_00806, ECUC_Dlt_00811, ECUC_Dlt_00812, ECUC_Dlt_00813, ECUC_Dlt_00814, ECUC_Dlt_00836, Dlt.ASR431.ECUC_Dlt_00870, Dlt.ASR431.ECUC_Dlt_00871, Dlt.ASR431.ECUC_Dlt_00895, Dlt.ASR431.ECUC_Dlt_00874, Dlt.ASR431.ECUC_Dlt_00874, Dlt.ASR431.ECUC_Dlt_00887, Dlt.ASR431.ECUC_Dlt_00896, Dlt.ASR431.ECUC_Dlt_00888, Dlt.ASR431.ECUC_Dlt_00864, Dlt.ASR431.ECUC_Dlt_00889, Dlt.ASR431.ECUC_Dlt_00886, Dlt.ASR431.ECUC_Dlt_00877, Dlt.ASR431.ECUC_Dlt_00884, Dlt.ASR431.ECUC_Dlt_00878, Dlt.ASR431.ECUC_Dlt_00883, Dlt.ASR431.ECUC_Dlt_00879, Dlt.ASR431.ECUC_Dlt_00893, Dlt.ASR431.ECUC_Dlt_00892, Dlt.ASR431.ECUC_Dlt_00913, Dlt.ASR431.ECUC_Dlt_00856, Dlt.ASR431.ECUC_Dlt_00854, Dlt.ASR431.ECUC_Dlt_00858, Dlt.ASR431.ECUC_Dlt_00859, Dlt.ASR431.ECUC_Dlt_00812, Dlt.ASR431.ECUC_Dlt_00911

- Message counter is increased individually per log channel

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

The message counter in the MCNT field of each Dlt message is currently incremented per log channel instead of globally.

Rationale:

The PRS_Dlt_00105 requirement specifies that a generic message counter shall be implemented, which shall count every log and trace message received via the Dlt API. This somewhat contradicts SWS_Dlt_-00671, which states that a message counter shall be implemented for each specific log channel. Due to this conflict, both requirements cannot be implemented. As such, the behavior stated in SWS_Dlt_00671 has been chosen for implementation.

Requirements:

AUTOSAR 4.3.1: PRS_Dlt_00105

- Size of internal Dlt buffer for message processing is set via a single configuration parameter

Affected AUTOSAR releases:

- R4.2 rev 1

Description:

The size of the internal Dlt buffer used for log and trace messages is determined by a specific chosen parameter.

Rationale:

The SWS_Dlt_00003 requirement specifies that the size of the internal Dlt buffer, used prior to the initialization of the module (and re-used later as normal buffer if desired), shall be determined by the configuration parameter DltInitBufferSize (user definable). The specifications however mention other buffers as well, each with its size determined by a different configuration parameter (for example DltMessageBufferSize, from requirement SWS_Dlt_00342). Because of this, for the sake of memory efficiency a single internal buffer has been implemented for general usage (pre-initialization, runtime, no external client connected), with its size determined by the parameter DltMessageBufferSize, as required by SWS_Dlt_00342. This leaves the DltInitBufferSize parameter unused, thus deviating from the SWS_Dlt_00003 requirement.

Requirements:

AUTOSAR 4.2.1: SWS_Dlt_00003

- Dlt will filter messages with a LogLevel higher than the configured value of the LogChannel threshold or the Log Level of the found ApplicationID/ContextId

Affected AUTOSAR releases:

- R4.3 rev 1

Description:

Log messages with a LogLevel higher than the configured value of log level threshold of the element for which the Dlt_SendLogMessage() API was called. Log messages with a LogLevel higher than the configured value of LogChannel threshold for the identified LogChannel shall be discarded and E_OK shall be returned. This shall be done on each LogChannel from the list of output LogChannels for the LogMessages.

Rationale:

These requirements are wrong (AUTOSAR ticket AR-96503), the logic for filtering messages was reversed when these requirements were introduced with AUTOSAR 4.3.0 SWS. The correct behavior is the one presented in AUTOSAR 4.2.2 SWS: Dlt shall check if the log level of the incoming log message is the same or below as the maximum log level stored for this Context ID - Application ID tuple (the log level of the incoming message shall be in the pass through range). If the check is not successful, the messages shall be discarded, otherwise the message shall be transmitted to the external client.

Requirements:

AUTOSAR 4.3.1: SWS_Dlt_00662, SWS_Dlt_00667

3.3.1.5. Limitations

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

- Limitation of number of session IDs

  Description:

  The number of session IDs which can be registered by SWCs is limited to 2147483647.

  Rationale:

  This is the maximum value supported by the `size()` method of Java's `Lists<>` class, which is needed for evaluating the system description files.

  Requirements:

  SWS_Dlt_225

- AppId/ContextId registration cannot differ only by SessionId

  Description:

When DltServiceAPI is set to "AUTOSAR_421" or "AUTOSAR_422", only one SessionId (port belonging to a SWC) can be registered per tuple of AppId/ContextId.

Rationale:

The current internal data structure does not allow to have the same AppId/ContextId mapped to different SessionIds to have more optimal data handling.

- Limitation of traffic shaping

  Description:

  The minimum limit for traffic shaping is 1kByte/sec.

  Rationale:

  The configuration parameter DltBandwidthForComModule expects the value in kbit/sec, but the limiting of traffic can only be done on whole bytes.

- Limitation of maximum number of contexts and maximum message length

  Description:

  When DltEnableBswDistribution is set to true, the maximum number of possible contexts registered is limited to 4095 and the maximum message length is limited to 8188. These values depend on the architecture on which the stack is configured.

  Rationale:

  When multi-core is enabled, data needs to be exchanged through the IOC module between the satellites and the master to ensure a consistent configuration. The size of the data exchanged through the IOC cannot exceed 65536. To meet this constraint, when BSW Distribution is enabled, maximum values were computed for DltMaxCountContextIds and DltMaxMessageLength. If these parameters are configured with a greater value, the project was risked not to compile as the compilation is stopped in the IOC code.

- Limitation related to the basic software distribution functionality

  Description:

  When DltEnableBswDistribution is set to true, control APIs can only be called from the master context in order to modify the status of the following run-time variables: DltDefaultTraceStatus, DltFilterMessages, DltDefaultMaxLogLevel, DltHeaderUseEcuId, DltHeaderUseExtendedHeader, DltHeaderUseSessionID, DltHeaderUseTimestamp. If these functions are called from the slave context, a negative return value is reported because these APIs cannot be called from the satellite cores.

  Rationale:

Run-time variables are only stored in the master core. Therefore, the necessity to only call the control APIs from the master core is justified.

Description:

If DltImplementFilterMessages and DltFilterMessages are enabled, then DLT filtering functionality is available for both master core and satellite cores. By using the filtering functionality on the slave core, a lot of inter-core communication can be avoided, because each message which does not pass the filter can simply be discarded by the slave and does not need to be forwarded to the master.

Description:

If DltImplementNVRamStorage is enabled, then the persistent storage functionality shall be possible from both master core and slave core. Note that on the satellite core, the slave sends the session ID to the master Dlt instance via SchM send call. On the master core this then triggers a function which receives the session ID of a session from the slave and triggers the Dlt_StorePersistent() API.

Description:

If the satellite core calls Dlt_SendLogMessage()/ Dlt_SendTraceStatus() but the processing of the log message/trace status fails on the master's side, the slave will never be informed about this.

Rationale:

If a Dlt_SendLogMessage()/Dlt_SendTraceStatus() request fails on the master core with a negative return value, the information is not passed back from the master to the slave core.

- Limitation functionality related to some input parameters

Description:

Dlt_RegisterContext() API will not use the input parameters app_description, len_app_description, context_description, len_context_description.

Rationale:

To reduce the size of memory footprint, app_description and context_description is ignored in the current implementation.

Requirements:

SWS_Dlt_00245, ECUC_Dlt_00836

- Reception PDU Ids

Description:

Only one Rx PDU Id can be used for receiving messages.

Rationale:

A maximum multiplicity of 1 for DltRxPdu should be sufficient for most use cases.

- Mapping of error codes for Dlt_SendLogMessage API.

Description:

For Autosar 4.1 and 4.2: If Det is enabled and Dlt_SendLogMessage() is called with an options parameter different from DLT_TYPE_LOG in the log_info structure, the function shall report a DLT_E_WRONG_PARAMETERS error to Det and exit with return value DLT_E_ERROR_UNKNOWN. If Det is enabled and Dlt_SendLogMessage() is called with a structure containing the relevant information for filtering the message (log_info) or the payload of the send message (log_data) as NULL_PTR, the function shall report a DLT_E_PARAM_POINTER error to Det and exit with return value DLT_E_ERROR_UNKNOWN. For Autosar 4.3: If Det is enabled and Dlt_SendLogMessage() is called with an options parameter different from DLT_TYPE_LOG in the log_info structure, the function shall report a DLT_E_WRONG_PARAMETERS error to Det and exit with return value DLT_E_UNKNOWN_SESSION_ID. If Det is enabled and Dlt_SendLogMessage() is called with a structure containing the relevant information for filtering the message (log_info) or the payload of the send message (log_data) as NULL_PTR, the function shall report a DLT_E_PARAM_POINTER error to Det and exit with return value DLT_E_UNKNOWN_SESSION_ID.

Rationale:

In Autosar 4.3 DLT_E_ERROR_UNKNOWN is no longer a valid return value for Dlt_SendLogMessage. DLT_E_UNKNOWN_SESSION_ID is the semantically closest out of all the available return values.

- Mapping of error codes for Dlt_SendTraceMessage API.

Description:

For Autosar 4.1 and 4.2: If Det is enabled and Dlt_SendTraceMessage() is called with an options parameter different from DLT_TYPE_APP_TRACE and DLT_TYPE_NW_TRACE in the trace_info structure, the function shall report a DLT_E_WRONG_PARAMETERS error to Det and exit with return value DLT_E_ERROR_UNKNOWN. If Det is enabled and Dlt_SendTraceMessage() is called with a structure containing the relevant information for filtering the message (trace_info) or the payload of the send message (trace_data) as NULL_PTR, the function shall report a DLT_E_PARAM_POINTER error to Det and exit with return value DLT_E_ERROR_UNKNOWN. For Autosar 4.3: If Det is enabled and Dlt_SendTraceMessage() is called with an options parameter different from DLT_TYPE_APP_TRACE and DLT_TYPE_NW_TRACE in the trace_info structure, the function shall report a DLT_E_WRONG_PARAMETERS error to Det and exit with return value DLT_E_UNKNOWN_SESSION_ID. If Det is enabled and Dlt_SendTraceMessage() is called with a structure containing the relevant information for filtering the message (trace_info) or the payload of the send message (trace_data) as NULL_PTR, the function shall report a DLT_E_PARAM_POINTER error to Det and exit with return value DLT_E_UNKNOWN_SESSION_ID.

Rationale:

In Autosar 4.3 DLT_E_ERROR_UNKNOWN is no longer a valid return code. DLT_E_UNKNOWN_SESSION_ID is the semantically closest out of all the available return values.

- Limitation of setter API functions, control messages and operations

  Description:

  If a setter function is called with a valid tuple, all identical tuples under all SWCs will be updated with the requested change.

  Rationale:

  The Autosar APIs Dlt_SetLogLevel(), Dlt_SetTraceStatus(), and Dlt_SetVerboseMode() control messages (from the DltControlService interface), do not provide a way to distinguish between identical tuples under different SWCs. This makes setting a specific tuple under a particular SWC impossible, in scenarios where there are multiple identical tuples under different SWCs. Because of this, the setter function APIs requested for a specific tuple will affect all identical tuples.

3.3.1.6. Open-source software

`Dlt` does not use open-source software.

4. ACG8 DLT user guide

4.1. Overview

This chapter provides you with `Dlt`-specific information:

- Section 4.2, “Background Information” explains the concepts of the `Dlt` module.
- Section 4.3, “Configuring the Dlt module” provides instructions on how to configure the `Dlt` module.

4.2. Background Information

4.2.1. Dlt core functions

4.2.1.1. Generic creation of log and trace messages

Log and trace messages are created by providing the following information:

- *SessionId*: SessionId of the SWC (e.g. 0x1000U)
- *Log/trace information*: contains AppId/ContextId tuple, message type (e.g. log, trace, control), log level, and trace status
- *Log/trace data*: the actual data for which the message is sent
- *Log/trace data length*: length of the data sent with the message

4.2.1.2. Generic reception of log and trace messages

Log and trace messages are received either by:

- direct API calls of `Dlt_SendLogMessage()` and `Dlt_SendTraceMessage()`
- SWCs that call the SendLogMessage and SendTraceMessage operations (members of the DLTService service interface when `DltServiceAPI` is set to AUTOSAR_421 or AUTOSAR_422 and of the DltSwcMessageService service interface when `DltServiceAPI` is set to AUTOSAR_431).

The Dlt processes, filters, and puts the received messages into an internal buffer. When the next Dlt_MainFunction() cycle executes, the messages received in the previous cycle are sent to the PduR.

4.2.1.3. Buffering and filtering management

The Dlt uses a circular internal buffer to store incoming messages. Multiple messages can be received and stored without immediate transmission to the PduR by a Dlt_MainFunction() call. The number of messages that can be stored depends on the buffer size configured with DltMessageBufferSize. If too many incoming messages are stored in this buffer without an actual transmission, the oldest messages can be lost. In this case, Dlt reports the Det error code DLT_E_MSG_LOOSE if DltDevErrorDetect is enabled.

The Dlt can filter both log and trace messages. The filter options are configured at precompile time and can be modified at run-time. The filtering allows you to change the general module default log level and trace status, or change the tuple-specific log level and trace status. Based on these, log and/or trace messages can be rejected or permitted for transmission.

4.2.1.4. Message filtering based on log level and trace status

The Dlt filters incoming log and trace messages based on their log level and trace statuses. Log messages support the following log levels:

- DLT_LOG_OFF (0x00U)
- DLT_LOG_FATAL (0x01U)
- DLT_LOG_ERROR (0x02U)
- DLT_LOG_WARN (0x03U)
- DLT_LOG_INFO (0x04U)
- DLT_LOG_DEBUG (0x05U)
- DLT_LOG_VERBOSE (0x06U)

DLT_LOG_FATAL (0x01U) is the most severe type of log message, while DLT_LOG_VERBOSE (0x06U) is the least severe type. For example, if the Dlt log level is set to DLT_LOG_WARN (0x03U), the filtering works as follows:

- An incoming log message with DLT_LOG_WARN (0x03U) is accepted and processed.
- An incoming log message with DLT_LOG_FATAL (0x01U) is accepted and processed.
- An incoming log message with DLT_LOG_DEBUG (0x05U) is rejected and Dlt_SendLogMessage() returns E_OK.

Trace messages support the following trace statuses:

- DLT_TRACE_STATUS_OFF (0x00U)
- DLT_TRACE_STATUS_ON (0x01U)

If the `Dlt` trace status is DLT_TRACE_STATUS_OFF, no trace message is sent.

4.2.1.5. Generic handling of control requests

If the `DltRxPduId` configuration parameter is enabled, the `Dlt` can receive control messages. These messages can be sent by a `Dlt` client to change the run-time behavior of the `Dlt`. They consist of changing tuple log level and trace status, getting information about tuples, and modifying log channel behavior. For a complete list of these control messages, together with their description and usage, see Section 4.2.8.1, “Controlling the Dlt via one communication channel by means of control messages”.

4.2.1.6. Parallel transmission of multiple frames within one PDU

The multiple frames transmission feature allows you to transmit multiple `Dlt` frames within one PDU. The feature is enabled with the `DltEnableMultipleFrames` configuration parameter. The number of `Dlt` frames contained in the PDU is defined by the length of the individual `Dlt` frames and the value of the `DltMaxMessageLength` configuration parameter. The maximum number of `Dlt` frames packed into one PDU is based on whether their total size does not exceed the value set for `DltMaxMessageLength`.

The individual frames are simply concatenated within the PDU. The receiver must analyze the length of the whole PDU and compare it to the length parameters in the header fields of the first frame in order to determine whether the PDU contains one or multiple frames. If the PDU contains multiple frames, each subsequent frame is found by using the length parameter of the previous frame as offset.

If data is overwritten and the `DltDevErrorDetect` configuration parameter is enabled, the `Dlt` reports the Det error DLT_E_MSG_LOOSE once, regardless of how many messages are overwritten.

4.2.1.7. Handling of log channels

When the `DltServiceAPI` configuration parameter is set to AUTOSAR_431, the `Dlt` supports the configuration of multiple log channels in the `DltLogChannel` list.

At run-time, the assignment of a registered AppId/ContextId tuple to a log channel can be controlled through the `Dlt_SetLogChannelAssignment()` API. The threshold of a log channel can be set using the API `Dlt_SetLogChannelThreshold()`.

If a tuple does not have any log channel assigned, the configured default log channel is used.

NOTE

The parameter `logChannelNames` that is used in the `Dlt_GetLogChannelNames()` API is of type Dlt_LogChannelNameArrayType instead of Dlt_LogChannelNameType. The size of this array type is equal to the number of configured log channels.

When the `Rte` is used, the RTE_PTR2ARRAYTYPE_PASSING macro shall be used to enforce the use of the array type and not the array base type for the `logChannelName` of the `SetLogChannelAssignment` operation. As such, every inclusion of the `Rte`-generated prototypes (`Rte_Dlt.h`) has RTE_PTR2ARRAYTYPE_PASSING's definition before it.

4.2.2. Dlt service interface according to AUTOSAR 4.2.1 and AUTOSAR 4.3.1

4.2.2.1. AUTOSAR 4.2.1/4.2.2

The following service interfaces are available when the `DltServiceAPI` configuration parameter is set to AUTOSAR_421 or AUTOSAR_422:

Service interface	Defined operation
DLTService	RegisterContext
	StorePersistent
	SendLogMessage
	SendTraceMessage
LogTraceSessionControl	SetLogLevel
	SetTraceStatus
VerboseModeControl	SetVerboseMode

4.2.2.2. AUTOSAR 4.3.1

The following service interfaces are available when the `DltServiceAPI` configuration parameter is set to AUTOSAR_431:

Service interface	Defined operation
DltControlService	GetDefaultLogLevel
	SetDefaultLogLevel

Service interface	Defined operation
	GetDefaultTraceStatus
	SetDefaultTraceStatus
	SetMessageFiltering
	ResetToFactoryDefault
	StoreConfiguration
	SetLogChannelAssignment
	SetLogChannelThreshold
	GetLogChannelThreshold
	GetLogChannelNames
	SetLogLevel
	SetTraceStatus
	GetTraceStatus
	GetLogInfo
DltSwcMessageService	RegisterContext
	UnregisterContext
	SendLogMessage
	SendTraceMessage
LogTraceSessionControl	LogLevelChangedNotification
	TraceStatusChangedNotification

For each SWC, you can enable only the interfaces that the SWC needs with the following parameters:

- `DltProvideLogTraceSessionControlPort`
- `DltProvideSwcMessageServicePort`
- `DltProvideControlServicePort`

Then the `Dlt` generates only the ports, events, and runnable entities relevant for each SWC. The interfaces are generated internally. You do not need to run the **Create Dlt Port Defined Argument Values** wizard.

If the `DltProvideLogTraceSessionControlPort` parameter is enabled, `Dlt` provides the operations `TraceStatusChangedNotification` and `LogLevelChangedNotification` for the `LogTraceSessionControl` interface. Each SWC that wants to receive this notification through the `Rte`, must provide the client/server interface `LogTraceSessionControl`, with the operations `LogLevelChangedNotification` and `TraceStatusChangedNotification`. These operations must be accessible by a P-Port.

If the `DltProvideSwcMessageServicePort` parameter is enabled, `Dlt` provides the following operations for the `DltSwcMessageService` interface:

- `RegisterContext`
- `UnregisterContext`
- `SendLogMessage`
- `SendTraceMessage`

Each SWC that wants to receive this notification through the `Rte` must provide the client/server interface `DltSwcMessageService`, with the operations `LogLevelChangedNotification` and `TraceStatusChangedNotification`. These operations must be accessible by a R-Port.

If the `DltProvideControlServicePort` parameter is enabled, `Dlt` provides the following operations for the `DltControlService` interface:

- `GetDefaultLogLevel`
- `SetDefaultLogLevel`
- `GetDefaultTraceStatus`
- `SetDefaultTraceStatus`
- `SetMessageFiltering`
- `ResetToFactoryDefault`
- `StoreConfiguration`
- `SetLogChannelAssignment`
- `SetLogChannelThreshold`
- `GetLogChannelThreshold`
- `GetLogChannelNames`
- `SetLogLevel`
- `SetTraceStatus`
- `GetTraceStatus`
- `GetLogInfo`

Each SWC that wants to send a control message through the `Rte` must provide the client/server interface `LogTraceSessionControl`, with the same operations as listed above for `DltControlService`. These operations must be accessible by a R-Port.

4.2.2.3. Available APIs

`Dlt` provides the following APIs for all AUTOSAR versions:

- Dlt_SendLogMessage
- Dlt_SendTraceMessage

- Dlt_RegisterContext
- Dlt_GetDefaultLogLevel
- Dlt_GetLogLevel
- Dlt_SetDefaultLogLevel
- Dlt_SetLogLevel
- Dlt_GetDefaultTraceStatus
- Dlt_GetTraceStatus
- Dlt_SetDefaultTraceStatus
- Dlt_SetTraceStatus
- Dlt_GetComInterfaceMaxBandwidth
- Dlt_GetGlobalLogging
- Dlt_GetMessageFilteringStatus
- Dlt_GetUseECUID
- Dlt_GetUseExtendedHeader
- Dlt_GetUseSessionID
- Dlt_GetUseTimestamp
- Dlt_GetVerboseModeStatus
- Dlt_SetComInterfaceMaxBandwidth
- Dlt_SetGlobalLogging
- Dlt_SetUseECUID
- Dlt_SetUseExtendedHeader
- Dlt_SetUseSessionID
- Dlt_SetUseTimestamp
- Dlt_SetVerboseMode
- Dlt_GetLogInfo
- Dlt_SetMessageFiltering

The following API is provided only if the `DltServiceAPI` configuration parameter is set to AUTOSAR_421 or AUTOSAR_422:

- Dlt_StorePersistent

The following APIs are provided only if the `DltServiceAPI` configuration parameter is set to AUTOSAR_431:

- Dlt_UnregisterContext

- Dlt_StoreConfiguration
- Dlt_GetLogChannelNames
- Dlt_GetLogChannelThreshold
- Dlt_SetLogChannelAssignment
- Dlt_SetLogChannelThreshold

4.2.2.4. Endianness of AppId, ContextId and log channel name

When AUTOSAR 4.3 is used (i.e. `DltServiceAPI` is set to AUTOSAR_431), Dlt_ApplicationIDType and Dlt_-ContextIDType are defined as a 4-byte array. The type used for log channel names is Dlt_LogChannelNameArrayType, which is a matrix with DLT_TXPDU_NO rows. Each row holds the 4 bytes of each log channel name. Because of this, the platform endianness is relevant.

