Document Title	Specification of Diagnostic Log and Trace
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	351
Document Classification	Standard

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History

Date	Release	Changed by	Change Description
2016-11-30	R4.3.0	AUTOSAR Release Management	• Major rework of the SWS Dlt • Dlt Protocol moved to PRS Dlt Protocol specification • Removed interaction with DCM
2015-07-31	4.2.2	AUTOSAR Release Management	• Minor corrections
2014-10-31	4.2.1	AUTOSAR Release Management	• Changed requirements: SWS_Dlt_00515, SWS_Dlt_00516, SWS_Dlt_00332, SWS_Dlt_0028
2014-03-31	4.1.3	AUTOSAR Release Management	• Changed SWS_Dlt_00477
2013-10-31	4.1.2	AUTOSAR Release Management	• Minor corrections • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Modeling of Services: introduction of formal descriptions of service interfaces • Reworked according to the new SWS_BSWGeneral

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2011-12-22	4.0.3	AUTOSAR Administration	• Added Dlt control messages for getting values of modifiable parameters • Modification and update of Dem and Dcm interfaces • Added FIBEX example for non verbose transmission mode
2010-09-30	3.1.5	AUTOSAR Administration	• Bug fixes and extension of Dlt control message specification • Update of communication with Dem (Dem_GetEventFreezeFrameData) • Update of interface to Dcm (Dlt_ReadData)
2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

This specification and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview 7

2 Acronyms and abbreviations 8

2.1 Term and definition 8

3 Related documentation 10

3.1 Input documents 10
3.2 Related standards and norms 10
3.3 Related specification 10

4 Constraints and assumptions 11

4.1 Limitations 11
4.2 Applicability to car domains 11

5 Dependencies to other modules 12

5.1 RTE 12
5.2 PDU Router 12
5.3 NvM 12
5.4 GPT 12
5.5 DET 12
5.6 DEM 12
5.7 File structure 13

6 Requirements traceability 14

7 Functional specification 16

7.1 Dlt specification 16
7.1.1 Dlt commands 16
7.1.2 Dlt interaction with software components 17
7.1.3 VFB trace 20
7.1.4 Log messages from DEM 22
7.1.5 Log messages from DET 23
7.1.6 Recommendation for generation of Message IDs 24
7.1.7 Startup behavior 24
7.1.8 Persistent storage of configuration 25
7.1.9 Sending of Log and Trace Messages 26
7.1.10 Receiving of Dlt commands 33
7.1.11 Sending of Dlt commands 37
7.2 Error classification 38
7.2.1 Development errors 38
7.2.2 Runtime errors 38
7.2.3 Transient faults 39
7.2.4 Production errors 39
7.2.5 Extended production errors 39

8 API specification 40

8.1 Imported types 40
8.2 Type definitions 40

8.2.1 Dlt_ConfigType ... 40
8.2.2 Dlt_MessageType ... 40
8.2.3 Dlt_MessageIDType ... 41
8.2.4 Dlt_MessageNetworkTraceInfoType ... 41
8.2.5 Dlt_ReturnType ... 41
8.3 Function definitions ... 42
8.3.1 Dlt_Init ... 42
8.3.2 Dlt_GetVersionInfo ... 42
8.3.3 Dlt_SendTraceMessage ... 43
8.3.4 Dlt_SendLogMessage ... 43
8.3.5 Dlt_RegisterContext ... 44
8.3.6 Dlt_UnregisterContext ... 45
8.3.7 Dlt_DemTriggerOnEventData ... 45
8.3.8 Dlt_DetForwardErrorTrace ... 45
8.3.9 Dlt_SetLogLevel ... 46
8.3.10 Dlt_SetTraceStatus ... 47
8.3.11 Dlt_GetLogInfo ... 47
8.3.12 Dlt_GetDefaultLogLevel ... 48
8.3.13 Dlt_StoreConfiguration ... 48
8.3.14 Dlt_ResetToFactoryDefault ... 49
8.3.15 Dlt_SetMessageFiltering ... 49
8.3.16 Dlt_SetDefaultLogLevel ... 50
8.3.17 Dlt_SetDefaultTraceStatus ... 50
8.3.18 Dlt_GetDefaultTraceStatus ... 51
8.3.19 Dlt_GetLogChannelNames ... 52
8.3.20 Dlt_GetTraceStatus ... 52
8.3.21 Dlt_SetLogChannelAssignment ... 53
8.3.22 Dlt_SetLogChannelThreshold ... 53
8.3.23 Dlt_GetLogChannelThreshold ... 54
8.3.24 Dlt_InjectCall_<SESSION> ... 54
8.4 Call-back notifications ... 55
8.4.1 Dlt_RxIndication ... 55
8.4.2 Dlt_TriggerTransmit ... 55
8.4.3 Dlt_TxConfirmation ... 56
8.4.4 Dlt_TpTxConfirmation ... 57
8.4.5 Dlt_CopyTxData ... 57
8.4.6 Dlt_DemTriggerOnEventData ... 59
8.5 Scheduled functions ... 59
8.5.1 Dlt_TxFunction ... 59
8.6 Expected interfaces ... 60
8.6.1 Mandatory interfaces ... 60
8.6.2 Optional interfaces ... 61
8.7 Client-Server-Interfaces ... 62
8.7.1 DltControlService ... 62
8.7.2 InjectionCallback ... 70
8.7.3 LogTraceSessionControl ... 72
8.7.4 DltSwcMessageService ... 74
8.8 Implementation Data Types ... 78
8.8.1 Dlt_ApplicationIDType ... 78
8.8.2 Dlt_ContextIDType ... 78

8.8.3 Dlt_SessionIDType 78
8.8.4 Dlt_LogInfoType 79
8.8.5 Dlt_ContextIdInfoType 79
8.8.6 Dlt_ApplicationIdInfoType 79
8.8.7 Dlt_MessageOptionsType 80
8.8.8 Dlt_MessageLogInfoType 80
8.8.9 Dlt_MessageLogLevelType 81
8.8.10 Dlt_MessageTraceType 81
8.8.11 Dlt_MessageArgumentCount 82
8.8.12 Dlt_MessageLogInfoType 82
8.8.13 Dlt_MessageTraceInfoType 83
8.8.14 Dlt_MessageLogChannelNameType 83
8.8.15 Dlt_AssignmentOperation 84
8.9 Ports 85
8.9.1 Dlt_ControlService_{SW-C} 85
8.9.2 Dlt_InjectCallback_{SW-C} 85
8.9.3 Dlt_SessionControlCallback_{SW-C} 85
8.9.4 Dlt_SwcMessageService_{SW-C} 86
9 Sequence diagrams 87
9.1 Dlt initialization 87
9.2 Overview of Dlt message transmission on one LogChannel 88
9.3 SetLogLevelFilter 90
9.4 Buffer overflow indication 91
10 Configuration specification 93
10.1 Containers and configuration parameters 93
10.1.1 Dlt 94
10.1.2 DltGeneral 94
10.1.3 DltSwc 98
10.1.4 DltSwcContext 100
10.1.5 DltConfigSet 101
10.1.6 DltProtocol 102
10.1.7 DltEcuId 104
10.1.8 DltEcuIdCalloutChoice 104
10.1.9 DltEcuIdValueChoice 105
10.1.10 DltLogLevelSetting 105
10.1.11 DltLogChannelAssignment 107
10.1.12 DltTraceStatusSetting 107
10.1.13 DltTraceStatusAssignment 108
10.1.14 DltLogOutput 109
10.1.15 DltLogChannel 110
10.1.16 DltTxPdu 114
10.1.17 DltRxPdu 115
10.2 Published Information 117

1 Introduction and functional overview

This specification describes the functionality and the configuration of the AUTOSAR Basic Software module Dlt.

It receives log information from DET, DEM, SW-Cs, or trace information of the RTE. The Dlt module transmits this data via communication busses to make this information visible outside the ECU.

For this purpose, the Dlt module defines the API to send and receive dedicated log/trace information on the bus.

In addition, the NvM module can be optionally used to store an updated filter setting of the Dlt module persistently. This enables the ECU to transmit log/trace information with the desired level without the need of an explicit setup request coming from the communication bus (via a logging tool) at every ECU startup.

The Dlt module is located on top of the PduR and below the RTE.

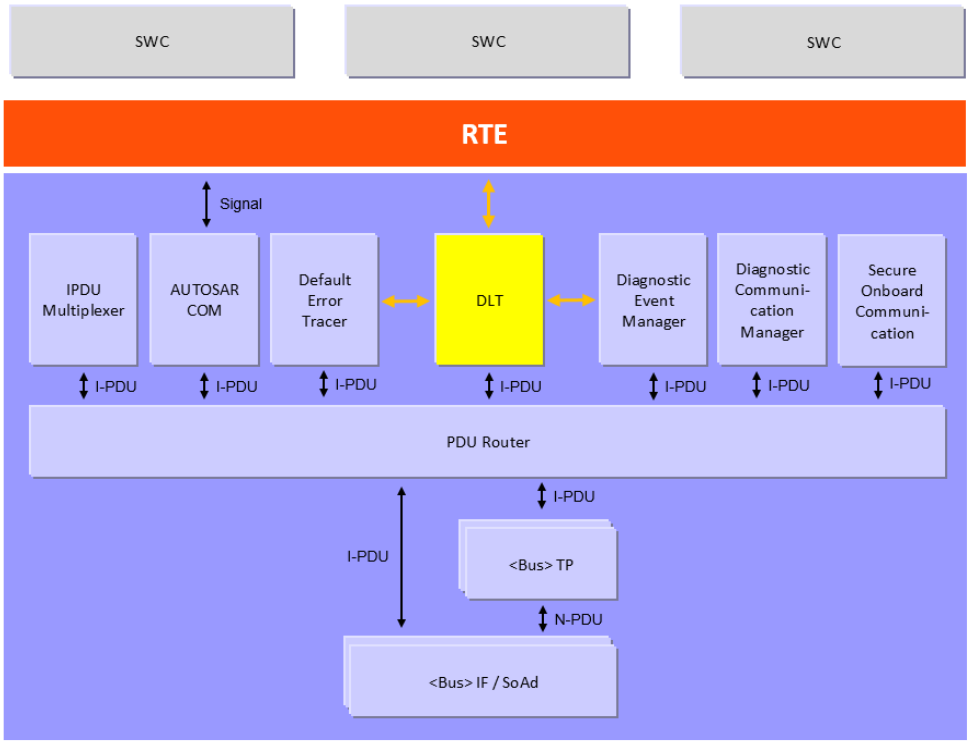


Figure 1 – Location of the Dlt module

Please note:
The Dlt Message Format, the available Dlt Commands, and the Dlt protocol (to communicate with an external logging and tracing tool) are defined in a separate document. Please refer to the [1] Dlt Protocol Specification for further information.

2 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
APID	Application ID
CTID	Context ID
Dlt	Diagnostic Log and Trace
MCNT	Message Counter
MSBF	Most Significant Byte First
MSBI	Message Bus Info
MSCI	Message Control Info
MSLI	Message Log Info
MSTP	Message Type
MSTI	Message Trace Info
NOAR	Number of Arguments
STMS	Timestamp
UEH	Use Extended Header
VERB	Verbose
VERS	Version Number
WEID	With ECU ID
WSID	With Session ID
WTMS	With Timestamp

2.1 Term and definition

Term	*Description:*
Log and trace message	A log and trace message contains all data and options to describe a log and trace event in a software. A log and trace message consists of a header and payload.
Dlt User	A Dlt User represents the source of a generated Dlt message. The possible users are SW-Cs, RTE (for VFB traces), DEM, or DET.
Log Message	A Log Message contains debug information like state changes or value changes.
Trace Message	A Trace messages contains information, which has passed via the VFB.
ECU ID	ECU IDis the name of an ECU, composed by four 8-bit ASCII characters (e.g., ABS0 or COMB).
Session	A session is a logical entity of source of log or trace messages. If an application / SW-C is instantiated several times, each instance gets a globally unique session ID with respect to the application / context ID. It is possible for an application / SWC to have several simultaneous log or trace sessions, if it has several ports opened to Dlt. Since Session ID is not specified in AUTOSAR for SW-Cs, the port defined argument values shall be used for this number.
Session ID	Session ID is the identification number of a log or trace session.
Application ID	Application ID is an abbreviation of an application / SW-C. It identifies the

	application / SW-C a log and trace message originates from. The Application ID is composed by four 8-bit ASCII characters.
Context ID	Context ID is a user defined identifier to group Log and Trace Messages generated by an application / SW-C. The following rules apply: • Each ApplicationID can own several Context IDs. • Context IDs are grouped by Application IDs. • Context IDs shall be unique within an Application ID. • The source of a log and trace message is identified using the tuple “ApplicationID” and “ContextId”. Four 8-bit ASCII characters compose the ContextId.
Message ID	Messaged ID is the identifier to characterize the information, which is transported by the message itself. A Message ID identifies a kind of log or trace message uniquely. It can be used for identifying the source (in source code) of a message and it can be used for characterizing the payload of a message. A Message ID is statically fixed at development or configuration time.
Log level	A log level defines a classification for the severity grade of a Log Message.
Trace status	The trace status provides information, if a trace message should be send.
Log Channel	A physical communication bus, which is used to transmit Dlt messages.
External client	The external client is a tool to control, monitor, and store log / trace messages provided by ECUs using the Dlt module.

3 Related documentation

3.1 Input documents

- [1] DLT Protocol Specification
PRS_DLTProtocol.pdf
- [2] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
- [3] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf
- [4] AUTOSAR Specification of RTE
AUTOSAR_SWS_RTE.pdf
- [5] AUTOSAR Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf
- [6] AUTOSAR Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager.pdf
- [7] AUTOSAR Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf
- [8] AUTOSAR Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf
- [9] AUTOSAR Specification of GPT Driver
AUTOSAR_SWS_GPTDriver.pdf

3.2 Related standards and norms

- IEC 7498-1 The Basic Model, IEC Norm, 1994

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General) which is also valid for Dlt.

Thus, the specification SWS BSW General shall be considered as additional required specification for the Dlt module.

4 Constraints and assumptions

4.1 Limitations

VFB Tracing: Currently, VFB Trace only supports the non-verbose mode. I.e., the Dlt module will send out the arguments in a raw format, simply doing a memory copy of the arguments to the trace message.

Note:
Currently, the Dlt data type model does NOT support arbitrarily nested complex data types, which AUTOSAR does. So there is no generic way to transform arguments given to the VFB Trace hook functions into Dlt data types needed for the verbose mode.

Also an ASAM Fibex description cannot be generated by the Dlt module as the in-memory representation might not be compliant to the SWCD data type description of the arguments.

4.2 Applicability to car domains

This basic software module can be used for all car domains.

5 Dependencies to other modules

5.1 RTE

The RTE (including the VFB and the BSW Scheduler) is used to interact with SW-Cs to generate Log and Trace messages and to call the Dlt module's Tx function cyclically.

5.2 PDU Router

In order to transmit Dlt messages on the communication bus, the Dlt module interacts with the PDU Router.

5.3 NvM

In order to load and store altered configurations like filter settings and/or Log Channel assignments, the NvM module can optionally be used.

5.4 GPT

In order to derive a time stamp, the GPT module can be used for this purpose.

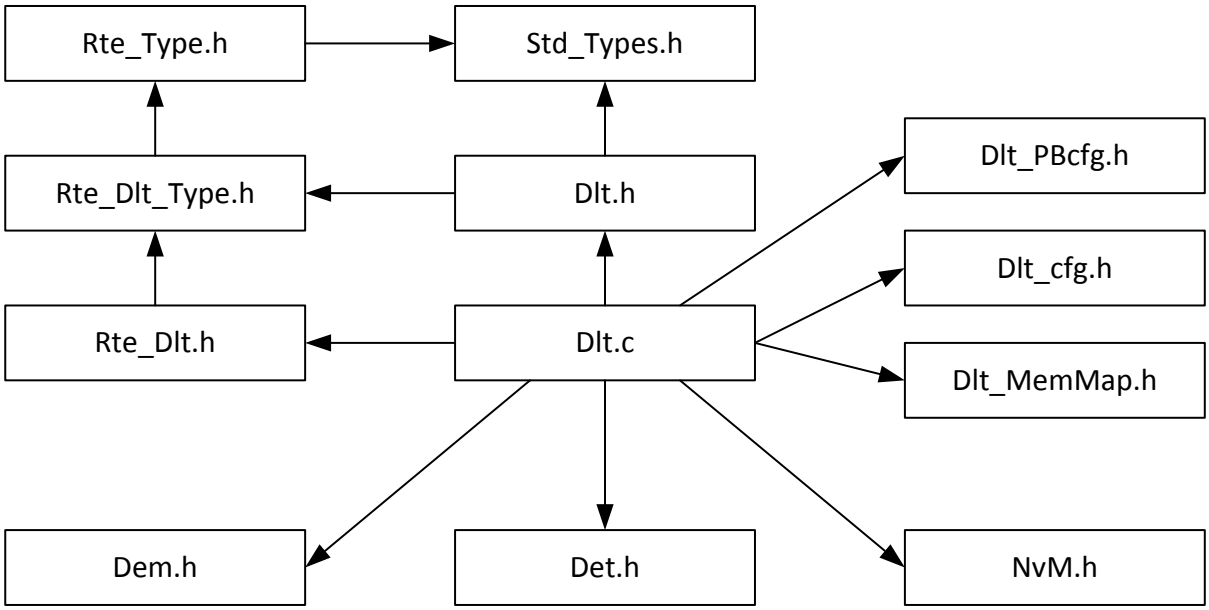
5.5 DET

In order to be able to report default errors and to forward DET errors to the communication bus, the Dlt module has to interact with the DET module. However, the interaction with DET is optional.

5.6 DEM

In order to be able to report development errors and to transmit DEM events on the communication bus, the Dlt module has to interact with the DEM module. However, the interaction with DEM is optional.

5.7 File structure



[SWS_Dlt_00482] ⌈The module header file Dlt.h shall include Rte_Dlt_Type.h to include the types, which are commonly used by BSW Modules and Software Components. Dlt.h and all Dlt*cfg.h files shall only contain types, that are not already defined in Rte_Dlt_Type.h. ⌋()

6 Requirements traceability

Requirement	Description	Satisfied by
PRS_Dlt_00635	-	SWS_Dlt_00643
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Dlt_00239
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Dlt_00239
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Dlt_00239
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Dlt_00238
SRS_BSW_00402	Each module shall provide version information	SWS_Dlt_00271
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Dlt_00239
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_Dlt_00239
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Dlt_00239
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_Dlt_00239, SWS_Dlt_00437
SRS_Dlt_00003	SWCs shall have the possibility to send log or trace messages to the DLT module.	SWS_Dlt_00241, SWS_Dlt_00243
SRS_Dlt_00004	The DLT shall provide the actual set of log levels and the trace status to a SWC.	SWS_Dlt_00252, SWS_Dlt_00254
SRS_Dlt_00006	Trace events from errors generated by BSW and SWCs shall be forwarded to the DLT module	SWS_Dlt_00430, SWS_Dlt_00432
SRS_Dlt_00007	The DEM shall forward error events to the DLT module	SWS_Dlt_00470, SWS_Dlt_00477, SWS_Dlt_00478, SWS_Dlt_00479, SWS_Dlt_00781
SRS_Dlt_00008	RTE shall provide an interface for DLT to trace RTE/VFB calls.	SWS_Dlt_00284
SRS_Dlt_00009	The DLT shall implement an interface to trace the RTE/VFB.	SWS_Dlt_00276, SWS_Dlt_00277, SWS_Dlt_00285
SRS_Dlt_00033	A list of all log and trace sources of an ECU shall be accessible from the external client.	SWS_Dlt_00021, SWS_Dlt_00245, SWS_Dlt_00769
SRS_Dlt_00034	DLT shall support a generic API for communicating over a DLT communication module.	SWS_Dlt_00263, SWS_Dlt_00264, SWS_Dlt_00265, SWS_Dlt_00515, SWS_Dlt_00516, SWS_Dlt_00517
SRS_Dlt_00036	The DLT shall provide a buffer for storing log and trace messages before initialization	SWS_Dlt_00003
SRS_Dlt_00038	A mechanism shall be implemented to be able to set the trace status and log levels of	SWS_Dlt_00252, SWS_Dlt_00254

	registered application IDs and context IDs of each SWC.	
SRS_Dlt_00039	The DLT shall provide the possibility to store configuration data in a persistent way.	SWS_Dlt_00078, SWS_Dlt_00453

7 Functional specification

7.1 Dlt specification

The following chapters describe the AUTOSAR specific data and control paths the Dlt module needs for the interaction with SW-Cs, PduR, and an external client (logging tool).

