

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

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R25-11

Document Change History			
Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	- Removed chapter 7.1.5 Log messages from DET - Removed SWS_Dlt_00432 - Minor corrections - Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	- Added error code DLT_E_ERROR to SWS_Dlt_00736 - Added error code DLT_E_NOT_SUPPORTED to SWS_Dlt_91011 - Minor corrections - Editorial changes
2023-10-23	R23-11	AUTOSAR Release Management	- Added Message Tags specifications - Minor corrections - Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	- Added DltProtocolVersion Parameter - Added Privacy flags and message tags - Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	- Bugfixes and corrections - Editorial changes





2020-11-30	R20-11	AUTOSAR Release Management	• Added subcontainer and definition for parameter DltLogLevelThreshold and for DltGeneralNvRAMSupport • Assigned new ID for Imported Types because of duplicated ID • Minor corrections and bugfixes • Editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Tracing to RS LogAndTrace • Interaction DLT <> DEM removed • Minor corrections
2017-12-08	4.3.1	AUTOSAR Release Management	• Introduced use of StbM • Added APIs regarding Rx data path • Removed redundant items • Editorial changes
2016-11-30	4.3.0	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
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 • Update of interface to Dcm
2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) 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 work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work 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 work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	10
2	Acronyms and Abbreviations	11
2.1	Term and definition	11
3	Related documentation	13
3.1	Input documents & related standards and norms	13
3.2	Related standards and norms	13
3.3	Related specification	14
4	Constraints and assumptions	15
4.1	Limitations	15
4.2	Applicability to car domains	15
5	Dependencies to other modules	16
5.1	RTE	16
5.2	PDU Router	16
5.3	NvM	16
5.4	GPT	16
5.5	StbM	16
5.6	DET	16
5.7	DEM	16
6	Requirements Tracing	17
7	Functional specification	18
7.1	Dlt specification	18
7.1.1	Dlt commands	18
7.1.2	Dlt interaction with software components	19
7.1.2.1	Registering ApplicationIDs and ContextIDs to Dlt	20
7.1.2.2	Unregistering ApplicationIDs and ContextIDs to Dlt	21
7.1.2.3	Using port defined argument values for the definition of Session Ids	21
7.1.3	VFB trace	22
7.1.3.1	Interfaces provided by Dlt for VFB traces	22
7.1.3.2	Generating hook functions	23
7.1.4	Log messages from DEM	24
7.1.5	Recommendation for generation of Message IDs	24
7.1.6	Startup behavior	25
7.1.7	Persistent storage of configuration	25
7.1.8	Sending of Log and Trace Messages	26
7.1.8.1	Generating the timestamp	29
7.1.8.2	Message filtering	29

7.1.8.3	Select target LogChannel	31
7.1.8.4	Check message length	31
7.1.8.5	Apply LogChannel LogLevelThreshold	31
7.1.8.6	Copying Dlt message to the LogChannel buffer	32
7.1.8.7	Apply the message attributes, if any are present and supported	33
7.1.8.8	Sending messages from LogChannel Buffer	33
7.1.8.9	Create Dlt message header	34
7.1.8.9.1	Assembling the Dlt Header	35
7.1.8.9.2	Assembling the Dlt Extended Header	36
7.1.8.10	Removing messages from LogChannel buffer	36
7.1.9	Receiving of Dlt commands	36
7.1.9.1	SetLogLevel	37
7.1.9.2	SetTraceStatus	37
7.1.9.3	GetLogInfo	37
7.1.9.4	GetDefaultLogLevel	38
7.1.9.5	StoreConfiguration	38
7.1.9.6	ResetToFactoryDefault	38
7.1.9.7	SetMessageFiltering	39
7.1.9.8	SetDefaultLogLevel	39
7.1.9.9	SetDefaultTraceStatus	39
7.1.9.10	GetDefaultTraceStatus	40
7.1.9.11	GetLogChannelNames	40
7.1.9.12	GetTraceStatus	40
7.1.9.13	SetLogChannelAssignment	40
7.1.9.14	SetLogChannelThreshold	41
7.1.9.15	GetLogChannelThreshold	41
7.1.10	Sending of Dlt commands	41
7.1.10.1	BufferOverflowNotification	41
7.2	Error Classification	42
7.2.1	Development Errors	42
7.2.2	Runtime Errors	42
7.2.3	Production Errors	43
7.2.4	Extended Production Errors	43
8	API specification	44
8.1	Imported types	44
8.2	Type definitions	44
8.2.1	Dlt_ConfigType	44
8.2.2	Dlt_MessageType	45
8.2.3	Dlt_MessageNetworkTraceInfoType	45
8.3	Function definitions	45
8.3.1	Dlt_Init	46
8.3.2	Dlt_GetVersionInfo	46
8.3.3	Dlt_SendTraceMessage	47

8.3.4	Dlt_SendLogMessage	48
8.3.5	Dlt_RegisterContext	49
8.3.6	Dlt_UnregisterContext	50
8.3.7	Dlt_SetLogLevel	50
8.3.8	Dlt_SetTraceStatus	51
8.3.9	Dlt_GetLogInfo	52
8.3.10	Dlt_GetDefaultLogLevel	52
8.3.11	Dlt_StoreConfiguration	53
8.3.12	Dlt_ResetToFactoryDefault	54
8.3.13	Dlt_SetMessageFiltering	54
8.3.14	Dlt_SetDefaultLogLevel	55
8.3.15	Dlt_SetDefaultTraceStatus	55
8.3.16	Dlt_GetDefaultTraceStatus	56
8.3.17	Dlt_GetLogChannelNames	56
8.3.18	Dlt_GetTraceStatus	57
8.3.19	Dlt_SetLogChannelAssignment	57
8.3.20	Dlt_SetLogChannelThreshold	58
8.3.21	Dlt_GetLogChannelThreshold	59
8.3.22	Dlt_SendLogMessageWithAttributes	59
8.3.23	Dlt_SendTraceMessageWithAttributes	60
8.4	Callback notifications	61
8.4.1	Dlt_RxIndication	61
8.4.2	Dlt_TriggerTransmit	61
8.4.3	Dlt_TxConfirmation	62
8.4.4	Dlt_TpTxConfirmation	63
8.4.5	Dlt_CopyTxData	63
8.4.6	Dlt_StartOfReception	64
8.4.7	Dlt_TpRxIndication	65
8.4.8	Dlt_CopyRxData	66
8.5	Scheduled functions	66
8.5.1	Dlt_TxFunction	66
8.6	Expected interfaces	67
8.6.1	Mandatory interfaces	67
8.6.2	Optional interfaces	67
8.6.3	Configurable interfaces	68
8.7	Service Interfaces	69
8.7.1	Client-Server-Interfaces	69
8.7.1.1	DltControlService	69
8.7.1.2	InjectionCallback	75
8.7.1.3	LogTraceSessionControl	75
8.7.1.4	DltSwcMessageService	77
8.7.2	Implementation Data Types	81
8.7.2.1	Dlt_ApplicationIDType	81