The interpretation of application IDs, context IDs and log channel names in interfaces that use them, is done either based on the platform endianness that the `Dlt` runs on, or in MSB-only format, if the parameter `Dlt-TupleEndianness` is enabled.

The following control messages have AppId/ContextId as parameters in the request payload:

- GetLogInfo
- SetLogChannelAssignment
- SetLogLevel
- SetTraceStatus
- GetTraceStatus
- SetVerboseMode
- GetVerboseModeStatus

The following operations have AppId/ContextId as parameters in the request payload:

- RegisterContext
- UnregisterContext
- SendLogMessage
- SendTraceMessage

The following control messages have LogChannelName as parameter in the request payload:

- SetLogChannelAssignment
- SetLogChannelThreshold
- GetLogChannelThreshold

The configuration parameter `DltTupleEndianness` is available under AUTOSAR 4.3.1 (`DltServiceAPI` is set to AUTOSAR_431). This parameter can change the interpretation of SWC tuples when using the APIs provided by the `Dlt` module that use application IDs and context IDs.

The `DltTupleEndianness` parameter has two states - enabled and disabled. By default, the parameter is disabled. In this case, all SWC tuples are interpreted according to the endianness of the platform. If the parameter is enabled, all AppId/ContextId tuples coming from direct API calls as well as DltSwcMessageService and DltControlService operations are interpreted in MSBF format only.

The parameter's state also affects the names of the log channels in the `Dlt` configuration during precompile time. The log channel IDs (`DltLogChannelId`) are interpreted with the same endianness as SWC tuples, based on the value configured in `DltTupleEndianness`.

4.2.2.5. Multiple instances of AppId/ContextId tuples

When the `DltServiceAPI` configuration parameter is set to AUTOSAR_431, the `Dlt` module supports multiple instances of the AppId/ContextId tuples configured under SWCs.

This allows multiple instances of the configured tuples to be accepted and stored internally as separate tuples.

Control messages that are received from a `Dlt` client do not provide the session ID as argument. Therefore, they only affect the first tuple of AppId/ContextId identified.

4.2.2.6. Registering software components and contexts using Dlt_RegisterContext()

SWCs can register applicationId/contextId tuples with the `Dlt` by using the `Dlt_RegisterContext()` API or the RegisterContext operation of the DLTService service interface (when `DltServiceAPI` is set to AUTOSAR_421 or AUTOSAR_422) or of the DltSwcMessageService service interface (when `DltServiceAPI` is set to AUTOSAR_431).

When `DltServiceAPI` is set to AUTOSAR_421 or AUTOSAR_422, any AppId/ContextId tuple can be registered, as the `Dlt` is not aware at run-time of the SWCs that want to register tuples with the module.

When `DltServiceAPI` is set to AUTOSAR_431, only SWCs that explicitly configured their AppId/ContextId tuples with the `Dlt` can register them at run-time.

4.2.2.7. Unregistering software components and contexts using Dlt_UnregisterContext() (only for AUTOSAR 4.3.1)

When `DltServiceAPI` is set to AUTOSAR_431, SWCs that registered AppId/ContextId tuples with the `Dlt` can unregister them at run-time. This provides the possibility for a SWC to completely unregister its AppId/ContextId tuples.

4.2.2.8. Sending log and trace messages from SWCs using Dlt_SendLogMessage() and Dlt_SendTraceMessage()

BSW modules (SessionIDs less than 0x1000U) and SWCs (SessionIDs greater than or equal to 0x1000U) can use the two APIs provided by the `Dlt` in order to send various types of log and/or trace messages.

These messages are subject to filtering, based on the current run-time configuration. If they pass all filters, they are sent to the `PduR` module by using `PduR_DltTransmit()`. This happens on every cycle of the `Dlt_MainFunction()` API.

4.2.2.9. Notifying SWCs about log level and trace status changes

The `Dlt` requires the existence of the LogTraceSessionControl service interface if the SWCs want to receive notifications about log level/trace status changes of AppId/ContextId tuples that are registered with the `Dlt` at run-time.

When `DltServiceAPI` is set to AUTOSAR_421 or AUTOSAR_422, the LogTraceSessionControl service interface provides the operations SetLogLevel and SetTraceStatus.

When `DltServiceAPI` is set to AUTOSAR_431, the LogTraceSessionControl service interface provides the operations LogLevelChangedNotification and TraceStatusChangedNotification. Also, the notify mechanism is only available if SWCs have the `DltSwcSupportLogLevelChangeNotification` configuration parameter enabled.

4.2.2.10. Dlt control by SWCs at run-time (only for AUTOSAR 4.3.1)

When `DltServiceAPI` is set to AUTOSAR_431, the `Dlt` provides the DltControlService service interface with the following operations:

- GetDefaultLogLevel
- SetDefaultLogLevel
- GetDefaultTraceStatus
- SetDefaultTraceStatus
- SetMessageFiltering
- ResetToFactoryDefault
- StoreConfiguration
- SetLogChannelAssignment
- SetLogChannelThreshold
- GetLogChannelThreshold

- GetLogChannelNames
- SetLogLevel
- SetTraceStatus
- GetTraceStatus
- GetLogInfo

This service interface allows SWCs to directly control the `Dlt` run-time behavior. This is not possible in AUTOSAR 4.2 where only control messages can change the behavior and SWCs do not have direct control.

Also, with AUTOSAR 4.3.1, it is possible to have a SWC whose sole purpose is to control the `Dlt` behavior, through the `DltProvideControlServicePort` configuration parameter. For more information about this mechanism see Section 4.2.2.2, “AUTOSAR 4.3.1”.

4.2.3. Virtual function bus tracing

4.2.3.1. Interface for VFB trace

When VFB (Virtual Functional Bus) tracing is enabled, `Dlt` provides a trace hook function for every `Rte` function call that is configured to have trace hook functions enabled.

For every hook function, `Dlt` assigns a unique Message ID, starting from value 0x10000000U and incrementing it for every new generated function.

The trace data of the trace message that is sent for every hook function is composed of:

- Message ID - 4 bytes, the value is written considering the MSBF bit of the HTYP field of the standard header
- Value of every function parameter - for each parameter, the size of its data type is copied, if `DltVfbSendHookFunctionParameters` is enabled

NOTE **No size for pointer data types**

When the parameter of a function is of type pointer (e.g. an array), `Dlt` does not know the size of the data that must be copied. So it copies only the address.

NOTE **No tracing benefit for start function**

For the output parameters of a function, `Dlt` copies the value provided by the parameter. So this data only makes sense for a return function, not for the start function.

VFB trace messages can be transmitted in two ways, based on the state of the parameter `DltVfbMainFunctionPeriod`:

- `DltVfbMainFunctionPeriod` is disabled: VFB trace messages are transmitted when the hook function is called.
- `DltVfbMainFunctionPeriod` is enabled: VFB trace messages are transmitted via the VFB MainFunction, with the periodicity configured in the parameter.

NOTE **Impact of a too small buffer**

It is possible that some of the VFB trace messages are not sent, if the size of the VFB buffer calculated based on the configured number of interruptions and the maximum size of a VFB message is not large enough for holding the new messages. Then this message is discarded. In this case, `Dlt` reports a production error.

NOTE **Buffer overflow notification**

If the transmission buffer is not large enough to hold all messages before sending them, it is possible that some of the VFB trace messages are overwritten, based on the behavior described by the AUTOSAR requirements, which request that the newest message replaces the older ones. In this case, `Dlt` reports a buffer overflow development error. It is possible that the first VFB trace messages are not sent.

4.2.3.2. Support for generating and implementing RTE hook functions

When all configuration conditions are met, the `Rte` module will generate hook functions.

If one hook function is configured in the `Rte` for tracing, the generated `Rte.c` file will include calls to several hook functions defined in the `Dlt`.

In each of the hook functions, the `Dlt` calls the `Dlt_SendTraceMessage()` API in order to send the traces to the `PduR`.

For VFB configuration steps, see Section 4.3.4, “Configuring VFB tracing”.

4.2.4. Persistent storage of configuration

4.2.4.1. Storing the run-time configuration in non-volatile (NV) memory

The persistent storage functionality allows you to modify the run-time configuration and ensure that this configuration is recovered after an ECU reset.

To guarantee that the configuration is restored at every start-up, the `Dlt` provides a persistent storage in NVRAM to store general configuration data and explicitly set (by an external client) log level, trace status, and verbose mode for a tuple of application ID and context ID.

When the DltServiceAPI configuration parameter is set to AUTOSAR_421 or AUTOSAR_422, the API Dlt_StorePersistent() is used for the persistent storage service.

This service, unlike those listed in Section 4.2.4.1.3, “Dlt_ApplGetConfigFactoryDefault() call-out”, must be assigned to a port interface in the Rte, i.e. to the DLTService operation StorePersistent.

When the DltServiceAPI configuration parameter is set to AUTOSAR_431, the operation StoreConfiguration of the interface DltControlService or the API Dlt_StoreConfiguration() are used for the persistent storage service. For this AUTOSAR version, the log channel assignment is added to the stored data.

The Dlt uses two types of non-volatile (NV) blocks:

- An NvM block of type DATASET is used for storing log level and trace status for a registered tuple of application ID and context ID.

Start address	Dataset NvM Block							
	ApplicationId (uint32)	ContextId (uint32)	Log Level (uint8)	Trace Status (uint8)	SessionId (uint32)	NewData + OsCoreId (uint8)	Verbose Mode (uint8)	Log Channel Assignments (uint8)
0x00U, block length = 16 bytes + no. of configured log channels	APP0	CTX0	4	ON	0x00U	0U	ON	0U, 1U, 1U...
Next Address, block length = 16 bytes + no. of configured log channels	APP1	CTX1	1	OFF	0x1000U	7U	OFF	0U, 0U, 1U...
Next Address, block length = 16 bytes + no. of configured log channels	APP2	CTX2	4	OFF	0x00U	14U	ON	1U, 1U, 0U...
The number of rows for the Dataset NvM block is equal to the number of registered pairs of AppId/ContextId via Dlt_RegisterContext()					SWC SessionId range: >= 0x1000U BSW SessionId range: < 0x1000U	NewData is stored on the first bit. When DltServiceAPI == AUTOSAR_431, OsCoreId is stored on the next 4 bits. This allows up to 16 core IDs to be configured .		Number of bytes occupied by the log channel assignments = Number of configured log channels, when DltServiceAPI == AUTOSAR_431.
Current Address = (previous address of APPN-2) + block length (16 bytes min.)	APPN-1	CTX8	5	ON	0x1001U	1U	OFF	1U, 1U, 1U...
Current Address = (previous address of APPN-1) + block length (16 bytes min.)	APPN	CTX9	5	OFF	0x1002U	1U	ON	1U, 0U, 1U...

Figure 4.1. Dlt's mapping of information persistently stored in a dataset block

The fields OsCoreId and Log channel assignment are stored only if DltServiceAPI is set to AUTOSAR_431.

- An NvM block of type NATIVE is used for storing general configuration parameters.

Start address	**Native NvM block**
0x00U, block length = 1 byte	DltFilterMessages (uint8)
0x01U, block length = 1 byte	DltDefaultMaxLogLevel (uint8)
0x02U, block length = 1 byte	DltHeaderUseTimestamp (uint8)
0x03U, block length = 1 byte	DltHeaderUseEcuId (uint8)
0x04U, block length = 1 byte	DltHeaderUseExtendedHeader (uint8)

Start address	Native NvM block
0x05U, block length = 1 byte	DltHeaderUseSessionId (uint8)
0x06U, block length = 1 byte	DltHeaderUseVerboseMode (uint8)
0x07U, block length = 1 byte Not supported for current implementation. Space reserved.	DltVfbTraceLogLevel (uint8)
0x08U, block length = 4 bytes	DltBandWidthForComModule (uint8)
0x0CU, block length = 1 byte	Empty byte
0x0DU, block length = 1 byte	DltDefaultTraceStatus (uint8)
0x0EU, block length = 1 byte	Dlt_GlobalLogStatus (uint8)
0x0FU - 0xYYU, variable block length	Dlt_LogChannelThresholdInfo (uint8)

Table 4.1. `Dlt's` mapping of information persistently stored in a native block

The field Dlt_LogChannelThresholdInfo is occupied only if `DltServiceAPI` is set to AUTOSAR_431.

For both blocks, the explicit synchronization mechanism (`NvMBlockUseSyncMechanism`) is automatically enabled. The configuration shown below is automatically generated by the Service Needs Calculator:

Dlt block name: NVM_BLOCK_DLT_GENERAL

- NvMBlockManagementType: NVM_BLOCK_NATIVE
- NvMBlockCrcType: NVM_CRC16
- NvMNvBlockLength: 0
- NvMCalcRamBlockCrc: false
- NvMBlockWriteProt: false
- NvMResistantToChangedSw: false
- NvMSelectBlockForReadAll: false
- NvMWriteBlockOnce: false
- NvMBlockJobPriority: 0
- NvMBlockUseSyncMechanism: true
- NvMInitBlockCallback: Dlt_NvMInitNativeBlockCbk
- NvMReadRamBlockFromNvCallback: Dlt_NvMReadRamBlockFromNvMNativeCbk
- NvMSingleBlockCallback: Dlt_NvMSingleBlockCallbackNative
- NvMWriteRamBlockToNvCallback: Dlt_NvMWriteRamBlockToNvMNativeCbk

Dlt block name: NVM_BLOCK_DLT_DATASET

- NvMBlockManagementType: NVM_BLOCK_DATASET
- NvMBlockCrcType: NVM_CRC16
- NvMNvBlockLength: 0
- NvMCalcRamBlockCrc: false
- NvMBlockWriteProt: false
- NvMResistantToChangedSw: false
- NvMSelectBlockForReadAll: false
- NvMWriteBlockOnce: false
- NvMBlockJobPriority: 1
- NvMBlockUseSyncMechanism: true
- NvMInitBlockCallback: Dlt_NvMInitDataSetBlockCbk
- NvMReadRamBlockFromNvCallback: Dlt_NvMReadRamBlockFromNvMDataSetCbk
- NvMSingleBlockCallback: Dlt_NvMSingleBlockCallbackDataSet
- NvMWriteRamBlockToNvCallback: Dlt_NvMWriteRamBlockToNvMDataSetCbk

NOTE

Variable block length value

The configured parameter `length` is set to 0U so that you are not limited to a specific value of the non-volatile block length. Rationale: The block length is variable and depends on the platform configuration.

4.2.4.1.1. Length of NvM blocks

The minimum value for the parameter `length` is defined as follows:

Block type	**`DltServiceAPI` set to**	**Minimum block length**
Native	AUTOSAR_421 or AUTOSAR_422	15U
	AUTOSAR_431	Depends on the number of channels configured: - at least 17U if only one channel is configured - 2 bytes for every additional channel
Dataset	AUTOSAR_421 or AUTOSAR_422	16U
	AUTOSAR_431	Depends on the number of channels configured:

Block type	DltServiceAPI set to	Minimum block length
		▶ at least 17U if only one channel is configured ▶ 1 byte for every additional channel

4.2.4.1.2. Number of NvM blocks

The number of non-volatile blocks is configured with the NvmNvBlockNum parameter and depends on the NvM block type.

- Native block type
  This block type can contain only one non-volatile block. The NvMNvBlockNum parameter must be set to 1.
- Dataset block type
  The value of NvMNvBlockNum should be set so that all expected tuples of application IDs/context IDs can be safely stored in the non-volatile memory.

  If DltServiceAPI is set to AUTOSAR_421 or AUTOSAR_422, configure NvMNvBlockNum to the same value as DltMaxCountContextIds.

NOTE **Number of blocks per dataset and number of dataset blocks**

The maximum number of dataset blocks is configured via the NvMDatasetSelectionBits parameter. Ensure that this maximum number is at least equal to DltMaxCountContextIds. This allows you to configure, in each dataset block created, the number of blocks to be less or equal to the maximum number of available datasets.

If DltServiceAPI is set to AUTOSAR_431, the parameter DltMaxCountContextIds is not available. So no check is performed in the configuration.

4.2.4.1.3. Dlt_ApplGetConfigFactoryDefault() call-out

The call-out Dlt_ApplGetConfigFactoryDefault() is a user-defined callout. Its purpose is to pass a set of parameters needed in order to re-configure the registered context to the factory default values. As specified, only the log level of a uniquely defined tuple of application ID and context ID given by the user shall be set to DltFactoryDefaultMaxLogLevel. Thus, no value is requested for it. The rest of the parameter values are to be decided by the user. A data set index has to be uniquely defined. This data set index is given in order to trigger the initialization callback, which registers the information provided in the register context tables.

In other words, the information passed via Dlt_ApplGetConfigFactoryDefault() is considered as a request for a reset to the factory default values. A suggestion on the implementation itself would be to define

an array with wanted values for each parameter that needs to be configured (all five input parameters of the callout have to be configured with values), and at each data set index a different context will be registered.

4.2.5. Dlt assistant (tooling)

4.2.5.1. Support for generating Dlt ports according to service needs

NOTE

Only for AUTOSAR 4.2.1 and 4.2.2

This feature is available only if the `DltServiceAPI` configuration parameter is set to AUTOSAR_421 or AUTOSAR_422.

The `Dlt` module configuration does not contain any information about the SWCs that use the `Dlt`. Thus, the ports that are required for communication cannot be created based on the module configuration.

The **Create Dlt Port Defined Argument Values** unattended wizard (`DltAs`) uses the software component descriptions (SWCDs) to obtain the necessary information. The wizard checks the <INTERNAL-BEHAVIOR> containers for <SERVICE-NEEDS> items. If they are of type <DLT-USER-NEEDS>, the wizard retrieves all assigned ports from the corresponding <ASSIGNED-PORTS> containers.

For advice on how to describe the `Dlt` service needs in the SWCD, see Section 4.3.3, “Configuring the Dlt service needs”.

4.2.6. Message transmission

4.2.6.1. Data transmission via a network-specific communication channel

The `Dlt` is located above the `PduR`. This guarantees that `Dlt` is independent from the network. It allows the `Dlt` to transmit communication data via any specific communication channel of a specific network (e.g. CAN, FlexRay) using underlying communication layers.

4.2.6.2. Segmentation/adaptation of log or trace messages

If a transport protocol (TP) is used, the `Dlt` can receive and/or transmit messages that are adapted/segmented according to the needs of the communication channel.

4.2.6.3. Delay of message transmission

When `DltServiceAPI` is set to AUTOSAR_431, the `DltGeneralStartUpDelayTimer` configuration parameter is available for configuring. After the `Dlt` initializes, this parameter introduces a delay of seconds in message transmission.

4.2.7. Verbose mode

4.2.7.1. Transmitting messages in verbose mode

The `Dlt` allows the transmission of messages in verbose mode, which provides full description of the data. Verbose mode requires the presence of the extended header in the messages.

4.2.7.2. Support of a configuration switch to enable/disable verbose mode

The `Dlt` uses the following parameters to control verbose mode:

- DltImplementVerboseMode
- DltUseVerboseMode
- DltImplementExtendedHeader
- DltHeaderUseExtendedHeader

4.2.7.3. Building Dlt messages with extended headers [SWS_Dlt_00457]

When the `DltImplementExtendedHeader` and `DltHeaderUseExtendedHeader` configuration parameters are enabled, all log and trace messages have the extended header embedded in them. The following extra fields are present in the message header:

- MSIN (Message Info): verbose mode (enabled/disabled), message type, message type info
- NOAR (Number of Arguments): number of arguments that the traced function has
- APID (Application ID): the AppId of the log/trace message
- CTID (Context ID): the ContextId of the log/trace message

These fields add an extra 10 bytes to the total size of a message.

4.2.8. Control messages

4.2.8.1. Controlling the Dlt via one communication channel by means of control messages

The `Dlt` module offers two ways to receive control requests, e.g. for setting the log level:

- via dedicated C APIs
  C API control messages are always available.
- via communication services as specified by AUTOSAR
  Control messages via communication services are enabled with the `DltRxPduId` configuration parameter.

4.2.8.1.1. Dlt control messages via C APIs

If `DltRxPduId` is disabled, the control messages can only be received via C API calls that are provided via the `Dlt_Control.h` header.

The following table shows the mappings between the AUTOSAR 4.2.1 control messages and the implemented C APIs.