7.1.1 Dlt commands

The Dlt Protocol specifies all sorts of Dlt Commands which are identified by unique Service IDs. The Dlt Commands are used to modify the behavior of the Dlt module at runtime, e.g., fetching information about the current Dlt configuration or altering filter settings.

[SWS_Dlt_00643] ⌈The AUTOSAR Dlt module shall support the following Dlt Commands identified by the following Services IDs:

Service ID	Dlt Command Name	Description
0x01	SetLogLevel	Set the Log Level
0x02	SetTraceStatus	Enable/Disable Trace Messages
0x03	GetLogInfo	Return the LogLevel for registered SW-Cs
0x04	GetDefaultLogLevel	Return the Log Level for wildcards
0x05	StoreConfiguration	Store the current configuration non volatile
0x06	ResetToFactoryDefault	Set the configuration back to default
0x0A	SetMessageFiltering	Enable/Disable the Dlt filters
0x11	SetDefaultLogLevel	Set the LogLevel for wildcards
0x12	SetDefaultTraceStatus	Enable/Disable Trace Messages for wildcards
0x15	GetDefaultTraceStatus	Get the current TraceLevel for wildcards
0x17	GetLogChannelNames	Return the name(s) of the LogChannel(s)
0x1F	GetTraceStatus	Get the current trace status (on/off)
0x20	SetLogChannelAssignment	Add/ Remove the given LogChannel as output path
0x21	SetLogChannelThreshold	Set the filter threshold for the given LogChannel
0x22	GetLogChannelThreshold	Get the filter threshold for the given LogChannel
0x23	BufferOverflowNotification	Indication of a buffer overflow within the DLT module

⌋ (PRS_Dlt_00635)

Note:
The layouts of the defined Dlt Commands, which can be received via Dlt Control Messages, are defined in the Dlt Protocol Specification [1].

7.1.2 Dlt interaction with software components

The Dlt module offers interfaces SW-Cs can use for sending Log and Trace Messages.

Optionally, SW-Cs can provide a port for notifications on log level threshold and trace status changes, which are provided by the Dlt module separately for every tuple of `ApplicationId/ContextId`. These notifications can be used to avoid already the generation of Log and Trace Messages by the SW-Cs, instead of having them to be filtered out later on by the Dlt module.

Since the Dlt module supports multiple instances of SW-Cs, which use the same tuples of `ApplicationId/ContextId`, an additional `SessionId` parameter allows distinguishing log/trace messages from different instances of the same SW-C.

To separate those SW-Cs technically from each other and to avoid that SW-Cs have to use unique `SessionId`s in calls to `SendLogMessage/SendTraceMessage` (details, see next chapters), the Dlt module provides a dedicated P-Port per configured SW-C (see configuration parameter `DltSwc`) where the `SessionId` is managed as a port-defined-argument.

If a configured SW-C is marked as being interested in notifications on log level and trace state changes, the Dlt module also provides a corresponding R-Port to notify the respective SW-C.

The information, which SW-C is responsible for which `ApplicationId/ContextId` tuples, is configured for the SW-C and/or updated by the SW-C during runtime with a call to `RegisterContext` and `UnregisterContext` respectively.

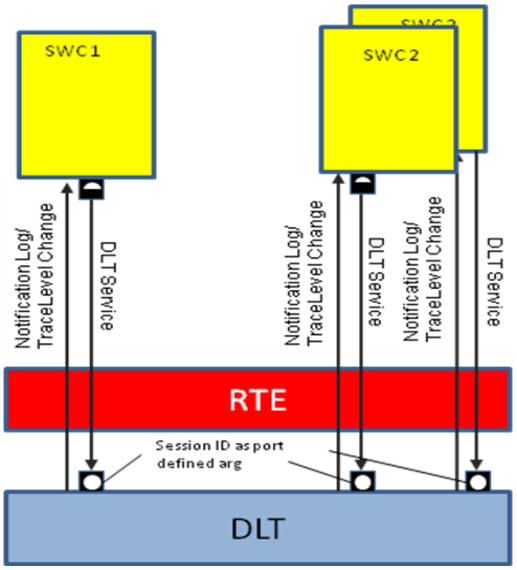


Figure 2 - Interaction with SW-C (Port configuration)

[SWS_Dlt_00644] ⌈The Dlt module shall provide a P-Port typed by interface `DltService` (see chapter 8), for each configured SW-C (see configuration container `DltSwc`). ⌋ ()

[SWS_Dlt_00645] ⌈The P-Port typed by interface has SessionId as a port-defined argument. ⌋ ()

[SWS_Dlt_00646] ⌈The Dlt module shall provide an R-Port typed by interface `LogTraceSessionControl` (see chapter 8), for each configured SW-C (see configuration container `DltSwc`), where the configuration parameter `DltSwcSupportLogLevelChangeNotification` is set to TRUE. ⌋ ()

[SWS_Dlt_00647] ⌈The `ApplicationId/ContextId` tuples for which the SW-C is responsible for and therefore needs to be notified in case of log level or trace state changes shall be deduced from configuration parameter `DltSwcContext`. ⌋ ()

7.1.2.1 Registering ApplicationIDs and ContextIds to Dlt

The Dlt module is able to inform SW-Cs about a log level change. For this purpose, they have to register at the Dlt module, using a tuple of `ApplicationId/ContextId` at runtime.

Note:
Because the developing of SW-C is not object of this specification, the Dlt module has to collect this information at runtime.

[SWS_Dlt_00765] ⌈The Dlt module shall remember all tuples of `ApplicationID`s and `ContextId`s of the SW-Cs, which register to the Dlt module. ⌋ ()

[SWS_Dlt_00766] ⌈The Dlt module shall manage a log level and a trace state for every tuple of `ContextId` and `ApplicationID`. ⌋ ()

Note:
In addition, a dynamic registration supports the possibility for the Dlt module to see which SW-C/runnable is active and which not. This is essential to know which SW-C to inform in case of a log level or trace status change.

When a SW-C is calling the `Dlt_RegisterContext` method of the `DLTService` interface, a port defined argument value is provided (`SessionID`) to the Dlt module. The value of this port defined argument corresponds to `LogTraceSessionControl` interface of the SW-C/runnable for providing information about the changing of a log level to the SW-C/runnable.

[SWS_Dlt_00021] ⌈The Dlt module shall remember the relation between the registered tuple of `ApplicationId/ContextId`, and the port interface where this tuple is registered.⌋(SRS_Dlt_00033)

[SWS_Dlt_00768] ⌈If the parameter `DltGeneralRegisterContextNotification` is set to TRUE, every time `Dlt_RegisterContext` is called, the Dlt module shall send the Dlt Control Message `GetLogInfo` containing the provided `ApplicationId/ContextId`.⌋()

7.1.2.2 Unregistering ApplicationIDs and ContextIds to Dlt

In case a SW-C is going to be stopped, it should unregister itself. This information can be used to inform an external client (e.g. a logging device) about the current SW-C status.

[SWS_Dlt_00773] ⌈The Dlt module shall delete all tuples of `ApplicationID`s and `ContextId`s of the SW-Cs which unregister to the Dlt module from the list of registered applications.⌋()

Note:
For these tuples, the Dlt module will not try to notify the corresponding SWC any more about LogLevel changes.

[SWS_Dlt_00774] ⌈If the parameter `DltGeneralRegisterContextNotification` is set to TRUE, every time `Dlt_UnregisterContext` is called, the Dlt module shall send the Dlt Control Message `GetLogInfo` containing the provided `ApplicationId/ContextId` with parameter “status” set to 6.⌋()

7.1.2.3 Port defined argument values and LogTraceSessionControl interface

For every function call of `Dlt_SendLogMessage`, `Dlt_SendTraceMessage`, `Dlt_RegisterContext` and `Dlt_UnregisterContext`, a port defined argument value needs to be provided.

[SWS_Dlt_00022] ⌈Port defined argument values shall be used by the Dlt module as `SessionIds`.⌋()

Note:
A session is the part of a SW-C for which a log level monitor is responsible. For each log level monitor the same `SessionId` (port defined argument value) shall be used.

[SWS_Dlt_00023] ⌈The port defined argument value corresponds to the defined

SessionID. The value shall start at 0x1000 (for BSW modules the module ID is taken). ⌋ ()

[SWS_Dlt_00332] ⌈Each port of a SW-C connected to the Dlt module shall have a unique SessionId as port defined argument. The range of SessionIds shall be continuous. ⌋ ()

7.1.3 VFB trace

The VFB trace is specified in the RTE. The meaning of VFB trace is an implicit (system inherent) forwarding of SW-C communication data (which flows over the RTE) to the Dlt module. Trace means in this case that no explicit call by the SW-C is made to forward this data to Dlt. This section describes the interaction of the RTE with the Dlt module to record a VFB trace and the internal control of the trace data.

7.1.3.1 Interfaces provided by Dlt for VFB traces

In case the Dlt module is used as a VFB trace client, the RTE has to be configured accordingly. This means that the RTE configuration parameter RteVfbTraceClientPrefix has to be configured with value “Dlt”.

The configuration, whether VFB tracing is enabled at all and which traceable events are supported/activated, is solely configured in the RTE module.

From its configuration, the RTE generator then updates in Generation Phase the RTEs Basic Software Module Description with BswModuleEntries for each configured VFB trace hook function. Those BswModuleEntries exactly describe the expected function prototype the configured trace clients have to provide:

- The expected function name is defined by the shortname.
- The rest of the expected signature is defined by the contained arguments.

The Dlt module has to provide the implementation for all BswModuleEntries, which are referenced by the attribute outgoingCallback of the BswModuleDescription of the RTE, whose shortname start with “Rte_Dlt”.

[SWS_Dlt_00284] ⌈The Dlt module shall be compliant to the VFB trace described in the AUTOSAR_RTE_SWS. ⌋ (SRS_Dlt_00008)

[SWS_Dlt_00276] ⌈The Dlt module shall provide the possibility to trace all kinds of trace events described in the SWS RTE. ⌋ (SRS_Dlt_00009)

[SWS_Dlt_00027] ⌈The Dlt module shall provide the implementation of the hook functions for every configured event given by an BswModuleEntry, which owns a shortname starting with “Rte_Dlt” provided by the BswModuleDefinition of the RTE. ⌋ ()

[SWS_Dlt_00335] ⌈The prototype of this hook function is to be taken from the `BswModuleEntry` of the `BSWModulDescription` of the RTE. ⌋ ()

7.1.3.2 Generating hook functions

[SWS_Dlt_00285] ⌈Because the interface `Dlt_SendTraceMessage` is a SW-C interface, an internal function which is equivalent to `Dlt_SendTraceMessage` shall be implemented to be called by the generated hook functions. ⌋ (SRS_Dlt_00009)

[SWS_Dlt_00277] ⌈In the hook function the internal representation of `Dlt_SendTraceMessage` shall be called. This call shall be in non-verbose mode. ⌋ (SRS_Dlt_00009)

[SWS_Dlt_00278] ⌈The payload for this hook function call shall be filled with the arguments provided by the hook function. All data transported with the arguments shall be provided. ⌋ ()

[SWS_Dlt_00632] ⌈The argument data shall be written in raw format to the payload. ⌋ ()

[SWS_Dlt_00279] ⌈Every hook function shall get its own `ContextId`.
In some cases some events can be bundled to the same `ContextId`. This shall mostly be done if a very large number of signals is traced. ⌋ ()

[SWS_Dlt_00337] ⌈The `ApplicationID` shall be “VFBT”. ⌋ ()

[SWS_Dlt_00484] ⌈The Message Type (MSTP) entry in the generated trace message shall be set to DLT_TYPE_NW_TRACE, the Message Trace Info (MSTI) entry in this case shall be set to DLT_NW_TRACE_IPC. ⌋ ()

[SWS_Dlt_00280] ⌈Because non-verbose mode is used, a unique Message ID as defined in [SWS_Dlt_00031] shall be used for each call to `Dlt_SendTraceMessage`. ⌋ ()

Note:
The description for the Message ID-payload shall be generated and provided. This description can be generated from the SW-C description file, were the interface is described.

[SWS_Dlt_00281] ⌈In each hook function the trace status of the `ContextId` shall be checked. ⌋ ()

```
if (vfb_actual_trace_status_contextXY){
     <internal_Dlt_SendTraceMessage>(…);
}
```

Figure 3 Requirement for hook function to check the trace status of the ContextId before call of Dlt_SendTraceMessage (`vfb_actual_trace_status_contextXY` is a freely named variable to hold the actual trace status for a specific ContextId)

[SWS_Dlt_00282] ⌈Dlt shall use for every VFB trace hook function an own `ContextId` and thus handle for every VFB trace `ContextId` a separate trace status. This can be done with a separate variable.⌋()

[SWS_Dlt_00283] ⌈A separate function shall be implemented to modify the trace status of VFB trace hook functions. This function shall be harmonized with the SW-C `LogTraceSessionControl` interface.⌋()

7.1.4 Log messages from DEM

DEM internally stores events generated by SW-Cs and BSW modules. These events are characterized by event IDs.
To an event in DEM belongs additional information. This information consists of a Diagnostic Trouble Code (DTC), Extended Data Records, and a Freeze Frame. Each time the state of an event changes DEM calls the `Dlt_DemTriggerOnEventStatus` function to notify Dlt of this change.
Always use "DEM_DTC_FORMAT_UDS" as parameter for getting the DTCCode.

[SWS_Dlt_00474] ⌈Dlt shall provide the function `Dlt_DemTriggerOnEventStatus`. Within this function, the DEM module provides the EventID of the event which status has changed. With this EventID, the Dlt module shall request additional information about the event.⌋()

[SWS_Dlt_00475] ⌈In `Dlt_DemTriggerOnEventStatus` the Dlt module shall compare the old to the new event status. If the event status are not the same Dlt shall build a Dlt log message with the new status and send it by calling `Dlt_SendLogMessage`.⌋()

Info: Log Messages generated for DEM events is not standardized. The parameter structure looks similar like this:

Number	*Type*	*Name*	*Description*
1	uint32	EventId	the EventId
2	uint32	DTCOfEvent	the DTC of the event
3	RAW	EventExtendedDataRecord	all extended data records
4	RAW	EventFreezeFrameData	the most resent FreezeFrame

[SWS_Dlt_00377] ⌈The `ApplicationID`, `ContextId` and Message ID of a Log Message sent for a DEM event shall have the following values:

ApplicationID = “DEM”
ContextId = “STD0”
MessageID = 0x00000001

⌋ ()

[SWS_Dlt_00477] ⌈The DTCOfEvent entry from [SWS_Dlt_00476] shall be requested from the DEM by calling the function `Dem_GetDTCOfEvent` with the EventId provided in `Dlt_DemTriggerOnEventData` and DTCFormatType set to the appropriate format of the DTC value. ⌋ (SRS_Dlt_00007)

[SWS_Dlt_00478] ⌈The EventExtendedDataRecord entry from [SWS_Dlt_00476] shall be filled by calling the `Dem_DltGetAllExtendedDataRecords` function of DEM with the EventId provided in `Dlt_DemTriggerOnEventData`. ⌋ (SRS_Dlt_00007)

[SWS_Dlt_00479] ⌈The `EventFreezeFrameData entry` from [SWS_Dlt_00476] shall be filled by calling the `Dem_DltGetMostRecentFreezeFrameRecordData` function of Dem with the EventId provided in `Dlt_DemTriggerOnEventData`. ⌋ (SRS_Dlt_00007)

NOTE:
The data in the ExtendedDataRecord and the FreezeFrame are not interpreted by the Dlt module. They are send as raw data and the interpretation should be done at the external client. There could be some description files, like specified in the ODX standard, used.

7.1.5 Log messages from DET

SW-Cs and BSW modules can report errors to the DET module. Such errors can be forwarded to the Dlt module as messages with a suitable content using the `Dlt_DetForwardErrorTrace`.

Note:
All parameters from the DET function `Det_ReportError` are forwarded to the Dlt function `Dlt_DetForwardErrorTrace` by the DET fan-out capability.

[SWS_Dlt_00430] ⌈The Dlt module shall provide the `Dlt_DetForwardErrorTrace` function for the fan-out capability of DET. ⌋ (SRS_Dlt_00006)

[SWS_Dlt_00376] ⌈The `ApplicationID`, `ContextId` and `MessageID` of the Log Message send by DET shall have the following values:

ApplicationID = “DET”
ContextId = “STD”

MessageID = 0x00000002
LogLevel = “Error”

⌋()

7.1.6 Recommendation for generation of Message IDs

The payload of non-verbose messages contains the Message ID. The Message ID shall be unique for an ECU. The problem is that Message IDs are provided by a SW-C (the user of Dlt) and at the point in time when coding of the log and trace message calls are done there is no instance to guarantee the uniqueness of used Message IDs.
A possible solution is to map all Log Messages in a virtual memory segment and then use the memory address as Message ID. Another solution is to have an authoring tool that is responsible for the uniqueness of the Message IDs.
In addition, it could be possible to fix Message ID values during the post build process, so uniqueness for the ECU can be guaranteed.
It is important to provide for every Message ID a description for the associated message.

[SWS_Dlt_00031] ⌈MessageIds used for DEM (0x00000001) and DET (0x00000002), and Trace Messages (0x00000003) are reserved and therefore not usable for SW-Cs. ⌋()

7.1.7 Startup behavior

The Dlt module specifies several configuration parameters, which can be reconfigured during runtime via API calls or via Dlt control messages.

This means, that those configuration parameters respectively data structures, which are based on them, have to be loaded into runtime variables during the startup of the Dlt module.

In addition, it might happen that SW-Cs and/or BSW modules are already generating log and trace data even though the Dlt module itself has not been initialized yet. For this scenario, the Dlt module offers the possibility to buffer even this data until the Dlt module is initialized.

The described functionalities result in the requirements below:

[SWS_Dlt_00003] ⌈The Dlt module shall be able to buffer data coming from calls to `Dlt_SendLogMessage` and/or `Dlt_SendTraceMessage` even if the Dlt module has not been initialized yet. ⌋(SRS_Dlt_00036)

[SWS_Dlt_00648] ⌈When the `Dlt_Init` is called, the optional timer `DltGeneralStartUpDelayTimer` shall be started if configured. ⌋()

[SWS_Dlt_00649] ⌈If the parameter `DltGeneralNvRAMSupport` is disabled, static Dlt module configuration shall be used for initialization.⌋()

[SWS_Dlt_00005] ⌈As soon as the Dlt module is initialized by `Dlt_Init` and the optional timer `DltGeneralStartUpDelayTimer` has expired, all the log and trace data, which has been buffered meanwhile, shall be processed as described in section “7.3.6. Sending of Log and Trace Messages”.⌋()

7.1.8 Persistent storage of configuration

The Dlt module offers the possibility to store configuration data in the NVRamManager module. Therefore, it is recommended to call the `Dlt_Init` function only after the NVRamManager module has been initialized.

The persistency functionality of the Dlt module supports the non-volatile saving of configuration values, which are modifiable during runtime.

The idea is to allow to customize the logging configuration during runtime and to assure that this configuration is recovered after an ECU reset or restart.

[SWS_Dlt_00451] ⌈If the parameter `DltGeneralNvRAMSupport` is set to TRUE, non-volatile memory blocks shall be used by the Dlt module to store the current Dlt configuration persistently.⌋()

[SWS_Dlt_00449] ⌈If the parameter `DltGeneralNvRAMSupport` is set to TRUE, the Dlt module has to verify the validity of the non-volatile blocks used.⌋()

[SWS_Dlt_00350] ⌈If the parameter `DltGeneralNvRAMSupport` is set to TRUE, the stored Dlt configuration shall be used as initial values.⌋()

Note:
Initial values in this case are the initial values for the persistent stored values for the first startup of the ECU.