8.7.2.2	Dlt_ContextIDType	81
8.7.2.3	Dlt_SessionIDType	81
8.7.2.4	Dlt_LogInfoType	82
8.7.2.5	Dlt_ContextIdInfoType	82
8.7.2.6	Dlt_ApplicationIdInfoType	83
8.7.2.7	Dlt_MessageOptionsType	83
8.7.2.8	Dlt_MessageLogInfoType	84
8.7.2.9	Dlt_MessageLogLevelType	84
8.7.2.10	Dlt_MessageTraceType	85
8.7.2.11	Dlt_MessageArgumentCount	85
8.7.2.12	Dlt_MessageTraceInfoType	86
8.7.2.13	Dlt_LogChannelNameInfoType	86
8.7.2.14	Dlt_AssignmentOperation	87
8.7.2.15	Dlt_MessageAttributesType	87
8.7.3	Ports	88
8.7.3.1	Dlt_ControlService	88
8.7.3.2	Dlt_InjectCallback_{SW-C}	88
8.7.3.3	Dlt_SessionControlCallback_{SW-C}	88
8.7.3.4	Dlt_SwcMessageService_{SW-C}	89
9	Sequence diagrams	90
9.1	Dlt initialization	90
9.2	Overview of Dlt message transmission on one LogChannel	91
9.3	SetLogLevelFilter	92
9.4	Buffer overflow indication	93
10	Configuration specification	94
10.1	Containers and configuration parameters	94
10.1.1	Dlt	94
10.1.2	DltGeneral	95
10.1.3	DltSwc	103
10.1.4	DltSwcContext	105
10.1.5	DltConfigSet	106
10.1.6	DltProtocol	108
10.1.7	DltEcuid	111
10.1.8	DltEcuidCalloutChoice	111
10.1.9	DltEcuidValueChoice	112
10.1.10	DltLogLevelSetting	113
10.1.11	DltLogLevelThreshold	114
10.1.12	DltLogChannelAssignment	115
10.1.13	DltTraceStatusSetting	117
10.1.14	DltTraceStatusAssignment	118
10.1.15	DltLogOutput	119
10.1.16	DltLogChannel	120

10.1.17 DltTxPdu	125
10.1.18 DltRxPdu	127
10.2 Published Information	129
A Mentioned Class Tables	130
B Change History	132
B.1 Change History of this document according to AUTOSAR Release R23-11	132
B.1.1 Added Specification Items in R23-11	132
B.1.2 Changed Specification Items in R23-11	137
B.1.3 Deleted Specification Items in R23-11	138
B.2 Change History of this document according to AUTOSAR Release R24-11	138
B.2.1 Added Specification Items in R24-11	138
B.2.2 Changed Specification Items in R24-11	138
B.2.3 Deleted Specification Items in R24-11	138
B.3 Change History of this document according to AUTOSAR Release R25-11	138
B.3.1 Added Specification Items in R25-11	138
B.3.2 Changed Specification Items in R25-11	139
B.3.3 Deleted Specification Items in R25-11	141

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 as shown in 1.1.

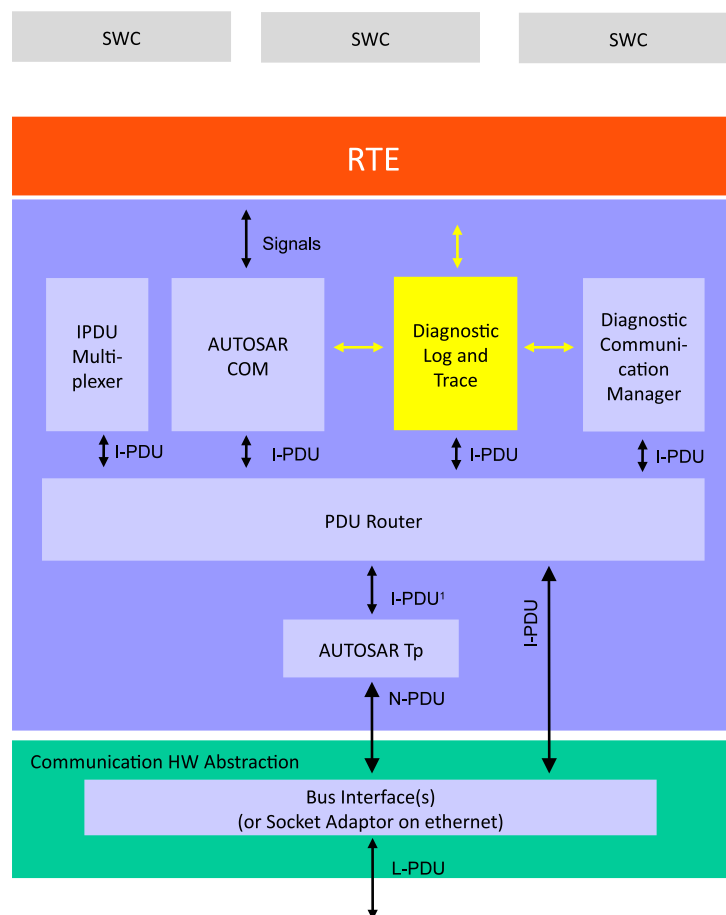


Figure 1.1: Location of the Dlt module

Please note: The Dlt Message Format, available Commands, and Protocol (to communicate with an external logging and tracing tool) are defined in [1].

2 Acronyms and Abbreviations

The glossary below includes acronyms, abbreviations and definitions relevant to the Diagnostic Log and Trace module that are not included in [2] or in [3].

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. The 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 ID is the name of an ECU, composed by four 8-bit ASCII characters (e.g., ABS0 or COMB).





Term	Description:
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	Message ID is the ID 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 and trace 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 sent.
Log Channel	A physical Communication bus which is used to transmit Dlt messages.
External client	An 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 & related standards and norms

- [1] Log and Trace Protocol Specification
AUTOSAR_FO_PRS_LogAndTraceProtocol
- [2] Glossary
AUTOSAR_FO_TR_Glossary
- [3] Requirements on Log and Trace
AUTOSAR_FO_RS_LogAndTrace
- [4] ISO 7498 – Information processing systems – Open Systems Interconnection –
Basic Reference Model
<https://www.iso.org>
ISO/IEC 7498-1:1994
- [5] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral
- [6] Specification of RTE Software
AUTOSAR_CP_SWS_RTE
- [7] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter
- [8] Specification of NVRAM Manager
AUTOSAR_CP_SWS_NVRAMManager
- [9] Specification of GPT Driver
AUTOSAR_CP_SWS_GPTDriver
- [10] Specification of Synchronized Time-Base Manager
AUTOSAR_CP_SWS_SynchronizedTimeBaseManager
- [11] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer
- [12] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

3.2 Related standards and norms

- [4, ISO-7498-1]

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software [\[5\]](#) which is also valid for Dlt.

4 Constraints and assumptions

4.1 Limitations

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.

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.

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 [6] (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[7].

5.3 NvM

In order to load and store altered configurations like filter settings and/or Log Channel assignments, the NvM module[8] can optionally be used.

5.4 GPT

In order to derive a time stamp, the GPT module[9] can be used for this purpose.

5.5 StbM

In order to get a synchronized time value (Local Time Base derived from Global Time Base) in standard/extended format., the StbM module[10] can be used for this purpose.

5.6 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[11]. However, the interaction with DET is optional.

5.7 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[12] using a CDD and/or a SW-C. No standardized interaction between DEM and DLT is available.