AUTOSAR 4.2.1 control message	**`Dlt` API function**	**Comment**
Set_LogLevel	*Dlt_SetLogLevel()*	
Set_DefaultLogLevel	*Dlt_SetDefaultLogLevel()*	
Set_TraceStatus	*Dlt_SetTraceStatus()*	
Set_DefaultTraceStatus	*Dlt_SetDefaultTraceStatus()*	
Get_LogInfo, option 3	not supported	
Get_LogInfo, option 4	*Dlt_GetLogLevel()*	for one AppId/ContextId tuple at a time
Get_LogInfo, option 5	*Dlt_GetTraceStatus()*	for one AppId/ContextId tuple at a time
Get_LogInfo, option 6	*Dlt_GetLogInfo()*	
Get_LogInfo, option 7	*Dlt_GetLogInfo()*	
Get_DefaultLogLevel	*Dlt_GetDefaultLogLevel()*	
Get_DefaultTraceStatus	*Dlt_GetDefaultTraceStatus()*	
Store_Config	*Dlt_StorePersistent()*	
ResetToFactoryDefault	not supported	

AUTOSAR 4.2.1 control message	**`Dlt` API function**	**Comment**
SetComInterfaceStatus	not supported	
GetComInterfaceStatus	not supported	
GetComInterfaceNames	not supported	
SetComInterfaceMaxBandwidth	*Dlt_SetComInterfaceMaxBandwidth()*	
GetComInterfaceMaxBandwidth	*Dlt_GetComInterfaceMaxBandwidth()*	
SetVerboseMode	*Dlt_SetVerboseMode()*	
GetVerboseModeStatus	*Dlt_GetVerboseModeStatus()*	
SetMessageFilterering	*Dlt_SetMessageFiltering()*	
GetMessageFiltereringStatus	*Dlt_GetMessageFilteringStatus()*	
SetTimingPackets	not supported	
GetLocalTime	not supported	
SetUseECUID	*Dlt_SetUseECUID()*	
GetUseECUID	*Dlt_GetUseECUID()*	
SetUseSessionID	*Dlt_SetUseSessionID()*	
GetUseSessionID	*Dlt_GetUseSessionID()*	
UseTimestamp	*Dlt_SetUseTimestamp()*	
GetUseTimestamp	*Dlt_GetUseTimestamp()*	
UseExtendedHeader	*Dlt_SetUseExtendedHeader()*	
GetUseExtendedHeader	*Dlt_GetUseExtendedHeader()*	
CallSW-CInjection	not supported	
GetSoftwareVersion	*Dlt_GetSoftwareVersion - User-Callout*	
MessageBufferOverflow	not supported	

Table 4.2. Control messages conversion to C APIs, AUTOSAR 4.2.1

The following table shows the mappings between the AUTOSAR 4.3.1 control messages and the implemented C APIs.

AUTOSAR 4.3.1 control message	**`Dlt` API function**	**Comment**
SetLogLevel	*Dlt_SetLogLevel()*	

AUTOSAR 4.3.1 control message	**`Dlt` API function**	**Comment**
SetTraceStatus	*Dlt_SetTraceStatus()*	
GetLogInfo	*Dlt_GetLogInfo()*	
GetDefaultLogLevel	*Dlt_GetDefaultLogLevel()*	
StoreConfiguration	*Dlt_StoreConfiguration()*	
RestoreToFactoryDefault	*Dlt_ResetToFactoryDefault()*	
SetMessageFiltering	*Dlt_SetMessageFiltering()*	
SetDefaultLogLevel	*Dlt_SetDefaultLogLevel()*	
SetDefaultTraceStatus	*Dlt_SetDefaultTraceStatus()*	
GetSoftwareVersion	*Dlt_GetSoftwareVersion()*	
GetLogChannelNames	*Dlt_GetLogChannelNames()*	
GetTraceStatus	*Dlt_GetTraceStatus()*	
SetLogChannelAssignment	*Dlt_SetLogChannelAssignment()*	
SetLogChannelThreshold	*Dlt_SetLogChannelThreshold()*	
GetLogChannelThreshold	*Dlt_GetLogChannelThreshold()*	
BufferOverflowNotification	not supported	
SyncTimeStamp	not supported	

Table 4.3. Control messages conversion to C APIs, AUTOSAR 4.3.1

4.2.8.1.1.1. Dlt_GetSoftwareVersion() call-out

`Dlt_GetSoftwareVersion()` is a user-defined call-out. Its purpose is to write a string representing the ECU software version to a buffer passed to it via one of its parameters. The other parameter is the maximum length of the software version string. You must check that the written string does not exceed the maximum length.

4.2.8.1.1.2. Dlt_AppGetEcuIdAddress() call-out

Dlt_AppGetEcuIdAddress() is an user-defined call-out that can be used when the "DltEcuIdChoice" configuration parameter is set to "Callout". It is called upon module initialization and it retrieves the address of a buffer that holds an user-defined EcuId. This EcuId is used by log and/or trace messages to compose the ECU field in the standard header. It is up to the user to make sure that the EcuId string does not exceed the maximum length (4 bytes). If the call-out retrieves a NULL pointer, the Dlt module will report the DLT_E_PARAM_POINTER error code to the Det, with a 0x89U ServiceId. If the EcuId cannot be retrieved, the corresponding bytes will be filled with 0x00U.

4.2.8.1.1.3. Dlt_GetLogInfo() buffer size

The `Dlt_GetLogInfo()` API takes the following arguments:

- *options*: This can either be `6U` for non-textual information, or `7U` for textual information about the given AppId/ContextId tuple(s). Any other value is considered invalid and the function returns with E_NOT_OK.
- *appId*: This can either be an already registered application ID or the value `0U`. With `0U`, all registered application IDs are retrieved.
- *contextId*: This can either be a valid context ID of an already registered application ID, or the value `0U`. With `0U`, if a valid application ID is given, all context IDs of the given application ID are retrieved.
- *status*: This returns the API status, which is separate from its return values.
- *logInfo*: This returns information about one or multiple AppId/ContextId tuples. This parameter is a buffer that needs its size well defined before the API can be called. The size is calculated based on the following formula:

  Size when non-textual descriptions are requested:
  *BufferSize = 2 bytes + (6 bytes * TotalNoOfAppIds) + (6 bytes * TotalNoOfContextIds)*

  Note: The buffer does not contain any information related to textual descriptions when they are not requested (i.e. the *options* argument equals `6U`).

  Size when textual descriptions are requested:
  *BufferSize = 2 bytes + ((7 bytes * TotalNoOfAppIds) + TotalDescLengthOfAppIds) + ((7 bytes * TotalNoOfContextIds) + TotalDescLengthOfContextIds)*

  - *TotalNoOfAppIds*: total number of requested application IDs. Its value can either be `1U` when a specific valid application ID is given. Or it is equal to the total number of registered application IDs when the *appId* argument is given as `0U`.
  - *TotalDescLengthOfAppIds*: total description length of all application IDs
  - *TotalNoOfContextIds*: total number of requested context IDs. Its value is based on the total number of registered context IDs across all given application IDs.
  - *TotalDescLengthOfContextIds*: total description length of all context IDs.

For a valid call of the API, the exact number of registered AppId/ContextId tuples must be known. If the given application ID is different from `0U` and it was previously registered, only the number of context IDs must be known. The first 2 bytes are the number of application IDs requested, while the rest of the buffer size is based on the total number of registered AppId/ContextId tuples.

Note: When the `DltServiceAPI` configuration parameter is set to AUTOSAR_421 or AUTOSAR_422, the `Dlt_GetLogInfo()` API is not capable of retrieving textual descriptions because they are not supported by `Dlt_RegisterContext()`. In this case, *TotalDescLengthOfAppIds* and *TotalDescLengthOfContextIds* from the buffer size calculation are not applicable. The description length for application IDs and context IDs is always written as `0U`.

4.2.8.1.2. Dlt control messages via communication services

If `DltRxPduId` is enabled, the control messages can also be received via the communication services, i.e. `PduR`.

When the `DltServiceAPI` configuration parameter is set to AUTOSAR_431, the `Dlt` supports the configuration of multiple log channels, for which a `DltITxPduHandleId` and a `DltTxPduIdRef` must be configured. The handle IDs for log channels are zero-based and consecutive.

PDUs are configured in the `EcuC` module and are referenced in the `DltTxPduIdRef` parameter of a log channel.

`Dlt` always responds to a received control message with the configured standard header and the extended header. The extended header is sent regardless of the `Dlt_HeaderUseExtendedHeader` value. The ECU ID is sent regardless of the `Dlt_HeaderUseEcuId` value.

If there is a validation error of the received message, e.g. a wrong ECU ID, the length is lower than expected, or the extended header is not sent, `Dlt` responds with a message that has the status ERROR and service ID = 0 in the payload.

If the underlying communication layer is Ethernet, the following limitations apply:

- `Dlt_Init()` must be called after `SoAd_Init()`.
- The `SoAdSocketAutomaticSoConSetup` parameter must be set to false. `Dlt_Init()` opens the connection itself.
- The `SoAdPduHeaderEnable` parameter must be set to false. The PDU header is not described in the `Dlt` protocol, and a `Dlt`-external client does not add it.
- If `UDP` is used, the `SoAdSocketUdpRetryEnabled` parameter must be set to true.

When a control message is sent via `PduR`, the message is stored in the internal reception buffer. The message is processed during the `Dlt_MainFunction()` call. The `Dlt` responds via a `PduR_DltTransmit()` call.

4.2.9. Support for synchronized time-base

4.2.9.1. Support of synchronized time stamps/chronological order of log and trace data

This feature is only available when `DltServiceAPI` is set to AUTOSAR_431.

If the configuration parameter `DltImplementTimestamp` is enabled and `DltGeneralStbMTimeBaseRef` is configured, time stamps are provided by the `StbM_GetCurrentTime()` API. Thus, this requires the existence of a `StbM` configuration in the current project. The **StbMSynchronizedTimeBase** container needs to have

an entry that enables the **StbMLocalTimeClock** container. This container needs a hardware counter (Rte_-Counter) mapped to `StbMLocalTimeHardware`.

If the configuration parameter `DltImplementTimestamp` is enabled and `DltGeneralStbMTimeBaseRef` is not configured, time stamps are provided by the `OS_GetTimeStamp()` API. Thus, this requires the existence of an `Os` in the current project. An alarm needs to be configured in the `Os` (OsAlarm) with a hardware counter mapped to it, as an alarm counter reference. Therefore, if the time stamp functionality in `Dlt` is wanted, the current project needs to integrate the `Os`. Otherwise, if `DltImplementTimestamp` is enabled but no `Os` is available, the project fails to compile.

4.2.10. Support for BSW distribution

For multi-core support, a *master-satellite* concept is implemented.

Each core that is able to register AppId/ContextId tuples and send `Dlt` messages gets its own instance of a `Dlt` satellite.

4.2.10.1. Supported APIs

The following APIs are requested and processed only on the master core:

- Dlt_ComTxConfirmation()
- Dlt_ComCopyTxData()
- Dlt_GetTraceStatus()
- Dlt_SetLogLevel()
- Dlt_SetTraceStatus()
- Dlt_GetLogLevel()
- Dlt_GetLogInfo()
- Dlt_GetComInterfaceMaxBandwidth()
- Dlt_SetComInterfaceMaxBandwidth()
- Dlt_GetUseExtendedHeader()
- Dlt_SetUseExtendedHeader()
- Dlt_GetUseTimestamp()
- Dlt_SetUseTimestamp()
- Dlt_GetDefaultLogLevel()
- Dlt_GetDefaultTraceStatus()

- Dlt_SetDefaultLogLevel()
- Dlt_SetDefaultTraceStatus()
- Dlt_SetMessageFiltering()
- Dlt_GetGlobalLogging()
- Dlt_GetMessageFilteringStatus()
- Dlt_GetUseECUID()
- Dlt_GetUseSessionID()
- Dlt_GetVerboseModeStatus()
- Dlt_SetGlobalLogging()
- Dlt_SetUseECUID()
- Dlt_SetUseSessionID()
- Dlt_SetVerboseMode()
- Dlt_GetLogChannelNames()
- Dlt_GetLogChannelThreshold()
- Dlt_SetLogChannelAssignment()
- Dlt_SetLogChannelThreshold()
- Dlt_ResetToFactoryDefault()
- Dlt_NvMWriteRamBlockToNvMDataSetCbk()
- Dlt_NvMWriteRamBlockToNvMNativeCbk()
- Dlt_NvMInitDataSetBlockCbk()
- Dlt_NvMInitNativeBlockCbk()
- Dlt_NvMReadRamBlockFromNvMDataSetCbk()
- Dlt_NvMReadRamBlockFromNvMNativeCbk()
- Dlt_NvMSingleBlockCallbackDataSet()
- Dlt_NvMSingleBlockCallbackNative()

The following APIs are requested on a satellite core, but processed on the master core:

- Dlt_StorePersistent() - only if `DltServiceAPI` is set to AUTOSAR_421 or AUTOSAR_422
- Dlt_StoreConfiguration() - only if `DltServiceAPI` is set to AUTOSAR_431

The following APIs can be requested on both master core and satellite cores:

- Dlt_SendLogMessage()
- Dlt_SendTraceMessage()

- Dlt_RegisterContext()
- Dlt_UnregisterContext() - only if `DltServiceAPI` is set to AUTOSAR_431

NOTE

AUTOSAR_431: Use core where tuple is registered

When AUTOSAR_431 is selected, the tables containing information about the registered tuples are updated locally, on the core where `Dlt_RegisterContext()` is called. All API calls for a register tuple should always be made on the same core, for BSW applications, because the information from the registration is only available on that core.

NOTE

No SWC cross-core calls

Cross-core calls are not supported for SWCs.

The validation of the data in the `Dlt_SendLogMessage()` and `Dlt_SendTraceMessage()` APIs is done locally, on the core on which they were called. But the transmission of the log/trace messages is always done on the master core, regardless of the core on which the API was called.

4.2.10.2. BSW distribution initialization

If `DltEnableBswDistribution` is set to true, BSW distribution support is enabled. To ensure a consistent initialization state for both master and satellite cores, the following sequence must be completed during the start-up phase of the project:

1. Initialize all `Dlt` satellite cores.
2. Initialize the `Dlt` master core.

This sequence protects the project against inconsistent data storage in the system and invalid run-time data exchange between the cores.

For configuring BSW distribution, see Section 4.3.5, “Configuring BSW distribution”.

4.2.11. Traffic shaper

4.2.11.1. Support of bandwidth management

Bandwidth management is defined as traffic shaping in the `Dlt` implementation. According to the specifications, a *sliding time window* is to be used for this purpose.

The `Dlt_MainFunction()` API uses pre-configured time periods during which it can measure traffic and bandwidth exhaustion. These time windows also enforce a specific number of bytes that can be transmitted.

4.2.11.2. Configurable bandwidth limits per communication channel

The maximum number of bytes that can be transmitted within the time window is calculated at precompile time, based on the bandwidth configured for the communication channel multiplied by the configured traffic shaping time period. This is divided by 1000 to obtain the maximum bytes allowed.

4.2.11.3. Delaying of message transmission if bandwidth is exhausted

If more than the maximum number of bytes is sent and awaiting transmission, the `Dlt_MainFunction()` calculates a certain number of *wait states*, considering the amount of data in excess. These wait states are used by the `Dlt_MainFunction()` to delay message transmission in order to balance the average bus load so the `Dlt` can meet the bandwidth limits.

4.3. Configuring the Dlt module

4.3.1. Configuring the Rte for AUTOSAR 4.2.1 and 4.2.2

Each software component (SWC) that shall be able to communicate with the `Dlt` must provide a `LogTraceSessionControl` client/server interface with the operations `Dlt_SetLogLevel()` and `Dlt_SetTraceStatus()`. These operations must be accessible by a P-Port that has a `Port Defined Argument Value` of type `Dlt_SessionIDType`. The port defined argument values must be consecutive and start from 0x1000.

If the `DltImplementVerboseMode` parameter is set to `true`, each SWC must also provide a `VerboseModeControl` client/server interface with the operation `Dlt_SetVerboseMode()`. This operation must be accessible by a P-Port that has a `Port Defined Argument Value` of type `Dlt_SessionIDType`.

Additionally, each SWC that shall be able to communicate with the `Dlt` must offer the following R-Ports with the `DLTService` interface assigned to *required interface*:

- `R_StorePersistent`
- `R_RegisterContext`

- `R_SendLogMessage`
- `R_SendTraceMessage`

To make the `Dlt` aware of the port defined argument values, run the **Create Dlt Port Defined Argument Values** wizard. Follow the sequence described here:

Mapping the SWC and Dlt ports

Step 1
Run the **generate_swcd** option from the **Generate code for the currently selected projects** button.

Step 2
Run the *System Description Importer* to import all the system descriptions generated previously.

Step 3
Run the **Create Dlt Port Defined Argument Values** wizard.

Step 4
Open the **Compositions and Connections** editor.

Step 5
Map the R-Ports `R_StorePersistent`, `R_RegisterContext`, `R_SendLogMessage` and `R_SendTraceMessage` of the SWCs to the corresponding `Dlt_service_gen` P-Ports of the `Dlt` module.

Step 6
Map the `LogTraceSessionControl` and, if `DltImplementVerboseMode` parameter is set to `true`, the `VerboseModeControl` P-Ports of the software components to the corresponding `PSCN` R-Port of the `Dlt` module.

Step 7
Run the **Create an ECU Extract** wizard.

Now you can configure the `Rte`.

Configuring the Rte

Step 1
In the `Os`, create a task for the runnable entities

- RE_SetLogLevel
- RE_SetTraceStatus
- RE_SetVerboseMode

Step 2
Set the task priority to a value that is greater than the priority of the task that the software cyclic counter is mapped to. In this way, the three runnable entities are called before the cyclic counter is incremented.

Step 3
In the `Rte`, map the three runnable entities to their associated task that you created in the `Os`.

Step 4
Map the RE_SetLogLevel, RE_SetTraceStatus and RE_SetVerboseMode runnable entities to internal trigger events.

For detailed information about the integration with the `Rte`, see the following documents:

- Specification of Diagnostic Log and Trace, AUTOSAR 4.2.1., chapter 7.6.3
- Specification of Diagnostic Log and Trace, AUTOSAR 4.3.1., chapter 7.1.2

4.3.2. Configuring Dlt for AUTOSAR 4.3.1

You can configure the AUTOSAR version to be used by setting the parameter `DltServiceAPI`. When using AUTOSAR 4.3, you must configure the SWCs in the `Dlt` (`DltSwc`). You must also configure the contexts you want to use for a specific SWC (`DltSwcContext`).

Configuring the SWC context

Step 1
Open the `Dlt` editor on the `DltSwcContext` tab.

Step 2
Configure a list of ApplicationId/ContextId tuples that are supported by this SWC:

Step 2.1
In `DltSwcApplicationId`, set the SWC application ID.

Step 2.2
In `DltSwcContextId`, set the context ID.

Step 3
Go to the `General` tab.

Step 4
In `DltSwcSessionId`, set the session ID.

If you enter a value that is smaller than 0x1000, the configured SWC is considered to be a BSW module. In this case, the parameter `DltSwcSupportLogLevelChangeNotification` is disabled because BSW modules cannot be notified about log and trace changes.

Step 5
Enable parameter `DltSwcSupportLogLevelChangeNotification` if the `Dlt` shall notify the SWC via LogLevelChangedNotification about log level changes.

Step 6
In `MaxSwcLogMessagelength` and `MaxSwcTraceMessageLength`, configure the maximum length of log and trace messages for this SWC.

Step 7
The tab `RtePorts` is available if the value of the `DltSwcSessionId` is greater than 0x1000. If the `DltRteUsage` parameter is enabled, you can configure the generation of the ports `DltProvideControlServicePort`, `DltProvideSwcMessageServicePort` and `DltProvideLogTraceSessionControlPort`.

NOTE

No registration without context configuration

At run-time, if `Dlt` receives a request to register a context, `Dlt` only allows it if the context was configured for this SWC. Otherwise, `Dlt` rejects the request.

4.3.2.1. Configuring Dlt functionalities

In the `DltConfigSet` tab, you can configure the global `Dlt` functionalities that can be enabled or disabled.

4.3.2.1.1. Configuring the log channels

In the `DltLogOutput`, configure the log/trace message output. `Dlt` supports the configuration of multiple log channels.

Configuring the log channels

Step 1
Go to tab `DltLogChannel` to configure the available log channels. For each channel, configure the following parameters:

- the channel name in `DltLogChannelId`
- the maximum length of the message in `DltLogChannelMaxMessageLength`
- the log level threshold in `DltLogChannelThreshold`
- the buffer size in `DltLogChannelBufferSize`
- the transmit cycle time in `DltLogChannelTransmitCycle`
- the handle IDs in `DltITxPduHandleId`. The handle IDs for log channels are 0-based and consecutive.
- the PDUs that are configured in the `EcuC` module. These PDUs are referenced in the `DltTxPduIdRef` parameter of a log channel.

Step 2
Go to the **General** tab.

Step 3
In `DltDefaultLogChannelRef`, configure the reference to the default log channel. If one tuple does not have a log channel assigned, this default channel is used.

Step 4
To assign a channel to a configured context, go to the `DltLogChannelAssignment` tab.

Step 5
For every entry in the list, reference a configured ApplicationId/ContextId tuple in `DltLogChannelAssignmentSwcContextRef`, and a configured log channel in `DltLogChannelRef`.

You can assign multiple log channels to one ApplicationId/ContextId tuple. In this case, if a request to send a log or trace message is received, the message is sent on every assigned log channel.

4.3.2.1.2. Configuring the trace status

In the **DltTraceStatusSetting** tab, you can configure the trace status for the configured ApplicationId/ContextId tuples.

Configuring the trace status

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

Step 2
Enable the parameter `DltDefaultTraceStatus` to set the default trace status. This status is used for every tuple that does not have a trace status configured.

Step 3
Go to the **DltTraceStatusAssignment** tab.

Step 4
For every configured ApplicationId/ContextId tuple, configure the trace status with `DltTraceStatus`.

4.3.2.1.3. Configuring the log level

In the **DltLogLevelSetting** tab, you can configure the log level for the configured ApplicationId/ContextId tuples.

Configuring the log level

Step 1
In the **General** tab, in parameter `DltDefaultLogLevel`, select the default log level. The default log level is used as the log level for every tuple that does not have a log level configured.

Step 2
In the **DltLogLevelThreshold** tab, set the log level for a configured ApplicationId/ContextId tuple.

4.3.3. Configuring the Dlt service needs

For background information, see Section 4.2.5.1, “Support for generating Dlt ports according to service needs”.

Describing the `Dlt` service needs in the SWCD

For each group of ports that belong to one session ID and are handled with one *PortDefinedArgumentValue* by the `Dlt` service:

Step 1
For each SessionId, create one *SwcServiceDependency* as part of the *SwcInternalBehavior*.

Step 2
Add the *DltUserNeeds* to this *SwcServiceDependency*.

Step 3
For each included port, add one *RoleBasedPortAssignment* with a reference to the *PortPrototype*. The role of *RoleBasedPortAssignment* can be left empty.

Step 4
Create a new *PortAPIOption* with the value of the SessionId as *PortDefinedArgumentValue*.