[SWS_Dlt_00078] ⌈Storing the current configuration to NvRAM shall only be done if the parameter `DltGeneralNvRAMSupport` is enabled and the storing has been explicitly requested by the Dlt Command “StoreConfiguration”.⌋(SRS_Dlt_00039)

Note:
To store the current configuration to NvRAM, the API `NvM_WriteBlock` is used.

7.1.9 Sending of Log and Trace Messages

The Dlt data path describes the flow a Dlt Log and Trace Message takes from the source to the sink. The source can be either a SW-C or a BSW module, whereas the PDU Router is representing the sink.

The following figure provides an overview of the separate steps to send a Dlt message on the communication bus:

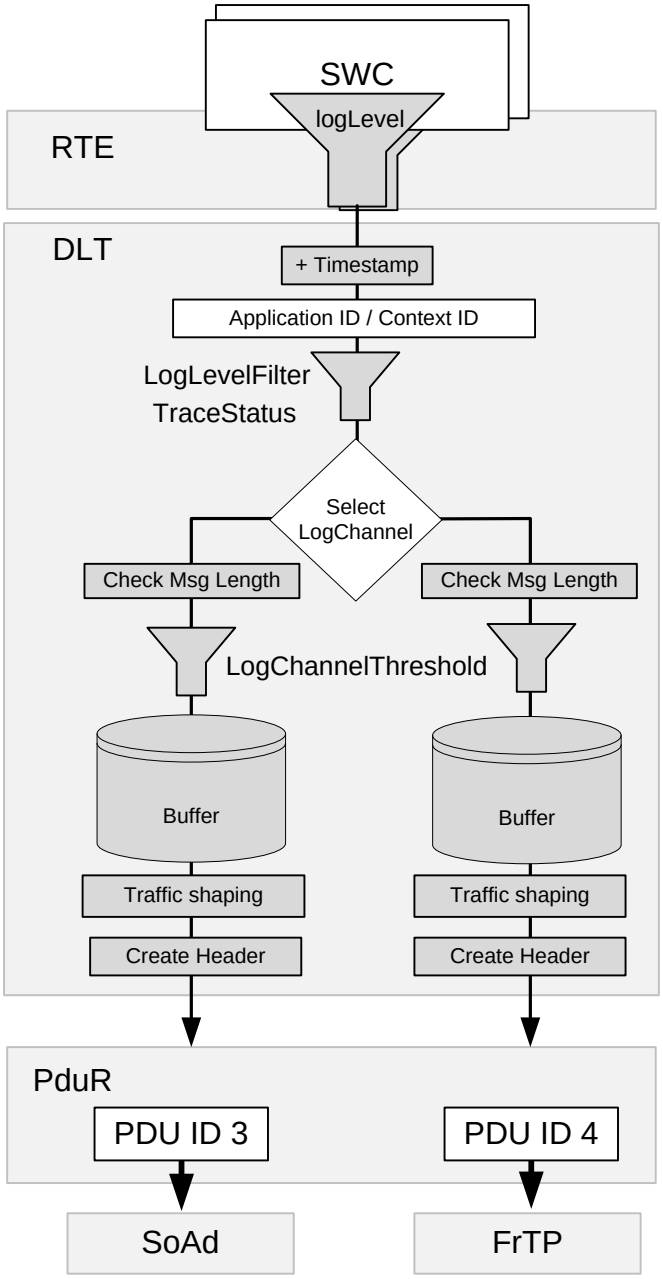


Figure 3 – Example Tx Data Path

[SWS_Dlt_00650] ⌈The following steps describe the logical order, in the context of calls to `Dlt_SendLogMessage` or `Dlt_SendTraceMessage`:

1. Generate timestamp (see 7.1.9.1)
2. Filter message (see 7.1.9.2)
3. Select target LogChannel(s) (see 7.1.9.3)
4. Check Message length (see 7.1.9.4)
5. Apply the current LogChannel threshold (see 7.1.9.5)
6. Copy Dlt message to LogChannel specific buffer (see 7.1.9.6)

⌋ ()

Note:
Because of optimizations in an implementation, the order might be changed. For instance, a typical optimization could be, that the Dlt header, which is created by Dlt module for each Dlt message, is NOT saved to the LogChannel specific buffer per Dlt message, but is created on-the-fly directly before sending the message to PduR.

[SWS_Dlt_00651] ⌈The following steps have to be taken deferred/decoupled from the context of calls to `Dlt_SendLogMessage` or `Dlt_SendTraceMessage`:

7. Send Dlt message to PduR according to TrafficShaping settings. (see 7.1.9.7)
8. Create Dlt Header according to header settings (see 7.1.9.8)
9. Remove the Dlt message from the LogChannel specific buffer (see 7.1.9.9)

⌋ ()

7.1.9.1 Generating the timestamp

Depending of the current configuration, a timestamp may be added to the Dlt message.

[SWS_Dlt_00652] ⌈Only if the parameter `DltHeaderUseTimestamp` is set to TRUE, the Dlt module shall fetch a timestamp. ⌋ ()

[SWS_Dlt_00653] ⌈If the parameter `DltHeaderUseTimestamp` is set to TRUE, but the Dlt module cannot fetch a timestamp for any reason, the timestamp shall be set to 0x00000000. ⌋ ()

[SWS_Dlt_00654] ⌈⌈If the parameter `DltHeaderUseTimestamp` is set to TRUE, the Dlt module shall call the API Gpt_GetTimeElapsed() with the configured channel reference (see `DltGeneralGptChannelRef`) to fetch the elapsted time. ⌋ ()

7.1.9.2 Message filtering

Message filtering means to accept or discard an incoming log or trace message based on the `ApplicationId/ContextId` tuple, which is assigned to that message.

Filtering differs slightly between Log Messages (`Dlt_SendLogMessage`) and trace messages (`Dlt_SendTraceMessage`).

[SWS_Dlt_00656] [For Dlt Log Messages, the highest LogLevel Threshold shall be defined as “Verbose”.] ()

[SWS_Dlt_00657] [For Dlt Log Messages, the lowest LogLevel Threshold shall be defined as “Filter off”.] ()

Note:
The `Dlt_MessageLogLevelType` defines all possible Log Message filter levels.

[SWS_Dlt_00658] [For Log Message filtering the Dlt internally manages LogLevel threshold to `ApplicationId/ContextId` tuple mappings (see configuration parameter `DltLogLevelThreshold`).] ()

[SWS_Dlt_00659] [For trace message filtering the Dlt internally manages trace activation state to `ApplicationId/ContextId` tuple mappings (see configuration parameter `DltTraceStatusAssignment`).] ()

Note:
The matching algorithm for finding the proper mapping element (containing a threshold log level value in the Log Message case respectively containing a trace activation state in the trace message case) is identical for Log Messages and trace messages.

[SWS_Dlt_00661] [The Dlt module shall find a matching mapping element (log level threshold respectively trace activation state) for the `ApplicationId/ContextId` tuple contained in a `Dlt_SendLogMessage` or `Dlt_SendTraceMessage` call. To do so, the following steps shall be performed:

1. Check whether a mapping element exists, where `ApplicationId/ContextId` tuple of mapping element equals to the `ApplicationId/ContextId` tuple of the log/trace message. If such a mapping element exists, the matching mapping element is found.
2. In case no match has been found in step 1, check whether a mapping element exists, where the `ApplicationID` equals the `ApplicationID` of the log/trace message and the `ContextId` of mapping element equals wildcard (value 0x00000000). If such a mapping element exists, the matching mapping element is found.
3. In case no match has been found in step 1 and 2, the matching mapping element is the current `DefaultLogLevelThreshold` respectively the current `DefaultTraceStatus`.

] ()

[SWS_Dlt_00662] [In the `Dlt_SendLogMessage` case, the found mapping element is a log level threshold. If the value of the log level threshold is higher than the log level of the Log Message, the message is not further processed and E_OK is returned.] ()

[SWS_Dlt_00663] [In the `Dlt_SendTraceMessage` case, the found mapping element is a trace activation state. If the value of the trace activation state is FALSE, the message is not further processed and E_OK is returned.] ()

7.1.9.3 Select target LogChannel

In this step, the Dlt module identifies on which LogChannel(s) the log or tarce message will be transmitted.

[SWS_Dlt_00664] [For LogChannel selection the Dlt module manages LogChannel to `ApplicationId/ContextId` tuple mappings. (see configuration parameter `DltLogChannelAssignmentSwcContextRef`).] ()

Note:
There can be several LogChannels configured for a given `ApplicationId/ContextId` tuple contained in a `Dlt_SendLogMessage` or `Dlt_SendTraceMessage` call.

[SWS_Dlt_00665] [To find the matching LogChannels for the `ApplicationId/ContextId` tuple contained in a `Dlt_SendLogMessage` or `Dlt_SendTraceMessage` call, the Dlt module shall do the following steps:

1. From all mapping elements, where `ApplicationId/ContextId` tuple of mapping element equals to the `ApplicationId/ContextId` tuple of the log/trace message, the LogChannel shall be added to the list of output LogChannels.
2. From all mapping elements, where `ApplicationID` of mapping element equals to the `ApplicationID` of the log/trace message AND the `ContextId` of mapping element equals wildcard (value 0x00000000), the LogChannel shall be added to the list of output LogChannels.
3. If the list of output LogChannels is still empty after step 1 and 2. The default LogChannel (see configuration parameter `DltDefaultLogChannelRef`) shall be added to the list of output LogChannels.

] ()

7.1.9.4 Check message length

[SWS_Dlt_00666] [If the Dlt message length including the required Dlt headers exceeds the configured value given by `DltLogChannelMaxMessageLength` for all assigned LogChannels, discard this Dlt message and return DLT_E_MSG_TOO_LARGE.] ()

Note:
If the message is short enough for at least one assigned LogChannel, continue to process this message for all LogChannels where the message is short enough.

7.1.9.5 Apply LogChannel LogLevelThreshold

In this step, the Dlt module decides for each identified target LogChannel, whether the Log Message or trace message might pass according to the LogChannel specific LogLevel threshold respectively TraceStatusFlag.

[SWS_Dlt_00667] [Log messages with a LogLevel lower than the configured value of LogChannel threshold for the identified LogChannel shall be discarded and E_OK shall be returned. This shall only be done if this holds true for every LogChannel the LogMessage is assigned to.] ()

[SWS_Dlt_00668] [Trace messages shall be filtered out, when the config parameter TraceStatusFlag is FALSE for the identified LogChannel. That means they do not proceed to the next processing step and E_OK is returned.] ()

7.1.9.6 Copying Dlt message to the LogChannel buffer

In this step the Dlt module copies the Dlt message to all buffers of the LogChannels, which the Dlt message is assigned to.

[SWS_Dlt_00669] [The Dlt module shall copy the log/trace message which has passed the message filters to all assigned target LogChannel buffers where the Dlt message length is not larger than `DltLogChannelMaxMessageLength` of the respective LogChannel.] ()

[SWS_Dlt_00670] [If there was not enough space to copy the complete message to any of the assigned LogChannel's buffer, DLT_E_NO_BUFFER shall be returned and the Dlt log and trace message shall be discarded.
In addition, check each assigned buffer whether it was already full before, i.e., check Dlt internal flags to store a buffer overflow event:

- If the buffer overflow flag is currently not set for this buffer:
  - Set the buffer overflow flag to indicate the occurrence of a buffer overflow
  - The Dlt log and trace message shall be discarded
- If the buffer overflow flag for this buffer was already set for this buffer:
  - The Dlt log and trace message shall be discarded

- Send Dlt Control Message(s) “BufferOverflowNotification” according to the configuration. Please refer to chapter 7.1.11.1.

] ()

Note:
The cyclicly called `Dlt_TxFunction` checks the status of the buffer overflow flag and the debounce time for sending buffer overflow notifications. This function also sets back the flag cyclically according to a buffer overflow notification.

[SWS_Dlt_00671] [If a new massage has been copied successfully to the assigned LogChannel’s buffer, the message counter shall be increased by 1. This message counter value shall be stored for the Dlt message.] ()

Note:
When the Dlt message is going to be transmitted, this message counter value will be written into the Message Counter Field (MCNT).

[SWS_Dlt_00672] [If a new massage has been copied successfully to at least one LogChannel buffer, DLT_OK shall be returned.] ()

7.1.9.7 Sending messages from LogChannel Buffer

[SWS_Dlt_00780] [The sending of Dlt messages via the PduR API shall be decoupled from the `Dlt_SendLogMessage` and `Dlt_SendTraceMessage` API call.] ()

Note:
The decoupling is done because of the following reasons:

1. Shortening runtime of calls from the SW-Cs/BSWs which trigger log/trace messages, to reduce blocking time.
2. In case traffic shaping functionality is enabled, the transmissions have to be processed by an asynchronous cyclic BSW entity anyway.
3. In case retry feature is enabled a decoupled BSW entity, which cares for retries, is needed anyway.

[SWS_Dlt_00673] [The Dlt module shall transmit Dlt messages collected in the LogChannel specific buffer from the context of the `Dlt_TxFunction` function.] ()

[SWS_Dlt_00674] [The Dlt Message Header shall be assembled before `PduR_DltTransmit` is called.] ()

Note:
For details regarding the assembling of the Dlt Message Header, please refer to the next section.

[SWS_Dlt_00675] [The Dlt module shall use the `PduR_DltTransmit` function to send the Dlt message with the configured TxPduId.] ()

[SWS_Dlt_00677] [The Dlt module shall monitor a transmit counter for each Dlt message in a LogChannel specific buffer. Each time it calls `PduR_DltTransmit` for a Dlt message in the buffer, it shall increment the transmit counter.] ()

7.1.9.8 Create Dlt message header

Assembling the Dlt Header

[SWS_Dlt_00678] [The UEH bit shall be set to '1' if the parameters `DltUseExtHeaderInNonVerbMode` is set to TRUE.
Otherwise, the UEH bit shall be set to '0'.] ()

[SWS_Dlt_00679] [The MSBF bit shall be set to '1' if the current platform is BIGENDIAN.] ()

[SWS_Dlt_00680] [The MSBF bit shall be set to '0' if the current platform is LITTLEENDIAN.] ()

[SWS_Dlt_00681] [The WEID bit shall be set to '1' if the parameter DltHeaderUseEcuId is set to TRUE. Else, the WEID bit shall be set to '0'.] ()

[SWS_Dlt_00682] [The WSID bit shall be set to '1' if the parameter `DltHeaderUseSessionID` is set to TRUE. Else, the WSID bit shall be set to '0'.] ()

[SWS_Dlt_00683] [The WTMS bit shall be set to '1' if the parameter `DltHeaderUseTimestamp` is set to TRUE. Else, the WSID bit shall be set to '0'.] ()

[SWS_Dlt_00684] [The VERS bits shall always be set to "001".] ()

[SWS_Dlt_00685] [The MCNT field shall be set to the stored value of this Dlt message when it is copied to the LogChannel's buffer.] ()

[SWS_Dlt_00686] [The optional ECU field shall only exist if `DltHeaderUseEcuId` is set to TRUE.] ()

[SWS_Dlt_00687] [The optional ECU field shall be set to the value configured in `DltProtocolEcuIdValue`. If the configured ECU IDis shorter than 4 byte, the remaining bytes shall be set to 0x00.] ()

[SWS_Dlt_00688] [The optional SEID field shall be set to the value configured via `DltSwcSessionId` and shall only exist if `DltHeaderUseSessionID` is set to TRUE.] ()

[SWS_Dlt_00689] [The optional TMSP field shall contain the derived timestamp if `DltHeaderContainsTimeStamp` is set to TRUE.] ()

[SWS_Dlt_00690] ⌈The LEN field shall be set to the overall length of the finally assembled Dlt Data Message, which shall be the sum of the length of the Header, the length of the optional Extended Header, and the length of the Payload. ⌋ ()

Assembling the Dlt Extended Header

[SWS_Dlt_00691] ⌈If the parameters `DltUseExtHeaderInNonVerbMode` is set to TRUE, the Dlt Extended Header has to be generated for Dlt Data Messages:
⌋ ()

[SWS_Dlt_00692] ⌈The VERB bit shall be set to '1 'if the parameter `DltUseExtHeaderInNonVerbMode` is set to TRUE. Else, the VERB bit shall be set to '0'. ⌋()

[SWS_Dlt_00693] ⌈The MSTP field shall be set to 0x0 if the Dlt message has to be assembled due to the API call `Dlt_SendLogMessage`.⌋()

[SWS_Dlt_00694] ⌈The MSTP field shall be set to 0x1 if the Dlt message has to be assembled due to the API call `Dlt_SendTraceMessage`.⌋()

[SWS_Dlt_00695] ⌈The MTIN field shall be set accordingly to the Dlt_MessageTraceInfoType value, which has been passed by the API `Dlt_SendLogMessage`. ⌋()

[SWS_Dlt_00696] ⌈The MTIN field shall be set accordingly to the Dlt_MessageTraceInfoType value, which has been passed by the API `Dlt_SendTraceMessage`. ⌋()

7.1.9.9 Removing messages from LogChannel buffer

[SWS_Dlt_00697] ⌈A Dlt message, for which `PduR_DltTransmit` has been called, shall be removed from the LogChannel specific buffer in the following cases:

- `PduR_DltTransmit` has returned with E_NOT_OK,
- A positive TX confirmation for this TxPduId has been received, or
- A negative TX confirmation for this TxPduId has been received and the transmit counter of the Dlt message is greater than the configured `DltLogChannelMaxNumOfRetries`.

⌋ ()

7.1.10 Receiving of Dlt commands

The Dlt module can receive Dlt commands via the Rx Data Path and/or via dedicated API calls (see chapter 8). These Dlt commands can be used to control the Dlt module.

[SWS_Dlt_00698] ⌈The Dlt module shall ignore all received Dlt control messages via the Rx Data Path in case the parameter `DltGeneralRxDataPathSupport` is set to FALSE. ⌋ ()

Note:
In case the Rx Data Path is disabled, the Dlt client can be controlled via the optional control APIs defined in chapter 8.

[SWS_Dlt_00699] [If `DltGeneralRxDataPathSupport` is set to TRUE, the Dlt module shall process received Dlt control messages.] ()

[SWS_Dlt_00700] [If a received Dlt command has been executed successfully, “OK” shall be returned.] ()

7.1.10.1 SetLogLevel

[SWS_Dlt_00701] [If the Dlt command “SetLogLevel” is requested, the new LogLevel shall be stored for the received tuple of `ApplicationId/ContextId`.] ()

[SWS_Dlt_00702] [If the Dlt command “SetLogLevel” is requested, but the received tuple of `ApplicationId/ContextId` is unknown, the Dlt command shall be answered with “ERROR”.] ()

7.1.10.2 SetTraceStatus

[SWS_Dlt_00703] [If the Dlt command “SetTraceStatus” is requested, the new trace status shall be stored for the received tuple of `ApplicationId/ContextId`.] ()

[SWS_Dlt_00704] [If the Dlt command “SetTraceStatus” is requested, but the addressed tuple of `ApplicationId/ContextId` is unknown, the Dlt command shall be answered with “ERROR”.] ()

7.1.10.3 GetLogInfo

[SWS_Dlt_00705] [If the Dlt command “GetLogInfo” is requested, the requested LogInfo shall be returned.] ( )

[SWS_Dlt_00706] [If the Dlt command “GetLogInfo” is requested, but the addressed tuple of `ApplicationId/ContextId` is unknown, the Dlt command shall be answered with “ERROR”.] ()

7.1.10.4 GetDefaultLogLevel

[SWS_Dlt_00708] [If the Dlt command “GetDefaultLogLevel” is requested, the current value of the parameter `DltDefaultLogLevel` shall be returned.] ()

7.1.10.5 StoreConfiguration

[SWS_Dlt_00709] [If the Dlt command “StoreConfiguration” is requested and the configuration parameter `DltGeneralNvRAMSupport` is set to TRUE, the following steps shall be performed:

- Call `NvM_WriteBlock` to store the current configuration of the LogChannelAssignment, LogChannelThreshold, and the LogLevelFilter.
  - If `NvM_WriteBlock` returned with E_OK, the Dlt command “StoreConfiguration” shall return with “OK”.
  - If `NvM_WriteBlock` returned with something else than E_OK, the Dlt command “StoreConfiguration” shall return with “ERROR”.