6 Requirements Tracing

The following tables reference requirements specified in an upper tracing level context and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement, this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_LT_00003]		[SWS_DIt_00241] [SWS_DIt_00243]
[RS_LT_00004]		[SWS_DIt_00252] [SWS_DIt_00254]
[RS_LT_00008]		[SWS_DIt_00284]
[RS_LT_00009]		[SWS_DIt_00276] [SWS_DIt_00277] [SWS_DIt_00285]
[RS_LT_00032]		[SWS_DIt_00643]
[RS_LT_00033]		[SWS_DIt_00021] [SWS_DIt_00245] [SWS_DIt_00769]
[RS_LT_00034]		[SWS_DIt_00516]
[RS_LT_00036]		[SWS_DIt_00003]
[RS_LT_00038]		[SWS_DIt_00252] [SWS_DIt_00254]
[RS_LT_00039]		[SWS_DIt_00078] [SWS_DIt_00453]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_DIt_00239]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_DIt_00239]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_DIt_00239]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_DIt_00271]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_DIt_00239]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_DIt_00239]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_DIt_00239]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_DIt_00239] [SWS_DIt_00437]

Table 6.1: Requirements Tracing

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] Supported Service ID to Dlt Command Name mapping

Upstream requirements: [RS_LT_00032](#)

[

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 Log Channel(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 Log Channel
0x22	GetLogChannelThreshold	Get the filter threshold for the given LogChannel
0x23	BufferOverflowNotification	Indication of a buffer overflow within the DLT module
0x24	SyncTimeStamp	Reports synchronized absolute time
0x13	GetSoftwareVersion	Get the ECU software version

]

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

7.1.2 Dlt interaction with software components

The Dlt module offers interfaces SW-Cs can use for sending Log and Trace Messages as shown in 7.1.

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 `DltSwcApplicationId/DltSwcContextId`. 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 `DltSwcApplicationId/DltSwcContextId`, an additional `DltSwcSessionId` 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 `DltSwcSessionId`s in calls to `SendLogMessage/SendTraceMessage` (details, see next chapters), the Dlt module provides a dedicated `PPortPrototype` per configured SW-C (see configuration parameter `DltSwcSessionId`) 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 `RPortPrototype` to notify the respective SW-C.

The information, which SW-C is responsible for which `DltSwcApplicationId/DltSwcContextId` tuples, is configured for the SW-C and/or updated by the SW-C during runtime with a call to `Dlt_RegisterContext` and `Dlt_UnregisterContext` respectively.

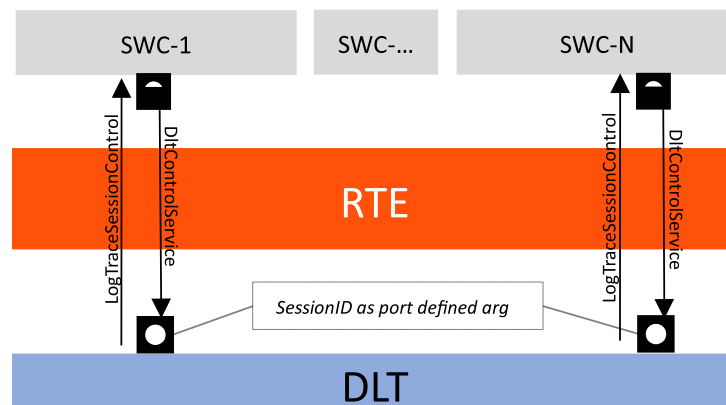


Figure 7.1: Interaction with SW-C (Port configuration)

[SWS_Dlt_00644] [The Dlt module shall provide a `PPortPrototype`, `SwcMessageService`, typed by interface `DltSwcMessageService` for each configured SW-C (see `DltSwc`).]

[SWS_Dlt_00645] [The `PPortPrototype` `SwcMessageService` typed by interface `DltSwcMessageService` has `Dlt_SessionIDType` as a port-defined argument.]

[SWS_Dlt_00646] [The Dlt module shall provide an `RPortPrototype`, `SessionControlCallback`, typed by interface `LogTraceSessionControl`, for each configured SW-C (see configuration container `DltSwc`), where the configuration parameter `DltSwcSupportLogLevelAndTraceStatusChangeNotification` is set to `TRUE`.]

[SWS_Dlt_00647] [The `DltSwcApplicationId/DltSwcContextId` 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 `DltSwcApplicationId/DltSwcContextId` at runtime.

Note: Because the development of SWCs are part of this specification, the Dlt module has to collect this information at runtime.

[SWS_Dlt_00765] [The Dlt module shall remember all tuples of `DltSwcApplicationIds` and `DltSwcContextIds` 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 `DltSwcApplicationId` and `DltSwcContextId`.]

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 `DltSwcMessageService` 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]

Upstream requirements: [RS_LT_00033](#)

[The Dlt module shall remember the relation between the registered tuple of [DltSwcApplicationId/DltSwcContextId](#), and the port interface where this tuple is registered.]

[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 [Dlt_GetLogInfo](#) containing the provided [DltSwcApplicationId/DltSwcContextId](#).]

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 [DltSwcApplicationIds](#) and [DltSwcContextIds](#) 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 [Dlt_GetLogInfo](#) containing the provided [DltSwcApplicationId/DltSwcContextId](#) with parameter `status` set to 5.]

7.1.2.3 Using port defined argument values for the definition of SessionIDs

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` starts with `"Rte_Dlt"`.

[SWS_Dlt_00284]

Upstream requirements: [RS_LT_00008](#)

[The Dlt module shall be compliant to the VFB trace described in the AUTOSAR_RTE_SWS.]

[SWS_Dlt_00276]

Upstream requirements: [RS_LT_00009](#)

[The Dlt module shall provide the possibility to trace all kinds of trace events described in the SWS RTE.]

[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 [short-Name](#) 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 BSWModuleDescription of the RTE.]

7.1.3.2 Generating hook functions

[SWS_Dlt_00285]

Upstream requirements: [RS_LT_00009](#)

[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.]

[SWS_Dlt_00277]

Upstream requirements: [RS_LT_00009](#)

[In the hook function the internal representation of [Dlt_SendTraceMessage](#) shall be called. This call shall be in non-verbose mode.]

[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 [DltSwcContextId](#).

In some cases some events can be bundled to the same ContextId. This shall mostly be done if a very large number of signals are traced.]

[SWS_Dlt_00337] [The ApplicationID shall be "VFBS".]

[SWS_Dlt_00484] [The Message Type (MSTP) entry in the generated trace message shall be set to [DLT_TYPE_APP_TRACE](#), the Message Trace Info (MSTI) entry in this case shall be set to [DLT_TRACE_VFB](#).]

[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, such that:

```

/*
  Check the trace status of the ContextId before calling Dlt_SendTraceMessage
  "vfb_actual_trace_status_contextXY" holds the trace status for a specific ContextId
*/
if (vfb_actual_trace_status_contextXY) {
  <internal_Dlt_SendTraceMessage>(...);
}

```

[SWS_Dlt_00282] [Dlt shall use for every VFB trace hook function an own `DltSwcContextId` and thus handle for every VFB trace `DltSwcContextId` 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

[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]

7.1.5 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.]

7.1.6 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]

Upstream requirements: [RS_LT_00036](#)

[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.]

[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 according to [\[SWS_Dlt_00651\]](#), using the configured filter settings.]

7.1.7 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]

Upstream requirements: [RS_LT_00039](#)

[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 [Dlt_StoreConfiguration](#).]