Step 5
Attach to *RoleBasedPortAssignment* all *PortPrototype* elements that belong to this SessionId.

4.3.4. Configuring VFB tracing

For background information, see Section 4.2.3, “Virtual function bus tracing”.

Configuring virtual function bus tracing

Step 1
Enable VFB tracing with the parameter `DltImplementVfbTrace`.

Step 2
To send trace messages periodically in the VFB MainFunction, enable the parameter `DltVfbMainFunctionPeriod`. Otherwise, the transmission is triggered in the hook function.

Step 3
To add the hook function parameter values in the trace message payload, enable the parameter `DltVfbSendHookFunctionParameters`.

Step 4
If you enabled `DltVfbSendHookFunctionParameters`, you must configure the maximum size of the trace data sent by a VFB hook function in the `DltVfbTracePayloadMaxSize` parameter.

Step 5
In parameter `DltVfbTraceNoOfInterrupts`, configure the maximum number of times that VFB hook function calls can interrupt other calls of this hook function. This value is used for calculating the necessary size of the VFB buffer. This buffer is allocated in RAM memory, so a large value configured will have an impact on the RAM usage.

Step 6
If the actual number of interruptions is greater than the one configured in `DltVfbTraceNoOfInterrupts`, it is possible that there is not enough space in the buffer to save the data of the hook function. In this case `Dlt` can report a production error.

Configure `DltReportToDem` to report an error to `Dem`, to `Det`, or to ignore the error.

Step 6.1
For a `Dem` error, reference a valid `Dem` event parameter in `DltDemEventParameterRefs`.

4.3.5. Configuring BSW distribution

The `Dlt` BSW distribution depends on the multi-core configuration of the OS. As a mandatory precondition, the project must integrate and configure the OS stack with multi-core support. To enable multi-core support, the parameter `OsNumberOfCores` must be configured to a value higher than 1.

Only if `OsNumberOfCores` is set to a value higher than 1, the `Dlt` BSW distribution feature with all associated configuration parameters is available.

Moreover, `OsNumberOfCores` defines the number of core configurations. For example, if `OsNumberOfCores` is configured to 3, there are three configurations in the **OsCoreConfig** tab.

In addition to the `Os` module, the BSW distribution functionality depends on the existence of the `Rte` module. The data transfer between the master and satellite instances is realized with the help of the scheduler `SchM`, which is integrated into the `Rte`.

This section describes the configuration dependencies of the basic software distribution functionality (multi-core support) and the interaction between the `Dlt` and the `Rte` module. For background information on the `Dlt` BSW distribution, see Section 4.2.10, “Support for BSW distribution”.

4.3.5.1. Configuring the Rte for Dlt BSW distribution

Configuring the Rte

Step 1
Open the **Rte Generic Editor**.

Step 2
In the **BSW Module Instances** tab, create BSW module instances for each core on which the module should run. The module does not necessarily need to run on all cores.

Step 3
In **RteBswRequiredSenderReceiverConnection**, define the ports between satellites and master based on the following port connection information.

	Master instance
RteBswProvidedVariableDataPrototypeRef	Dlt_MasterSendSyncTablePort_<satellite_id>
RteBswRequiredVariableDataPrototypeRef	Dlt_SlaveReceiveSyncTablePort_<satellite_id>

Table 4.4. Port connection for the master

For each satellite, a `Dlt_MasterSendSyncTablePort_<satellite_id>` port and a `Dlt_SlaveReceiveSyncTablePort_<satellite_id>` port is generated. Each master port needs to be connected to its corresponding satellite port.

	send Message	**send Context**	**send SessionID**
RteBswProvidedVariableDataPrototypeRef	Dlt_SlaveSendMessagePort	Dlt_SlaveSendContextMessagePort	Dlt_SlaveSendSessionIdPort
RteBswRequiredVariableDataPrototypeRef	Dlt_MasterReceiveMessagePort	Dlt_MasterReceiveContextMessagePort	Dlt_MasterReceiveSessionIdPort

Table 4.5. Port connection for each satellite

4.3.6. Configuring the Dlt main function

The `Dlt` provides one scheduled function, `Dlt_MainFunction()`. This function is used for timing measurements and proper bandwidth management. Thus, this function must be called regularly on a fixed time base.

Information about this function is exported to the basic software module description (BSWMD) of `Dlt` to allow a simple mapping of `Dlt_MainFunction()` to a suitable, periodically scheduled task with the help of EB tresos AutoCore Generic 8 RTE (`Rte`).

Mapping Dlt_MainFunction() to an Rte task

Step 1
Set the `DltMainFunctionPeriod` configuration parameter to the desired `Dlt_MainFunction()` call interval in seconds. Note that this interval also defines the minimum interval between the transmission of two `Dlt` messages.

Step 2
Run the **generate_swcd** option from the **Generate code for the currently selected projects** button.

Step 3
Run the *System Description Importer* to import all the system descriptions generated previously.

Step 4
Use the `Rte Editor` to assign the `Dlt_MainFunction()` to a proper task.

For more information on the `Rte Editor`, see the EB tresos AutoCore Generic 8 RTE user guide.

5. ACG8 DLT module references

5.1. Overview

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

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

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

5.1.1. Notation in EB module references

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

5.1.1.1. Default value of configuration parameters

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

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

5.1.1.2. Range information of configuration parameters

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

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

5.2. Dlt

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
DltGeneral	1..1	Flags for adding removing functionality from code.
ReportToDem	1..1	**Label:** Production error handling Production error handling
DltMemory	1..1	Configuration parameters for reserving memory for some internal storing and buffer.
DltVfbTrace	0..0	All functions to trace from the VFB by the Dlt.
DltConfigSet	1..1	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption. It is supported starting with AUTOSAR_431
DltMultipleConfigurationContainer	1..1	This container contains the configuration parameters and sub containers of the AUTOSAR Dlt module.
DltDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
DltSwc	1..n	This container purpose is to register a number of Sw Components. It is supported starting with AUTOSAR_431
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

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

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT	
Label	Config Variant	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VariantPreCompile	
Range	VariantPreCompile	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion

Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

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

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

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL

Default value	8
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	7
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	55
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

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

Parameter Name	**Release**

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

5.2.1.2. DltGeneral

Containers included		
Container name	**Multiplicity**	**Description**
DltDemEventParameterRefs	1..1	**Label:** Dem Events References

Parameters included	
Parameter name	**Multiplicity**
DltRteUsage	1..1
DltGeneralStartUpDelayTimer	0..1
DltDevErrorDetect	1..1
DltEnableTrafficShaper	1..1
DltImplementAppIdContextIdQuery	1..1
DltImplementExtendedHeader	1..1
DltImplementFilterMessages	1..1
DltImplementNVRamStorage	1..1
DltImplementSWCInjection	1..1
DltImplementTimestamp	1..1
DltImplementVerboseMode	1..1
DltImplementVfbTrace	1..1
DltVersionInfoApi	1..1
DltGeneralRegisterContextNotification	1..1
DltVfbTraceLogLevel	1..1
DltGptChannel	1..1
DltGeneralStbMTimeBaseRef	0..1
DltCSPortsQueueLength	1..1

Parameters included	
DltControlMessageMaxResponseSize	1..1
DltTupleEndianness	1..1
DltMessageOptionsBitField	1..1
DltVfbMainFunctionPeriod	0..1
DltVfbSendHookFunctionParameters	1..1
DltVfbTracePayloadMaxSize	1..1
DltVfbTraceNoOfInterrupts	1..1

Parameter Name	DltRteUsage	
Label	Enable Rte Usage	
Description	This parameter enables the usage of the RTE for this module. For an easy integration it is recommended to disable the usage of the RTE at the beginning of the integration work.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	DltGeneralStartUpDelayTimer	
Description	Defines the delay of starting the transmission of messages after the Dlt module has been initialized in seconds.	
Multiplicity	0..1	
Type	FLOAT	
Default value	0.1	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltDevErrorDetect
Description	Enables/Disables development error detection.
Multiplicity	1..1
Type	BOOLEAN

Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltEnableTrafficShaper**	
Description	Enable Traffic Shaper feature for Dlt.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	EB	

Parameter Name	**DltImplementAppIdContextIdQuery**	
Description	If set the functionality for Verbose Mode shall be available. Note: The functionality related to this parameter is not supported by the current implementation.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltImplementExtendedHeader**	
Description	If set the extended functionality for the header shall be available.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltImplementFilterMessages**
Description	This flag is for code generation of Dlt. If set the functionality for filtering messages shall be included in the code.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltImplementNVRamStorage**	
Description	If set the functionality for storing and loading information in and from NVRam shall be available.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltImplementSWCInjection**	
Description	If the remote call from functions over the Dlt in SW-C shall be available. Note: The functionality related to this parameter is not supported by the current implementation.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltImplementTimestamp**	
Description	If set the timestamp functionality for the header shall be available. In the AUTOSAR 4.3.1 specifications this parameter is called diferently (DltGeneralTimeStampSupport).	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltImplementVerboseMode	
Description	If set the timestamp functionality for the header shall be available. Note: when DltServiceAPI is set to "AUTOSAR_431", this parameter is disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltImplementVfbTrace	
Description	If set the header files and the implementation of VFB-trace shall be generated.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltVersionInfoApi	
Description	Pre-processor switch for enabling Version Info API support.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltGeneralRegisterContextNotification	
Description	If this parameter is set to TRUE, a Dlt Control Message is sent every time a SWC registeres and/or de-registers at/from the Dlt Module. Else, this notification is not sent. TODO This parameter is not enabled yet	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**DltVfbTraceLogLevel**	
Description	The log level of the log messages generated by the VFB-Trace.. Note: The functionality related to this parameter is not supported by the current implementation.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltGptChannel**	
Description	Reference to the hardware free running timer of the GPT module for timestamps (if no HWFRT is applied, calls to add timestamps are ignored).	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltGeneralStbMTimeBaseRef**	
Description	If DltImplementTimestamp is set to TRUE and DltGeneralStbMTimeBaseRef is configured, the Dlt module shall fetch the time from the StbM module by calling StbM_GetCurrentTime with the here referenced StbMSynchronizedTimeBase.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltCSPortsQueueLength**
Description	Configuration parameter which allows defining the queue length of the *Client / Server ComSpec Operations*. If parameter is set to value 0 no SERVER-COM-SPECs are created and the queue length will be calculated by RTE based on the number of connected ports, otherwise the configured DltCSPortsQueueLength value is assigned.

Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltControlMessageMaxResponseSize**
Description	Allows customization of the response buffer's size for control messages. If the GetLogInfo control message is to be used, the following formula must be used to calculate the response size: Size when no textual descriptions are requested: ResponseBufferSize = 2 bytes + (6 bytes * TotalNoOfAppIds) + (6 bytes * TotalNoOfContextIds) + ReservedBytes ReservedBytes (4 bytes) is always filled with 0 Size when textual descriptions are requested: ResponseBufferSize = 2 bytes + (7 bytes * TotalNoOfAppIds) + (7 bytes * TotalNoOfContextIds) + ReservedBytes ReservedBytes (4 bytes) is always filled with 0 Note: Textual descriptions are not supported when the DltServiceAPI configuration parameter is set to "AUTOSAR_421" or "AUTOSAR_422". As such, the description length fields for AppIds (1 byte) and ContextIds (1 byte) will be written and filled with 0.
Multiplicity	1..1
Type	INTEGER
Default value	4
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltTupleEndianness**
Description	If enabled, the Application and Context Id of a tuple, under Autosar 4.3.1 (DltServiceAPI == AUTOSAR_431), is taken in MSB order. If this parameter is disabled, the Application/Context IDs will be interpreted according to the platform endianness.

	For Autosar versions different than 4.3.1 (DltServiceAPI == AUTOSAR_421) this parameter is disabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltMessageOptionsBitField**
Label	AUTOSAR release for Message Options bit-field
Description	Defines what AUTOSAR release to use for Message Options' bit-field. On AUTOSAR 4.2.1: ► Bit 3 is used for verbose_mode ► Bits 0, 1, 2 are used for message_type On AUTOSAR 4.3.1: ► Bit 0 is used for verbose_mode ► Bits 1, 2, 3 are used for message_type Values: ► `AUTOSAR_421` = AUTOSAR 4.2.1's bit-field (default) ► `AUTOSAR_431` = AUTOSAR 4.3.1's bit-field
Multiplicity	1..1
Type	ENUMERATION
Default value	AUTOSAR_421
Range	AUTOSAR_421 AUTOSAR_431
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltVfbMainFunctionPeriod**
Label	Period of Dlt_VfbMainFunction()
Description	Period of Dlt_VfbMainFunction() calls in seconds.
Multiplicity	0..1

Type	FLOAT	
Default value	0.1	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltVfbSendHookFunctionParameters**	
Label	Send VFB Hook Function Parameters	
Description	Enable this parameter in order to send parameter information of the configured hook functions. If disabled, only the Message ID will be sent as payload in the trace message.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltVfbTracePayloadMaxSize**	
Label	VFB Hook Function Payload Max. Size	
Description	Maximum size of the trace data (payload) sent by a VFB hook function.	
Multiplicity	1..1	
Type	INTEGER	
Default value	4	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltVfbTraceNoOfInterrupts**	
Label	VFB Trace Number of Max. Interrupts	
Description	The maximum number of interrupts possible within the system.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.3. DltDemEventParameterRefs

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

Parameter Name	DLT_E_VFB_BUFFER_FULL	
Label	Error for VFB full buffer	
Description	Reference to the DemEventParameter that shall be issued when the buffer allocated for the configured trace hook function's parameters has no available space left.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.4. ReportToDem

Parameters included	
Parameter name	**Multiplicity**
DltVfbBufferFullReportToDem	1..1
DltVfbBufferFullDemDetErrorId	1..1

Parameter Name	DltVfbBufferFullReportToDem
Label	Dlt VFB buffer full
Description	Defines the handling of the production error: *DLT_E_VFB_BUFFER_FULL* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM

	DET DISABLE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltVfbBufferFullDemDetErrorId**	
Label	Dlt VFB buffer full DET ErrorId	
Description	Defines the `ErrorId` of the Dlt VFB buffer full error when it is reported to Det instead of Dem. Range: ► 30 .. 255	
Multiplicity	1..1	
Type	INTEGER	
Default value	30	
Range	<=255 >=30	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.5. DltMemory

Parameters included	
Parameter name	**Multiplicity**
DltInitBufferSize	1..1
DltMaxCountAppIds	1..1
DltMaxCountContextIds	1..1
DltMaxCountContextIdsPerAppId	1..1
DltMessageBufferSize	1..1
DltReceptionBufferSize	1..1

Parameter Name	**DltInitBufferSize**
Description	Buffer size for the C-init buffer.

	This buffer is for storing messages from other BSW modules before Dlt is initialized.
Multiplicity	1..1
Type	INTEGER
Default value	0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**DltMaxCountAppIds**
Description	The maximum count of registrable Application Ids. There shall be a table to manage registered Application IDs, this is the number of lines to hold in this table. To enable the parameter , AUTOSAR_42 or lower version shall be configured.
Multiplicity	1..1
Type	INTEGER
Default value	10
Range	<=4294967295 >=1
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**DltMaxCountContextIds**
Description	The maximum count of registrable Context Ids. There shall be a table to manage registered Context IDs, this is the number of lines to hold in this table. Note: To enable the parameter , AUTOSAR_42 or lower version shall be configured.
Multiplicity	1..1
Type	INTEGER
Default value	100
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**DltMaxCountContextIdsPerAppId**
Description	Each Context ID belongs to a specific Application ID. Dlt shall handle the correlation between them. The table of the registered Application IDs shall hold for

	every Application ID several references to the proper Context IDs. This is the maximum count for Context IDs per Application ID. Note: To enable the parameter , AUTOSAR_42 or lower version shall be configured, otherwise its default value will be equal to 1 since in AUTOSAR431 each ApplicationId/ContextId tuple shall be preconfigured and used within the application
Multiplicity	1..1
Type	INTEGER
Default value	10
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**DltMessageBufferSize**
Description	Buffer size for storing Dlt messages for waiting to transmit over the Network (send buffer). This parameter can be enabled only when AUTOSAR_42 or lower version is used
Multiplicity	1..1
Type	INTEGER
Default value	1024
Range	<=4294967295 >=20
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**DltReceptionBufferSize**
Description	Defines the buffer size for the receiving buffer. This buffer is for storing messages from communication lower layer. Note: As some external Dlt clients might send all Control Messages at once, this buffer must be large enough to accomodate them all. Length of a Control Message varies but, to get a sense of how much size one might have, 30 bytes could be initially considered for one Control Message. Dependency on parameter(s): ► Dlt Rx Pdu Id (`DltRxPduId`): this parameter must be enabled in order to be able to configure the Rx buffer size.
Multiplicity	1..1
Type	INTEGER

Default value	200	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.6. DltVfbTrace

Parameters included	
Parameter name	**Multiplicity**
DltVfbTraceFunction	1..1
DltBswModuleEntryRef	1..1

Parameter Name	**DltVfbTraceFunction**	
Description	The Dlt generator shall enable VFB tracing for a given hook function when there is a #define in the Dlt-VFB configuration header file for the hook function name and tracing is globally enabled.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltBswModuleEntryRef**	
Description	Foreign reference to the BSWModuleEntry describing the trace function implementation.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.7. DltConfigSet

Containers included		
Container name	**Multiplicity**	**Description**
DltLogOutput	0..1	This container contains settings for log/trace message output.

Containers included		
DltLogLevelSetting	1..1	Contains settings for thresholds.
DltTraceStatusSetting	1..1	Contains settings for trace status

5.2.1.8. DltLogOutput

Containers included		
Container name	**Multiplicity**	**Description**
DltLogChannel	1..255	This container contains settings for log/trace message output.
DltLogChannelAssignment	0..n	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned log channel.

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

Parameter Name	**DltDefaultLogChannelRef**	
Description	Reference to the default log channel, which has to be used for a log/trace output, if no other match has been found.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.9. DltLogChannel

Containers included		
Container name	**Multiplicity**	**Description**
DltTxPdu	1..1	Contains the configuration parameters of the AUTOSAR Dlt module's Tx Pdus.

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
DltLogChannelBufferOverflowTimer	1..1
DltLogChannelBufferSize	1..1
DltLogChannelId	1..1
DltLogChannelMaxMessageLength	1..1
DltLogChannelMaxNumOfRetries	1..1
DltLogChannelThreshold	1..1
DltLogChannelTrafficShapingBandwidth	1..1
DltLogChannelTransmitCycle	1..1
DltLogTraceStatusFlag	1..1

Parameter Name	DltLogChannelBufferOverflowTimer	
Description	Specifies the cycle time in seconds for resetting the buffer overflow flag in case a buffer overflow occurred. The implementation for this parameter is not done yet	
Multiplicity	1..1	
Type	FLOAT	
Default value	1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	DltLogChannelBufferSize	
Description	Buffer size in bytes for the LogChannel specific message buffer.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	DltLogChannelId	
Description	This is the 4 ASCII character long name of the log channel	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	DltLogChannelMaxMessageLength
Description	The maximum length of a Dlt log or trace message.

Multiplicity	1..1	
Type	INTEGER	
Range	>=0 <=65535	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltLogChannelMaxNumOfRetries**	
Description	The maximum number of retries for sending a message on a log channel , in case the message wasn't successfully sent . This parameter is not used yet	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	>=0 <=255	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltLogChannelThreshold**	
Description	This is the default log level for the channel in use	
Multiplicity	1..1	
Type	ENUMERATION	
Range	DLT_LOG_DEBUG DLT_LOG_ERROR DLT_LOG_FATAL DLT_LOG_INFO DLT_LOG_OFF DLT_LOG_VERBOSE DLT_LOG_WARN	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltLogChannelTrafficShapingBandwidth**
Description	Set the maximum possible bandwith in bit/s. The implementation for this parameter is not done yet
Multiplicity	1..1

Type	INTEGER	
Default value	8	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltLogChannelTransmitCycle**	
Description	Specifies the cycle time in ms of the transmit functionality of this log channel.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	>=1 <=1000	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltLogTraceStatusFlag**	
Description	Parameter to turn on/off sending trace information on this LogChannel completely.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.10. DltTxPdu

Parameters included	
Parameter name	**Multiplicity**
DltITxPduHandleId	1..1
DltTxPduIdRef	1..1
DltITxPduUsesTp	1..1

Parameter Name	**DltITxPduHandleId**
Description	The numerical value used as the ID of this I-PDU. This handle Id is used for the APIs calls Dlt_TxConfirmation, Dlt_TriggerTransmit, Dlt_TriggerIPDUSend or Dlt_TriggerIPDUSendWithMetaData, Dlt_CopyTxData and Dlt_TpTxConfirmation to transmit respectively confirm transmissions of I-PDUs, as well as

	the PduId passed to the Tx-I-PDU-callout configured with DltIPduCallout and/or DltIPduTriggerTransmitCallout.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltTxPduIdRef**	
Description	Reference to the global Pdu structure to allow harmonization of handle IDs in the COM-Stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Parameter Name	**DltITxPduUsesTp**	
Description	If set to TRUE, the PDU is transmitted using the TP API. If FALSE, the IF API is used. Implementation for this parameter is not done yet	
Multiplicity	1..1	
Type	BOOLEAN	
Range	true false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.11. DltLogChannelAssignment

Parameters included	
Parameter name	**Multiplicity**
DltLogChannelAssignmentSwcContextRef	1..1
DltLogChannelRef	1..1

Parameter Name	**DltLogChannelAssignmentSwcContextRef**
Description	Reference to an ApplicationId/ContextId pair that is assigned to a DltLogChannel.
Multiplicity	1..1
Type	REFERENCE

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltLogChannelRef**	
Description	Reference to a DltLogChannel that is assigned to an ApplicationId / ContextId pair.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.12. DltLogLevelSetting

Containers included		
Container name	**Multiplicity**	**Description**
DltLogLevelThreshold	0..n	This is the effective trace status used for a valid tuple of ApplicationId/ContextId.