] ( )

[SWS_Dlt_00710] [If the Dlt command “StoreConfiguration” is requested and the configuration parameter `DltGeneralNvRAMSupport` is set to FALSE, the Dlt command “StoreConfiguration” shall return “NOT_SUPPORTED”.] ()

7.1.10.6 RestoreToFactoryDefault

[SWS_Dlt_00711] [If the Dlt command “RestoreToFactoryDefault” is requested and if the parameter `DltGeneralNvRAMSupport` is set to FALSE, reset the following runtime parameters to the values stored in the Dlt module’s static configuration:

- DltDefaultLogLevel
- DltThreshold
- DltDefaultTraceStatus
- DltLogChannelThreshold
- DltDefaultLogChannelRef

] ()

[SWS_Dlt_00712] [If the Dlt command “RestoreToFactoryDefault” is requested and if the parameter `DltGeneralNvRAMSupport` is set to TRUE, delete the stored configuration of the NvM by calling `NvM_EraseNvBlock` and reset the following runtime parameters to the values stored in the Dlt module’s static configuration:

- DltDefaultLogLevel
- DltThreshold
- DltDefaultTraceStatus
- DltLogChannelThreshold
- DltDefaultLogChannelRef

] ()

[SWS_Dlt_00713] [If the Dlt command “RestoreToFactoryDefault” is requested and if the parameter `DltGeneralNvRAMSupport` is set to FALSE, “OK” shall be returned if the Dlt module reset the current configuration values to the default configuration successfully.] ()

[SWS_Dlt_00714] [If the Dlt command “RestoreToFactoryDefault” is requested and the parameter `DltGeneralNvRAMSupport` is set to TRUE, response with “ERROR”

- if the Dlt module could not reset the current configuration to the static default configuration or
- if the stored configuration of the NvM could not be deleted.] ()

7.1.10.7 SetMessageFiltering

[SWS_Dlt_00775] [If the Dlt command “SetMessageFiltering” is requested, all the Dlt filters shall be enabled/disabled as requested, and the Dlt command shall be answered with “OK”.] ( )

7.1.10.8 SetDefaultLogLevel

[SWS_Dlt_00715] [If the Dlt command “SetDefaultLogLevel” is requested, the parameter `DltDefaultLogLevel` shall be updated to the received new LogLevel.] ()

7.1.10.9 SetDefaultTraceStatus

[SWS_Dlt_00716] [If the Dlt command “SetDefaultTraceStatus” is requested, the parameter `DltDefaultTraceStatus` shall be updated to the received new TraceStatus.] ()

7.1.10.10 GetDefaultTraceStatus

[SWS_Dlt_00717] [If the Dlt command “GetDefaultTraceStatus” is requested, the current value of the parameter `DltDefaultTraceStatus` shall be returned.] ()

7.1.10.11 GetLogChannelNames

[SWS_Dlt_00718] [If the Dlt command “GetLogChannelNames” is requested, the number of configured LogChannels and all LogChannel names given by the parameter `DltLogChannelName` shall be returned.] ()

7.1.10.12 GetTraceStatus

[SWS_Dlt_00719] [If the Dlt Command “GetTraceStauts” is requested, the TraceStatus shall be returned for the received tuple of `ApplicationId/ContextId`.] ()

7.1.10.13 SetLogChannelAssignment

[SWS_Dlt_00720] [If the Dlt command “SetLogChannelAssignment” is requested with parameter `addRemoveOp` set to `DLT_ASSIGN_ADD`, add the tuple of `ApplicationId/ContextId` to the LogChannel with the name provided by the

parameter `logChannelName`. The Dlt command shall return “OK” even if the tuple was already assigned to the requested LogChannel before. ⌋ ()

[SWS_Dlt_00721] ⌈If the Dlt command “SetLogChannelAssignment” is requested with parameter `addRemoveOp` set to `DLT_ASSIGN_REMOVE`, remove the tuple of `ApplicationId/ContextId` from the LogChannel with the name provided by the parameter `logChannelName`. The Dlt command shall return “OK” even if the tuple was not assigned to the requested LogChannel before. ⌋ ()

Note:
If a tuple of `ApplicationId/ContextId` is not assigned explicitly to any specific LogChannel (any more), the mandatory default LogChannel (see `DltDefaultLogChannelRef`) will be used for transmission.

[SWS_Dlt_00722] ⌈If the Dlt command “SetLogChannelAssignment” is requested with an unknown tuple of `ApplicationId/ContextId` or an unknown LogChannel name, the Dlt command shall return “ERROR”. ⌋ ()

7.1.10.14 SetLogChannelThreshold

[SWS_Dlt_00723] ⌈If the Dlt command “SetLogChannelThreshold” is requested, the LogChannelThreshold of the addressed LogChannel shall be set to the value received by the parameter `newThreshold`. ⌋ ()

[SWS_Dlt_00724] ⌈If the Dlt command “SetLogChannelThreshold” is requested and the `logChannelName` and/or the `newThreshold` is unknown, the Dlt command shall return “ERROR”. ⌋ ()

7.1.10.15 GetLogChannelThreshold

[SWS_Dlt_00725] ⌈If the Dlt command “GetLogChannelThreshold” is requested, the LogChannelThreshold of the addressed LogChannel shall be returned. ⌋ ()

[SWS_Dlt_00726] ⌈If the Dlt command “GetLogChannelThreshold” is requested and the `logChannelName` or the `newThreshold` is unknown, the Dlt command shall return “ERROR”. ⌋ ()

7.1.11 Sending of Dlt commands

Typically, the Dlt module receives Dlt commands generated by a Dlt logging tool, which are answered by the Dlt module. Only two Dlt commands are triggered for sending by the Dlt module itself:

- GetLogInfo (only in case one or more SW-Cs register/unregister themselves)
- BufferOverflowNotification (in case of a buffer overflow)

7.1.11.1 BufferOverflowNotification

The buffer overflow notification is used to inform the Dlt Logging tool about the loss of Dlt messages. The amount of BufferOverflowNotifications on the bus can be limited/debounced by configuration. This notification contains a counter which indicates the amount of lost Dlt messages since the last BufferOverflowNotification.

[SWS_Dlt_00776] ⌈If the Dlt module detects a buffer overflow, it shall send a Dlt command “BufferOverflowNotification” cyclically (see DltLogChannelBufferOverflowTimer) as long as the buffer is still full. ⌋ ()

[SWS_Dlt_00777] ⌈The parameter overflowCounter of the Dlt control message “BufferOverflowNotification” shall be set to the number of lost Dlt messages since the last transmission of the “BufferOverflowNotification”. ⌋ ()

7.2 Error classification

7.2.1 Development errors

[SWS_Dlt_00727] ⌈The following development error types shall be supported:

Type of error	*Related error code*	*Value [hex]*
API service called with wrong parameter	DLT_E_PARAM	0x01
Null pointer has been passed as an argument	DLT_E_PARAM_POINTER	0x02
Initialization failed	DLT_E_INIT_FAILED	0x03
Registration failed	DLT_E_REGISTRATION	0x04

⌋ ()

7.2.2 Runtime errors

[SWS_Dlt_00728] ⌈The folowing runtime error types shall be supported:

Type of error	*Related error code*	*Value [hex]*
Message could not be sent	DLT_E_SKIPPED_TRANSMISSION	0x05
A deprecated parameter with a value different to 0 for a Dlt command has been received	DLT_E_DEPRECATED_PARAMETER	0x06
Multiple Control Requests at the same time	DLT_E_MULTIPLE_REQUESTS	0x07

⌋ ()

7.2.3 Transient faults

There are no transient faults.

7.2.4 Production errors

There are no production errors.

7.2.5 Extended production errors

There are no extended production errors.

8 API specification

8.1 Imported types

In this section all types imported from the following modules are listed:

The following types are imported from the specified modules:

[SWS_Dlt_00729] [

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
	PduIdType
	PduInfoType
	PduLengthType
	RetryInfoType
Dem	Dem_DTCFormatType
	Dem_EventIdType
	Dem_UdsStatusByteType
Gpt	Gpt_ChannelType
	Gpt_ValueType
NvM	NvM_BlockIdType
Std_Types	Std_ReturnType
	Std_VersionInfoType

] ()

8.2 Type definitions

8.2.1 Dlt_ConfigType

[SWS_Dlt_00437] [

Name:	`Dlt_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	The content of the initialization data structure is implementation specific
Description:	This is the type of the data structure containing the initialization data for Dlt.	

] (SRS_BSW_00414)

8.2.2 Dlt_MessageType

[SWS_Dlt_00224] [

Name:	`Dlt_MessageType`		
Type:	`Enumeration`		
Range:	`DLT_TYPE_LOG`	0x00	A log message
	`DLT_TYPE_APP_TRACE`	0x01	A trace message
	`DLT_TYPE_NW_TRACE`	0x02	A message forwarded from a communication bus (like

			CAN, FlexRay)
	DLT_TYPE_CONTROL	0x03	A message for internal use/control sent between Dlt module and external client.
Description:	This type describes the type of the message.		

] ()

8.2.3 Dlt_MessageIDType

[SWS_Dlt_00228] [

Name:	Dlt_MessageIDType		
Kind:	Array		
Type:	uint8		
Size:	4		
Range:	0x00000000-0xFFFFFFFF	--	--
Description:	Contains the unique MessageId for a message. This is only relevant in non-verbose mode.		

] ()

8.2.4 Dlt_MessageNetworkTraceInfoType

[SWS_Dlt_00233] [

Name:	Dlt_MessageNetworkTraceInfoType		
Type:	Enumeration		
Range:	DLT_NW_TRACE_IPC	0x01	Inter process communication
	DLT_NW_TRACE_CAN	0x02	CAN communication
	DLT_NW_TRACE_FLEXRAY	0x03	Flexray communication
	DLT_NW_TRACE_MOST	0x04	MOST communication
	DLT_NW_TRACE_ETHERNET	0x05	Ethernet communication
	DLT_NW_TRACE_SOMEIP	0x06	SOME/IP communication
Description:	This type describes transported type of a Dlt BUSMESSAGE.		

] ()

8.2.5 Dlt_ReturnType

[SWS_Dlt_00238] [

Range:	DLT_E_MSG_TOO_LARGE	0x02	The message is too big for the internal Dlt buffer.
	DLT_E_CONTEXT_ALREADY_REG	0x03	The software module context has already registered.
	DLT_E_UNKNOWN_SESSION_ID	0x04	Unknown session id.
	DLT_E_NO_BUFFER	0x05	Buffer overflow.
	DLT_E_CONTEXT_NOT_YET_REG	0x06	The software module context has not registered before.
	DLT_E_ERROR	0x09	--
Description:	--		

] (SRS_BSW_00377)

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Dlt_Init

[SWS_Dlt_00239] ⌈

Service name:	Dlt_Init	

Syntax:	`void Dlt_Init(`
	`  const Dlt_ConfigType* config`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	config	Pointer to a DLT configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
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.	

⌋ (SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00101, SRS_BSW_00407, SRS_BSW_00358, SRS_BSW_00414)

[SWS_Dlt_00453] ⌈If the parameter `DltGeneralNvRAMSupport` is set to TRUE, the Dlt module shall use the API `NvM_ReadBlock` of the NVRAM module for restoring the values from persistent storage for the variables required by [SWS_Dlt_00077] in the `Dlt_Init` function. ⌋ (SRS_Dlt_00039)

8.3.2 Dlt_GetVersionInfo

[SWS_Dlt_00271] ⌈

Service name:	Dlt_GetVersionInfo	

Syntax:	`void Dlt_GetVersionInfo(`
	`  Std_VersionInfoType* versioninfo`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

⌋ (SRS_BSW_00402)

8.3.3 Dlt_SendTraceMessage

[SWS_Dlt_00243] ⌈

Service name:	Dlt_SendTraceMessage	
Syntax:	`Std_ReturnType Dlt_SendTraceMessage( Dlt_SessionIDType sessionId, const Dlt_MessageTraceInfoType* traceInfo, const uint8* traceData, uint16 traceDataLength )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	sessionId	Number of the module (Module ID within BSW, Port defined argument value within SW-C)
	traceInfo	Structure containing the relevant information for filtering the message.
	traceData	Buffer containing the parameters to be traced. The contents of this pointer represents the payload of the Trace Message to be sent.
	traceDataLength	Length of the data buffer traceData
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The required operation succeeded. DLT_E_MSG_TOO_LARGE: The message is too large for all assigned LogChannels. DLT_E_NO_BUFFER: Not enough buffer available, the Dlt message cannot be buffered for at least one LogChannel. DLT_E_UNKNOWN_SESSION_ID: The provided session id is unknown.
Description:	The service represents the interface to be used by basic software modules or by software components to trace parameters.	

⌋ (SRS_Dlt_00003) ⌋

8.3.4 Dlt_SendLogMessage

[SWS_Dlt_00241] ⌈

Service name:	Dlt_SendLogMessage	
Syntax:	`Std_ReturnType Dlt_SendLogMessage( Dlt_SessionIDType sessionId, const Dlt_MessageLogInfoType* logInfo, const uint8* logData, uint16 logDataLength )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	sessionId	For SW-C this is not visible (Port defined argument value), for BSW-modules it is the module number
	logInfo	Structure containing the relevant information for filtering the message.
	logData	Buffer containing the parameters to be logged. The contents of this pointer represents the payload of the Log Message to be sent.

	logDataLength	Length of the data buffer logData.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	DLT_OK: The required operation succeeded. DLT_E_MSG_TOO_LARGE: The message is too large for all assigned LogChannels DLT_E_NO_BUFFER: The LogMessage could not be buffered at any assigned LogChannel DLT_E_UNKNOWN_SESSION_ID: The provided session id is unknown.
Description:	The service represents the interface to be used by basic software modules or by software component to send Log Messages.	

] (SRS_Dlt_00003)

8.3.5 Dlt_RegisterContext

[SWS_Dlt_00245] ⌈

Service name:	Dlt_RegisterContext	

Syntax:	`Std_ReturnType Dlt_RegisterContext(`
	`  Dlt_SessionIDType sessionId,`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId,`
	`  const uint8* appDescription,`
	`  uint8 lenAppDescription,`
	`  const uint8* contextDescription,`
	`  uint8 lenContextDescription`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	sessionId	number of the module (Module ID within BSW, Port defined argument value within SW-C)
	appId	the ApplicationId
	contextId	the ContextId
	appDescription	Points to description string for the provided ApplicationId. At maximum 255 characters are interpreted.
	lenAppDescription	The length of the description for the ApplicationId string (number of characters of description string).
	contextDescription	Points to description string for the provided context. At maximum 255 characters are interpreted.
	lenContextDescription	The length of the description string (number of characters of description string).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The required operation succeeded. DLT_E_CONTEXT_ALREADY_REG: The software module context has already registered. DLT_E_UNKNOWN_SESSION_ID: The provided session id is unknown.
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 ContextId is being registered for an already registered ApplicationId then appDescription can be NULL and len_appDescription zero.	

] (SRS_Dlt_00033)

8.3.6 Dlt_UnregisterContext

[SWS_Dlt_00769] ⌈

Service name:	Dlt_UnregisterContext	

Syntax:	`Std_ReturnType Dlt_UnregisterContext(`
	`  Dlt_SessionIDType sessionId,`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId`
	`)`	

Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	sessionId	number of the module (Module ID within BSW, Port defined argument value within SW-C)
	appId	the ApplicationId
	contextId	the ContextId
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The required operation succeeded. 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.
Description:	The service has to be called when a software module is going to be stopped.	

⌋ (SRS_Dlt_00033)

8.3.7 Dlt_DemTriggerOnEventData

[SWS_Dlt_00781] ⌈

Service name:	Dlt_DemTriggerOnEventData	

Syntax:	`Std_ReturnType Dlt_DemTriggerOnEventData(`
	`  Dem_EventIdType EventId`
	`)`	

Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: error occured
Description:	Triggers on changes of the event related data in the event memory.	

⌋ (SRS_Dlt_00007)

8.3.8 Dlt_DetForwardErrorTrace

[SWS_Dlt_00432] ⌈

Service name:	Dlt_DetForwardErrorTrace
Syntax:	`void Dlt_DetForwardErrorTrace(`

	`uint16 moduleId,`
	`uint8 instanceId,`
	`uint8 apiId,`
	`uint8 errorId`
	)	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	moduleId	Module ID of calling module.
	instanceId	The identifier of the index based instance of a module, starting from 0. If the module is a single instance module it shall pass 0 as the instanceId.
	apiId	ID of API service in which error is detected (defined in SWS of calling module)
	errorId	ID of detected development error (defined in SWS of calling module).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service to forward error reports from Det to Dlt.	

] (SRS_Dlt_00006)

8.3.9 Dlt_SetLogLevel

[SWS_Dlt_00252] [

Service name:	Dlt_SetLogLevel	

Syntax:	`Std_ReturnType Dlt_SetLogLevel(`
	`Dlt_ApplicationIDType appId,`
	`Dlt_ContextIDType contextId,`
	`Dlt_MessageLogLevelType newLogLevel`
	)	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	appId	ID of the SW-C
	contextId	ID of the context
	newLogLevel	new log level to set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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.	

] (SRS_Dlt_00004, SRS_Dlt_00038)

8.3.10 Dlt_SetTraceStatus

[SWS_Dlt_00254] [

Service name:	Dlt_SetTraceStatus	

Syntax:	`Std_ReturnType Dlt_SetTraceStatus(`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId,`
	`  boolean newTraceStatus`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	appId	ID of the SW-C
	contextId	ID of the context
	newTraceStatus	New trace status
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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.	

] (SRS_Dlt_00004, SRS_Dlt_00038)

8.3.11 Dlt_GetLogInfo

[SWS_Dlt_00732] [

Service name:	Dlt_GetLogInfo	

Syntax:	`Std_ReturnType Dlt_GetLogInfo(`
	`  uint8 options,`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId,`
	`  uint8* status,`
	`  Dlt_LogInfoType* logInfo`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
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 (inout):	None	
Parameters (out):	status	--
	logInfo	Details about the returned Application IDs
Return value:	Std_ReturnType	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.	

] ()

8.3.12 Dlt_GetDefaultLogLevel

[SWS_Dlt_00733] [

Service name:	Dlt_GetDefaultLogLevel	
Syntax:	`Std_ReturnType Dlt_GetDefaultLogLevel( Dlt_MessageLogLevelType* defaultLogLevel )`	
Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	defaultLogLevel	Returns the stored LogLevel setting
Return value:	Std_ReturnType	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.	

] ()

[SWS_Dlt_00734] [A call to `Dlt_GetDefaultLogLevel` shall return with E_NOT_OK if the Dlt module cannot return the current value of the parameter `DltDefaultLogLevel`.] ()

[SWS_Dlt_00735] [A call to `Dlt_GetDefaultLogLevel` shall return with E_NOT_OK if the Dlt module cannot return the current value of the parameter `DltDefaultLogLevel`.] ()

8.3.13 Dlt_StoreConfiguration

[SWS_Dlt_00736] [

Service name:	Dlt_StoreConfiguration	
Syntax:	`Std_ReturnType Dlt_StoreConfiguration( void )`	
Service ID[hex]:	0x1a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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().	

] ()

[SWS_Dlt_00737] [If the parameter `DltGeneralNvRAMSupport` is set to FALSE, a call to `Dlt_StoreConfiguration` shall return with DLT_NOT_SUPPORTED.] ()

[SWS_Dlt_00729] [If the parameter `DltGeneralNvRAMSupport` is set to TRUE, a call to `Dlt_StoreConfiguration` shall return with DLT_E_ERROR in case the call to `NvM_WriteBlock` returned with E_NOT_OK.] ()

[SWS_Dlt_00738] [If the parameter `DltGeneralNvRAMSupport` is set to TRUE, a call to `Dlt_StoreConfiguration` shall return with DLT_OK in case the call to `NvM_WriteBlock` returned with E_OK.] ()

8.3.14 Dlt_ResetToFactoryDefault

[SWS_Dlt_00739] [

Service name:	Dlt_ResetToFactoryDefault	
Syntax:	`Std_ReturnType Dlt_ResetToFactoryDefault( void )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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.	

] ()

8.3.15 Dlt_SetMessageFiltering

[SWS_Dlt_00770] [

Service name:	Dlt_SetMessageFiltering	
Syntax:	`Std_ReturnType Dlt_SetMessageFiltering( boolean status )`	
Service ID[hex]:	0x1b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	status	TRUE: enable message filtering FALSE: disable message filtering

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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.	