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

7.1.8 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.

Figure [7.2](#) provides an overview of the separate steps to send a Dlt message on the communication bus:

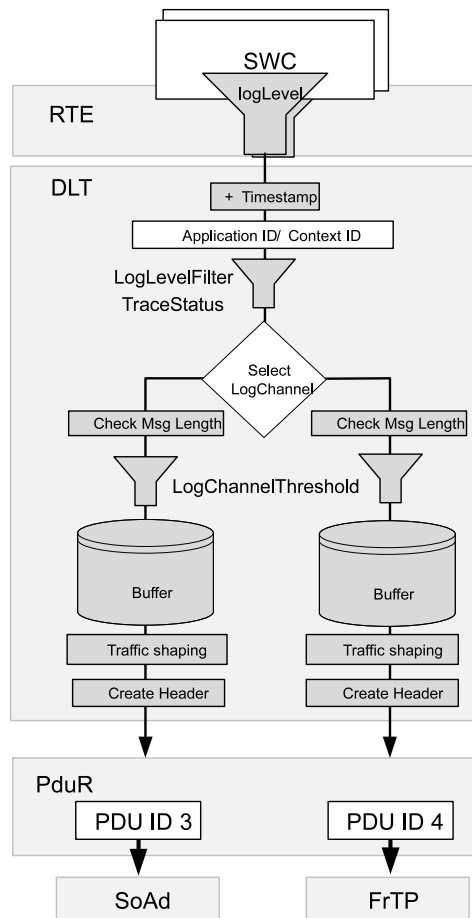


Figure 7.2: Example Tx Data Path

Sending of Log and Trace messages is done with the dedicated functions [Dlt_SendLogMessage](#) and [Dlt_SendTraceMessage](#).

Two additional functions exist that allow to give additional attributes to Log and Trace messages. These functions are named [Dlt_SendLogMessageWithAttributes](#) and [Dlt_SendTraceMessageWithAttributes](#) respectively. These two are pure supersets of the previously mentioned ones, which remain in the Standard for backwards-compatibility and convenience purposes, for the common case where no additional attributes are needed.

Please note that throughout this document, whenever the functions [Dlt_SendLogMessage](#) or [Dlt_SendTraceMessage](#) are being mentioned, these need to be understood as a shorthand notation for [Dlt_SendLogMessage](#) or [Dlt_SendLogMessageWithAttributes](#) and [Dlt_SendTraceMessage](#) or [Dlt_SendTraceMessageWithAttributes](#) respectively, unless otherwise noted.

[SWS_Dlt_00787]

Status: DRAFT

[Calling the function `Dlt_SendLogMessageWithAttributes` with the parameter attributes set to `NULL` shall be equivalent to calling the function `Dlt_SendLogMessage` with the remaining parameters.]

[SWS_Dlt_00782]

Status: DRAFT

[Calling the function `Dlt_SendTraceMessageWithAttributes` with the parameter attributes set to `NULL` shall be equivalent to calling the function `Dlt_SendTraceMessage` with the remaining parameters.]

[SWS_Dlt_00783]

Status: DRAFT

[If the configuration parameter `DltProtocolVersion` is set to 1, a call to `Dlt_SendLogMessageWithAttributes` or `Dlt_SendTraceMessageWithAttributes` where the attribute's argument is `non-NULL`, shall return with `DLT_E_NOT_SUPPORTED`.]

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

- Generate timestamp (see chapter "Generating the timestamp")
- Filter message (see chapter "Message filtering")
- Select target LogChannel(s) (see chapter "Select target LogChannel")
- Check Message length (see chapter "Check message length")
- Apply the current LogChannel threshold (see chapter "Apply LogChannel LogLevel/Threshold")
- Copy Dlt message to LogChannel specific buffer (see chapter "Copying Dlt message to the LogChannel buffer")
- Apply the message attributes, if any are present and supported (see chapter "Apply the message attributes, if any are present and supported")

]

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`:

- Send Dlt message to `PduR` according to `TrafficShaping` settings (see chapter *"Sending messages from LogChannel Buffer"*)
- Create Dlt Header according to header settings (see chapter *"Create Dlt message header"*)
- Remove the Dlt message from the LogChannel specific buffer (see chapter *"Removing messages from LogChannel buffer"*)

]

7.1.8.1 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`, shall the Dlt module 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` and `DltGeneralGptChannelRef` is configured, the Dlt module shall call the API `Gpt_GetTimeElapsed` with the configured channel reference (see `DltGeneralGptChannelRef`) to fetch the elapsed time.]

[SWS_Dlt_00655] [If the parameter `DltHeaderUseTimestamp` is set to `TRUE` and `DltGeneralStbMTimeBaseRef` is configured, the Dlt module shall call the API `StbM_GetCurrentTime` with the configured time base reference (see `DltGeneralStbMTimeBaseRef`) to fetch the current synchronized time and calculate the elapsed time.]

7.1.8.2 Message filtering

Message filtering means to accept or discard an incoming log or trace message based on the `DltSwcApplicationId/DltSwcContextId` 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 `DLT_LOG_VERBOSE`.]

[SWS_Dlt_00657] [For Dlt Log Messages, the lowest LogLevel Threshold shall be defined as `DLT_LOG_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 `DltSwcApplicationId/DltSwcContextId` tuple mappings (see configuration parameter `DltLogLevelThreshold`).]

[SWS_Dlt_00659] [For trace message filtering the Dlt internally manages trace activation state to `DltSwcApplicationId/DltSwcContextId` 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 `DltSwcApplicationId/DltSwcContextId` tuple contained in a `Dlt_SendLogMessage` or `Dlt_SendTraceMessage` call. To do so, the following steps shall be performed:

- Check whether a mapping element exists, where `DltSwcApplicationId/DltSwcContextId` tuple of mapping element equals to the `DltSwcApplicationId/DltSwcContextId` tuple of the log/trace message. If such a mapping element exists, the matching mapping element is found.
- In case no match has been found in step 1, check whether a mapping element exists, where the `DltSwcApplicationId` equals the ApplicationID of the log/trace message and the `DltSwcContextId` of mapping element equals wildcard (value `0x00000000`). If such a mapping element exists, the matching mapping element is found.
- 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 log level value of the Log Message is numerically higher than this log level threshold, the Log 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.]

7.1.8.3 Select target LogChannel

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

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

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

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

- From all mapping elements, where `DltSwcApplicationId/DltSwcContextId` tuple of mapping element equals to the `DltSwcApplicationId/DltSwcContextId` tuple of the log/trace message, the LogChannel shall be added to the list of output LogChannels.
- 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.
- If the list of output LogChannels is still empty after step 1 and 2. The default Log Channel (see configuration parameter `DltDefaultLogChannelRef`) shall be added to the list of output LogChannels.

]

7.1.8.4 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.

7.1.8.5 Apply LogChannel LogLevelThreshold

In this step, the Dlt module decides, individually for each identified log and trace channel, whether the current log or trace message may pass or not.

[SWS_Dlt_00667] [Log messages with a log level numerically higher than the configured value of LogChannel threshold for the identified LogChannel shall be discarded and `E_OK` shall be returned. This shall be done on each LogChannel from the list of output LogChannels for the Log Message, considering [\[SWS_Dlt_00665\]](#).]