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

Parameter Name	**DltDefaultLogLevel**
Description	This is the effective trace status used in case no filter matches the given ApplicationId and ContextId. This can be seen as a fall-through filter definition with wildcard for ApplicationId and ContextId, which will be used, when no other filter matches.
Multiplicity	1..1
Type	ENUMERATION
Range	DLT_LOG_DEBUG
	DLT_LOG_ERROR
	DLT_LOG_FATAL
	DLT_LOG_INFO
	DLT_LOG_OFF

	DLT_LOG_VERBOSE	
	DLT_LOG_WARN	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.13. DltLogLevelThreshold

Parameters included	
Parameter name	**Multiplicity**
DltLogLevelThresholdSwcContextRef	1..1
DltSwCLogLevel	1..1

Parameter Name	**DltLogLevelThresholdSwcContextRef**	
Description	Reference to an ApplicationId/ContextId pair to which a DltTraceStatus is assigned.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltSwCLogLevel**	
Description	LogLevel for a mapping ApplicationId/ContextId.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	DLT_LOG_DEBUG	
	DLT_LOG_ERROR	
	DLT_LOG_FATAL	
	DLT_LOG_INFO	
	DLT_LOG_OFF	
	DLT_LOG_VERBOSE	
	DLT_LOG_WARN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.14. DltTraceStatusSetting

Containers included		
Container name	**Multiplicity**	**Description**
DltTraceStatusAssignment	0..n	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned trace status.

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

Parameter Name	**DltDefaultTraceStatus**	
Description	This is the effective trace status used in case no filter matches the given ApplicationId and ContextId. This can be seen as a fall-through filter definition with wildcard for ApplicationId and ContextId, which will be used, when no other filter matches.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.15. DltTraceStatusAssignment

Parameters included	
Parameter name	**Multiplicity**
DltTraceStatusAssignmentSwcContextRef	1..1
DltTraceStatus	1..1

Parameter Name	**DltTraceStatusAssignmentSwcContextRef**	
Description	Reference to an ApplicationId/ContextId pair to which a DltTraceStatus is assigned.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltTraceStatus**

Description	Trace status for the given ApplicationId/ContextId tuple.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.16. DltMultipleConfigurationContainer

Containers included		
Container name	**Multiplicity**	**Description**
DltBandwidth	1..1	Configuration parameters controlling network and diagnostic interfaces bandwidth. If DltImplementNVRamStorage is enabled these parameters are the initial values for the NVRam. If DltImplementNVRamStorage is not set, Link-Time or Post-Build configuration shall be uesd.
DltMessageFiltering	1..1	Configuration parameters for setting message filtering properties in Dlt module.
DltBswDistribution	1..1	Configuration parameters controlling the BSW distribution of the Dlt module on multi-core platforms.
DltPduConfig	1..1	Configuration parameters referencing the Pdus for transmitting and receiving messages.
DltServiceAPI	1..1	Support for AUTOSAR API releases
DltProtocol	1..1	Configuration parameters for handling the specific protocol variants.

Parameters included	
Parameter name	**Multiplicity**
DltNvRamBlockRef	1..1
DltNvRamDataSetBlockRef	1..1
DltMainFunctionPeriod	1..1

Parameter Name	**DltNvRamBlockRef**
Description	Reference to the NvM Block which is used to store the Dlt parameters.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE

Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltNvRamDataSetBlockRef**	
Description	Reference to the NvM DataSetBlock which is used to store the Dlt parameters.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltMainFunctionPeriod**	
Description	Period of Dlt_MainFunction calls in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.1	
Range	<=1.000 >=0.001	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.17. DltBandwidth

Parameters included	
Parameter name	**Multiplicity**
DltBandwidthForComModule	1..1
DltBandwidthForDiagChannel	1..1
DltTimePeriodTrafficShaping	1..1

Parameter Name	**DltBandwidthForComModule**
Description	For communication over the Dlt Communication Module the maximum bandwidth shall be set. Unit: kbits/s. ***Note*** : Parameter is editable only if Traffic Shaping functionality is enabled via DltEnableTrafficShaper.

Multiplicity	1..1	
Type	INTEGER	
Default value	8	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltBandwidthForDiagChannel**	
Description	For communication over the DCM and as follows over the diagnostic interface the maximum bandwidth shall be set. ***Note*** : Functionality for DltBandwidthForDiagChannel is currently not implemented.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=4294967295 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltTimePeriodTrafficShaping**	
Description	For implementing a traffic shaping, a time window for measuring shall be provided. This parameter specifies the size of this Window in milliseconds. Upper limit: 300000ms (5 minutes). ***Note*** : Parameter is editable only if Traffic Shaping functionality is enabled.	
Multiplicity	1..1	
Type	FLOAT	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.18. DltMessageFiltering

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
DltDefaultMaxLogLevel	1..1
DltDefaultTraceStatus	1..1
DltFactoryDefaultMaxLogLevel	1..1
DltFilterMessages	1..1

Parameter Name	DltDefaultMaxLogLevel	
Description	The maximum log level a received message (from SW-C to Dlt) can have. This can also be modified at runtime and stored persistently in NVRAM. If DltImplementNVRamStorage is enabled this parameter is the initial value for the corresponding NVRam entry. If DltImplementNVRamStorage is not set, Link-Time or Post-Build configuration shall be used. The value 0 means logging is disabled. Note: This parameter is disabled with AUTOSAR_43 or later version. With the Autosar_43 or later default value for log messages is configured with DltDefaultLogLevel from DltLogLevelSetting.	
Multiplicity	1..1	
Type	INTEGER	
Default value	3	
Range	<=6 >=0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltDefaultTraceStatus	
Description	Tells if trace messages shall be forwarded by Dlt. This functionality can also be modified at runtime and changed can stored persistently in NVRAM. If DltImplementNVRamStorage is enabled this parameter is the initial value for the corresponding NVRam entry. If DltImplementNVRamStorage is not set, Link-Time or Post-Build configuration shall be used. Note: This parameter is disabled with the Autosar_43 or later. Instead DltDefaultTraceStatus from the DltTraceStatusSetting is used	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltFactoryDefaultMaxLogLevel	
Description	The maximum log levels a received message (from SW-C to Dlt) can have. This is for setting DltDefaultMaxLogLevel to factory defaults. The value 0 means logging is disabled.	
Multiplicity	1..1	
Type	INTEGER	
Default value	3	
Range	<=6 >=0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltFilterMessages	
Description	This flag gives the Dlt the instruction to filter or not to filter incoming log or trace messages. If it is not set all incoming messages are forwarded to the communication channel. So also the caller of the DltSendXXXMessage can leave the field trace_info or log_info empty.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.19. DltBswDistribution

Containers included		
Container name	**Multiplicity**	**Description**
DltSatelliteCore	0..n	Core IDs of satellite cores.

Parameters included	
Parameter name	**Multiplicity**
DltEnableBswDistribution	1..1
DltNumberOfSatellites	1..1

Parameters included	
DltNumberOfMessagesPerSatellite	1..1
DltCoreSyncQueueSize	1..1
DltMasterCore	1..1

Parameter Name	DltEnableBswDistribution	
Description	Enable BSW distribution features for Dlt.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DltNumberOfSatellites	
Description	Number of satellite cores (in addition to the master core).	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	DltNumberOfMessagesPerSatellite	
Description	Number of messages per satellite core. The queue size for the reception policy of the master is generated according to the number of satellites and according to the value configured for this parameter. This way a minimisation of the possibility of losing message, in case the data received events are mapped to a task which is not immediately planned and the master cannot read fast enough, is performed.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=65535 >=1	
Configuration class	**PreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**DltCoreSyncQueueSize**	
Description	The queue size for the reception policy of sync data for each satellite. Increasing this value will decrease the possibility of losing sync data, in case sync data is generated too fast and the satellite RTE task is not scheduled in time to empty the queue. When this queue size is too small, the DET error DLT_E_CORE_SYNC_FAILED is reported.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=65535 >=1	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltMasterCore**	
Description	Core ID of the Dlt master.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.20. DltSatelliteCore

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

Parameter Name	**DltSatelliteCoreId**
Description	Core ID of the core a satellite is running on.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.21. DltPduConfig

Parameters included	
Parameter name	**Multiplicity**
DltRxPduId	0..1
DltRxPduRef	1..1
DltTxPduId	1..1
DltTxPduRef	1..1
DltEnableMultipleFrames	1..1
DltDiscardMsgTxFail	1..1

Parameter Name	**DltRxPduId**	
Description	PduId of Pdu for receiving messages. If enabled, Dlt can receive Control Messages.	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltRxPduRef**	
Description	Reference to Pdu for receiving messages. This parameter is only available if DltRxPduId is enabled.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltTxPduId**
Description	PduId of Pdu for transmitting messages.

Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=65535 >=0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltTxPduRef**	
Description	Reference to Pdu for transmitting messages. Note : This parameter is disabled if AUTOSAR_43 or newer version is configured Default value will be the Pdu wich is assigned to the default log channel	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltEnableMultipleFrames**	
Description	If enabled, multiple Dlt frames are transmitted within one Pdu. The number of frames per Pdu depends on the size of the individual frames and the DltMaxMessageLength setting, i.e. as many frames are transmitted as fit into one Pdu.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltDiscardMsgTxFail**
Description	Defines if DLT messages are discarded or not when transmit request to PduR return an error. If enabled the messages will be discarded by DLT when PduR_DltTransmit returns E_NOT_OK.
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.22. DltServiceAPI

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

Parameter Name	**DltServiceAPI**	
Label	Default AUTOSAR return values	
Description	Defines the default AUTOSAR return values. ► `AUTOSAR_421` = AUTOSAR 4.2.1 return values (default) ► `AUTOSAR_422` = AUTOSAR 4.2.2 return values ► `AUTOSAR_431` = AUTOSAR 4.3.1 return values	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	AUTOSAR_421	
Range	AUTOSAR_421 AUTOSAR_422 AUTOSAR_431	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.23. DltProtocol

Parameters included	
Parameter name	**Multiplicity**
DltHeaderUseEcuId	1..1
DltEcuId	1..1
DltEcuIdCallout	1..1
DltHeaderPayloadEndianes	1..1

Parameters included	
DltHeaderUseExtendedHeader	1..1
DltHeaderUseSessionID	1..1
DltHeaderUseTimestamp	1..1
DltMaxMessageLength	1..1
DltUseVerboseMode	1..1
DltGetSoftwareVersion	1..1
DltGetSoftwareVersionLength	1..1
DltEcuIdChoice	1..1

Parameter Name	DltHeaderUseEcuId	
Description	Corresponds to field WEID (With ECU ID). If set ECU ID shall be placed in the header, else not.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltEcuId	
Description	This is the name of the ECU for use within the Dlt protocol. The meaning is described in the AUTOSAR SWS document. This name is transmitted within the Dlt protocol and contains 4 characters.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	DltEcuIdCallout	
Description	This is the name of the user callout to call in order to get the ECU id.	
Multiplicity	1..1	
Type	STRING	
Default value	Dlt_AppGetEcuIdAddress	
Configuration class	**PreCompile:**	VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**DltHeaderPayloadEndianes**	
Description	Determines the endianess of the CPU (Most Significant Byte).	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	LittleEndian	
Range	BigEndian	
	LittleEndian	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltHeaderUseExtendedHeader**	
Description	Corresponds to field UEH (Use Extended Header). If set the Extended Header shall be attached, else not.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltHeaderUseSessionID**	
Description	Corresponds to field WSID (with Session ID). If set the Session ID shall be placed in the header, else not.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltHeaderUseTimestamp**
Description	Corresponds to field WTMS (With Timestamp). If set the timestamp shall be placed in the header, else not.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltMaxMessageLength**	
Description	The maximum length of a Dlt log or trace message.	
Multiplicity	1..1	
Type	INTEGER	
Default value	128	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltUseVerboseMode**	
Description	If this flag is set Dlt shall use the Verbose Mode for generating the header of the transport protocol. Also it shall store the information provided by registering Context ID and Application ID (description) at runtime if flag is set. If it is not set, the Non Verbose Mode shall be used.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltGetSoftwareVersion**
Label	GetSoftwareVersion control message
Description	Enables the control message GetSoftwareVersion.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**DltGetSoftwareVersionLength**
Label	GetSoftwareVersion control message length
Description	The length of the ECU software version string in bytes.

Multiplicity	1..1
Type	INTEGER

Parameter Name	**DltEcuIdChoice**
Label	Get EcuId via
Description	Enables the user to provide EcuId at runtime via a user defined callout function
Multiplicity	1..1
Type	ENUMERATION
Default value	Value
Range	Value Callout

5.2.1.24. DltDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
DltDefProgEnabled	1..1
DltPrecondAssertEnabled	1..1
DltPostcondAssertEnabled	1..1
DltStaticAssertEnabled	1..1
DltUnreachAssertEnabled	1..1
DltInvariantAssertEnabled	1..1

Parameter Name	**DltDefProgEnabled**
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module Dlt. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required
Multiplicity	1..1
Type	BOOLEAN

Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltPrecondAssertEnabled**	
Label	Enable Precondition Assertions	
Description	Enables handling of precondition assertion checks reported from the module Dlt. Dependency on parameter(s): ► Enable Development Error Detection (`DltDevErrorDetect`): must be enabled ► Enable Defensive Programming (`DltDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltPostcondAssertEnabled**	
Label	Enable Postcondition Assertions	
Description	Enables handling of postcondition assertion checks reported from the module Dlt. Dependency on parameter(s): ► Enable Development Error Detection (`DltDevErrorDetect`): must be enabled ► Enable Defensive Programming (`DltDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**DltStaticAssertEnabled**
Label	Enable Static Assertions

Description	Enables handling of static assertion checks reported from the module Dlt. Dependency on parameter(s): ► Enable Development Error Detection (`DltDevErrorDetect`): must be enabled ► Enable Defensive Programming (`DltDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltUnreachAssertEnabled**
Label	Enable Unreachable Code Assertions
Description	Enables handling of unreachable code assertion checks reported from the module Dlt. Dependency on parameter(s): ► Enable Development Error Detection (`DltDevErrorDetect`): must be enabled ► Enable Defensive Programming (`DltDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltInvariantAssertEnabled**
Label	Enable Invariant Assertions
Description	Enables handling of invariant assertion checks reported from functions of the module Dlt. Dependency on parameter(s): ► Enable Development Error Detection (`DltDevErrorDetect`): must be enabled ► Enable Defensive Programming (`DltDefProgEnabled`): must be enabled

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.25. DltSwc

Containers included		
Container name	**Multiplicity**	**Description**
DltSwcContext	1..n	This container contains the configuration of ApplicationId / ContextId pairs which are supported by this SWC.

Parameters included	
Parameter name	**Multiplicity**
DltSwcSessionId	1..1
DltSwcSupportLogLevelChangeNotification	1..1
MaxSwcLogMessageLength	1..1
MaxSwcTraceMessageLength	1..1
DltProvideControlServicePort	1..1
DltProvideSwcMessageServicePort	1..1
DltProvideLogTraceSessionControlPort	1..1

Parameter Name	**DltSwcSessionId**	
Description	An ECU wide unique ID to identify the port a SWC (instance) uses. It is supported starting with AUTOSAR_431 Note: Base value ID for SWCs will be 0x1000. From this value DltSwcSupportLogLevelChangeNotification parameter will be enabled and can be used. Values < 0x1000 will be for BSW modules and they cannot be notified about log and trace changes	
Multiplicity	1..1	
Type	INTEGER	
Range	<=4294967295 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**DltSwcSupportLogLevelChangeNotification**	
Description	Flag indicating, whether Dlt has to provide a R-Port for the notification of the SWC about LogLevel changes. If this parameter is set to true, at each log information change, Dlt shall notify the SwC via LogLevelChangedNotification about it. Note : Parameter in use only for DltSwcSessionId >= 0x1000	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**MaxSwcLogMessageLength**	
Description	Defines the maximum allowed length (unit16) for LogMessages of a SwC.	
Multiplicity	1..1	
Type	INTEGER	
Range	>=0 \<=65535	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**MaxSwcTraceMessageLength**	
Description	Defines the maximum allowed length (unit16) for TraceMessages of a SwC.	
Multiplicity	1..1	
Type	INTEGER	
Range	>=0 \<=65535	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltProvideControlServicePort**
Label	Generate ControlService Port
Description	Enables the generation of the `DltControlService` port

	Dependency on parameter(s): ► `DltRteUsage` must be enabled. This port includes the following operations: ► SetDefaultLogLevel ► GetDefaultLogLevel ► SetDefaultTraceStatus ► GetDefaultTraceStatus ► SetLogLevel ► GetLogInfo ► SetTraceStatus ► GetTraceStatus ► SetMessageFiltering ► GetLogChannelNames ► GetLogChannelThreshold ► SetLogChannelAssignment ► SetLogChannelThreshold ► ResetToFactoryDefault ► StoreConfiguration
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltProvideSwcMessageServicePort**
Label	Generate SwcMessageService Port
Description	Enables the generation of the `DltSwcMessageService` port Dependency on parameter(s): ► `DltRteUsage` must be enabled. This port includes the following operations: ► RegisterContext ► UnregisterContext

	► SendLogMessage ► SendTraceMessage
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**DltProvideLogTraceSessionControlPort**
Label	Generate LogTraceSessionControl Port
Description	Enables the generation of the `LogTraceSessionControl` port Dependency on parameter(s): ► `DltRteUsage` must be enabled. ► `DltSwcSupportLogLevelChangeNotification` must be enabled. This port includes the following operations: ► LogLevelChangedNotification ► TraceStatusChangedNotification
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.26. DltSwcContext

Parameters included	
Parameter name	**Multiplicity**
DltSwcApplicationId	1..1
DltSwcContextId	1..1

Parameter Name	**DltSwcApplicationId**
Description	SWC Application Id.
Multiplicity	1..1

Type	INTEGER	
Default value	0	
Range	>=1 <=4294967295	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**DltSwcContextId**	
Description	SWC Context Id.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	>=1 <=4294967295	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.27. PublishedInformation

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

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

5.2.2. Application programming interface (API)

5.2.2.1. Type definitions

5.2.2.1.1. Dlt_ApplicationIDType

Purpose	This type describes the Application ID.
Type	`uint8[4U]`

5.2.2.1.2. Dlt_AssignmentOperation

Purpose	Assignment operation used for setting log channel assignments.
Type	`uint8`

5.2.2.1.3. Dlt_AssignmentOperationType

Purpose	Assignment operation used for setting log channel assignments.
Type	`uint8`

5.2.2.1.4. Dlt_ContextIDType

Purpose	This type describes the Context ID.
Type	`uint8[4U]`

5.2.2.1.5. Dlt_CtrlReturnType

Purpose	
Type	`uint8`

5.2.2.1.6. Dlt_FilterMessagesType

Purpose	

Type	`uint8`

5.2.2.1.7. Dlt_GlobalLogStatusType

Purpose	
Type	`uint8`

5.2.2.1.8. Dlt_Internal_ApplicationIDType

Purpose	Internal representation of Dlt_ApplicationIDType.
Type	`uint32`

5.2.2.1.9. Dlt_Internal_ContextIDType

Purpose	Internal representation of Dlt_ContextIDType.
Type	`uint32`

5.2.2.1.10. Dlt_MessageArgumentCountType

Purpose	
Type	`uint16`

5.2.2.1.11. Dlt_MessageCommonInfoType

Purpose		
Type	struct	
Members	`Dlt_MessageArgumentCountType arg_count`	
	`Dlt_MessageLogTraceType log_-level_trace_info`	
	`Dlt_MessageOptionsType options`	
	`Dlt_ContextIDType context_id`	

	`Dlt_ApplicationIDType app_id`	

5.2.2.1.12. Dlt_MessageControlInfoType

Purpose	
Type	`uint8`

5.2.2.1.13. Dlt_MessageIDType

Purpose	Contains the unique Message ID for a message.
Type	`uint8`

5.2.2.1.14. Dlt_MessageLogInfoType

Purpose		
Type	struct	
Members	`Dlt_MessageArgumentCountType arg_count`	
	`Dlt_MessageLogLevelType log_-level`	
	`Dlt_MessageOptionsType options`	
	`Dlt_ContextIDType context_id`	
	`Dlt_ApplicationIDType app_id`	

5.2.2.1.15. Dlt_MessageLogLevelType

Purpose	
Type	`uint8`

5.2.2.1.16. Dlt_MessageLogTraceType

Purpose	
Type	`uint8`

5.2.2.1.17. Dlt_MessageNetworkTraceInfoType

Purpose	
Type	`uint8`

5.2.2.1.18. Dlt_MessageOptionsType

Purpose	Bitfield.
Type	`uint8`

5.2.2.1.19. Dlt_MessageTraceInfoType

Purpose		
Type	struct	
Members	`Dlt_MessageTraceType trace_info`	
	`Dlt_MessageOptionsType options`	
	`Dlt_ContextIDType context_id`	
	`Dlt_ApplicationIDType app_id`	

5.2.2.1.20. Dlt_MessageTraceStatusType

Purpose	
Type	`uint8`

5.2.2.1.21. Dlt_MessageTraceType

Purpose	
Type	`uint8`

5.2.2.1.22. Dlt_MessageTypeType

Purpose	This type describes the type of the message.
Type	`uint8`

5.2.2.1.23. Dlt_ReturnType

Purpose	
Type	`uint8`

5.2.2.1.24. Dlt_SessionIDType

Purpose	This type describes the Session ID.
Type	`uint32`

5.2.2.1.25. Dlt_TxConnectionType

Purpose		
Type	struct	
Members	`PduLengthType PositionInMessage`	Holds the index of valid data in bytes to be transmitted from the current message. This also represents the amount of data which has already been copied to the lower layer from the current message, though not confirmed.
	`PduLengthType ConfirmedPosi-tionInMessage`	Holds the index of confirmed data in bytes to be transmitted from this message. This also represents the amount of data which has already been copied to the lower layer from the current message and the transmission of which has been confirmed by the lower layer.