] ()

8.3.16 Dlt_SetDefaultLogLevel

[SWS_Dlt_00740] [

Service name:	Dlt_SetDefaultLogLevel	

Syntax:	`Std_ReturnType Dlt_SetDefaultLogLevel(`
	`  Dlt_MessageLogLevelType newLogLevel`
	`)`	

Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	newLogLevel	sets the new filter value
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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.	

] ()

[SWS_Dlt_00741] [If a call to `Dlt_SetDefaultLogLevel` successfully set the addressed LogChannel to the requested LogLevel, it shall return with E_OK.] ()

[SWS_Dlt_00742] [If a call to `Dlt_SetDefaultLogLevel` could not set the addressed LogChannel to the requested LogLevel, it shall return with E_NOT_OK.] ()

8.3.17 Dlt_SetDefaultTraceStatus

[SWS_Dlt_00743] [

Service name:	Dlt_SetDefaultTraceStatus

Syntax:	`Std_ReturnType Dlt_SetDefaultTraceStatus(`
	`  boolean newTraceStatus,`
	`  Dlt_LogChannelNameType logChannelName`
	`)`

Service ID[hex]:	0x12
Sync/Async:	Synchronous
Reentrancy:	Reentrant

Parameters (in):	newTraceStatus	enabling/disabling of Trace messages
	logChannelName	Name of the addressed LogChannel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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.	

] ()

[SWS_Dlt_00744] [If a call to `Dlt_SetDefaultTraceStatus` successfully set the addressed LogChannel to the requested new TraceStatus, it shall return with E_OK.] ()

[SWS_Dlt_00745] [If a call to `Dlt_SetDefaultTraceStatus` could not set the addressed LogChannel to the requested TraceStatus, it shall return with E_NOT_OK.] ()

8.3.18 Dlt_GetDefaultTraceStatus

[SWS_Dlt_00746] [

Service name:	Dlt_GetDefaultTraceStatus	
Syntax:	`Std_ReturnType Dlt_GetDefaultTraceStatus( Dlt_LogChannelNameType logChannelName, boolean* traceStatus )`	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	logChannelName	Name of the addressed LogChannel
Parameters (inout):	None	
Parameters (out):	traceStatus	current trace status (enabled/disabled)
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: Default Trace Status could not be returned
Description:	Returns the current Trace Status of the addressed LogChannel.	

] ()

[SWS_Dlt_00747] [If a call to `Dlt_GetDefaultTraceStatus` returned the current Trace Status of the requested LogChannel, it shall return with E_OK.] ()

[SWS_Dlt_00748] [If a call to `Dlt_GetDefaultTraceStatus` could not return the addressed TraceStatus of the addressed LogChannel, it shall return with E_NOT_OK.] ()

8.3.19 Dlt_GetLogChannelNames

[SWS_Dlt_00749] ⌈

Service name:	Dlt_GetLogChannelNames	

Syntax:	`Std_ReturnType Dlt_GetLogChannelNames(`
	`  uint8* numberOfLogChannels,`
	`  Dlt_LogChannelNameType* logChannelNames`
	`)`	

Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	numberOfLogChannels	Returns the number of configured LogChannels
	logChannelNames	Returns a list of configured LogChannel names
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: LogChannelNames could not be returned
Description:	Returns all configured LogChannel names.	

⌋ ()

8.3.20 Dlt_GetTraceStatus

[SWS_Dlt_00750] ⌈

Service name:	Dlt_GetTraceStatus	

Syntax:	`Std_ReturnType Dlt_GetTraceStatus(`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId,`
	`  boolean* traceStatus`
	`)`	

Service ID[hex]:	0x1f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	appId	ApplicationId
	contextId	ContextId
Parameters (inout):	None	
Parameters (out):	traceStatus	current Trace Status of the tuple ApplicationId/ContextId
Return value:	Std_ReturnType	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.	

⌋ ()

8.3.21 Dlt_SetLogChannelAssignment

[SWS_Dlt_00751] ⌈

Service name:	Dlt_SetLogChannelAssignment	

Syntax:	`Std_ReturnType Dlt_SetLogChannelAssignment(`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId,`
	`  Dlt_LogChannelNameType logChannelName,`
	`  Dlt_AssignmentOperation addRemoveOp`
	`)`	

Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	appId	ID of the addressed application / SW-C
	contextId	ID of the addressed context
	logChannelName	Name of the addressed LogChannel
	addRemoveOp	Operation to add/remove the addressed tuple ApplicationId/ContextId to/from the addressed LogChannel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: LogChannel assignment failed
Description:	Adds/removes the addressed tuple ApplicationId/ContextId to/from the addressed LogChannel.	

⌋ ()

8.3.22 Dlt_SetLogChannelThreshold

[SWS_Dlt_00752] ⌈

Service name:	Dlt_SetLogChannelThreshold	

Syntax:	`Std_ReturnType Dlt_SetLogChannelThreshold(`
	`  Dlt_LogChannelNameType logChannelName,`
	`  Dlt_MessageLogLevelType newThreshold,`
	`  boolean newTraceStatus`
	`)`	

Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different LogChannelNames	
Parameters (in):	logChannelName	Name of the addressed LogChannel
	newThreshold	Threshold for LogMessages
	newTraceStatus	TRUE: enable TraceMessages FALSE: disable TraceMessages
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: Setting of LogChannelThreshold failed
Description:	Sets the filter threshold for the given LogChannel.	

⌋ ()

8.3.23 Dlt_GetLogChannelThreshold

[SWS_Dlt_00753] ⌈

Service name:	Dlt_GetLogChannelThreshold	

Syntax:	`Std_ReturnType Dlt_GetLogChannelThreshold(`
	`  Dlt_LogChannelNameType logChannelName,`
	`  Dlt_MessageLogLevelType* logChannelThreshold,`
	`  boolean* traceStatus`
	`)`	

Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different LogChannelNames	
Parameters (in):	logChannelName	Addressed LogChannel name
Parameters (inout):	None	
Parameters (out):	logChannelThreshold	Current LogChannelThreshold
	traceStatus	Current TraceStatus. TRUE: TraceMessages enabled. FALSE: TraceMessages disabled.
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: LogChannelThreshold could not be returned
Description:	Returns the filter threshold for the given LogChannel.	

⌋ ()

8.3.24 Dlt_InjectCall_<SESSION>

[SWS_Dlt_00259] ⌈

Service name:	Dlt_InjectCall_<SESSION>	

Syntax:	`void Dlt_InjectCall_<SESSION>(`
	`  Dlt_ApplicationIDType appId,`
	`  Dlt_ContextIDType contextId,`
	`  uint32 serviceId,`
	`  uint32 dataLength,`
	`  const uint8* data`
	`)`	

Service ID[hex]:	0x14	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	appId	the Application ID
	contextId	the Context ID
	serviceId	the service ID for the injection (user defined)
	dataLength	length of the data puffer provided
	data	pointer to data puffer with data belonging to the injection (service ID). The contents of the data is user defined
Parameters (inout):	None	

Parameters (out):	None
Return value:	None
Description:	Callback is called by Dlt to inject a function call in the SW-C. The behaviour trigged by this function should depend on the service_id also the interpretation of the user data. Both are specific to the application.

] ()

8.4 Call-back notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file Dlt_`Cbk.h`.

8.4.1 Dlt_RxIndication

[SWS_Dlt_00272] [

Service name:	Dlt_RxIndication	

Syntax:	`void Dlt_RxIndication(`
	`  PduIdType RxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x42	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received PDU from a lower layer communication interface module.	

] ()

8.4.2 Dlt_TriggerTransmit

[SWS_Dlt_00754] [

Service name:	Dlt_TriggerTransmit

Syntax:	`Std_ReturnType Dlt_TriggerTransmit(`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`

Service ID[hex]:	0x41
Sync/Async:	Synchronous

Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout):	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	

⌋ ()

[SWS_Dlt_00755] ⌈If development error detection is enabled for this module, the module shall check all parameters for being valid. If the check fails, the function shall raise a development error and return. ⌋ ()

8.4.3 Dlt_TxConfirmation

[SWS_Dlt_00273] ⌈

Service name:	Dlt_TxConfirmation	

Syntax:	`void Dlt_TxConfirmation(`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	

⌋ ()

8.4.4 Dlt_TpTxConfirmation

[SWS_Dlt_00756] [

Service name:	Dlt_TpTxConfirmation	

Syntax:	`void Dlt_TpTxConfirmation(`
	`  PduIdType id,`
	`  Std_ReturnType result`
	`)`	

Service ID[hex]:	0x48	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the transmitted I-PDU.
	result	Result of the transmission of the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	

] ()

8.4.5 Dlt_CopyTxData

[SWS_Dlt_00516] [

Service name:	Dlt_CopyTxData	

Syntax:	`BufReq_ReturnType Dlt_CopyTxData(`
	`  PduIdType id,`
	`  const PduInfoType* info,`
	`  RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	`)`	

Service ID[hex]:	0x43	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
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 transmit 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 specifies the offset in bytes from the current data copy position.
Parameters (inout):	None	
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. FrIsoTp) to determine the size of the following CFs.
Return value:	BufReq_ReturnType	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.	

] ()

8.4.6 Dlt_DemTriggerOnEventData

[SWS_Dlt_00781] [

Service name:	Dlt_DemTriggerOnEventData	
Syntax:	`Std_ReturnType Dlt_DemTriggerOnEventData( Dem_EventIdType EventId )`	
Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	EventId	Identification of an event by assigned EventId
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error occurred E_NOT_OK: error occured
Description:	Triggers on changes of the event related data in the event memory.	

] (SRS_Dlt_00007)

8.5 Scheduled functions

8.5.1 Dlt_TxFunction

[SWS_Dlt_91005] [

Service name:	Dlt_TxFunction
Syntax:	`void Dlt_TxFunction( void )`
Service ID[hex]:	0x80
Description:	--

] ()

[SWS_Dlt_00758] [If the configuration parameter `DltGeneralTrafficShapingSupport` is set to TRUE, the Dlt messages shall be transmitted with the maximum bandwidth per LogChannel as configured using the parameter `DltLogChannelTrafficShapingBandwidth.`] ()

[SWS_Dlt_00759] [If the configuration parameter `DltGeneralTrafficShapingSupport` is set to FALSE, all buffered Dlt messages shall be transmitted at once.] ()

[SWS_Dlt_00760] [The `Dlt_TxFunction` shall check the status of the flag, which indicates that a BufferOverflow occurred:

- If a buffer overflow occurred, the Dlt command “BufferOverflowNotification” shall be sent only once, until the overflow flag is cleared again.
- After a time interval given by the parameter `DltLogChannelBufferOverflowTimer`, the buffer overflow flag shall be cleared.

This shall be done for every configured LogChannel separately. ⌋ ()

[SWS_Dlt_00761] ⌈If a Dlt message could not be sent, every time the `Dlt_TxFunction` is called, it shall retry to send this message one time. This shall be done for every message separately and taking care to not exceed the amount of retries given by `DltLogChannelMaxNumOfRetries`. ⌋ ()

8.6 Expected interfaces

In this section all external interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

This section defines all external interfaces which are required to fulfill the core functionality of the module.

The module relies on the following interfaces:

[SWS_Dlt_00762] ⌈

API function	*Description*
PduR_DltTransmit	Requests transmission of a PDU.

⌋ ()

8.6.2 Optional interfaces

This section defines all external interfaces which are required to fulfill an optional functionality of the module.

The module relies on the following optional interfaces:

[SWS_Dlt_00763] ⌈

API function	*Description*
Dem_DltGetAllExtendedDataRecords	Gets the data of all extended data records of an event.
Dem_DltGetMostRecentFreezeFrameRecordData	Gets the data of an most recent freeze frame record by event. The OBD-II freeze frame is not returned by this function.
Dem_GetDTCOfEvent	Gets the DTC of an event.
Det_ReportError	Service to report development errors.
Gpt_EnableNotification	Enables the interrupt notification for a channel (relevant in normal mode).
Gpt_StartTimer	Starts a timer channel.
NvM_EraseNvBlock	Service to erase a NV block.
NvM_ReadBlock	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.

⌋ ()

8.7 Client-Server-Interfaces

8.7.1 DltControlService

[SWS_Dlt_00772] ⌈

Name	DltControlService	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	7	DLT_E_NOT_SUPPORTED
	9	DLT_E_ERROR

Operations

GetDefaultLogLevel			
Comments	--		
Variation	--		
Parameters	defaultLoglevel	Comment	--
		Type	Dlt_MessageLogLevelType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	The default LogLevel could not be returned	

GetDefaultTraceStatus			
Comments	--		
Variation	--		
Parameters	logChannelName	Comment	--
		Type	Dlt_LogChannelNameType
		Variation	--

		Direction	IN
	traceStatus	Comment	--
		Type	boolean
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	Default LogLevel could not be set	
GetLogChannelNames			
Comments	--		
Variation	--		
Parameters	numberOfLogChannels	Comment	--
		Type	uint8
		Variation	--
		Direction	OUT
	logChannelNames	Comment	--
		Type	Dlt_LogChannelNameType*
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	LogChannelNames could not be returned	
GetLogChannelThreshold			
Comments	--		
Variation	--		
Parameters	logChannelName	Comment	--
		Type	Dlt_LogChannelNameType
		Variation	--
		Direction	IN
	logChannelThreshold	Comment	--
		Type	Dlt_MessageLogLevelType

		Variation	--
		Direction	IN
	traceStatusPtr	Comment	--
		Type	boolean
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	LogChannelThreshold could not be returned	
GetLogInfo			
Comments	--		
Variation	--		
Parameters	options	Comment	--
		Type	uint8
		Variation	--
		Direction	IN
	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	status	Comment	--
		Type	uint8
		Variation	--
		Direction	OUT
	logInfo	Comment	--
		Type	Dlt_LogInfoType
		Variation	--

		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	LogLevel could not be set	
GetTraceStatus			
Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	traceStatus	Comment	--
		Type	boolean
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	TraceStatus could not be returned	
ResetToFactoryDefault			
Comments	--		
Variation	--		
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	Configuration has not been reset	
SetDefaultLogLevel			
Comments	--		
Variation	--		

Parameters	newDefaultLogLevel	Comment	--
		Type	Dlt_MessageLogLevelType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	Default LogLevel could not be set	

SetDefaultTraceStatus			
Comments	--		
Variation	--		
Parameters	newTraceStatus	Comment	--
		Type	boolean
		Variation	--
		Direction	IN
	logChannel	Comment	--
		Type	uint8
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	Default Trace Status could not be set.	

SetLogChannelAssignment			
Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--

		Direction	IN
	logChannelName	Comment	--
		Type	Dlt_LogChannelNameType
		Variation	--
		Direction	IN
	addRemoveOp	Comment	--
		Type	Dlt_AssignmentOperation
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	LogChannel assignment failed	

SetLogChannelThreshold			
Comments	--		
Variation	--		
Parameters	logChannelName	Comment	--
		Type	Dlt_LogChannelNameType
		Variation	--
		Direction	IN
	newLogLevelThreshold	Comment	--
		Type	Dlt_MessageLogLevelType
		Variation	--
		Direction	IN
	newTraceStatus	Comment	--
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	Setting of LogChannelThreshold failed	

SetLogLevel

Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	newLogLevel	Comment	--
		Type	Dlt_MessageLogLevelType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	LogLevel could not be set	

SetMessageFiltering			
Comments	--		
Variation	--		
Parameters	status	Comment	--
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	TraceStatus could not be changed	

SetTraceStatus			
Comments	--		
Variation	--		
Parameters	appId	Comment	--

		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	newTraceStatus	Comment	--
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_ERROR	TraceStatus could not be changed	

StoreConfiguration		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	DLT_E_NOT_SUPPORTED	Service is not supported
	DLT_E_ERROR	The configuration could not be stored

] ()

8.7.2 InjectionCallback

[SWS_Dlt_00498] ⌈

Name	InjectionCallback	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

InjectCall			
Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	serviceId	Comment	--
		Type	uint32
		Variation	--
		Direction	IN
	dataLength	Comment	--
		Type	uint32
		Variation	--
		Direction	IN
	data	Comment	--
		Type	uint8*

		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

] ()

8.7.3 LogTraceSessionControl

[SWS_Dlt_00496] ⌈

Name	LogTraceSessionControl	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

LogLevelChangedNotification			
Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	logLevel	Comment	--
		Type	Dlt_MessageLogLevelType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	

TraceStatusChangedNotification	
Comments	--
Variation	--

Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	newTraceStatus	Comment	--
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	

] ()

8.7.4 DltSwcMessageService

[SWS_Dlt_00495] ⌈

Name	DltSwcMessageService	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	2	DLT_E_MSG_TOO_LARGE
	3	DLT_E_CONTEXT_ALREADY_REG
	4	DLT_E_UNKNOWN_SESSION_ID
	5	DLT_E_NO_BUFFER
	6	DLT_E_CONTEXT_NOT_YET_REG
	9	DLT_E_ERROR

Operations

RegisterContext			
Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType
		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
	appDescription	Comment	--
		Type	uint8[]
		Variation	--
		Direction	IN
	lenAppDescription	Comment	--

		Type	uint8
		Variation	--
		Direction	IN
	contextDescription	Comment	--
		Type	uint8[]
		Variation	--
		Direction	IN
	lenContextDescription	Comment	--
		Type	uint8
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_CONTEXT_ALREADY_REG	The software module context has already registered.	
	DLT_E_UNKNOWN_SESSION_ID	The provided session id is unknown.	

SendLogMessage			
Comments	--		
Variation	--		
Parameters	logInfo	Comment	--
		Type	Dlt_MessageLogInfoType
		Variation	--
		Direction	IN
	logData	Comment	--
		Type	uint8[]
		Variation	--
		Direction	IN
	logDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN

Possible Errors	E_OK	Operation successful
	DLT_E_MSG_TOO_LARGE	The message is too big for the internal Dlt buffer.
	DLT_E_UNKNOWN_SESSION_ID	The provided session id is unknown.
	DLT_E_NO_BUFFER	Buffer overflow.

SendTraceMessage			
Comments	--		
Variation	--		
Parameters	traceInfo	Comment	--
		Type	Dlt_MessageTraceInfoType
		Variation	--
		Direction	IN
	traceData	Comment	--
		Type	uint8[]
		Variation	--
		Direction	IN
	traceDataLength	Comment	--
		Type	uint16
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_MSG_TOO_LARGE	The message is too big for the internal Dlt buffer.	
	DLT_E_UNKNOWN_SESSION_ID	The provided session id is unknown.	
	DLT_E_NO_BUFFER	Buffer overflow.	

UnregisterContext			
Comments	--		
Variation	--		
Parameters	appId	Comment	--
		Type	Dlt_ApplicationIDType

		Variation	--
		Direction	IN
	contextId	Comment	--
		Type	Dlt_ContextIDType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	DLT_E_UNKNOWN_SESSION_ID	The provided session id is unknown.	
	DLT_E_CONTEXT_NOT_YET_REG	The software module context has not registered before.	