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

7.1.8.6 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 ("*BufferOverflowNotification*")

]

Note: The cyclically called `Dlt_TxFunction` checks the status of the buffer overflow flag and the de-bounce 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 message 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 message has been copied successfully to at least one LogChannel buffer, DLT_OK shall be returned.]

7.1.8.7 Apply the message attributes, if any are present and supported

Optional attributes can be added to a message when using the APIs `Dlt_SendLogMessageWithAttributes` or `Dlt_SendTraceMessageWithAttributes`, and if the configuration parameter `DltProtocolVersion` is set to 2 or higher.

The attributes are given as an additional function argument of type pointer to `Dlt_MessageAttributesType`.

The `Dlt_MessageAttributesType` structure has been designed to be extensible; any future extension of this structure will be provided as new fields, either with an in-band "invalid" state (e.g. a null pointer), or with a separate bool flag denoting the existence of a meaningful value for the subsequent field.

Therefore, prior to calling a function defined in this standard which reads values from a `Dlt_MessageAttributesType` structure (such as `Dlt_SendLogMessageWithAttributes`), the application shall ensure that all members of the structure, including any additional non-standardized members, are initialized with default initialization.

This can be done, for instance, with:

```
Dlt_MessageAttributesType attributes = { 0 };
```

The `messageTags` field of the `Dlt_MessageAttributesType` type constitutes a pointer to an array of strings. This array has to be "terminated" with a null-pointer.

An implementation might typically read this field with code such as:

```
const char** tags = attributes->messageTags;
int i;
for (i = 0; tags && (tags[i] != NULL); ++i) {
    const char* t = tags[i];
    ....
}
```

7.1.8.8 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:

- Shortening runtime of calls from the SW-Cs/BSWs which trigger log/trace messages, to reduce blocking time.

- In case traffic shaping functionality is enabled, the transmissions have to be processed by an asynchronous cyclic BSW entity anyway.
- 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 Log Channel 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 `DltTxPduId`.]

[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.]

7.1.8.9 Create Dlt message header

[SWS_Dlt_00676] [If the parameter `DltProtocolVersion` is set to 2 or higher, then the WTGS bit shall be set to `TRUE` if the value of the `messageTags` field of the `Dlt_MessageAttributesType` value that has been passed to the API `Dlt_SendLogMessageWithAttributes` or `Dlt_SendTraceMessageWithAttributes` is a non-NULL pointer. Otherwise, the WTGS bit shall be set to `FALSE`.]

[SWS_Dlt_00660] [If the parameter `DltProtocolVersion` is set to 2 or higher, and the WTGS bit has been set to `TRUE`, then the TAGS field shall be written as follows:

1. The NOTG field shall be set to the number of non-NULL pointers contained in the array pointed to by the `messageTags` field of the `Dlt_MessageAttributesType` value that has been passed to the API `Dlt_SendLogMessageWithAttributes` or `Dlt_SendTraceMessageWithAttributes`.

2. The strings pointed to by the `messageTags` array entries shall be written according to the rules defined by `PRS_Dlt_01031`.

Otherwise, the TAGS field shall be omitted.]

7.1.8.9.1 Assembling the Dlt Header

[SWS_Dlt_00678] [The UEH bit shall be set to 1 if at least one of the parameters `DltUseVerboseMode` or `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 `DltEcuIdValue`. If the configured ECU ID is 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 `DltHeaderUseTimestamp` 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.]

[SWS_Dlt_00784] [If the parameter `DltProtocolVersion` is set to 2 or higher, then the WPVL bit shall be set to the value of the `withPrivacyLevel` field of the `Dlt_MessageAttributesType` value that has been passed to the API `Dlt_SendLogMessageWithAttributes` or `Dlt_SendTraceMessageWithAttributes`. Otherwise, the WPVL bit shall be set to `FALSE`.]

[SWS_Dlt_00785] [If the parameter `DltProtocolVersion` is set to 2 or higher, and the WPVL bit has been set to `TRUE`, then the PRLV field shall be set to the value of the `privacyLevel` field of the `Dlt_MessageAttributesType` value that has been passed to the API `Dlt_SendLogMessageWithAttributes` or `Dlt_SendTraceMessageWithAttributes`. Otherwise, the PRLV field shall be omitted.]

7.1.8.9.2 Assembling the Dlt Extended Header

[SWS_Dlt_00691] [If the parameter `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 `DltUseVerboseMode` 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_MessageLogInfoType` 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`.]

7.1.8.10 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 `DltTxPduId` has been received and the transmit counter of the Dlt message is greater than the configured `DltLogChannelMaxNumOfRetries`.

]

7.1.9 Receiving of Dlt commands

The Dlt module can receive Dlt commands via the Rx Data Path and/or via dedicated API calls (see 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 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.]

7.1.9.1 SetLogLevel

[SWS_Dlt_00701] [If the Dlt command `Dlt_SetLogLevel` is requested, the new Log Level shall be stored for the received tuple of `DltSwcApplicationId/DltSwcContextId`.]

[SWS_Dlt_00702] [If the Dlt command `Dlt_SetLogLevel` is requested, but the received tuple of `DltSwcApplicationId/DltSwcContextId` is unknown, the Dlt command shall be answered with `DLT_E_ERROR`.]

7.1.9.2 SetTraceStatus

[SWS_Dlt_00703] [If the Dlt command `Dlt_SetTraceStatus` is requested, the new trace status shall be stored for the received tuple of `DltSwcApplicationId/DltSwcContextId`.]

[SWS_Dlt_00704] [If the Dlt command `Dlt_SetTraceStatus` is requested, but the addressed tuple of `DltSwcApplicationId/DltSwcContextId` is unknown, the Dlt command shall be answered with `DLT_E_ERROR`.]

7.1.9.3 GetLogInfo

[SWS_Dlt_00705] [If the Dlt command `Dlt_GetLogInfo` is requested, the requested `logInfo` shall be returned.]

[SWS_Dlt_00706] [If the Dlt command `Dlt_GetLogInfo` is requested, but the addressed tuple of `DltSwcApplicationId/DltSwcContextId` is unknown, the Dlt command shall be answered with `DLT_E_ERROR`.]

7.1.9.4 GetDefaultLogLevel

[SWS_Dlt_00708] [If the Dlt command `Dlt_GetDefaultLogLevel` is requested, the current value of the parameter `DltDefaultLogLevel` shall be returned.]

7.1.9.5 StoreConfiguration

[SWS_Dlt_00709] [If the Dlt command `Dlt_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 `Dlt_StoreConfiguration` shall return with `E_OK`.
 - If `NvM_WriteBlock` returned with something else than `E_OK`, the Dlt command `Dlt_StoreConfiguration` shall return with `DLT_E_ERROR`.

]

[SWS_Dlt_00710] [If the Dlt command `Dlt_StoreConfiguration` is requested and the configuration parameter `DltGeneralNvRAMSupport` is set to `FALSE`, the Dlt command `Dlt_StoreConfiguration` shall return `DLT_E_NOT_SUPPORTED`.]

7.1.9.6 ResetToFactoryDefault

[SWS_Dlt_00711] [If the Dlt command `Dlt_ResetToFactoryDefault` 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 `Dlt_ResetToFactoryDefault` 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 `Dlt_ResetToFactoryDefault` is requested and if the parameter `DltGeneralNvRAMSupport` is set to `FALSE`, `E_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 `Dlt_ResetToFactoryDefault` 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.