5.2.2.2. Macro constants

5.2.2.2.1. DLT_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.2.2.2.2. DLT_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	2U

5.2.2.2.3. DLT_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	1U

5.2.2.2.4. DLT_ASSIGN_ADD

Purpose	This marco is used to add a channel assignment to a registered context.
Value	0x01U

5.2.2.2.5. DLT_ASSIGN_REMOVE

Purpose	This marco is used to remove a channel assignment to a registered context.
Value	0x02U

5.2.2.2.6. DLT_CONTROL_REQUEST

Purpose	Request Control Message.
Value	0x01U

5.2.2.2.7. DLT_CONTROL_RESPONSE

Purpose	Respond Control Message.
Value	0x02U

5.2.2.2.8. DLT_CTRL_ERROR

Purpose	

Value	0x02U

5.2.2.2.9. DLT_CTRL_NOT_SUPPORTED

Purpose	
Value	0x01U

5.2.2.2.10. DLT_CTRL_OK

Purpose	
Value	0x00U

5.2.2.2.11. DLT_E_COM_FAILURE

Purpose	Development error for error in communication module.
Value	0x03U

5.2.2.2.12. DLT_E_CONTEXT_ALREADY_REG

Purpose	
Value	0x02U

5.2.2.2.13. DLT_E_CORE_SYNC_FAILED

Purpose	Development error for core synchronization.
Value	0x0BU

5.2.2.2.14. DLT_E_ERROR_IN_PROV_SERVICE

Purpose	Development error for error in provided service.
Value	0x02U

5.2.2.2.15. DLT_E_ERROR_TO_MANY_CONTEXT

Purpose	Development error for too many registered contexts.
Value	0x04U

5.2.2.2.16. DLT_E_ERROR_UNKNOWN

Purpose	
Value	0x06U

5.2.2.2.17. DLT_E_IF_BUSY

Purpose	
Value	0x05U

5.2.2.2.18. DLT_E_IF_NOT_AVAILABLE

Purpose	
Value	0x04U

5.2.2.2.19. DLT_E_MSG_LOOSE

Purpose	Development error for message buffer overflow.
Value	0x05U

5.2.2.2.20. DLT_E_MSG_TOO_LARGE

Purpose	
Value	0x01U

5.2.2.2.21. DLT_E_NOT_INITIALIZED

Purpose	Development error for wrong initialization state.

Value	0x08U

5.2.2.2.22. DLT_E_NOT_INITIALIZED_PERSISTENT

Purpose	Development error for wrong initialization state.
Value	0x09U

5.2.2.2.23. DLT_E_NOT_IN_VERBOSE_MODE

Purpose	
Value	0x08U

5.2.2.2.24. DLT_E_NOT_SUPPORTED

Purpose	This marco is used to twhron a not supported error in case the Service is not supported.
Value	0x07U

5.2.2.2.25. DLT_E_OK

Purpose	
Value	0x00U

5.2.2.2.26. DLT_E_PARAM_POINTER

Purpose	Development error for NULL pointer passed to an API.
Value	0x06U

5.2.2.2.27. DLT_E_PENDING

Purpose	
Value	0x07U

5.2.2.2.28. DLT_E_REQUEST_NOT_ACCEPTED

Purpose	Development error for read/write request unsuccessful.
Value	0x0AU

5.2.2.2.29. DLT_E_UNKNOWN_SESSION_ID

Purpose	
Value	0x03U

5.2.2.2.30. DLT_E_WRONG_PARAMETERS

Purpose	Development error for wrong parameters.
Value	0x01U

5.2.2.2.31. DLT_FILTER_MESSAGES_OFF

Purpose	
Value	0x00U

5.2.2.2.32. DLT_FILTER_MESSAGES_ON

Purpose	
Value	0x01U

5.2.2.2.33. DLT_GETLOGINFO_OPTIONS_NO_DESC

Purpose	
Value	0x06U

5.2.2.2.34. DLT_GETLOGINFO_OPTIONS_WITH_DESC

Purpose	

Value	0x07U

5.2.2.2.35. DLT_LOGGING_DISABLED

Purpose	
Value	0x00U

5.2.2.2.36. DLT_LOGGING_ENABLED

Purpose	
Value	0x01U

5.2.2.2.37. DLT_LOG_DEBUG

Purpose	Message of LogLevel type Debug.
Value	0x05U

5.2.2.2.38. DLT_LOG_DEFAULT

Purpose	Message of LogLevel type Default.
Value	0xffU

5.2.2.2.39. DLT_LOG_ERROR

Purpose	Application error.
Value	0x02U

5.2.2.2.40. DLT_LOG_FATAL

Purpose	Fatal system error.
Value	0x01U

5.2.2.2.41. DLT_LOG_INFO

Purpose	Message of LogLevel type Information.
Value	0x04U

5.2.2.2.42. DLT_LOG_OFF

Purpose	Message of LogLevel type Information.
Value	0x00U

5.2.2.2.43. DLT_LOG_VERBOSE

Purpose	Message of LogLevel type Verbose.
Value	0x06U

5.2.2.2.44. DLT_LOG_WARN

Purpose	Correct behavior cannot be ensured.
Value	0x03U

5.2.2.2.45. DLT_MODULE_ID

Purpose	AUTOSAR module identification.
Value	55U

5.2.2.2.46. DLT_NW_TRACE_CAN

Purpose	CAN Communications bus.
Value	0x02U

5.2.2.2.47. DLT_NW_TRACE_FLEXRAY

Purpose	FlexRay Communications bus.

Value	0x03U

5.2.2.2.48. DLT_NW_TRACE_IPC

Purpose	Inter-Process-Communication.
Value	0x01U

5.2.2.2.49. DLT_NW_TRACE_MOST

Purpose	Most Communications bus.
Value	0x04U

5.2.2.2.50. DLT_SID_ComCopyRxData

Purpose	Service Id for Dlt_ComCopyRxData().
Value	0x44U

5.2.2.2.51. DLT_SID_ComCopyTxData

Purpose	Service Id for Dlt_ComCopyTxData().
Value	0x43U

5.2.2.2.52. DLT_SID_ComRxIndication

Purpose	Service Id for Dlt_ComRxIndication().
Value	0x42U

5.2.2.2.53. DLT_SID_ComStartOfReception

Purpose	Service Id for Dlt_ComStartOfReception().
Value	0x46U

5.2.2.2.54. DLT_SID_GetComInterfaceMaxBandwidth

Purpose	Service Id for Dlt_GetComInterfaceMaxBandwidth().
Value	0xE8U

5.2.2.2.55. DLT_SID_GetDefaultLogLevel

Purpose	Service Id for Dlt_ASR42/43_GetDefaultLogLevel().
Value	0x18U

5.2.2.2.56. DLT_SID_GetDefaultTraceStatus

Purpose	Service Id for Dlt_ASR42/43_GetDefaultTraceStatus().
Value	0x19U

5.2.2.2.57. DLT_SID_GetEcuIdAddress

Purpose	Service Id for Dlt_AppGetEcuIdAddress().
Value	0x89U

5.2.2.2.58. DLT_SID_GetGlobalLogging

Purpose	Service Id for Dlt_GetGlobalLogging().
Value	0xADU

5.2.2.2.59. DLT_SID_GetLogChannelNames

Purpose	Service Id for Dlt_GetLogChannelNames ().
Value	0x17U

5.2.2.2.60. DLT_SID_GetLogChannelThreshold

Purpose	Service Id for Dlt_GetLogChannelThreshold ().

Value	0x22U

5.2.2.2.61. DLT_SID_GetLogInfo

Purpose	Service Id for Dlt_GetLogInfo().
Value	0x0AU

5.2.2.2.62. DLT_SID_GetLogLevel

Purpose	Service Id for Dlt_GetLogLevel().
Value	0xAAU

5.2.2.2.63. DLT_SID_GetMessageFilteringStatus

Purpose	Service Id for Dlt_GetMessageFilteringStatus().
Value	0xE6U

5.2.2.2.64. DLT_SID_GetTraceStatus

Purpose	Service Id for Dlt_ASR42/43_GetTraceStatus().
Value	0x1FU

5.2.2.2.65. DLT_SID_GetUseECUID

Purpose	Service Id for Dlt_GetUseECUID().
Value	0xE5U

5.2.2.2.66. DLT_SID_GetUseExtendedHeader

Purpose	Service Id for Dlt_GetUseExtendedHeader().
Value	0xE2U

5.2.2.2.67. DLT_SID_GetUseSessionID

Purpose	Service Id for Dlt_GetUseSessionID().
Value	0xE4U

5.2.2.2.68. DLT_SID_GetVerboseModeStatus

Purpose	Service Id for Dlt_GetVerboseModeStatus().
Value	0xE7U

5.2.2.2.69. DLT_SID_GetVersionInfo

Purpose	Service Id for Dlt_GetVersionInfo().
Value	0x02U

5.2.2.2.70. DLT_SID_Init

Purpose	Service Id for Dlt_Init().
Value	0x01U

5.2.2.2.71. DLT_SID_InternalReadFromDataSetApiId

Purpose	Internal API Id for Dlt_NvMReadRamBlockFromNvMDataSetCbk().
Value	0x84U

5.2.2.2.72. DLT_SID_InternalReadFromNativeApiId

Purpose	Internal API Id for Dlt_NvMReadRamBlockFromNvMNativeCbk().
Value	0x83U

5.2.2.2.73. DLT_SID_InternalRegisterContext

Purpose	Service Id for Dlt_InternalRegisterContext().

Value	0x86U

5.2.2.2.74. DLT_SID_InternalWriteToDataSetApiId

Purpose	Internal API Id for Dlt_NvMWriteRamBlockToNvMDataSetCbk().
Value	0x82U

5.2.2.2.75. DLT_SID_InternalWriteToNativeApiId

Purpose	Internal API Id for Dlt_NvMWriteRamBlockToNvMNativeCbk().
Value	0x81U

5.2.2.2.76. DLT_SID_NvMSingleBlockCallbackNative

Purpose	Defines API id of function Dlt_NvMSingleBlockCallbackNative().
Value	0x88U

5.2.2.2.77. DLT_SID_RegisterContext

Purpose	Service Id for Dlt_RegisterContext().
Value	0x05U

5.2.2.2.78. DLT_SID_ResetToFactoryDefault

Purpose	Service Id for Dlt_ResetToFactoryDefault().
Value	0x06U

5.2.2.2.79. DLT_SID_SendLogMessage

Purpose	Service Id for Dlt_SendLogMessage().
Value	0x03U

5.2.2.2.80. DLT_SID_SendTraceMessage

Purpose	Service Id for Dlt_SendTraceMessage().
Value	0x04U

5.2.2.2.81. DLT_SID_SetComInterfaceMaxBandwidth

Purpose	Service Id for Dlt_SetComInterfaceMaxBandwidth().
Value	0xF8U

5.2.2.2.82. DLT_SID_SetDefaultLogLevel

Purpose	Service Id for Dlt_SetDefaultLogLevel().
Value	0x11U

5.2.2.2.83. DLT_SID_SetDefaultTraceStatus

Purpose	Service Id for Dlt_ASR42/43_SetDefaultTraceStatus().
Value	0x12U

5.2.2.2.84. DLT_SID_SetGlobalLogging

Purpose	Service Id for Dlt_SetGlobalLogging().
Value	0xACU

5.2.2.2.85. DLT_SID_SetLogChannelAssignment

Purpose	Service Id for Dlt_SetLogChannelAssignment ().
Value	0x20U

5.2.2.2.86. DLT_SID_SetLogChannelThreshold

Purpose	Service Id for Dlt_SetLogChannelThreshold ().
Value	0x21U

5.2.2.2.87. DLT_SID_SetLogLevel

Purpose	Service Id for Dlt_SetLogLevel().
Value	0x08U

5.2.2.2.88. DLT_SID_SetMessageFiltering

Purpose	Service Id for Dlt_ASR42/43_SetMessageFiltering().
Value	0x1BU

5.2.2.2.89. DLT_SID_SetTraceStatus

Purpose	Service Id for Dlt_SetTraceStatus().
Value	0x09U

5.2.2.2.90. DLT_SID_SetUseECUID

Purpose	Service Id for Dlt_SetUseECUID().
Value	0xF3U

5.2.2.2.91. DLT_SID_SetUseExtendedHeader

Purpose	Service Id for Dlt_SetUseExtendedHeader().
Value	0xF0U

5.2.2.2.92. DLT_SID_SetUseSessionID

Purpose	Service Id for Dlt_SetUseSessionID().
Value	0xF2U

5.2.2.2.93. DLT_SID_SetVerboseMode

Purpose	Service Id for Dlt_SetVerboseMode().
Value	0x93U

5.2.2.2.94. DLT_SID_SingleBlockCallbackDataSet

Purpose	Service Id for Dlt_NvMSingleBlockCallbackDataSet().
Value	0x85U

5.2.2.2.95. DLT_SID_StoreConfiguration

Purpose	Service Id for Dlt_StoreConfiguration().
Value	0x1AU

5.2.2.2.96. DLT_SID_StorePersistent

Purpose	Service Id for Dlt_StorePersistent().
Value	0x87U

5.2.2.2.97. DLT_SID_TriggerTransmit

Purpose	Service Id for Dlt_TriggerTransmit().
Value	0x41U

5.2.2.2.98. DLT_SID_UnregisterContext

Purpose	Service Id for Dlt_UnregisterContext().
Value	0x16U

5.2.2.2.99. DLT_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	1U

5.2.2.2.100. DLT_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	8U

5.2.2.2.101. DLT_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	7U

5.2.2.2.102. DLT_TRACE_FUNCTION_IN

Purpose	Call of a function.
Value	0x02U

5.2.2.2.103. DLT_TRACE_FUNCTION_OUT

Purpose	Return of a function.
Value	0x03U

5.2.2.2.104. DLT_TRACE_STATE

Purpose	State of a State Machine.
Value	0x04U

5.2.2.2.105. DLT_TRACE_STATUS_DEFAULT

Purpose	
Value	0xffU

5.2.2.2.106. DLT_TRACE_STATUS_OFF

Purpose	
Value	0x00U

5.2.2.2.107. DLT_TRACE_STATUS_ON

Purpose	
Value	0x01U

5.2.2.2.108. DLT_TRACE_VARIABLE

Purpose	Value of variable.
Value	0x01U

5.2.2.2.109. DLT_TRACE_VFB

Purpose	RTE events.
Value	0x05U

5.2.2.2.110. DLT_TYPE_APP_TRACE

Purpose	A trace message.
Value	0x01U

5.2.2.2.111. DLT_TYPE_CONTROL

Purpose	A message for internal use/control send between Dlt module and external client.
Value	0x03U

5.2.2.2.112. DLT_TYPE_LOG

Purpose	A log message.
Value	0x00U

5.2.2.2.113. DLT_TYPE_NW_TRACE

Purpose	A message forwarded from a communication bus (like CAN, FlexRay).
Value	0x02U

5.2.2.2.114. DLT_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.2.2.2.115. Dlt_GetDefaultLogLevel

Purpose	
Value	Dlt_ASR43_GetDefaultLogLevel

5.2.2.2.116. Dlt_GetDefaultTraceStatus

Purpose	
Value	Dlt_ASR43_GetDefaultTraceStatus

5.2.2.2.117. Dlt_GetLogChannelNames

Purpose	
Value	Dlt_ASR43_GetLogChannelNames

5.2.2.2.118. Dlt_GetLogChannelThreshold

Purpose	
Value	Dlt_ASR43_GetLogChannelThreshold

5.2.2.2.119. Dlt_GetLogInfo

Purpose	
Value	Dlt_ASR43_GetLogInfo

5.2.2.2.120. Dlt_GetTraceStatus

Purpose	
Value	Dlt_ASR43_GetTraceStatus

5.2.2.2.121. Dlt_NW_TRACE_ETHERNET

Purpose	Ethernet Communications bus.
Value	0x05U

5.2.2.2.122. Dlt_NW_TRACE_SOMEIP

Purpose	Inter-SOME/IP Communication.
Value	0x06U

5.2.2.2.123. Dlt_RegisterContext

Purpose	
Value	Dlt_ASR43_RegisterContext

5.2.2.2.124. Dlt_ResetToFactoryDefault

Purpose	
Value	Dlt_ASR43_ResetToFactoryDefault

5.2.2.2.125. Dlt_SendLogMessage

Purpose	
Value	Dlt_ASR43_SendLogMessage

5.2.2.2.126. Dlt_SendTraceMessage

Purpose	
Value	Dlt_ASR43_SendTraceMessage

5.2.2.2.127. Dlt_SetDefaultLogLevel

Purpose	
Value	Dlt_ASR43_SetDefaultLogLevel

5.2.2.2.128. Dlt_SetDefaultTraceStatus

Purpose	
Value	Dlt_ASR43_SetDefaultTraceStatus

5.2.2.2.129. Dlt_SetLogChannelAssignment

Purpose	
Value	Dlt_ASR43_SetLogChannelAssignment

5.2.2.2.130. Dlt_SetLogChannelThreshold

Purpose	
Value	Dlt_ASR43_SetLogChannelThreshold

5.2.2.2.131. Dlt_SetLogLevel

Purpose	
Value	Dlt_ASR43_SetLogLevel

5.2.2.2.132. Dlt_SetMessageFiltering

Purpose	
Value	Dlt_ASR43_SetMessageFiltering

5.2.2.2.133. Dlt_SetTraceStatus

Purpose	
Value	Dlt_ASR43_SetTraceStatus

5.2.2.2.134. Dlt_StoreConfiguration

Purpose	
Value	Dlt_ASR43_StoreConfiguration

5.2.2.2.135. Dlt_UnregisterContext

Purpose	
Value	Dlt_ASR43_UnregisterContext

5.2.2.3. Functions

5.2.2.3.1. Dlt_ASR42_GetDefaultLogLevel

Purpose	Get default log level.
Synopsis	`Dlt_MessageLogLevelType` **`Dlt_ASR42_GetDefaultLogLevel`** `( void );`
Service ID	DLT_SID_GetDefaultLogLevel
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	Default log level setting
Description	This service gets the default log level applied to each context for which the log level has not been set explicitly.

5.2.2.3.2. Dlt_ASR42_GetDefaultTraceStatus

Purpose	Get default trace status.
Synopsis	`Dlt_MessageTraceStatusType` **`Dlt_ASR42_GetDefaultTraceStatus`** `( void );`
Service ID	DLT_SID_GetDefaultTraceStatus
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	Default trace status setting
Description	This service gets the default trace status applied to each context for which the trace status has not been set explicitly.

5.2.2.3.3. Dlt_ASR42_GetTraceStatus

Purpose	Get trace status.
Synopsis	`Dlt_CtrlReturnType` **`Dlt_ASR42_GetTraceStatus`** `( Dlt_Internal_ApplicationIDType AppId , Dlt_Internal_ContextIDType ContextId , Dlt_MessageTraceStatusType * TraceStatus );`
Service ID	DLT_SID_GetTraceStatus
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (in)	`AppId`	The application Id for which the trace status shall be retrieved
	`ContextId`	The context Id for which the trace status shall be retrieved
Parameters (out)	`TraceStatus`	Trace status of the given application and context Ids
Return Value	Result of retrieving the trace status	
	`DLT_CTRL_OK`	Retrieving the trace status succeeded
	`DLT_CTRL_ERROR`	Retrieving the trace status failed
Description	This service returns the trace status setting for a given context	

5.2.2.3.4. Dlt_ASR42_SetDefaultLogLevel

Purpose	Set default log level.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_ASR42_SetDefaultLogLevel`** `( Dlt_MessageLogLevelType NewLevel );`	
Service ID	DLT_SID_SetDefaultLogLevel	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`NewLevel`	The new default log level
Return Value	Result of changing the log level	
	`DLT_CTRL_OK`	Changing the log level succeeded
	`DLT_CTRL_ERROR`	Changing the log level failed
Description	This service sets the default log level applied to each context for which the log level has not been set explicitly.	

5.2.2.3.5. Dlt_ASR42_SetDefaultTraceStatus

Purpose	Set default trace status.
Synopsis	`Dlt_CtrlReturnType` **`Dlt_ASR42_SetDefaultTraceStatus`** `( Dlt_MessageTraceStatusType NewStatus );`
Service ID	DLT_SID_SetDefaultTraceStatus
Sync/Async	Synchronous

Reentrancy	Non Reentrant	
Parameters (in)	`NewStatus`	The new default trace status
Return Value	Result of changing the trace status	
	`DLT_CTRL_OK`	Changing the trace status succeeded
	`DLT_CTRL_ERROR`	Changing the trace status failed
Description	This service sets the default trace status applied to each context for which the trace status has not been set explicitly.	

5.2.2.3.6. Dlt_ASR42_SetLogLevel

Purpose	Set log level.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_ASR42_SetLogLevel`** `( Dlt_Internal_ApplicationIDType AppId , Dlt_Internal_ContextIDType ContextId , Dlt_MessageLogLevelType NewLevel );`	
Service ID	DLT_SID_SetLogLevel	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`AppId`	The application Id for which the log level is set
	`ContextId`	The context Id for which the log level is set
	`NewLevel`	The new log level for the given application and context Ids
Return Value	Result of changing the log level	
	`DLT_CTRL_OK`	Changing the log level succeeded
	`DLT_CTRL_ERROR`	Changing the log level failed
Description	This service sets the maximum level a log message may have to be accepted by the Dlt module.	

5.2.2.3.7. Dlt_ASR42_SetMessageFiltering

Purpose	Enable / disable message filtering.
Synopsis	`Dlt_CtrlReturnType` **`Dlt_ASR42_SetMessageFiltering`** `( Dlt_FilterMessagesType NewStatus );`

Service ID	DLT_SID_SetMessageFiltering	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`NewStatus`	New status of message filtering DLT_FILTER_MESSAGES_ON: enable message filtering DLT_FILTER_MESSAGES_OFF: disable message filtering
Return Value	Result of changing the message filtering status	
	`DLT_CTRL_OK`	Changing the message filtering status succeeded
	`DLT_CTRL_ERROR`	Changing the message filtering status failed
Description	This function enables and disables message filtering. If message filtering is disabled, all messages are processed by the Dlt, independently from the log level or trace status settings.	