] ()

8.8 Implementation Data Types

8.8.1 Dlt_ApplicationIDType

[SWS_Dlt_00226] ⌈

Name	Dlt_ApplicationIDType		
Kind	Array	Element type	uint8
Size	4 Elements		
Description	This type describes the ApplicationId. 0x00000000 means the so-called wildcard.		
Variation	--		

⌋ ()

8.8.2 Dlt_ContextIDType

[SWS_Dlt_00227] ⌈

Name	Dlt_ContextIDType		
Kind	Array	Element type	uint8
Size	4 Elements		
Description	This type describes the ContextId. 0x00000000 means the so-called wildcard.		
Variation	--		

⌋ ()

8.8.3 Dlt_SessionIDType

[SWS_Dlt_00225] ⌈

Name	Dlt_SessionIDType
Kind	Type
Derived from	uint32
Description	This type identifies the session.
Variation	--

⌋ ()

8.8.4 Dlt_LogInfoType

[SWS_Dlt_91002] ⌈

Name	Dlt_LogInfoType		
Kind	Structure		
Elements	appIdCount	uint16	Number of AppIds
	appIdInfo	Dlt_ApplicationIdInfoType	Details of Application
Description	--		
Variation	--		

⌋ ()

8.8.5 Dlt_ContextIdInfoType

[SWS_Dlt_91003] ⌈

Name	Dlt_ContextIdInfoType		
Kind	Structure		
Elements	contextId	Dlt_ContextIDType	the ContextId
	logLevel	Dlt_MessageLogLevelType	the log message filter level
	traceStatus	uint8	0: off 1: on
	lenContextDescription	uint16	Length of Context Description
	contextDesc	Array of uint8	Context Description
		Size	
Description	Context Information		
Variation	--		

⌋ ()

8.8.6 Dlt_ApplicationIdInfoType

[SWS_Dlt_91004] ⌈

Name	Dlt_ApplicationIdInfoType
Kind	Structure

Elements	appId	Dlt_ApplicationIDType	Application ID
	contextIdCount	uint16	Length of contextInfoList
	contextInfoList	Dlt_ContextIdInfoType	List of Context information
	appDescLen	uint16	Length of Application Description
	appDesc	uint8	Application Description
Description	Information about Applications		
Variation	--		

] ()

8.8.7 Dlt_MessageOptionsType

[SWS_Dlt_00229] [

Name	Dlt_MessageOptionsType		
Kind	Type		
Derived from	uint8		
Description	Bitfield		
Range	verbose_mode		Bit 0: If set Verbose mode is used (yet not relevant)
	message_type		Bit 1-3 Dlt_MessageTypeType: determines type of msg (log,trace,...)
Variation	--		

] ()

8.8.8 Dlt_MessageLogInfoType

[SWS_Dlt_00236] [

Name	Dlt_MessageLogInfoType		
Kind	Structure		
Elements	argCount	Dlt_MessageArgumentCount	--
	logLevel	Dlt_MessageLogLevelType	--
	options	Dlt_MessageOptionsType	--
	contextId	Dlt_ContextIDType	--
	appId	Dlt_ApplicationIDType	--

Description	--
Variation	--

] ()

8.8.9 Dlt_MessageLogLevelType

[SWS_Dlt_00230] [

Name	Dlt_MessageLogLevelType		
Kind	Enumeration		
Range	DLT_LOG_OFF	0x00	Turn off logging
	DLT_LOG_FATAL	0x01	Fatal system error
	DLT_LOG_ERROR	0x02	Errors occurring in a SW-C with impact to correct functionality
	DLT_LOG_WARN	0x03	Log messages where a incorrect behavior can not be ensured
	DLT_LOG_INFO	0x04	Log messages providing information for better understanding of the internal behavior of a software
	DLT_LOG_DEBUG	0x05	Log messages, which are usable only for debugging of a software
	DLT_LOG_VERBOSE	0x06	Log messages with the highest communicative level, here all possible states, information and everything else can be logged
Description	This type describes the log level for each log message.		
Variation	--		

] ()

8.8.10 Dlt_MessageTraceType

[SWS_Dlt_00231] [

Name	Dlt_MessageTraceType		
Kind	Enumeration		
Range	DLT_TRACE_VARIABLE	0x01	For tracing the value of a variable
	DLT_TRACE_FUNCTION_IN	0x02	For tracing the calling of a function
	DLT_TRACE_FUNCTION_OUT	0x03	For tracing the returning of a function

	DLT_TRACE_STATE	0x04	For tracing a state of a state machine
	DLT_TRACE_VFB	0x05	For tracing RTE Events
Description	This type describes labels for trace messages.		
Variation	--		

] ()

8.8.11 Dlt_MessageArgumentCount

[SWS_Dlt_00235] [

Name	Dlt_MessageArgumentCount
Kind	Type
Derived from	uint16
Description	This type describes the count of arguments provided to a message.
Variation	--

] ()

8.8.12 Dlt_MessageLogInfoType

[SWS_Dlt_00236] [

Name	Dlt_MessageLogInfoType		
Kind	Structure		
Elements	argCount	Dlt_MessageArgumentCount	--
	logLevel	Dlt_MessageLogLevelType	--
	options	Dlt_MessageOptionsType	--
	contextId	Dlt_ContextIDType	--
	appId	Dlt_ApplicationIDType	--
Description	--		
Variation	--		

] ()

8.8.13 Dlt_MessageTraceInfoType

⌈**[SWS_Dlt_00237]** ⌈

Name	Dlt_MessageTraceInfoType		
Kind	Structure		
Elements	traceInfo	Dlt_MessageTraceType	--
	options	Dlt_MessageOptionsType	--
	context	Dlt_ContextIDType	--
	appId	Dlt_ApplicationIDType	--
Description	--		
Variation	--		

⌋ ()

8.8.14 Dlt_MessageLogChannelNameType

[SWS_Dlt_00232] ⌈

Name	Dlt_LogChannelNameType		
Kind	Array	Element type	uint8
Size	4 Elements		
Description	This type describes the LogChannel name.		
Variation	--		

⌋ ()

8.8.15 Dlt_AssignmentOperation

[SWS_Dlt_00730] ⌈

Name	Dlt_AssignmentOperation		
Kind	Enumeration		
Range	DLT_ASSIGN_ADD	0x01	Adding a LogChannel assignment
	DLT_ASSIGN_REMOVE	0x02	Removing a LogChannel assignment
Description	Adding or removing a LogChannel assignment for the given tuple of ApplicationId/ContextId.		
Variation	--		

⌋ ()

8.9 Ports

8.9.1 Dlt_ControlService_{SW-C}

[SWS_Dlt_00499] ⌈

Name	ControlService_{SW-C}		
Kind	ProvidedPort	Interface	DltControlService
Description	Through this port SW-Cs can control log settings and other configurationitems of DLT.		
Variation	SW-C = {ecuc(Dlt/DltSwc.SHORT-NAME)}		

⌋ ()

8.9.2 Dlt_InjectCallback_{SW-C}

[SWS_Dlt_00778] ⌈

Name	InjectCallback_{SW-C}		
Kind	RequiredPort	Interface	InjectionCallback
Description	Callback Port to registered Application, which processes Injection.		
Variation	SW-C = {ecuc(Dlt/DltSwc.SHORT-NAME)}		

⌋ ()

8.9.3 Dlt_SessionControlCallback_{SW-C}

[SWS_Dlt_00779] ⌈

Name	SessionControlCallback_{SW-C}		
Kind	RequiredPort	Interface	LogTraceSessionControl
Description	Port used by Dlt to notify registered SW-C about LogLevel/TraceLevel Changes.		
Variation	SW-C = {ecuc(Dlt/DltSwc.SHORT-NAME)}		

⌋ ()

8.9.4 Dlt_SwcMessageService_{SW-C}

[SWS_Dlt_91001] ⌈

Name	SwcMessageService_{SW-C}		
Kind	ProvidedPort	Interface	DltSwcMessageService
Description	Through this port SW-Cs can register/unregister their contexts and send out log and trace messages.		
Port Defined Argument Value(s)	Type	Dlt_SessionIDType	
	Value	{ecuc(Dlt/DltSwc/DltSwcSessionId.value)}	
Variation	SW-C = {ecuc(Dlt/DltSwc.SHORT-NAME)}		

⌋ ()

9 Sequence diagrams

9.1 Dlt initialization

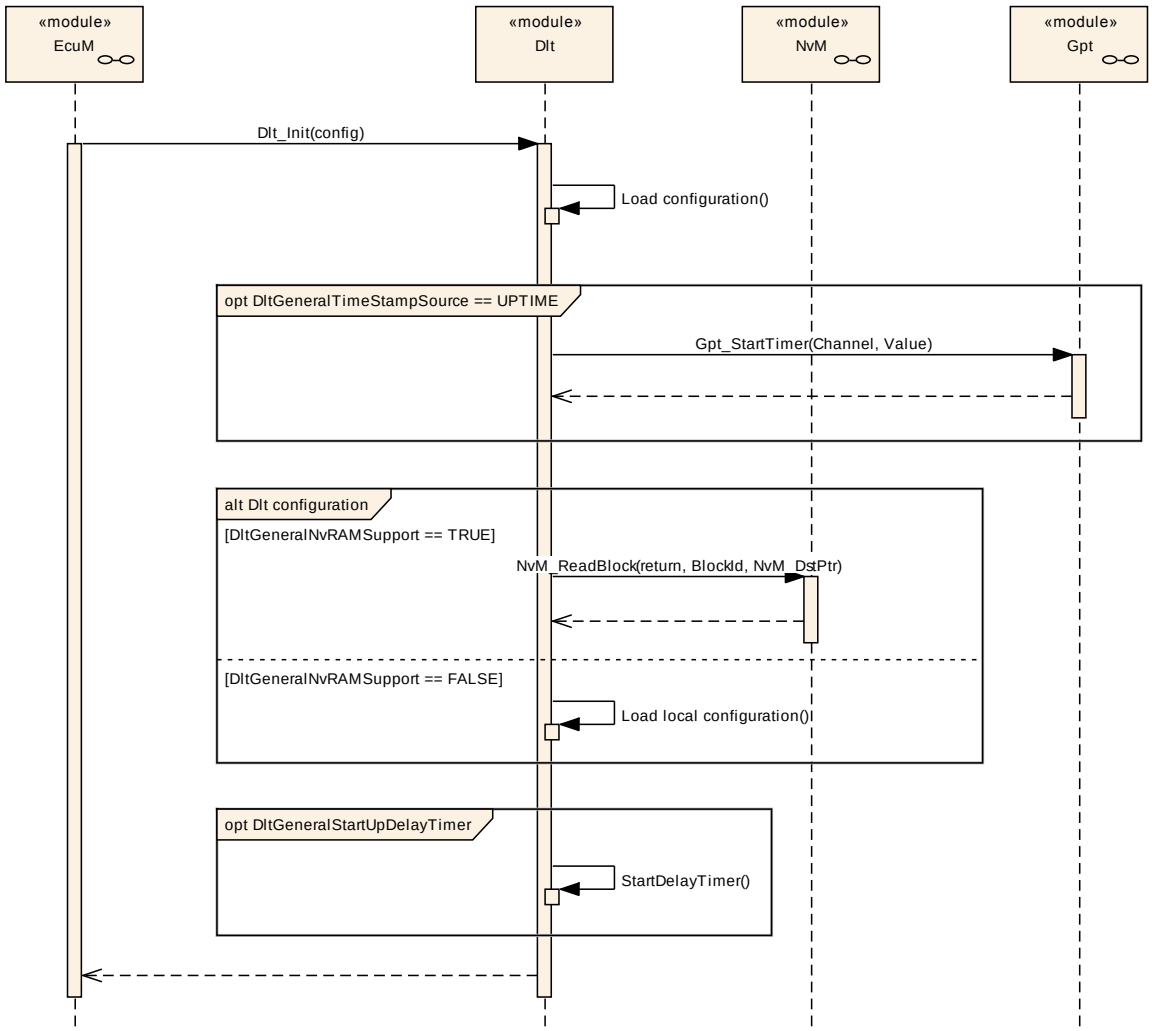


Figure 9-1: Dlt initialization

9.2 Overview of Dlt message transmission on one LogChannel

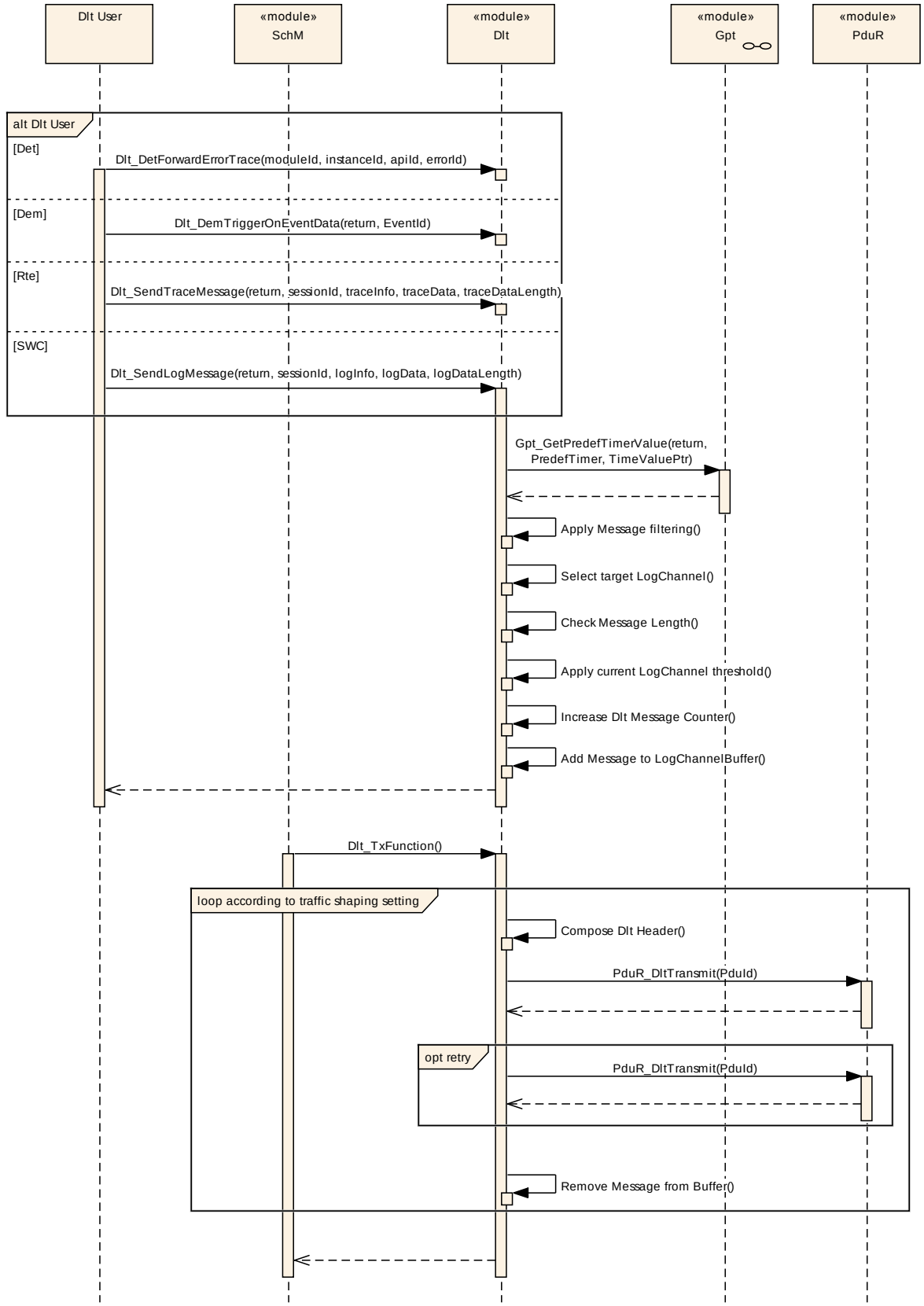


Figure 9-2: Overview of Dlt message transmission on one LogChannel

9.3 SetLogLevelFilter

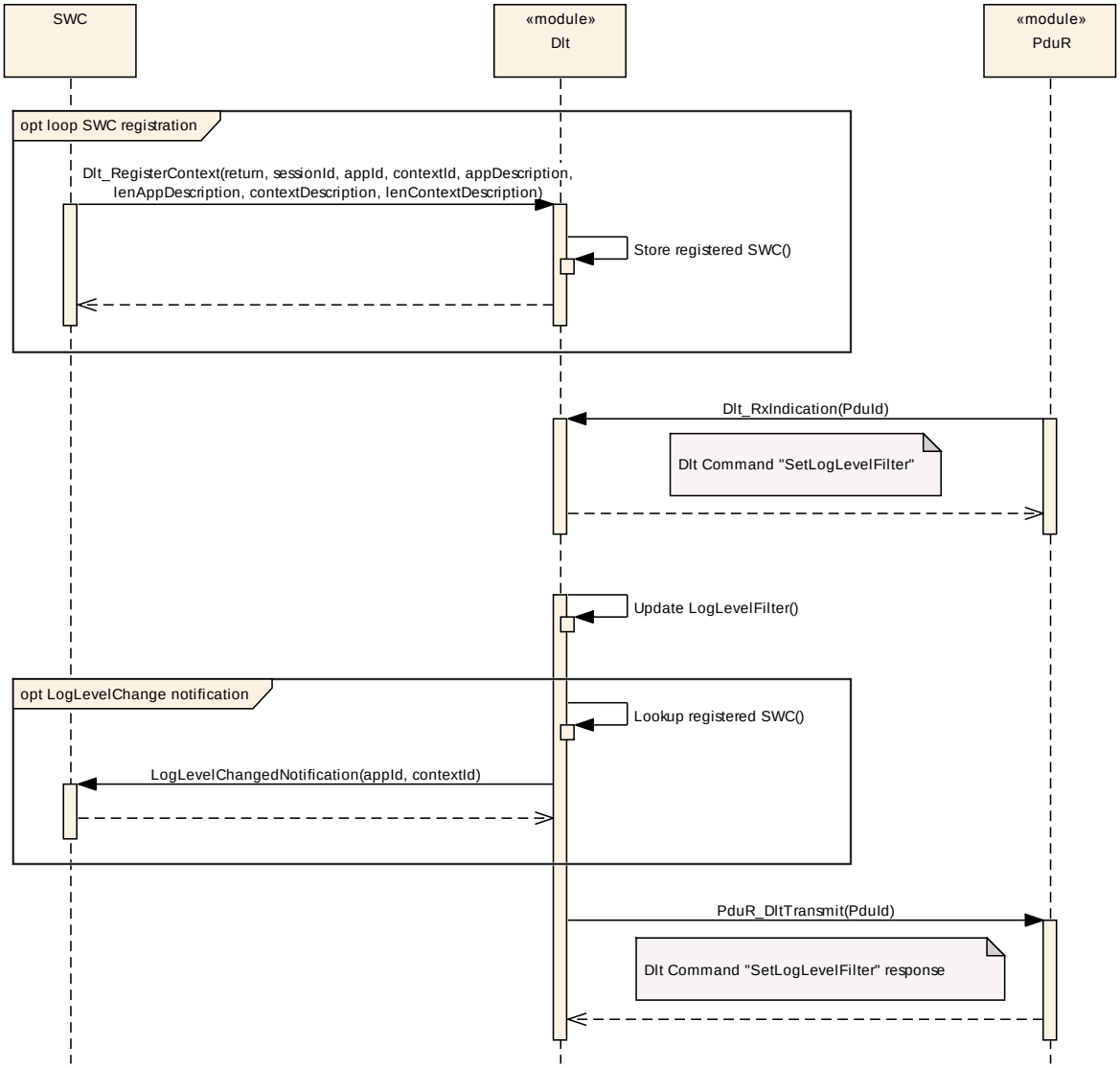


Figure 9-3: SetLogLevelFilter

9.4 Buffer overflow indication

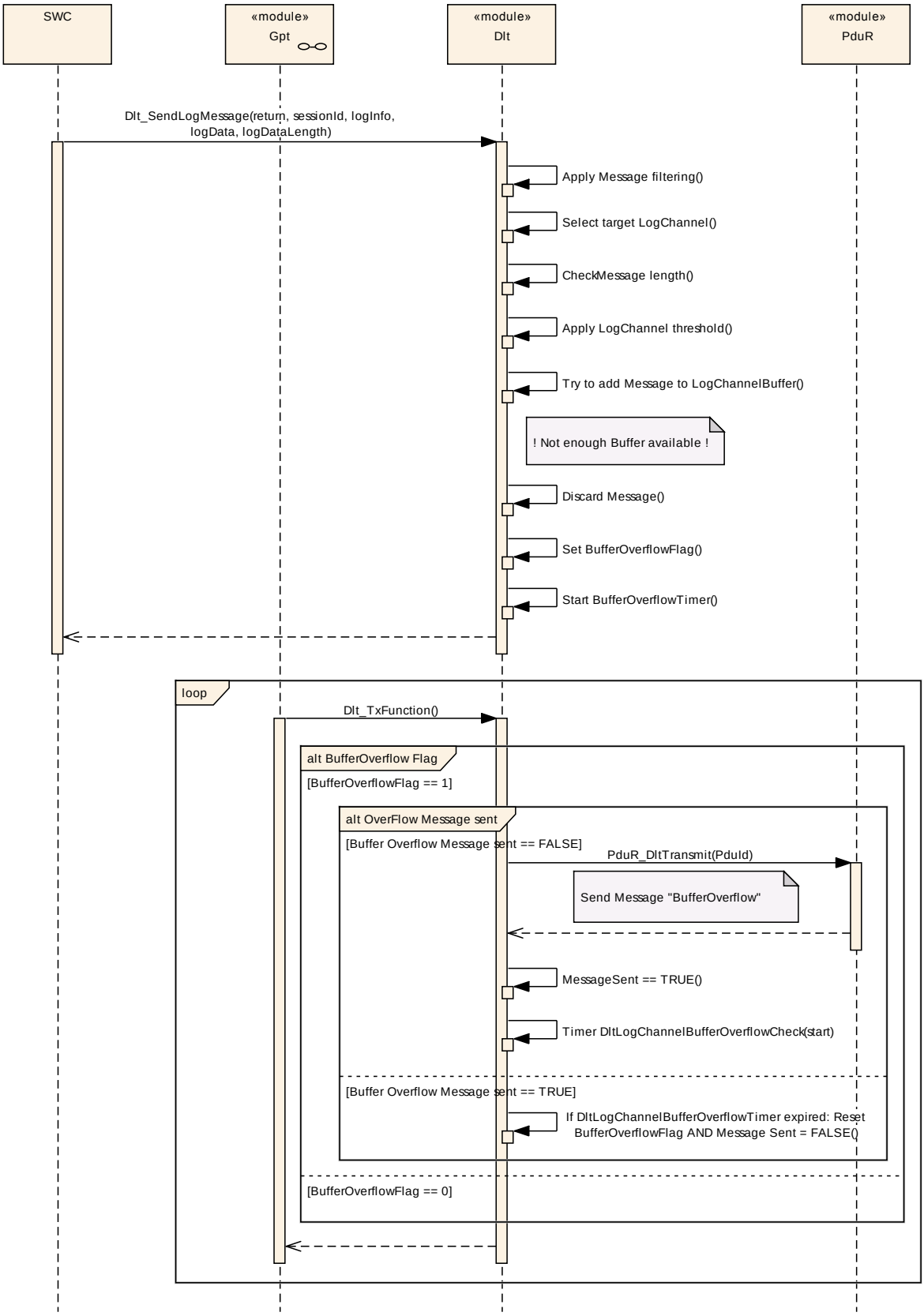


Figure 9-4: Buffer overflow indication

10 Configuration specification

Chapter 10.1 specifies the structure (containers) and the parameters of the module Dlt.