]

7.1.9.7 SetMessageFiltering

[SWS_Dlt_00775] [If the Dlt command `Dlt_SetMessageFiltering` is requested, all the Dlt filters shall be enabled/disabled as requested, and the Dlt command shall be answered with `E_OK`. Disabled filters will allow all messages to pass.]

7.1.9.8 SetDefaultLogLevel

[SWS_Dlt_00715] [If the Dlt command `Dlt_SetDefaultLogLevel` is requested, the parameter `DltDefaultLogLevel` shall be updated to the received `newLogLevel`.]

7.1.9.9 SetDefaultTraceStatus

[SWS_Dlt_00716] [If the Dlt command `Dlt_SetDefaultTraceStatus` is requested, the parameter `DltDefaultTraceStatus` shall be updated to the received `newTraceStatus`.]

7.1.9.10 GetDefaultTraceStatus

[SWS_Dlt_00717] [If the Dlt command `Dlt_GetDefaultTraceStatus` is requested, the current value of the parameter `DltDefaultTraceStatus` shall be returned.]

7.1.9.11 GetLogChannelNames

[SWS_Dlt_00718] [If the Dlt command `Dlt_GetLogChannelNames` is requested, the number of configured LogChannels and requested number of LogChannel names given by the parameter `DltLogChannelName` shall be returned.]

7.1.9.12 GetTraceStatus

[SWS_Dlt_00719] [If the Dlt Command `Dlt_GetTraceStatus` is requested, the `DltLogTraceStatusFlag` shall be returned for the received tuple of `DltSwcApplicationId/DltSwcContextId`.]

7.1.9.13 SetLogChannelAssignment

[SWS_Dlt_00720] [If the Dlt command `Dlt_SetLogChannelAssignment` is requested with parameter `addRemoveOp` set to `DLT_ASSIGN_ADD`, add the tuple of `DltSwcApplicationId/DltSwcContextId` to the LogChannel with the name provided by the parameter `logChannelName`. The Dlt command shall return `E_OK` even if the tuple was already assigned to the requested LogChannel before.]

[SWS_Dlt_00721] [If the Dlt command `Dlt_SetLogChannelAssignment` is requested with parameter `addRemoveOp` set to `DLT_ASSIGN_REMOVE`, remove the tuple of `DltSwcApplicationId/DltSwcContextId` from the LogChannel with the name provided by the parameter `logChannelName`. The Dlt command shall return `E_OK` even if the tuple was not assigned to the requested LogChannel before.]

Note: If a tuple of `DltSwcApplicationId/DltSwcContextId` 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 `Dlt_SetLogChannelAssignment` is requested with an unknown tuple of `DltSwcApplicationId/DltSwcContextId` or an unknown LogChannel name, the Dlt command shall return `DLT_E_ERROR`.]

7.1.9.14 SetLogChannelThreshold

[SWS_Dlt_00723] [If the Dlt command `Dlt_SetLogChannelThreshold` is requested, the `DltLogChannelThreshold` of the addressed LogChannel shall be set to the value received by the parameter `newThreshold`

]

[SWS_Dlt_00724] [If the Dlt command `Dlt_SetLogChannelThreshold` is requested and the `logChannelName` and/or the `newThreshold` is unknown, the Dlt command shall return `DLT_E_ERROR`.]

7.1.9.15 GetLogChannelThreshold

[SWS_Dlt_00725] [If the Dlt command `Dlt_GetLogChannelThreshold` is requested, the `DltLogChannelThreshold` of the addressed LogChannel shall be returned.]

[SWS_Dlt_00726] [If the Dlt command `Dlt_GetLogChannelThreshold` is requested and the `logChannelName` or the `logChannelThreshold` is unknown, the Dlt command shall return `DLT_E_ERROR`.]

7.1.10 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:

- `Dlt_GetLogInfo` (only in case one or more SW-Cs register/unregister themselves)
- `BufferOverflowNotification` ([PRS_Dlt_00769] in case of a buffer overflow).

7.1.10.1 BufferOverflowNotification

The buffer overflow notification ([PRS_Dlt_00769]) is used to inform the Dlt Logging tool about the loss of Dlt messages. The amount of `BufferOverflowNotifications` on the bus can be limited/de-bounced 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 `BufferOverflowNotification.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

Chapter [5, General Specification of Basic Software Modules] 7.2 “Error Handling” describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.2.1 Development Errors

[SWS_Dlt_00727] Definition of development errors in module Dlt [

Type of error	Related error code	Error value
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] Definition of runtime errors in module Dlt [

Type of error	Related error code	Error value
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 Production Errors

There are no production errors.

7.2.4 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:

[SWS_Dlt_91009] Definition of imported datatypes of module Dlt [

Module	Header File	Imported Type
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
Gpt	Gpt.h	Gpt_ChannelType
	Gpt.h	Gpt_ValueType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
StbM	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_TimeTupleType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_VirtualLocalTimeType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

]

8.2 Type definitions

8.2.1 Dlt_ConfigType

[SWS_Dlt_00437] Definition of datatype Dlt_ConfigType

Upstream requirements: [SRS_BSW_00414](#)

[

Name	Dlt_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	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.	





Available via	Dlt.h
----------------------	-------

]

8.2.2 Dlt_MessageType

[SWS_Dlt_00224] Definition of datatype Dlt_MessageType [

Name	Dlt_MessageType		
Kind	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.		
Available via	Dlt.h		

]

8.2.3 Dlt_MessageNetworkTraceInfoType

[SWS_Dlt_00233] Definition of datatype Dlt_MessageNetworkTraceInfoType [

Name	Dlt_MessageNetworkTraceInfoType		
Kind	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.		
Available via	Dlt.h		

]

8.3 Function definitions

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

8.3.1 Dlt_Init

[SWS_Dlt_00239] Definition of API function Dlt_Init

Upstream requirements: [SRS_BSW_00344](#), [SRS_BSW_00404](#), [SRS_BSW_00405](#), [SRS_BSW_00101](#), [SRS_BSW_00407](#), [SRS_BSW_00358](#), [SRS_BSW_00414](#)

Service Name	Dlt_Init	
Syntax	<pre>void Dlt_Init (const Dlt_ConfigType* config)</pre>	
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.	
Available via	Dlt.h	

[SWS_Dlt_00453]

Upstream requirements: [RS_LT_00039](#)

[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 [\[SWS_Dlt_00239\]](#) in the `Dlt_Init` function.]