5.2.2.3.8. Dlt_ASR42_SetTraceStatus

Purpose	Set trace status.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_ASR42_SetTraceStatus`** `( Dlt_Internal_ApplicationIDType AppId , Dlt_Internal_ContextIDType ContextId , Dlt_MessageTraceStatusType NewStatus );`	
Service ID	DLT_SID_SetTraceStatus	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`AppId`	The application Id for which the trace status is set
	`ContextId`	The context Id for which the trace status is set
	`NewStatus`	The new trace status for the given application and context Ids
Return Value	Result of changing the trace status	
	`DLT_CTRL_OK`	Changing the trace status succeeded
	`DLT_CTRL_ERROR`	Changing the trace status failed
Description	This service enables or disables the processing of trace messages by the Dlt module.	

5.2.2.3.9. Dlt_ASR43_GetDefaultLogLevel

Purpose	Get default log level.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_GetDefaultLogLevel`** `( Dlt_MessageLogLevelType * defaultLogLevel );`	
Parameters (out)	`defaultLogLevel`	Returns the stored LogLevel setting
Return Value	Default log level setting	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	The default LogLevel could not be returned
Description	Returns the Default Log Level currently used by the Dlt module. The returned Log Level might differ from the one which is stored non volatile. ServiceID{DLT_SID_GetDefaultLogLevel} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.10. Dlt_ASR43_GetDefaultTraceStatus

Purpose	Get default trace status.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_GetDefaultTraceStatus`** `( boolean * traceStatus );`	
Parameters (out)	`traceStatus`	current global default trace status (enabled/disabled)
Return Value	Default trace status setting	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	Default Trace Status could not be returned
Description	Returns the current global default trace status. ServiceID{DLT_SID_GetDefaultTraceStatus} Reentrancy{Reentrant} Synchronicity{Synchronous}	

5.2.2.3.11. Dlt_ASR43_GetLogChannelNames

Purpose	Get Log Channel Names.

Synopsis	`Std_ReturnType` **`Dlt_ASR43_GetLogChannelNames`** `( uint8 * numberOfLogChannels , Dlt_LogChannelNameArrayType * logChannelNames );`	
Parameters (out)	`numberOfLogChannels`	Returns the number of configured LogChannels
	`logChannelNames`	Returns a list of configured LogChannel names
Return Value	Result of changing the log level	
	`DLT_CTRL_OK`	Changing the log level succeeded
	`DLT_CTRL_ERROR`	Changing the log level failed
Description	Returns the filter threshold for the given LogChannel. ServiceID{DLT_SID_GetLogChannelNames} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.12. Dlt_ASR43_GetLogChannelThreshold

Purpose	Get Log Channel Threshold.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_GetLogChannelThreshold`** `( const Dlt_LogChannelNameType * logChannelName , Dlt_MessageLogLevelType * logChannelThreshold , boolean * traceStatus );`	
Parameters (in)	`logChannelName`	Addressed LogChannel name
Parameters (out)	`logChannelThreshold`	current LogChannelTreshold
	`traceStatus`	Current TraceStatus. TRUE: TraceMessages enabled. FALSE: TraceMessages disabled.
Return Value	Result of changing the log level	
	`DLT_CTRL_OK`	Changing the log level succeeded
	`DLT_CTRL_ERROR`	Changing the log level failed
Description	Returns the filter threshold for the given LogChannel. ServiceID{DLT_SID_GetLogChannelThreshold} Reentrancy{Reentrant for different LogChannelNames} Synchronicity{Synchronous}	

5.2.2.3.13. Dlt_ASR43_GetLogInfo

Purpose	Get log info.

Synopsis	`Std_ReturnType` **`Dlt_ASR43_GetLogInfo`** `( uint8 options , const Dlt_ApplicationIDType * appId , const Dlt_ContextIDType * contextId , uint8 * status , uint8 * logInfo );`	
Parameters (in)	`options`	Used to filter the response in respect to the ApplicationId, ContextId and Trace Status information
	`appId`	Representation of the ApplicationId
	`contextId`	Representation of the ContextId
Parameters (out)	`status`	
	`logInfo`	Details about the returned Application IDs
Return Value	Result of retrieving the log info	
	`E_OK`	No error occurred
	`E_NOT_OK`	LogInfo could not be returned
Description	Called to request information about registered ApplicationIds, their ContextIds and the corresponding log level. ServiceID{DLT_SID_GetLogInfo} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.14. Dlt_ASR43_GetTraceStatus

Purpose	Get trace status.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_GetTraceStatus`** `( const Dlt_ApplicationIDType * appId , const Dlt_ContextIDType * contextId , boolean * traceStatus );`	
Parameters (in)	`appId`	ApplicationId
	`contextId`	ContextId
Parameters (out)	`traceStatus`	current Trace Status of the tuple ApplicationId/ContextId
Return Value	Result of retrieving the trace status	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	TraceStatus could not be returned
Description	Returns the current Trace Status for a given tuple ApplicationId/ContextId. ServiceID{DLT_SID_GetTraceStatus} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.15. Dlt_ASR43_RegisterContext

Purpose	Register new logging or tracing context.	
Synopsis	`Dlt_ReturnType` **`Dlt_ASR43_RegisterContext`** `( Dlt_SessionIDType session_id , const Dlt_ApplicationIDType * app_id , const Dlt_ContextIDType * context_id , const uint8 * app_description , uint8 len_app_description , const uint8 * context_description , uint8 len_context_description );`	
Parameters (in)	`session_id`	number of the module (Module ID within BSW, port defined argument value within SW-C)
	`app_id`	the Application ID
	`context_id`	the Context ID
	`app_description`	Points to description string for the provided application id. At maximum 255 characters are interpreted.
	`len_app_description`	The length of the description for the application id string (number of characters of description string).
	`context_description`	Points to description string for the provided context id. At maximum 255 characters are interpreted.
	`len_context_description`	The length of the description string (number of characters of description string).
Return Value	Result of registering the context	
	`Autosar_4.2`	DLT_E_IF_NOT_AVAILABLE The module has not been initialized, thus the interface is not available
	`Autosar_4.2`	DLT_E_ERROR_UNKNOWN Too many contexts have been registered
	`Autosar_4.3`	or below version DLT_E_CONTEXT_ALREADY_REG The software module context has already been registered
	`Autosar_4.3`	or below version DLT_E_UNKNOWN_SESSION_ID The session Id is unknown, i.e. it is not specified by any SW-C In Autosar_4.3 version, each time a registration of a context was not succeeded because of Dlt not be-

		ing initialized or because of a overrun number of registered contexts, or of a wrong parameter used, the DLT_E_UNKNOWN_SESSION_ID shall be returned.
	DLT_E_OK	The required operation succeeded
Description	The service has to be called when a software module wants to use services offered by DLT software component for a specific context. If a context id is being registered for an already registered application id then app_description can be NULL and len_app_description zero. ServiceID{DLT_SID_RegisterContext} Reentrancy{Reentrant} Synchronicity{Synchronous} Possible development errors DLT_E_ERROR_TO_MANY_CONTEXT - the number of contexts to be registered has reached the maximum value DLT_E_NOT_INITIALIZED - Api was called before initializing Dlt module DLT_E_PARAM_POINTER - null pointer was used for context_description DLT_E_WRONG_PARAMETERS -wrong len_context_description was used DLT_E_UNKNOWN_SESSION_ID - wrong session id was used	

5.2.2.3.16. Dlt_ASR43_ResetToFactoryDefault

Purpose	Reset the configuration to factory defaults.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_ResetToFactoryDefault`** `( void );`	
Return Value	Result of resetting the configuration to factory defaults	
	`E_OK:`	Configuration has been reset successfully
	`E_NOT_OK:`	Configuration has not been reset
Description	The service Dlt_ResetToFactoryDefault sets the LogLevel and TraceStatus back to the persistently stored default values. If the feature NvMRAM support is enabled, all stored Dlt values in the NvM are deleted. ServiceID{DLT_SID_ResetToFactoryDefault} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.17. Dlt_ASR43_SendLogMessage

Purpose	Send a log message to the DLT module.
Synopsis	`Dlt_ReturnType` **`Dlt_ASR43_SendLogMessage`** `( Dlt_SessionIDType session_id , const Dlt_MessageLogInfoType * log_info , const uint8 * log_data , uint16 log_data_length );`

Parameters (in)	session_id	For SW-C this is not visible (Port defined argument value), for BSW-modules it is the module number
	log_info	Structure containing the relevant information for filtering the message
	log_data	Buffer containing the parameters to be logged. The contents of this pointer represents the payload of the send log message
	log_data_length	Length of the data buffer log_data
Return Value	Result of sending the log message	
	DLT_E_OK	The required operation succeeded
	DLT_E_MSG_TOO_LARGE	The message is too large for sending over the network
	DLT_E_IF_NOT_AVAILABLE	The interface is not available
	DLT_E_UNKNOWN_SESSION_ID	The provided session id is unknown
	DLT_E_NOT_IN_VERBOSE_MODE	Unable to send the message in verbose mode
Description	The service represents the interface to be used by basic software modules or by software component to send log messages. ServiceID{DLT_SID_SendLogMessage} Reentrancy{Reentrant} Synchronicity{Synchronous} Possible development errors: ▶ DLT_E_PARAM_POINTER - in case a null pointer was used for log_info or log_data input ▶ DLT_E_WRONG_PARAMETERS - a wrong parameter is used in any case ▶ DLT_E_UNKNOWN_SESSION_ID - session_id is used with a wrong/not registered value ▶ DLT_E_NO_BUFFER - log_data_length value is greater than configured maximum length for log messages	

5.2.2.3.18. Dlt_ASR43_SendTraceMessage

Purpose	Send a trace message to the DLT module.
Synopsis	`Dlt_ReturnType Dlt_ASR43_SendTraceMessage ( Dlt_SessionIDType session_id , const Dlt_MessageTraceInfoType * trace_info , const uint8 * trace_data , uint16 trace_data_length );`

Parameters (in)	`session_id`	number of the module (Module ID within BSW, port defined argument value within SW-C)
	`trace_info`	Structure containing the relevant information for filtering the message
	`trace_data`	Buffer containing the parameters to be traced. The contents of this pointer represents the payload of the send log message
	`trace_data_length`	Length of the data buffer trace_data
Return Value	Result of sending the trace message	
	`DLT_E_OK`	The required operation succeeded
	`DLT_E_MSG_TOO_LARGE`	The message is too large for sending over the network
	`DLT_E_IF_NOT_AVAILABLE`	The interface is not available
	`DLT_E_UNKNOWN_SESSION_ID`	The provided session id is unknown
	`DLT_E_NOT_IN_VERBOSE_MODE`	Unable to send the message in verbose mode
Description	The service represents the interface to be used by basic software modules or by software component to trace parameters. ServiceID{DLT_SID_SendTraceMessage} Reentrancy{Reentrant} Synchronicity{Synchronous} Possible development errors: ▶ DLT_E_PARAM_POINTER - in case a null pointer was used for log_info or log_data input ▶ DLT_E_WRONG_PARAMETERS - a wrong parameter is used in any case ▶ DLT_E_UNKNOWN_SESSION_ID - session_id is used with a wrong/not registered value ▶ DLT_E_NO_BUFFER - log_data_length value is greater than configured maximum length for log messages	

5.2.2.3.19. Dlt_ASR43_SetDefaultLogLevel

Purpose	Set default log level.
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetDefaultLogLevel`** `( Dlt_MessageLogLevelType newLogLevel );`

Parameters (in)	`newLogLevel`	sets the new filter value
Return Value	Result of changing the log level	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	Default LogLevel could not be set
Description	Called to modify the pass through range for Log Messages for all not explicit set ContextIds. ServiceID{DLT_SID_SetDefaultLogLevel} Reentrancy{Reentrant} Synchronicity{Synchronous}	

5.2.2.3.20. Dlt_ASR43_SetDefaultTraceStatus

Purpose	Set default trace status.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetDefaultTraceStatus`** `( boolean newTraceStatus );`	
Parameters (in)	`newTraceStatus`	enabling/disabling of Trace messages
Return Value	Result of changing the trace status	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	Default Trace Status could not be set
Description	Called to enable or disable trace messages for all not explicitly set ContextIds. ServiceID{DLT_SID_SetDefaultTraceStatus} Reentrancy{Reentrant} Synchronicity{Synchronous}	

5.2.2.3.21. Dlt_ASR43_SetLogChannelAssignment

Purpose	Set log channel assignment.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetLogChannelAssignment`** `( const Dlt_ApplicationIDType * appId , const Dlt_ContextIDType * contextId , const Dlt_LogChannelNameType * logChannelName , Dlt_AssignmentOperationType addRemoveOp );`	
Parameters (in)	`appId`	ID of the addressed application / SW-C
	`contextId`	The context Id for which the log level is set
	`logChannelName`	Name of the addressed LogChannel

	addRemoveOp	Operation to add/remove the addressed tuple ApplicationId/ContextId to/from the addressed LogChannel
Return Value	Result of changing the log level	
	DLT_CTRL_OK	Changing the log level succeeded
	DLT_CTRL_ERROR	Changing the log level failed
Description	Adds/removes the addressed tuple ApplicationId/ContextId to/from the addressed LogChannel. ServiceID{DLT_SID_SetLogChannelAssignment} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.22. Dlt_ASR43_SetLogChannelThreshold

Purpose	Set log Channel Threshold.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetLogChannelThreshold`** `( const Dlt_LogChannelNameType * logChannelName , Dlt_MessageLogLevelType newThreshold , boolean newTraceStatus );`	
Parameters (in)	logChannelName	Name of the addressed LogChannel
	newThreshold	Threshold for LogMessages
	newTraceStatus	TRUE: enable TraceMessages FALSE: disable TraceMessages
Return Value	Result of changing the log level	
	DLT_CTRL_OK	Changing the log level succeeded
	DLT_CTRL_ERROR	Changing the log level failed
Description	Sets the filter threshold for the given LogChannel. ServiceID{DLT_SID_SetLogChannelThreshold} Reentrancy{Reentrant for different LogChannelNames} Synchronicity{Synchronous}	

5.2.2.3.23. Dlt_ASR43_SetLogLevel

Purpose	Set log level.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetLogLevel`** `( const Dlt_ApplicationIDType * appId , const Dlt_ContextIDType * contextId , Dlt_MessageLogLevelType newLogLevel );`	
Parameters (in)	appId	ID of the SW-C
	contextId	ID of the context

	`newLogLevel`	new log level to set
Return Value	Result of changing the log level	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	LogLevel could not be changed
Description	This service is used to change the LogLevel for the given tuple of ApplicationID/ContextID. ServiceID{DLT_SID_SetLogLevel} Reentrancy{Reentrant} Synchronicity{Synchronous}	

5.2.2.3.24. Dlt_ASR43_SetMessageFiltering

Purpose	Enable / disable message filtering.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetMessageFiltering`** `( boolean status );`	
Parameters (in)	`status`	TRUE: enable message filtering FALSE: disable message filtering
Return Value	Result of changing the message filtering status	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	Setting of message filtering failed
Description	Switches on/off the message filtering functionality of the Dlt module. ServiceID{DLT_SID_SetMessageFiltering} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.25. Dlt_ASR43_SetTraceStatus

Purpose	Set trace status.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_SetTraceStatus`** `( const Dlt_ApplicationIDType * appId , const Dlt_ContextIDType * contextId , boolean newTraceStatus );`	
Parameters (in)	`appId`	ID of the SW-C
	`contextId`	ID of the context
	`newTraceStatus`	New trace status
Return Value	Result of changing the trace status	

	`E_OK:`	No error occurred
	`E_NOT_OK:`	Trace status could not be changed
Description	The service Dlt_SetTraceStatus sets the trace status for a specific tuple of ApplicationID and ContextID. ServiceID{DLT_SID_SetTraceStatus} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.26. Dlt_ASR43_StoreConfiguration

Purpose	Store the general configuration, log level and trace status set for a pair of Application ID and Context ID in NvM blocks.	
Synopsis	`Std_ReturnType` **`Dlt_ASR43_StoreConfiguration`** `( void );`	
Return Value	Result of configuration storage	
	`E_OK:`	No error occurred
	`E_NOT_OK:`	The configuration could not be stored
	`DLT_E_NOT_SUPPORTED:`	Service is not supported
Description	Copies the current Dlt configuration to NvRAM by calling NvM_WriteBlock(). ServiceID{DLT_SID_StoreConfiguration} Reentrancy{Non Reentrant} Synchronicity{Synchronous}	

5.2.2.3.27. Dlt_ASR43_UnregisterContext

Purpose	Unregister already logged application/context ids.	
Synopsis	`Dlt_ReturnType` **`Dlt_ASR43_UnregisterContext`** `( Dlt_SessionIDType sessionId , const Dlt_ApplicationIDType * appId , const Dlt_ContextIDType * contextId );`	
Parameters (in)	`sessionId`	number of the module (Module ID within BSW, port defined argument value within SW-C)
	`appId`	the Application ID
	`contextId`	the Context ID
Return Value	Result of unregistering the context	
	`DLT_E_CONTEXT_NOT_YET_REG:`	The software module context has not registered before.

	DLT_E_UNKNOWN_SESSION_ID:	The provided session id is unknown.
	DLT_E_OK	The required operation succeeded
Description	The service has to be called when a software module wants to use services offered by DLT software component for a specific context. ServiceID{DLT_SID_UnregisterContext} Reentrancy{Reentrant} Synchronicity{Synchronous} Possible development errors DLT_E_NOT_INITIALIZED - Api was called before initializing Dlt module DLT_E_CONTEXT_NOT_YET_REG - context was not registered yet DLT_E_UNKNOWN_SESSION_ID - wrong session id was used	

5.2.2.3.28. Dlt_ComCopyRxData

Purpose	Copy received data from PduR buffer.	
Synopsis	`BufReq_ReturnType Dlt_ComCopyRxData ( PduIdType id , const PduInfoType * info , PduLengthType * bufferSizePtr );`	
Service ID	DLT_SID_ComCopyRxData	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduDataLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied
Return Value	The success state of Dlt_ComCopyRxData	
	BUFREQ_OK	Data copied successfully
	BUFREQ_E_NOT_OK	Data was not copied because an error occurred.
Description	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. 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.	

5.2.2.3.29. Dlt_ComCopyTxData

Purpose	Copy transmit data to PduR buffer.	
Synopsis	`BufReq_ReturnType` **`Dlt_ComCopyTxData`** `( PduIdType id , PduInfoType * info , RetryInfoType * retry , PduLengthType * availableDataPtr );`	
Parameters (in)	`id`	Identification of the transmitted I-PDU
	`info`	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. 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 upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
	`retry`	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmitted data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt speci-

		fies the offset in bytes from the current data copy position.
Parameters (out)	`availableDataPtr`	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIso Tp) to determine the size of the following CFs.
Return Value	The success state of Dlt_ComCopyTxData	
	`BUFREQ_OK`	Data has been copied to the transmit buffer completely as requested.
	`BUFREQ_E_BUSY`	Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied.
	`BUFREQ_E_NOT_OK`	Data has not been copied. Request failed.
Description	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case, the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	

5.2.2.3.30. Dlt_ComRxIndication

Purpose	
Synopsis	`void` **`Dlt_ComRxIndication`** `( PduIdType DltRxPduId , NotifResultType Result );`

5.2.2.3.31. Dlt_ComStartOfReception

Purpose	Trigger the reception of a message.
Synopsis	`BufReq_ReturnType` **`Dlt_ComStartOfReception`** `( PduIdType id , PduLengthType TpSduLength , PduLengthType * bufferSizePtr );`
Service ID	DLT_SID_ComStartOfReception
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (in)	`id`	Identification of the I-PDU.
	`TpSduLength`	Total length of the N-SDU to be received.
Parameters (out)	`bufferSizePtr`	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return Value	The success state of Dlt_ComStartofReception	
	`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 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.
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).	

5.2.2.3.32. Dlt_ComTxConfirmation

Purpose	
Synopsis	`void` **`Dlt_ComTxConfirmation`** `( PduIdType DltTxPduId , NotifResultType Result );`

5.2.2.3.33. Dlt_GetComInterfaceMaxBandwidth

Purpose	Get maximum bandwidth for the communication interface.
Synopsis	`uint32` **`Dlt_GetComInterfaceMaxBandwidth`** `( void );`
Service ID	DLT_SID_GetComInterfaceMaxBandwidth

Sync/Async	Synchronous
Reentrancy	Non Reentrant
Return Value	The current maximum bandwidth in bits / second
Description	This function returns the maximum bandwidth available for the Dlt on the communication interface. If the Dlt module is not initialized, no messages will be sent (only stored in the internal buffer), thus this function will return 0 in this case.

5.2.2.3.34. Dlt_GetGlobalLogging

Purpose	Get global message logging status.	
Synopsis	`Dlt_GlobalLogStatusType` **`Dlt_GetGlobalLogging`** `( void );`	
Service ID	DLT_SID_GetGlobalLogging	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Return Value	Current message logging status	
	`DLT_LOGGING_ENABLED`	Message logging is enabled
	`DLT_LOGGING_DISABLED`	Message logging is disabled
Description	This function retrieves the current message logging status.	

5.2.2.3.35. Dlt_GetLogInfoInternal

Purpose	Get log info internal.	
Synopsis	`Std_ReturnType` **`Dlt_GetLogInfoInternal`** `( Dlt_SessionIDType SessionID , uint8 options , Dlt_Internal_ApplicationIDType AppID , Dlt_Internal_ContextIDType ContextID , uint8 * status , uint8 * logInfo );`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`SessionID`	Used to distinguish identical tuples
	`options`	Used to filter the response in respect to the ApplicationId, ContextId and Trace Status information
	`AppID`	Representation of the ApplicationId

	ContextID	Representation of the ContextId
Parameters (out)	status	
	logInfo	Details about the returned Application IDs
Return Value	Result of retrieving the log info	
	E_OK	No error occurred
	E_NOT_OK	LogInfo could not be returned
Description	Called to request information about registered ApplicationIds, providing support to Dlt_GetLogInfo by extending the API with a session ID parameter where available.	