Chapter 10.2 specifies additionally published information of the module Dlt.

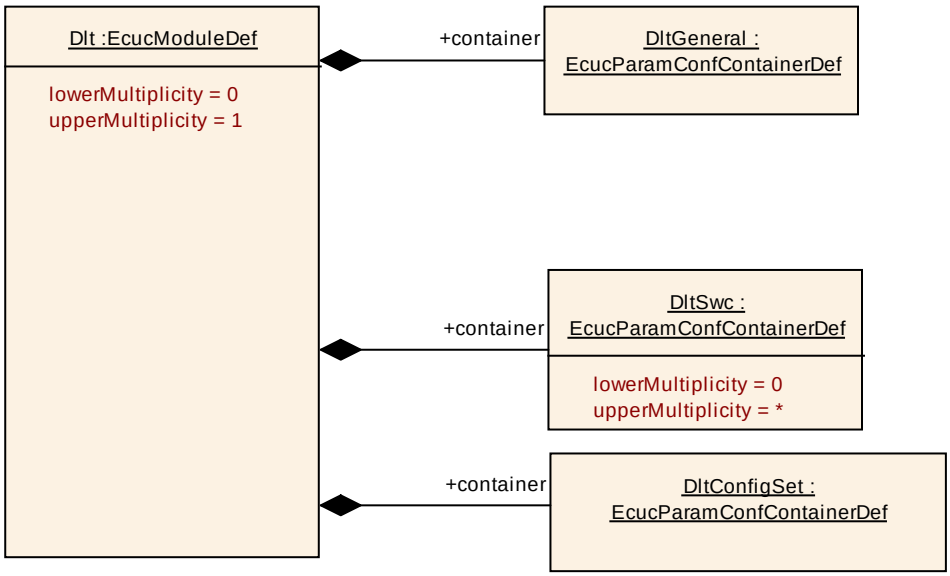
10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe chapters 0 and chapter 8.

10.1.1 Dlt

SWS Item	**ECUC_Dlt_00800 :**
Module Name	*Dlt*
Module Description	--
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DltConfigSet	1	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption.
DltGeneral	1	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption.
DltSwc	0..*	Contains necessary configuration parameters of the AUTOSAR Dlt module to interact with SWCs.



10.1.2 DltGeneral

SWS Item	**ECUC_Dlt_00809 :**
Container Name	DltGeneral
Description	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption.
Configuration Parameters	

SWS Item	ECUC_Dlt_00840 :		
Name	DltGeneralDevErrorDetect		
Description	If the Default Error Tracer (Det) shall be used, this parameter shall be set to TRUE. Otherwise, it shall be set to FALSE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Dlt_00847 :		
Name	DltGeneralInjectionSupport		
Description	Enables or disables the Dlt Injection feature.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Dlt_00846 :		
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.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Dlt_00848 :		
Name	DltGeneralRxDataPathSupport		
Description	Enables or disables the Rx Data Path to control the Dlt module.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: At least one RxPdu needs to be configured if DltGeneralRxDataPathSupport = TRUE		

SWS Item	ECUC_Dlt_00897 :
Name	DltGeneralStartUpDelayTimer
Description	Configurable delay in s of starting the transmission of Log and Trace

	messages after the Dlt module has been initialized.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 10]		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dlt_00850 :**		
Name	DltGeneralTimeStampSupport		
Description	If a Time Stamp shall be added to the Dlt messages, this configuration parameter shall be set to TRUE. Otherwise, it shall be set to FALSE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dlt_00849 :**		
Name	DltGeneralTrafficShapingSupport		
Description	Enables or disables the TrafficShaping feature to limit the maximum bandwidth for Dlt messages. If enabled, the maximum bandwidth can be configured per LogChannel.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

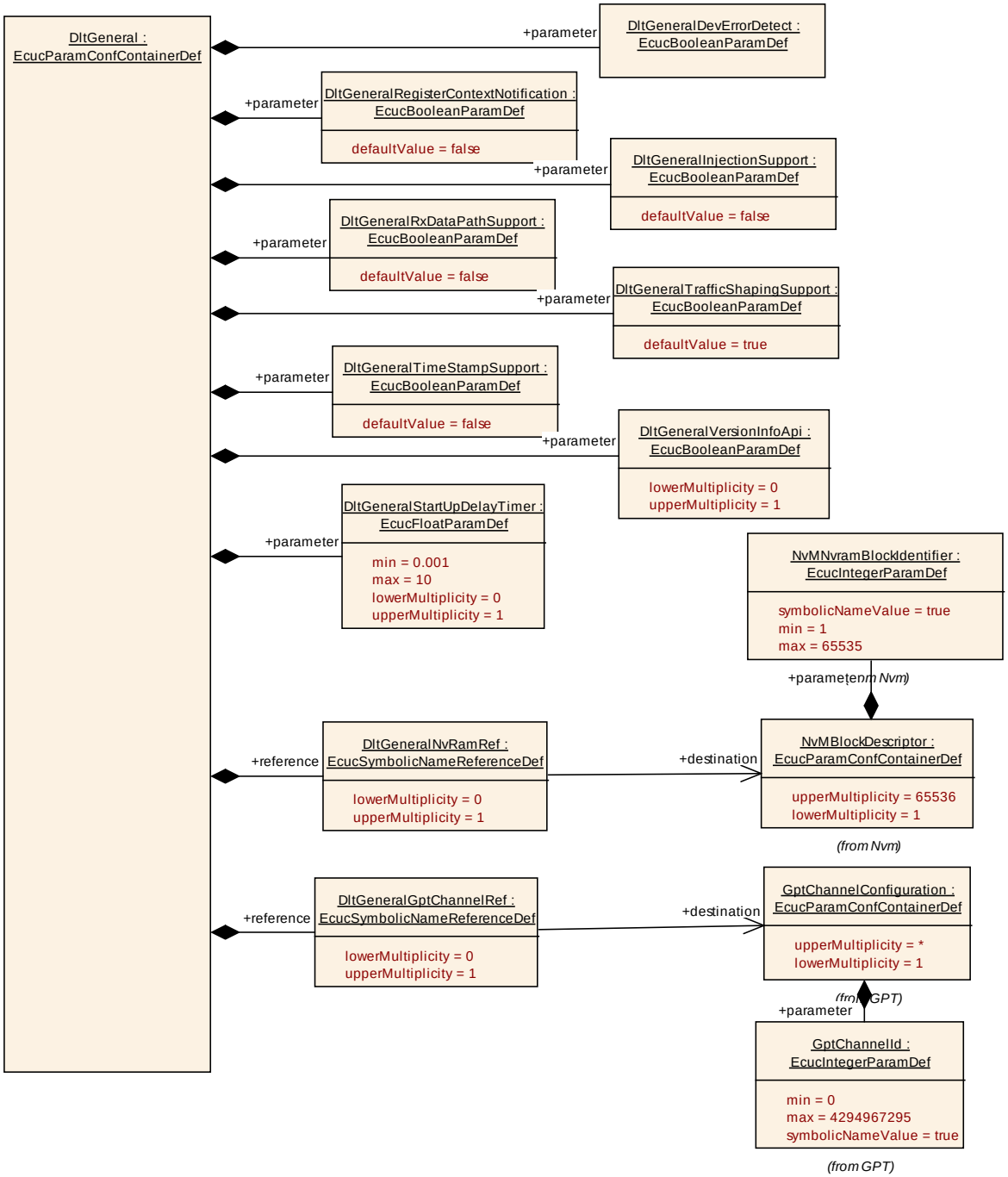
SWS Item	**ECUC_Dlt_00844 :**		
Name	DltGeneralVersionInfoApi		
Description	Pre-processor switch for enabling Version Info API support. • True: version information API activated • False: version information API deactivated		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Multiplicity	false		

Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dlt_00905 :**		
Name	DltGeneralGptChannelRef		
Description	If TimeStampSupport is used the Dlt module shall fetch the time from the Gpt module by calling Gpt_GetTimeElapsed with the here referenced GptChannel. The tick duration can be deduced from the GptChannelTickFrequency parameter of the GptChannelConfiguration container. This is necessary to calculate the microsecond resolution timestamp output in the Dlt message. A GPT timer shall be used which starts with value 0 at ECU start-up according to the PRS Dlt Protocol Specification.		
Multiplicity	0..1		
Type	Symbolic name reference to [GptChannelConfiguration]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: DltGeneralTimeStampSupport is set to TRUE		

SWS Item	**ECUC_Dlt_00845 :**		
Name	DltGeneralNvRamRef		
Description	If the Dlt module shall be able to store modified parameters during runtime persistently, this reference shall be set and shall point to the NvmBlock.		
Multiplicity	0..1		
Type	Symbolic name reference to [NvMBlockDescriptor]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.1.3 DltSwc

SWS Item	**ECUC_Dlt_00856 :**
Container Name	DltSwc
Description	Contains necessary configuration parameters of the AUTOSAR Dlt module to interact with SWCs.
Post-Build Variant	true

Multiplicity			
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Dlt_00852 :**		
Name	DltSwcSessionId		
Description	An ECU wide unique ID to identify the port a SWC (instance) uses.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

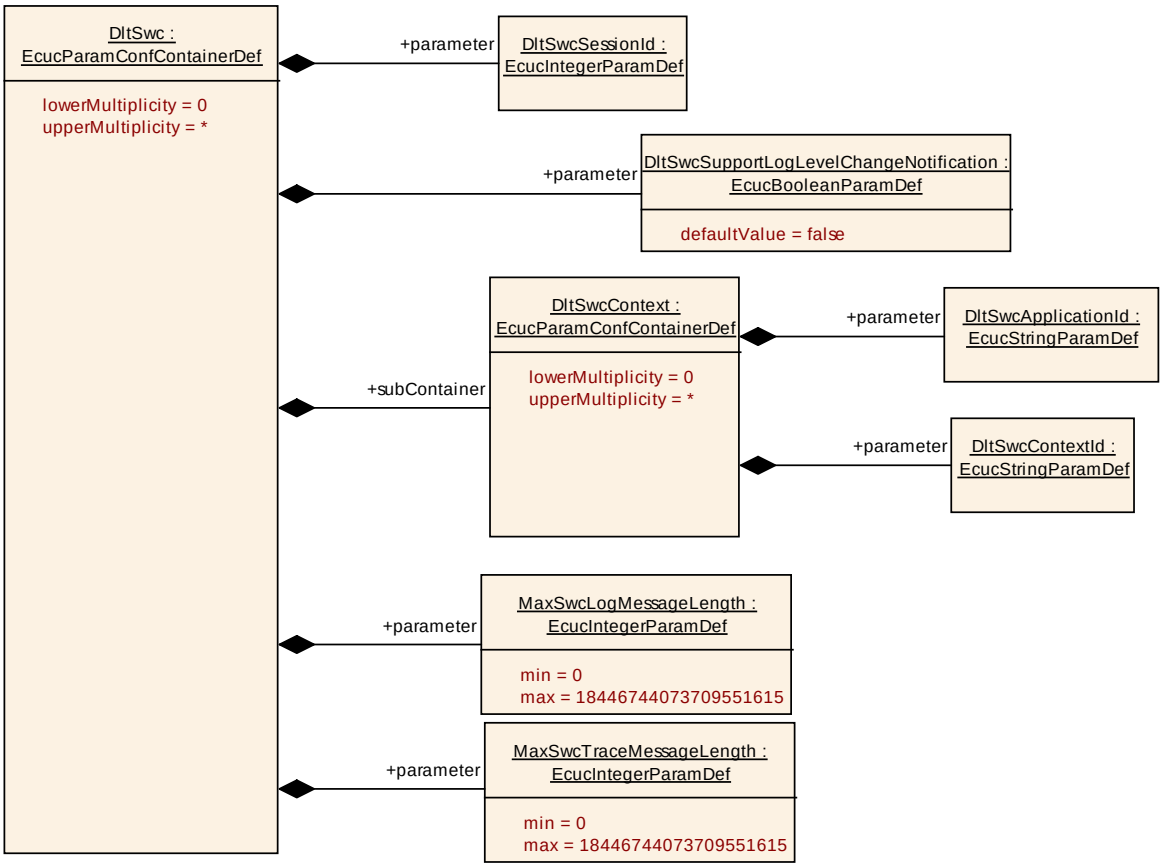
SWS Item	**ECUC_Dlt_00853 :**		
Name	DltSwcSupportLogLevelChangeNotification		
Description	Flag indicating, whether Dlt has to provide a R-Port for the notification of the SWC about LogLevel changes.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dlt_00909 :**		
Name	MaxSwcLogMessageLength		
Description	Defines the maximum allowed length (unit16) for LogMessages.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dlt_00910 :**		
Name	MaxSwcTraceMessageLength		
Description	Defines the maximum allowed length (unit16) for TraceMessages.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 ..		

	18446744073709551615		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DltSwcContext	0..*	This container contains the configuration of ApplicationId / ContextId pairs which are supported by this SWC.



10.1.4 DltSwcContext

SWS Item	**ECUC_Dlt_00854 :**
Container Name	DltSwcContext
Description	This container contains the configuration of ApplicationId / ContextId pairs which are supported by this SWC.
Post-Build Variant	true

Multiplicity			
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Dlt_00858 :**		
Name	DltSwcApplicationId		
Description	Abbreviation for the SWC (4 characters)		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dlt_00859 :**		
Name	DltSwcContextId		
Description	Abbreviation for the ContextId (4 characters)		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

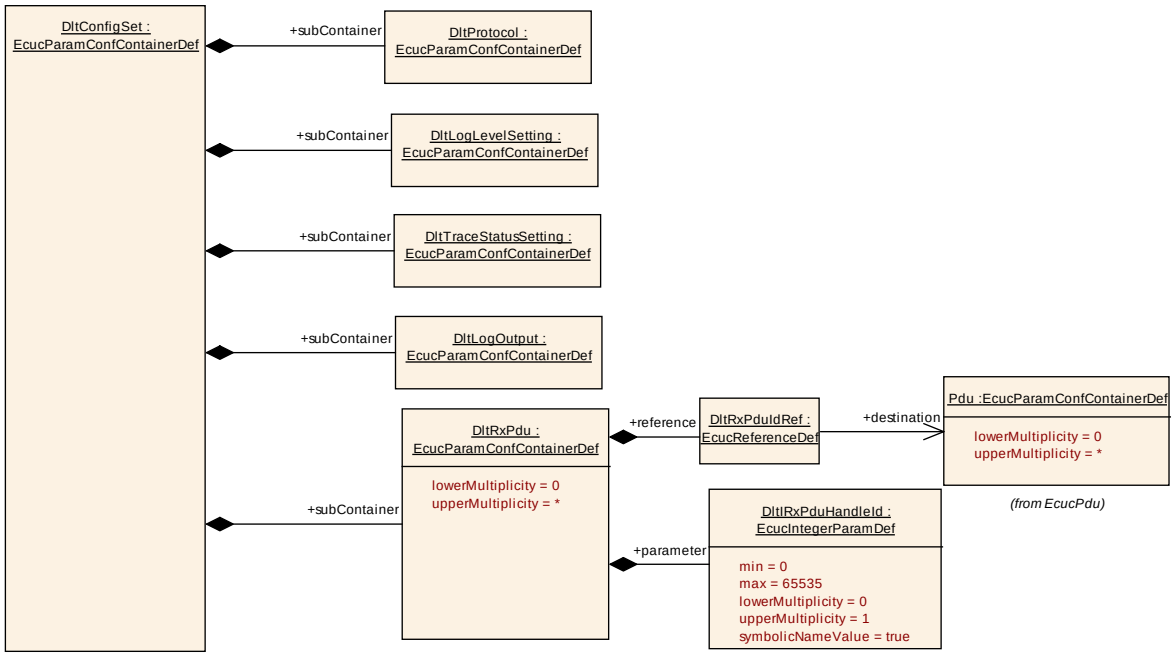
No Included Containers

10.1.5 DltConfigSet

SWS Item	**ECUC_Dlt_00842 :**
Container Name	DltConfigSet
Description	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DltLogLevelSetting	1	Contains settings for thresholds.
DltLogOutput	1	Contains settings for log/trace message output

DltProtocol	1	Configuration parameters for handling the specific protocol variants.
DltRxPdu	0..*	Contains the Pdu IDs to be used for Dlt control messages reception.
DltTraceStatusSetting	1	Contains settings for trace status



10.1.6 DltProtocol

SWS Item	**ECUC_Dlt_00832 :**
Container Name	DltProtocol
Description	Configuration parameters for handling the specific protocol variants.
Configuration Parameters	

SWS Item	**ECUC_Dlt_00811 :**		
Name	DltHeaderUseEcuId		
Description	Corresponds to field WEID (With ECU ID). If set ECU ID shall be placed in the header, else not. If the parameter DltGeneralNvRamRef is used this parameter defines the initial value for the corresponding NVRam entry. If the parameter DltGeneralNvRamRef is not set, Link-Time or Post-Build configuration shall be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: ECU

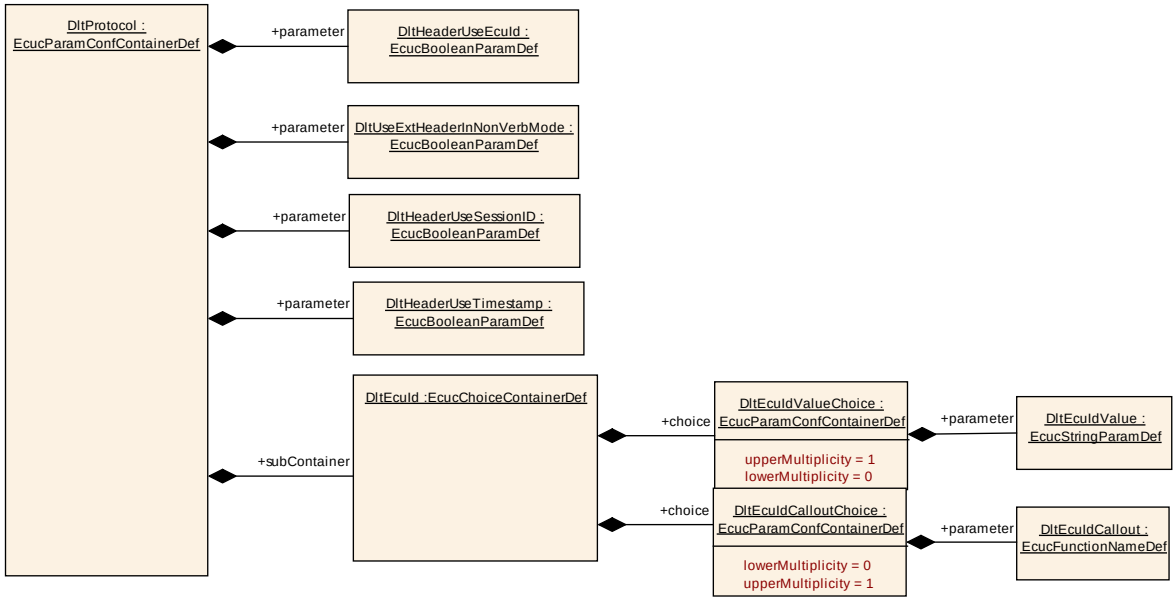
SWS Item	**ECUC_Dlt_00813 :**		
Name	DltHeaderUseSessionID		
Description	Corresponds to field WSID (with Session ID). If set the Session ID shall be placed in the header, else not. If the parameter DltGeneralNvRamRef is used this parameter defines the initial value for the corresponding NVRam entry. If the parameter DltGeneralNvRamRef is not set, Link-Time or Post-Build configuration shall be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00814 :**		
Name	DltHeaderUseTimestamp		
Description	Corresponds to field WTMS (With Timestamp). If set the timestamp shall be placed in the header, else not. If the parameter DltGeneralNvRamRef is used this parameter defines the initial value for the corresponding NVRam entry. If the parameter DltGeneralNvRamRef is not set, Link-Time or Post-Build configuration shall be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: Can only be true if DltImplementTimestamp is true.		