8.3.2 Dlt_GetVersionInfo

[SWS_Dlt_00271] Definition of API function Dlt_GetVersionInfo

Upstream requirements: [SRS_BSW_00402](#)

Service Name	Dlt_GetVersionInfo	
Syntax	<pre>void Dlt_GetVersionInfo (Std_VersionInfoType* versioninfo)</pre>	
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.	
Available via	Dlt.h	

8.3.3 Dlt_SendTraceMessage

[SWS_Dlt_00243] Definition of API function Dlt_SendTraceMessage

Upstream requirements: [RS_LT_00003](#)

Service Name	Dlt_SendTraceMessage	
Syntax	<pre>Std_ReturnType Dlt_SendTraceMessage (Dlt_SessionIDType sessionId, const Dlt_MessageTraceInfoType* traceInfo, const uint8* traceData, uint16 traceDataLength)</pre>	
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.	
Available via	Dlt.h	

8.3.4 Dlt_SendLogMessage

[SWS_Dlt_00241] Definition of API function Dlt_SendLogMessage

Upstream requirements: [RS_LT_00003](#)

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.	
Available via	Dlt.h	

8.3.5 Dlt_RegisterContext

[SWS_Dlt_00245] Definition of API function Dlt_RegisterContext

Upstream requirements: [RS_LT_00033](#)

Service Name	Dlt_RegisterContext	
Syntax	<pre>Std_ReturnType Dlt_RegisterContext (Dlt_SessionIDType sessionId, Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, const uint8* appDescription, uint8 appDescLen, const uint8* contextDescription, uint8 contextDescLen)</pre>	
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.
	appDescLen	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.
	contextDescLen	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.	
Available via	Dlt.h	

8.3.6 Dlt_UnregisterContext

[SWS_Dlt_00769] Definition of API function Dlt_UnregisterContext

Upstream requirements: [RS_LT_00033](#)

Service Name	Dlt_UnregisterContext	
Syntax	<pre>Std_ReturnType Dlt_UnregisterContext (Dlt_SessionIDType sessionId, Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId)</pre>	
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.	
Available via	Dlt.h	

8.3.7 Dlt_SetLogLevel

[SWS_Dlt_00252] Definition of API function Dlt_SetLogLevel

Upstream requirements: [RS_LT_00004](#), [RS_LT_00038](#)

Service Name	Dlt_SetLogLevel	
Syntax	<pre>Std_ReturnType Dlt_SetLogLevel (Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, Dlt_MessageLogLevelType newLogLevel)</pre>	
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.	
Available via	Dlt.h	

8.3.8 Dlt_SetTraceStatus

[SWS_Dlt_00254] Definition of API function Dlt_SetTraceStatus

Upstream requirements: [RS_LT_00004](#), [RS_LT_00038](#)

Service Name	Dlt_SetTraceStatus	
Syntax	<pre>Std_ReturnType Dlt_SetTraceStatus (Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, boolean newTraceStatus)</pre>	
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.	
Available via	Dlt.h	

8.3.9 Dlt_GetLogInfo

[SWS_Dlt_00732] Definition of API function Dlt_GetLogInfo [

Service Name	Dlt_GetLogInfo	
Syntax	<pre>Std_ReturnType Dlt_GetLogInfo (uint8 options, Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, uint8* status, Dlt_LogInfoType* logInfo)</pre>	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	options	Used to filter the response in respect to the ApplicationId, Context Id 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.	
Available via	Dlt.h	

]

8.3.10 Dlt_GetDefaultLogLevel

[SWS_Dlt_00733] Definition of API function Dlt_GetDefaultLogLevel [

Service Name	Dlt_GetDefaultLogLevel	
Syntax	<pre>Std_ReturnType Dlt_GetDefaultLogLevel (Dlt_MessageLogLevelType* defaultLogLevel)</pre>	
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.	
Available via	Dlt.h	

]

[SWS_Dlt_00734] [A call to `Dlt_GetDefaultLogLevel` shall return with `E_OK` if the Dlt module provided 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 provide the current value of the parameter `DltDefaultLogLevel`.]

8.3.11 Dlt_StoreConfiguration

[SWS_Dlt_00736] Definition of API function `Dlt_StoreConfiguration` [

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 DLT_E_ERROR
Description	Copies the current Dlt configuration to NvRAM by calling <code>NvM_WriteBlock()</code> . No return value expected from <code>NvM_WriteBlock()</code>	
Available via	Dlt.h	

]

[SWS_Dlt_00737] [If the parameter `DltGeneralNvRAMSupport` is set to `FALSE`, a call to `Dlt_StoreConfiguration` shall return with `DLT_E_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 `E_OK` in case the call to `NvM_WriteBlock` returned with `E_OK`.]

8.3.12 Dlt_ResetToFactoryDefault

[SWS_Dlt_00739] Definition of API function Dlt_ResetToFactoryDefault [

Service Name	Dlt_ResetToFactoryDefault	
Syntax	<pre>Std_ReturnType Dlt_ResetToFactoryDefault (void)</pre>	
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. No return value expected from NvM	
Available via	Dlt.h	

]

8.3.13 Dlt_SetMessageFiltering

[SWS_Dlt_00770] Definition of API function Dlt_SetMessageFiltering [

Service Name	Dlt_SetMessageFiltering	
Syntax	<pre>Std_ReturnType Dlt_SetMessageFiltering (boolean status)</pre>	
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. If disabled, all the messages will pass the filter.	
Available via	Dlt.h	

]

8.3.14 Dlt_SetDefaultLogLevel

[SWS_Dlt_00740] Definition of API function Dlt_SetDefaultLogLevel [

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.	
Available via	Dlt.h	

]

[SWS_Dlt_00741] [If a call to [Dlt_SetDefaultLogLevel](#) successfully sets the requested [DltDefaultLogLevel](#), it shall return with E_OK]

[SWS_Dlt_00742] [If a call to [Dlt_SetDefaultLogLevel](#) could not set the requested [DltDefaultLogLevel](#), it shall return with E_NOT_OK]

8.3.15 Dlt_SetDefaultTraceStatus

[SWS_Dlt_00743] Definition of API function Dlt_SetDefaultTraceStatus [

Service Name	Dlt_SetDefaultTraceStatus	
Syntax	Std_ReturnType Dlt_SetDefaultTraceStatus (boolean newTraceStatus)	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	newTraceStatus	enabling/disabling of Trace messages
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.	
Available via	Dlt.h	

]

[SWS_Dlt_00744] [If a call to [Dlt_SetDefaultTraceStatus](#) successfully sets the requested new [DltDefaultTraceStatus](#), it shall return E_OK]

[SWS_Dlt_00745] [If a call to `Dlt_SetDefaultTraceStatus` could not set the requested `DltDefaultTraceStatus`, it shall return with `E_NOT_OK`]

8.3.16 Dlt_GetDefaultTraceStatus

[SWS_Dlt_00746] Definition of API function `Dlt_GetDefaultTraceStatus` [

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

]

[SWS_Dlt_00747] [If a call to `Dlt_GetDefaultTraceStatus` could provide the current `DltDefaultTraceStatus`, it shall return with `E_OK`]

[SWS_Dlt_00748] [If a call to `Dlt_GetDefaultTraceStatus` could not provide the `DltDefaultTraceStatus`, it shall return with `E_NOT_OK`]

8.3.17 Dlt_GetLogChannelNames

[SWS_Dlt_00749] Definition of API function `Dlt_GetLogChannelNames` [

Service Name	Dlt_GetLogChannelNames	
Syntax	<pre>Std_ReturnType Dlt_GetLogChannelNames (uint8* numberOfLogChannels, Dlt_LogChannelNameInfoType logChannelNames)</pre>	
Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	numberOfLogChannels	Contains the number of requested LogChannels names. On Return it holds the number of configured LogChannels
Parameters (out)	logChannelNames	Returns a list of configured LogChannel names. The size of the list is limited by <code>MaxNumberOfChannels</code> .