5.2.2.3.36. Dlt_GetLogLevel

Purpose	Get log level.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_GetLogLevel`** `( const Dlt_ApplicationIDType * AppId , const Dlt_ContextIDType * ContextId , Dlt_MessageLogLevelType * LogLevel );`	
Service ID	DLT_SID_GetLogLevel	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	AppId	The application Id for which the log level shall be retrieved
	ContextId	The context Id for which the log level shall be retrieved
Parameters (out)	LogLevel	Log level setting for the given application and context Ids
Return Value	Result of retrieving the log level	
	DLT_CTRL_OK	Retrieving the log level succeeded
	DLT_CTRL_ERROR	Retrieving the log level failed
Description	This service returns the maximum level a log message may have to be accepted by the Dlt module.	

5.2.2.3.37. Dlt_GetMessageFilteringStatus

Purpose	Get message filtering status.

Synopsis	`Dlt_FilterMessagesType` **`Dlt_GetMessageFilteringStatus`** `( void );`	
Service ID	DLT_SID_GetMessageFilteringStatus	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Return Value	Current message filtering status	
	`DLT_FILTER_MESSAGES_ON`	Message filtering is enabled
	`DLT_FILTER_MESSAGES_OFF`	Message filtering is disabled
Description	This function retrieves the current message filtering status.	

5.2.2.3.38. Dlt_GetUseECUID

Purpose	Get EcuID header field status.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_GetUseECUID`** `( void );`	
Service ID	DLT_SID_GetUseECUID	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Return Value	Current EcuID header field status	
	`1`	EcuID is included in the Dlt message header
	`0`	EcuID is not included in the Dlt message header
	`2`	Call to the API to retrieve EcuID usage failed
Description	This function returns whether the EcuID is included in the Dlt message header (Used to get status of DltHeaderUseEcuId).	

5.2.2.3.39. Dlt_GetUseExtendedHeader

Purpose	Get extended header use status.
Synopsis	`Dlt_CtrlReturnType` **`Dlt_GetUseExtendedHeader`** `( void );`
Service ID	DLT_SID_GetUseExtendedHeader
Sync/Async	Synchronous

Reentrancy	Non Reentrant	
Return Value	Current Dlt header format status	
	`1`	Extended Dlt message headers are used
	`0`	Standard Dlt message headers are used
	`2`	Call to API failed to retrieve status of extended header usage.
Description	This function returns whether extended Dlt message headers are used.	

5.2.2.3.40. Dlt_GetUseSessionID

Purpose	Get SessionID header field status.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_GetUseSessionID`** `( void );`	
Service ID	DLT_SID_GetUseSessionID	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Return Value	Current SessionID header field status	
	`1`	SessionID is included in the Dlt message header
	`0`	SessionID is not included in the Dlt message header
	`2`	Call to the API to retrieve SessionID usage failed
Description	This function returns whether the SessionID is included in the Dlt message header.	

5.2.2.3.41. Dlt_GetVerboseModeStatus

Purpose	Get verbose mode status.
Synopsis	`Dlt_CtrlReturnType` **`Dlt_GetVerboseModeStatus`** `( const Dlt_ApplicationIDType * AppId , const Dlt_ContextIDType * ContextId , Dlt_MessageLogLevelType * ModeStatus );`
Service ID	DLT_SID_GetVerboseModeStatus
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (in)	`AppId`	The application Id for which the verbose mode is set
	`ContextId`	The context Id for which the verbose mode is set
Parameters (out)	`ModeStatus`	Mode setting for the given application and context Ids
Return Value	Result of retrieving the mode	
	`DLT_CTRL_OK`	Retrieving the mode succeeded
	`DLT_CTRL_ERROR`	Retrieving the mode failed
Description	This function returns whether the verbose mode is ON or OFF for a given application and context Id pair	

5.2.2.3.42. Dlt_GetVersionInfo

Purpose	This service returns the version information of this module.	
Synopsis	`void` **`Dlt_GetVersionInfo`** `( Std_VersionInfoType * versioninfo );`	
Service ID	DLT_SID_GetVersionInfo	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`versioninfo`	Pointer to where to store the version information of this module.
Description	The version information includes: ► Module Id ► Vendor Id ► Vendor specific version numbers (BSW00407). This function shall be pre-compile time configurable On/Off by the configuration parameter: DLT_VERSION_INFO_API	

5.2.2.3.43. Dlt_Init

Purpose	This service initializes the Dlt module.
Synopsis	`void` **`Dlt_Init`** `( const Dlt_ConfigType * config );`

Service ID	DLT_SID_Init	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`config`	Pointer to a DLT configuration structure
Description	Dlt is using the NVRamManager and is to be initialized very late in the ECU startup phase. The Dlt_init() function should be called after the NVRamManager is initialized.	

5.2.2.3.44. Dlt_MainFunction

Purpose	Periodically trigger message transmission.
Synopsis	`void` **`Dlt_MainFunction`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Description	This function periodically triggers the transmission of Dlt frames to the PduR. The function also performs bandwidth management tasks.

5.2.2.3.45. Dlt_NvMInitDataSetBlockCbk

Purpose	Callback-Function from NvM that will be called if no ROM data is available for initialization of the NVRAM block.
Synopsis	`Std_ReturnType` **`Dlt_NvMInitDataSetBlockCbk`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non reentrant
Return Value	returns E_OK

5.2.2.3.46. Dlt_NvMInitNativeBlockCbk

Purpose	Callback-Function from NvM that will be called if no ROM data is available for initialization of the NVRAM block.
Synopsis	`Std_ReturnType` **`Dlt_NvMInitNativeBlockCbk`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non reentrant
Return Value	returns E_OK

5.2.2.3.47. Dlt_NvMReadRamBlockFromNvMDataSetCbk

Purpose	Callback function from NvM for data set block to request the data to be copied from NvM RAM during NvM_ReadBlock().	
Synopsis	`Std_ReturnType` **`Dlt_NvMReadRamBlockFromNvMDataSetCbk`** `( const void * NvMBuffer );`	
Service ID	DLT_SID_InternalReadFromDataSetApiId	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`NvMBuffer`	Pointer to NvM RAM mirror
Return Value	always E_OK	
Description	This callback function shall copy information stored in the data set NvM block (the values of log level and trace status explicitly set by a pair of ApplicationID and ContextID registered via Dlt_RegisterContext) to the Dlt RAM. The approach used in this case is to copy the information one by one thus not using TS_MemCpy and more resources shall be saved, because only data is copied directly from NvM buffer to Dlt runtime variables without using additional buffers in Dlt.	

5.2.2.3.48. Dlt_NvMReadRamBlockFromNvMNativeCbk

Purpose	Callback function from NvM for native data block to request the data to be copied from the NvM RAM during NvM_ReadBlock().	
Synopsis	`Std_ReturnType` **`Dlt_NvMReadRamBlockFromNvMNativeCbk`** `( const void * NvMBuffer );`	
Service ID	DLT_SID_InternalReadFromNativeApiId	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`NvMBuffer`	Pointer to NvM RAM mirror
Return Value	always E_OK	
Description	This callback function shall copy information stored in the native NvM block (runtime configurable variables) to the Dlt RAM. The approach used in this case is to copy the information one by one thus not using TS_MemCpy and more resources shall be saved, because only data is copied directly from NvM buffer to Dlt runtime variables without using additional buffers in Dlt.	

5.2.2.3.49. Dlt_NvMSingleBlockCallbackDataSet

Purpose	Callback-Function from NvM for data set block information notifying successful completion of single block request.	
Synopsis	`Std_ReturnType` **`Dlt_NvMSingleBlockCallbackDataSet`** `( uint8 ServiceId , NvM_RequestResultType JobResult );`	
Service ID	DLT_SID_SingleBlockCallbackDataSet	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`ServiceId`	
	`JobResult`	
Return Value	returns always E_OK	

5.2.2.3.50. Dlt_NvMSingleBlockCallbackNative

Purpose	Callback-Function from NvM for general data data set block notifying successful completion of single block request.	
Synopsis	`Std_ReturnType` **`Dlt_NvMSingleBlockCallbackNative`** `( uint8 ServiceId , NvM_RequestResultType JobResult );`	
Service ID	DLT_SID_NvMSingleBlockCallbackNative	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`ServiceId`	
	`JobResult`	
Return Value	returns always E_OK	

5.2.2.3.51. Dlt_NvMWriteRamBlockToNvMDataSetCbk

Purpose	Callback function from NvM for data set block to request the data to be copied from Dlt module's RAM into non-volatile RAM during NvM_WriteBlock().
Synopsis	`Std_ReturnType` **`Dlt_NvMWriteRamBlockToNvMDataSetCbk`** `( void * NvMBuffer );`
Service ID	DLT_SID_InternalWriteToDataSetApiId
Sync/Async	Synchronous
Reentrancy	Non reentrant

Parameters (in)	`NvMBuffer`	Pointer to NvM RAM mirror
Return Value	always E_OK	
Description	This callback function shall copy information stored in Dlt module's RAM (the values of log level and trace status explicitly set by an external client for a pair of ApplicationID and ContextID registered via Dlt_RegisterContext) The approach used in this case is to copy the information one by one and not using TS_MemCpy thus more resources shall be saved, because no auxiliary buffer is used to sore the data to be copied.	

5.2.2.3.52. Dlt_NvMWriteRamBlockToNvMNativeCbk

Purpose	Callback function from NvM for native data block to request data to be copied into NvM RAM during NvM_WriteBlock().	
Synopsis	`Std_ReturnType` **`Dlt_NvMWriteRamBlockToNvMNativeCbk`** `( void * NvMBuffer );`	
Service ID	DLT_SID_InternalWriteToNativeApiId	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`NvMBuffer`	Pointer to NvM RAM mirror
Return Value	always E_OK	
Description	This callback function shall copy information (the runtime configurable variables) from Dlt module's RAM in the native NvM block. The approach used it this case is to copy the information one by one and not using TS_MemCpy thus resources are saved, because no auxiliary buffer is used.	

5.2.2.3.53. Dlt_SetComInterfaceMaxBandwidth

Purpose	Set the maximum bandwidth for the communication interface.
Synopsis	`Dlt_CtrlReturnType` **`Dlt_SetComInterfaceMaxBandwidth`** `( uint32 max_bandwidth );`
Service ID	DLT_SID_SetComInterfaceMaxBandwidth
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (in)	`max_bandwidth`	The maximum bandwidth in bits / second This must be a multiple of 8.
Return Value	Result of changing the maximum bandwidth	
	`DLT_CTRL_OK`	Changing the maximum bandwidth succeeded
	`DLT_CTRL_ERROR`	Changing the maximum bandwidth failed
Description	This function sets the maximum allowed bandwidth available for the Dlt on the communication interface. If this bandwidth is exceeded, the Dlt will stop transmitting messages until the bandwidth is balanced again.	

5.2.2.3.54. Dlt_SetGlobalLogging

Purpose	Globally enable / disable message logging.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_SetGlobalLogging`** `( Dlt_GlobalLogStatusType logStatus );`	
Service ID	DLT_SID_SetGlobalLogging	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`logStatus`	New status of message logging (DLT_LOGGING_ENABLED: enable message logging, DLT_LOGGING_DISABLED: disable message logging)
Return Value	Result of changing the message logging status	
	`DLT_CTRL_OK`	Changing the message logging status succeeded
	`DLT_CTRL_ERROR`	Changing the message logging status failed
Description	This function enables and disables message logging. If message logging is disabled all messages are discarded by the Dlt, independently from the log level or trace status settings.	

5.2.2.3.55. Dlt_SetUseECUID

Purpose	Enable / disable the transmission of the EcuID in message header.

Synopsis	`Dlt_CtrlReturnType` **`Dlt_SetUseECUID`** `( uint8 NewStatus );`	
Service ID	DLT_SID_SetUseECUID	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`NewStatus`	New status of including the EcuID (1: include the EcuID (ON), 0: do not include the EcuID (OFF))
Return Value	Result of changing the status	
	`DLT_CTRL_OK`	Changing the status succeeded.
	`DLT_CTRL_ERROR`	Changing the status failed.
Description	This function enables and disables the inclusion of the EcuID in the Dlt message header (the ECUID is attached in the Standard Header).	

5.2.2.3.56. Dlt_SetUseExtendedHeader

Purpose	Enable / disable use of an extended message header.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_SetUseExtendedHeader`** `( uint8 NewStatus );`	
Service ID	DLT_SID_SetUseExtendedHeader	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`NewStatus`	New status of using extended header (1: use extended header (ON), 0: use standard header (OFF))
Return Value	Result of changing the status	
	`DLT_CTRL_OK`	Changing the status succeeded
	`DLT_CTRL_ERROR`	Changing the status failed
Description	This function enables and disables the use of an extended Dlt message header.	

5.2.2.3.57. Dlt_SetUseSessionID

Purpose	Enable / disable SessionID in message header.

Synopsis	`Dlt_CtrlReturnType` **`Dlt_SetUseSessionID`** `( uint8 NewStatus );`	
Service ID	DLT_SID_SetUseSessionID	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`NewStatus`	New status of including the SessionID (1: include the SessionID (ON), 0: do not include the SessionID (OFF))
Return Value	Result of changing the status	
	`DLT_CTRL_OK`	Changing the status succeeded
	`DLT_CTRL_ERROR`	Changing the status failed
Description	This function enables and disables the inclusion of the SessionID in the Dlt message header.	

5.2.2.3.58. Dlt_SetVerboseMode

Purpose	Switch to Verbose Mode.	
Synopsis	`Dlt_CtrlReturnType` **`Dlt_SetVerboseMode`** `( const Dlt_ApplicationIDType * AppId , const Dlt_ContextIDType * ContextId , Dlt_MessageLogLevelType NewStatus );`	
Service ID	DLT_SID_SetVerboseMode	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`AppId`	The application Id for which the verbose mode is set
	`ContextId`	The context Id for which the verbose mode is set
	`NewStatus`	New status of verbose mode (1: enable verbose mode (ON), 0: disable verbose mode (OFF))
Return Value	Result of changing the status	
	`DLT_CTRL_OK`	Changing the status succeeded
	`DLT_NOT_SUPPORTED`	Verbose mode is not implemented
	`DLT_CTRL_ERROR`	Changing the status failed
Description	This function switches verbose mode	

5.2.3. Integration notes

5.2.3.1. Exclusive areas

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

5.2.3.1.1. SCHM_DLT_EXCLUSIVE_AREA_MASTER

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details.

5.2.3.1.2. SCHM_DLT_EXCLUSIVE_AREA_SlaveIndex

Protected data structures	All shared data that shall be protected from mutual access by the exclusive areas defined for it. Each core will have an internal behavior.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details.

5.2.3.2. Production errors

Production errors are not reported by the `Dlt` module.

5.2.3.3. Memory mapping

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

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

Memory section
VAR_INIT_8
VAR_INIT_16
MC_SHARED_VAR_INIT_8
VAR_INIT_32
VAR_INIT_UNSPECIFIED
VAR_CLEARED_8
VAR_CLEARED_16
VAR_CLEARED_32
VAR_CLEARED_UNSPECIFIED
MC_SHARED_VAR_CLEARED_UNSPECIFIED
CONST_8
CONST_16
CONST_32
CONST_UNSPECIFIED
VAR_UNSPECIFIED
CODE
APPL_CODE

5.2.3.4. Integration requirements

WARNING 

Integration requirements list is not exhaustive

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

5.2.3.4.1. intgr.Dlt.PDAV

Description	To integrate the DLT module with a Software Component(SW-C), the right ports needs to be defined for the SW-C. The SW-C description shall be built as follows: For each group of ports which belong to one SessionID and shall be handled with one PortDefinedArgumentValue by the Dlt service:

	- For each used SessionID create one SwcServiceDependency as part of the SwcInternalBehavior. - Add the DltUserNeeds to this SwcServiceDependency. - For each included Port add one RoleBasedPortAssignment with a reference to the PortPrototype. - The role of RoleBasedPortAssignment can be left empty. - Create a new PortAPIOption with the value of the SessionID as PortDefinedArgumentValue. - Attach to RoleBasedPortAssignment all PortPrototype elements which shall belong to this SessionID. To create the port defined argument values for DLT, the Tresos plugin DltAsExt is required. The Create Dlt Port Defined Argument Values unattended wizard creates the required ports, based on the SW-C description.
Rationale	The DLT SWS specifies this method to create the port defined arguments values. The DLT module does not know all components from the module configuration that send log messages. The DLT software component description is updated by the Create Dlt Port Defined Argument Values unattended wizard directly in the the Tresos database when the complete project configuration is loaded.

5.2.3.4.2. intgr.Dlt.NvMBlockLength

Description	If the NvM blocks are created via Calculate Service Needs Unattended Wizard, the NvMNvBlockLength should be adjusted to the correct size based on the information from the DLT user's guide.

5.2.3.4.3. intgr.Dlt.UniqueAppIdContextId

Description	AppId/ContextId registration cannot differ only by SessionId. If an AppId/ContextId tuple is registered from a SW-C with a sessionId, it cannot be registered from another or same SW-C with different sessionId.
Rationale	To have more optimal data handling, the current internal data structures do not offer this flexibility.

5.2.3.4.4. intgr.Dlt.TrafficShaping.MinimumThroughput

Description	The minimum limit for traffic shaping is 1kByte/sec.

Rationale	The configuration parameter DltBandwidthForComModule expects the value in kbit/sec, but in the current implementation, the calculation is based on bytes.

5.2.3.4.5. intgr.Dlt.Multicore.CoreInitialization

Description	When basic software distribution support is enabled for Dlt, in order to ensure a consistent initialization state for both Dlt master and Dlt satellite cores, the following initialization sequence must be ensured during the start-up phase of the project: 1. Initialize all Dlt satellite cores 2. Initialize Dlt master core.
Rationale	This sequence is necessary in order to protect the project of inconsistent data storage in the system and invalid run-time data exchange between the cores.

5.2.3.4.6. intgr.Dlt.Multicore.Limits

Description	In case of multi-core configuration, the maximum number of possible contexts registered is limited to 4095 and the maximum message length is limited to 8188. These values depend on the architecture on which the stack is configured.
Rationale	When multi-core is enabled, IOC communication is involved between the cores. The above mentioned limits were introduced as a result of the limitations of the IOC communication channel.

5.2.3.4.7. intgr.Dlt.Multicore.APIRestriction

Description	In case of multi-core configuration, the control APIs can only be called from the master context. If the control functions are called from the slave context, a negative return value is reported.
Rationale	Run-time variables are only stored in the master core. The data is not replicated on slave cores.

5.2.3.4.8. intgr.Dlt.Multicore.CallingContext.MasterCoreOnly

Description	If the Dlt BSW distribution functionality is enabled, the following Dlt APIs should always be called only in the context of a Dlt master instance: - Dlt_ComCopyTxData() - Dlt_ComTxConfirmation() - Dlt_ComCopyRxData() - Dlt_ComRxIndication() - Dlt_-

	MainFunction() - Dlt_IssueWriteRequestToNvM() - Dlt_ComStartOfReception() NvM callbacks: - Dlt_NvMWriteRamBlockToNvMDataSetCbk() - Dlt_NvMWriteRamBlockToNvMNativeCbk() - Dlt_NvMInitDataSetBlockCbk() - Dlt_NvMInitNativeBlockCbk() - Dlt_NvMReadRamBlockFromNvMDataSetCbk() - Dlt_NvMReadRamBlockFromNvMNativeCbk() - Dlt_NvMSingleBlockCallbackDataSet() - Dlt_NvMSingleBlockCallbackNative() Also, the Dlt events which are triggering the following Runnable Entities should be mapped to an Os task which belongs to the same Os core identifier as the Dlt master instance: - Dlt_SetLogLevelRunnableEntity() - Dlt_SetTraceStatusRunnableEntity() - Dlt_SetVerboseModeRunnableEntity()
Rationale	Communication is always processed in the context of the master Dlt instance.

5.2.3.4.9. intgr.Dlt.ParameterDescription

Description	In Autosar 4.2 or older version Dlt_RegisterContext() API will not use the input parameters app_description, len_app_description, context_description, len_context_description.
Rationale	To reduce the size of memory footprint, app_description and context_description is ignored in the current implementation.

5.2.3.4.10. intgr.Dlt.TaskMapping.LogTraceStatusChangedNotification

Description	In case of multi-core configuration, Rte_Task and Dlt_MainFunction need to be mapped to same core.
Rationale	Dlt_ASR42_SetLogLevel and Dlt_ASR42_SetTraceStatus functions are writing the global variable ContextIdTable.Flags. The same variable is read from Dlt_SetLogLevelRunnableEntity and Dlt_SetTraceStatusRunnableEntity.

5.2.3.4.11. intgr.Dlt.SingleTuple.LogChannelTotalNumberLimitation

Description	If only one AppId/ContextId tuple is present in the configuration (DltServiceAPI = AUTOSAR_431), the maximum number of configured log channels cannot exceed 131.
Rationale	The internal handling of the control messages limits the total number of configurable log channels. If the GetLogChannelNames command is sent to the Dlt and only one tuple is present in the configuration, the Dlt will correctly respond with the total number of configured log channels but will not respond with the log channel names beyond 131.

5.2.3.4.12. intgr.Dlt.Multicore.ContextRegistration

Description	When basic software distribution support is enabled for Dlt, the exact same context shall not be registered from two different cores.
Rationale	There is no conceivable use case where the exact same context should be registered from two distinct cores which would warrant supporting this feature. Registering the context on two cores succeeds because the cores do not check the masters table before registering, but do send an update to the master afterwards which will fail due to the context being already registered. The second core will have the context registered but no other core will have any knowledege about it and hence, unpredictable behaviour may arise.

5.2.3.4.13. intgr.Dlt.RtePrototypes.ArrayBaseType

Description	The definition of the RTE_PTR2ARRAYTYPE_PASSING macro is necessary before Rte-generated header files (Rte_Dlt.h) are included into the project.
Rationale	The Rte provides this macro in order to enforce the usage of the array type instead of the array base type for arrays passed to functions. This maintains consistency with the AUTOSAR specifications (for operations like SetLogChannelAssignment).

5.2.3.4.14. intgr.Dlt.Multicore.ContextUnregister

Description	When DltServiceAPI parameter is set to AUTOSAR_431 and basic software distribution support is enabled for Dlt, the unregister of a context shall be done on the same core on which it was registered.
Rationale	Information about the tuples registered is kept locally, on every core, with the tuples that are registered on that core. If the unregister API is called from a different core, the tuple will not be found in the local tables.