SWS Item	**ECUC_Dlt_00812 :**		
Name	DltUseExtHeaderInNonVerbMode		
Description	Non Verbose messages (opposed to verbose messages) do not need an extended header. If this flag is set to true the extended header shall also be used for non verbose messages. If DltGeneralNvRAMSupport is enabled this parameter is the initial value for the corresponding NVRam entry. If DltGeneralNvRAMSupport is not set, Link-Time or Post-Build configuration shall be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: Can only be true if DltImplementExtendedHeader is true.		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***

DltEcuId	1	This is a choice container to choose between a EcuId value or a callout to get the EcuId.



10.1.7 DltEcuId

SWS Item	**ECUC_Dlt_00860 :**
Choice container Name	DltEcuId
Description	This is a choice container to choose between a EcuId value or a callout to get the EcuId.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
DltEcuIdCalloutChoice	0..1	EcuId via user defined callout.
DltEcuIdValueChoice	0..1	EcuId value configuration

10.1.8 DltEcuIdCalloutChoice

SWS Item	**ECUC_Dlt_00902 :**
Container Name	DltEcuIdCalloutChoice
Description	EcuId via user defined callout.
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**ECUC_Dlt_00862 :**

Name	DltEcuIdCallout		
Description	If this choice is used the EcuId shall be fetched by calling the here configured callout function.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.1.9 DltEcuIdValueChoice

SWS Item	**ECUC_Dlt_00901 :**
Container Name	DltEcuIdValueChoice
Description	EcuId value configuration
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**ECUC_Dlt_00861 :**		
Name	DltEcuIdValue		
Description	If this choice is used the EcuId shall be taken from the configured string. This is the name of the ECU for use within the Dlt protocol. If you want to use a number representation type this as character.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

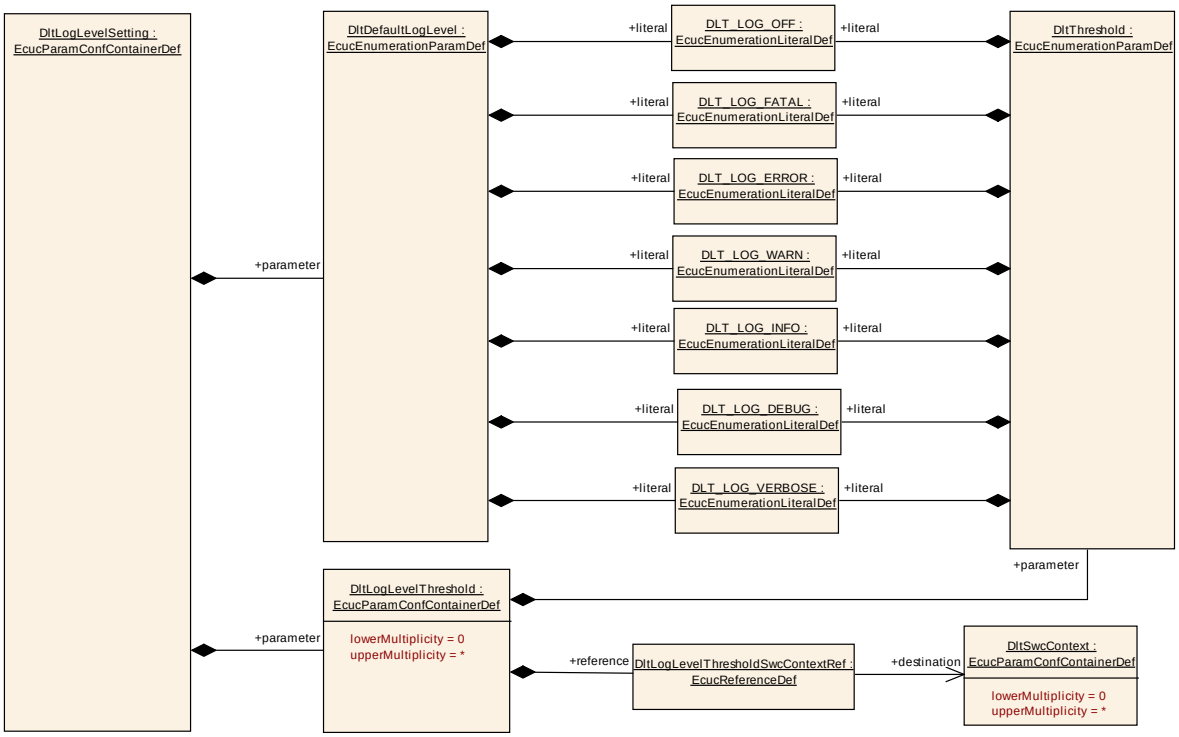
10.1.10 DltLogLevelSetting

SWS Item	**ECUC_Dlt_00863 :**
Container Name	DltLogLevelSetting

Description	Contains settings for thresholds.
Configuration Parameters	

SWS Item	**ECUC_Dlt_00864 :**		
Name	DltDefaultLogLevel		
Description	This is the effective log level used in case no filter matches the given AppicationId and ContextId. This can be seen as a fall-through filter definition with wildcard for AppicationId and ContextId, which will be used, when no other filter matches.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DLT_LOG_DEBUG	--	
	DLT_LOG_ERROR	--	
	DLT_LOG_FATAL	--	
	DLT_LOG_INFO	--	
	DLT_LOG_OFF	--	
	DLT_LOG_VERBOSE	--	
	DLT_LOG_WARN	--	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers



10.1.11 DltLogChannelAssignment

SWS Item	**ECUC_Dlt_00887 :**		
Container Name	DltLogChannelAssignment		
Description	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned log channel.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Dlt_00896 :**		
Name	DltLogChannelAssignmentSwcContextRef		
Description	Reference to an ApplicationId/ContextId pair that is assigned to a DltLogChannel.		
Multiplicity	1		
Type	Reference to [DltSwcContext]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**ECUC_Dlt_00888 :**		
Name	DltLogChannelRef		
Description	Reference to a DltLogChannel that is assigned to an ApplicationId / ContextId pair.		
Multiplicity	1		
Type	Reference to [DltLogChannel]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

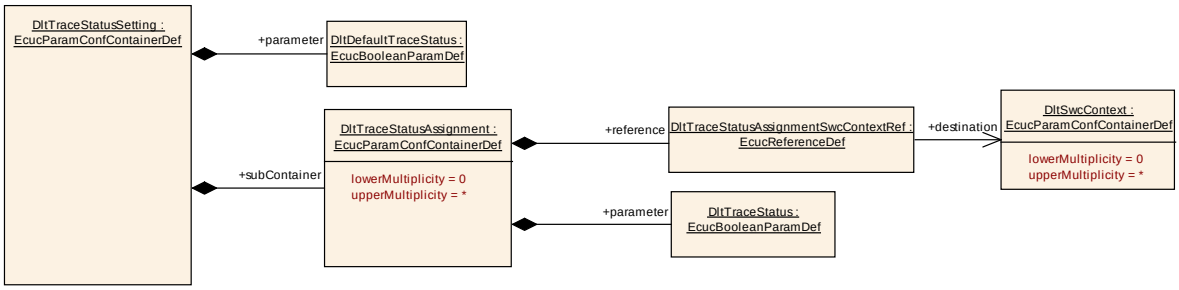
10.1.12 DltTraceStatusSetting

SWS Item	**ECUC_Dlt_00869 :**
Container Name	DltTraceStatusSetting
Description	Contains settings for trace status
Configuration Parameters	

SWS Item	**ECUC_Dlt_00870 :**
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

Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DltTraceStatusAssignment	0..*	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned trace status.



10.1.13 DltTraceStatusAssignment

SWS Item	**ECUC_Dlt_00871 :**		
Container Name	DltTraceStatusAssignment		
Description	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned trace status.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Dlt_00874 :**		
Name	DltTraceStatus		
Description	Trace status for the given ApplicationId/ContextId tuple.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00895 :**		
Name	DltTraceStatusAssignmentSwcContextRef		
Description	Reference to an ApplicationId/ContextId pair to which a DltTraceStatus is assigned.		
Multiplicity	1		
Type	Reference to [DltSwcContext]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

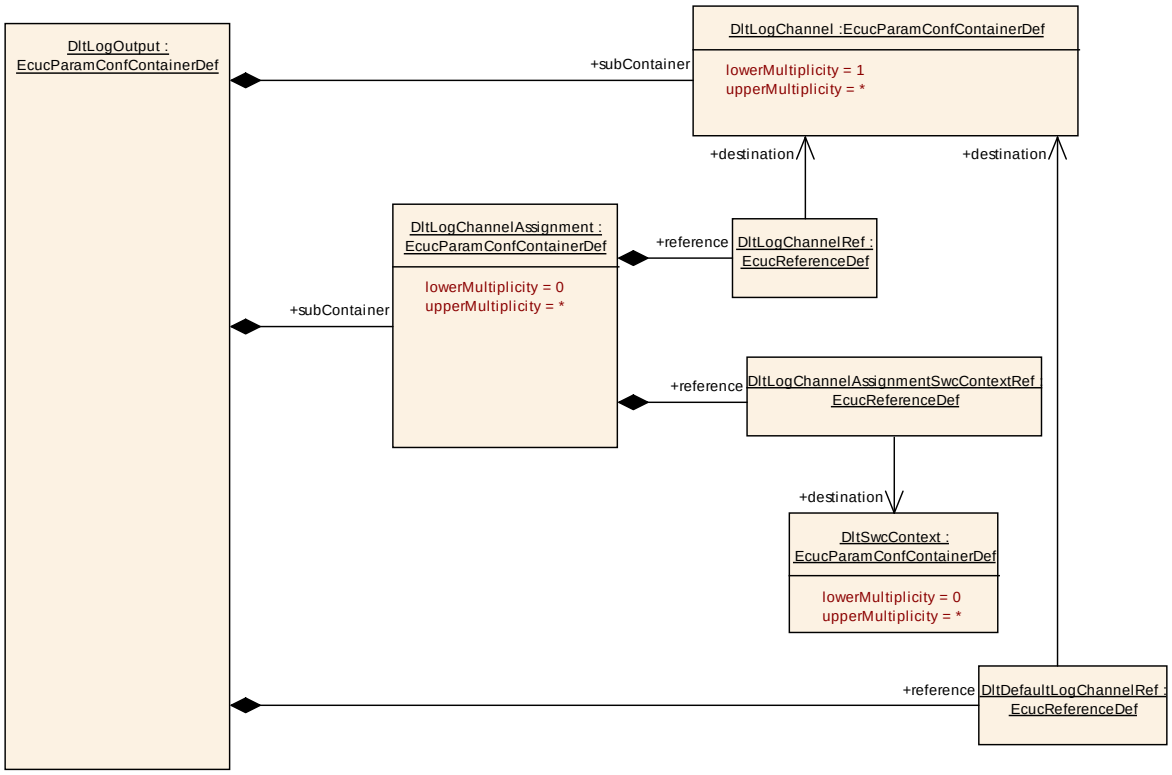
No Included Containers

10.1.14 DltLogOutput

SWS Item	**ECUC_Dlt_00875 :**		
Container Name	DltLogOutput		
Description	Contains settings for log/trace message output		
Configuration Parameters			

SWS Item	**ECUC_Dlt_00889 :**		
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		
Type	Reference to [DltLogChannel]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DltLogChannel	1..*	Contains settings for log/trace message output
DltLogChannelAssignment	0..*	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned log channel.



10.1.15 DltLogChannel

SWS Item	**ECUC_Dlt_00876 :**
Container Name	DltLogChannel
Description	Contains settings for log/trace message output
Configuration Parameters	

SWS Item	**ECUC_Dlt_00886 :**		
Name	DltLogChannelBufferOverflowTimer		
Description	Specifies the cycle time in seconds for resetting the buffer overflow flag in case a buffer overflow occurred.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0.001 .. 1]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00881 :**	
Name	DltLogChannelBufferSize	
Description	Buffer size in bytes for the LogChannel specific message buffer.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	

Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00877 :**		
Name	DltLogChannelId		
Description	This is the 4 ASCII character long name of the log channel as used in the Dlt control messages as parameter name Dlt_interface		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00882 :**		
Name	DltLogChannelMaxMessageLength		
Description	The maximum length of a Dlt log or trace message.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00884 :**		
Name	DltLogChannelMaxNumOfRetries		
Description	The maximum length of a Dlt log or trace message.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	0		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00878 :**	
Name	DltLogChannelThreshold	
Description	LogLevel Threshold	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DLT_LOG_DEBUG	--

	DLT_LOG_ERROR	--	
	DLT_LOG_FATAL	--	
	DLT_LOG_INFO	--	
	DLT_LOG_OFF	--	
	DLT_LOG_VERBOSE	--	
	DLT_LOG_WARN	--	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

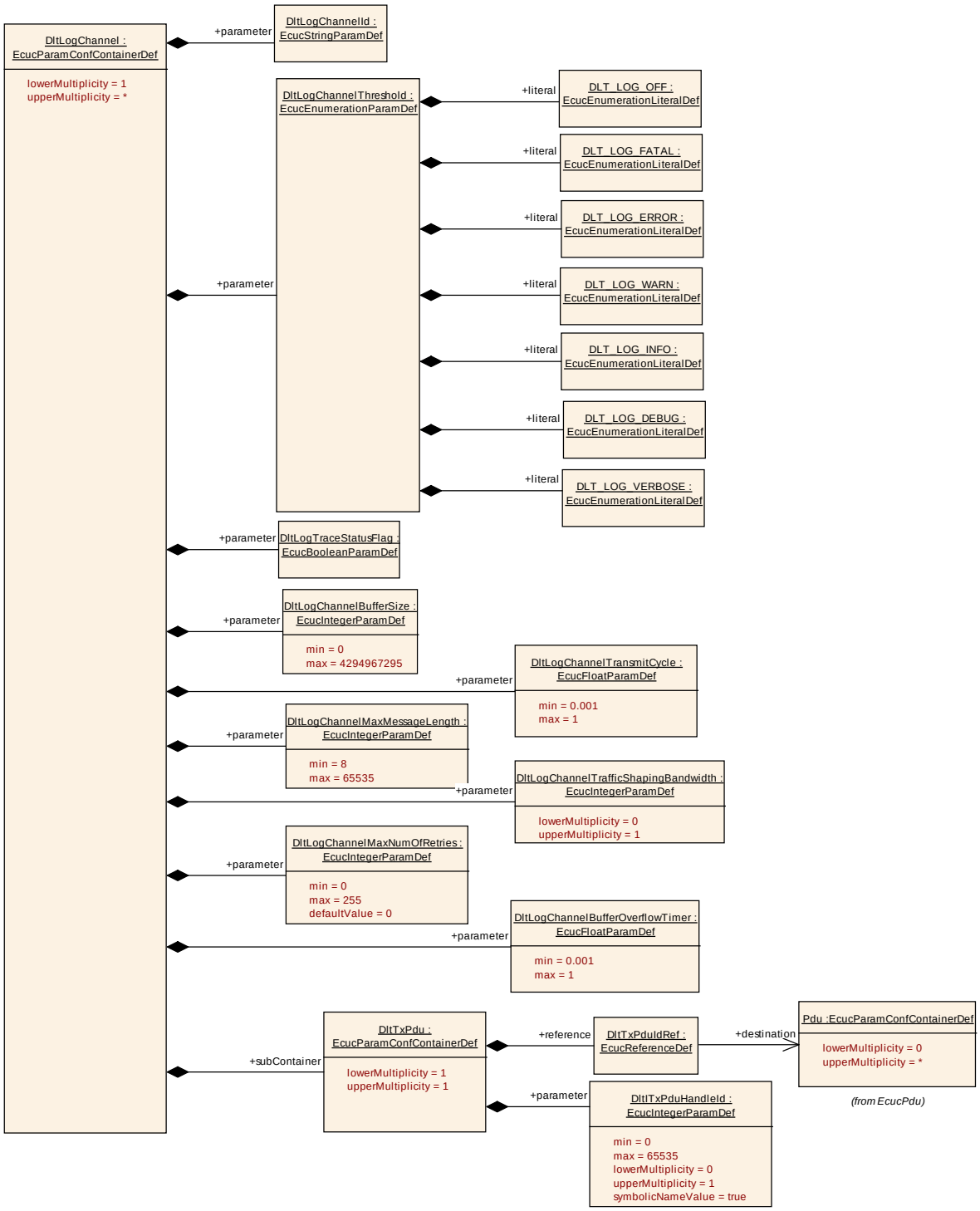
SWS Item	**ECUC_Dlt_00883 :**		
Name	DltLogChannelTrafficShapingBandwidth		
Description	Set the maximum possible bandwith in bit/s.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: DltGeneralTrafficShapingSupport enabled		

SWS Item	**ECUC_Dlt_00885 :**		
Name	DltLogChannelTransmitCycle		
Description	Specifies the cycle time in seconds of the transmit functionality of this log channel.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0.001 .. 1]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00879 :**
Name	DltLogTraceStatusFlag
Description	Parameter to turn on/off on this LogChannel completely.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--
Post-Build Variant Value	true

Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DltTxPdu	1	Contains the configuration parameters of the AUTOSAR Dlt module's Tx Pdus.



10.1.16 DltTxPdu

SWS Item	**ECUC_Dlt_00907 :**
Container Name	DltTxPdu
Description	Contains the configuration parameters of the AUTOSAR Dlt module's Tx Pdus.
Configuration Parameters	

SWS Item	**ECUC_Dlt_00893 :**		
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	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00892 :**		
Name	DltTxPduIdRef		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.17 DltRxPdu

SWS Item	**ECUC_Dlt_00900 :**		
Container Name	DltRxPdu		
Description	Contains the Pdu IDs to be used for Dlt control messages reception.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Dlt_00899 :**
Name	DltIRxPduHandleId

Description	The numerical value used as the ID of this I-PDU. The DltRxPduHandleId is required by the API calls Dlt_RxIndication, Dlt_TpRxIndication, Dlt_StartOfReception and Dlt_CopyRxData to receive I-PDUs from the PduR (DltIPduDirection: Receive), as well as the PduId passed to an Rx-I-PDU-callout.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dlt_00898 :**		
Name	DltRxPduIdRef		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

Additional module-specific published parameters are listed below if applicable.