Return value	Std_ReturnType	E_OK: No error occurred E_NOT_OK: LogChannelNames could not be returned
Description	The caller provides the number of logChannelNames to be returned. The function returns the requested amount of LogChannelNames and updates numberOfLogChannels as the outgoing information on how many LogChannels are actually configured.	
Available via	Dlt.h	

]

8.3.18 Dlt_GetTraceStatus

[SWS_Dlt_00750] Definition of API function Dlt_GetTraceStatus [

Service Name	Dlt_GetTraceStatus	
Syntax	<pre>Std_ReturnType Dlt_GetTraceStatus (Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, boolean* traceStatus)</pre>	
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.	
Available via	Dlt.h	

]

8.3.19 Dlt_SetLogChannelAssignment

[SWS_Dlt_00751] Definition of API function Dlt_SetLogChannelAssignment [

Service Name	Dlt_SetLogChannelAssignment	
Syntax	<pre>Std_ReturnType Dlt_SetLogChannelAssignment (Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, Dlt_LogChannelNameType logChannelName, Dlt_AssignmentOperation addRemoveOp)</pre>	
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.	
Available via	Dlt.h	

8.3.20 Dlt_SetLogChannelThreshold

[SWS_Dlt_00752] Definition of API function Dlt_SetLogChannelThreshold [

Service Name	Dlt_SetLogChannelThreshold	
Syntax	<pre>Std_ReturnType Dlt_SetLogChannelThreshold (Dlt_LogChannelNameType logChannelName, Dlt_MessageLogLevelType newThreshold, boolean newTraceStatus)</pre>	
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.	
Available via	Dlt.h	

8.3.21 Dlt_GetLogChannelThreshold

[SWS_Dlt_00753] Definition of API function Dlt_GetLogChannelThreshold [

Service Name	Dlt_GetLogChannelThreshold	
Syntax	<pre>Std_ReturnType Dlt_GetLogChannelThreshold (Dlt_LogChannelNameType logChannelName, Dlt_MessageLogLevelType* logChannelThreshold, boolean* traceStatus)</pre>	
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.	
Available via	Dlt.h	

]

8.3.22 Dlt_SendLogMessageWithAttributes

[SWS_Dlt_91011] Definition of API function Dlt_SendLogMessageWithAttributes [

Service Name	Dlt_SendLogMessageWithAttributes	
Syntax	<pre>Std_ReturnType Dlt_SendLogMessageWithAttributes (Dlt_SessionIDType sessionId, const Dlt_MessageLogInfoType* logInfo, const uint8* logData, uint16 logDataLength, const Dlt_MessageAttributesType* attributes)</pre>	
Service ID [hex]	0x81	
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.
	attributes	Structure containing optional message attributes
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. DLT_E_NOT_SUPPORTED: Operation/Command unsupported
Description	The service represents the interface to be used by basic software modules or by software component to send Log Messages with attributes.	
Available via	Dlt.h	

]

8.3.23 Dlt_SendTraceMessageWithAttributes

[SWS_Dlt_91012] Definition of API function Dlt_SendTraceMessageWithAttributes [

Service Name	Dlt_SendTraceMessageWithAttributes	
Syntax	<pre>Std_ReturnType Dlt_SendTraceMessageWithAttributes (Dlt_SessionIDType sessionId, const Dlt_MessageTraceInfoType* traceInfo, const uint8* traceData, uint16 traceDataLength, const Dlt_MessageAttributesType* attributes)</pre>	
Service ID [hex]	0x82	
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
	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.
	attributes	Structure containing optional message attributes
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. DLT_E_NOT_SUPPORTED: Operation/Command unsupported
Description	The service represents the interface to be used by basic software modules or by software components to trace parameters, with attributes.	





Available via	Dlt.h
----------------------	-------

8.4 Callback notifications

This is a list of functions provided for other modules.

8.4.1 Dlt_RxIndication

[SWS_Dlt_00272] Definition of callback function Dlt_RxIndication [

Service Name	Dlt_RxIndication	
Syntax	<pre>void Dlt_RxIndication (PduIdType RxPduId, const PduInfoType* PduInfoPtr)</pre>	
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.	
Available via	Dlt.h	

8.4.2 Dlt_TriggerTransmit

[SWS_Dlt_00754] Definition of callback function Dlt_TriggerTransmit [

Service Name	Dlt_TriggerTransmit	
Syntax	<pre>Std_ReturnType Dlt_TriggerTransmit (PduIdType TxPduId, PduInfoType* PduInfoPtr)</pre>	
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 SduLength. 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.	
Available via	Dlt.h	

]

[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] Definition of callback function Dlt_TxConfirmation [

Service Name	Dlt_TxConfirmation	
Syntax	<pre>void Dlt_TxConfirmation (PduIdType TxPduId, Std_ReturnType result)</pre>	
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.	
Available via	Dlt.h	

]

8.4.4 Dlt_TpTxConfirmation

[SWS_Dlt_00756] Definition of callback function Dlt_TpTxConfirmation [

Service Name	Dlt_TpTxConfirmation	
Syntax	<pre>void Dlt_TpTxConfirmation (PduIdType id, Std_ReturnType result)</pre>	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	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	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	
Available via	Dlt.h	

]

8.4.5 Dlt_CopyTxData

[SWS_Dlt_00516] Definition of callback function Dlt_CopyTxData

Upstream requirements: [RS_LT_00034](#)

[

Service Name	Dlt_CopyTxData	
Syntax	<pre>BufReq_ReturnType Dlt_CopyTxData (PduIdType id, const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr)</pre>	
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. FrlsoTp) 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.	
Available via	Dlt.h	

8.4.6 Dlt_StartOfReception

[SWS_Dlt_91006] Definition of callback function Dlt_StartOfReception [

Service Name	Dlt_StartOfReception	
Syntax	<pre>BufReq_ReturnType Dlt_StartOfReception (PduIdType id, const PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr)</pre>	
Service ID [hex]	0x46	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.





	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. BUFREQ_E_OVFL: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSduLength equal to 0.	
Available via	Dlt.h	

8.4.7 Dlt_TpRxIndication

[SWS_Dlt_91007] Definition of callback function Dlt_TpRxIndication [

Service Name	Dlt_TpRxIndication	
Syntax	<pre>void Dlt_TpRxIndication (PduIdType id, Std_ReturnType result)</pre>	
Service ID [hex]	0x45	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	result	E_OK: The PDU was received. E_NOT_OK: Reception of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.	
Available via	Dlt.h	

8.4.8 Dlt_CopyRxData

[SWS_Dlt_91008] Definition of callback function Dlt_CopyRxData [

Service Name	Dlt_CopyRxData	
Syntax	<pre>BufReq_ReturnType Dlt_CopyRxData (PduIdType id, const PduInfoType* info, PduLengthType* bufferSizePtr)</pre>	
Service ID [hex]	0x44	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied.
Return value	BufReq_ReturnType	BUFREQ_OK: Data copied successfully BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Description	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.	
Available via	Dlt.h	

]

8.5 Scheduled functions

8.5.1 Dlt_TxFunction

[SWS_Dlt_91005] Definition of scheduled function Dlt_TxFunction [

Service Name	Dlt_TxFunction	
Syntax	<pre>void Dlt_TxFunction (void)</pre>	
Service ID [hex]	0x80	
Description	–	
Available via	SchM_Dlt.h	

]

[SWS_Dlt_00758] [If the configuration parameter [DltGeneralTrafficShaping-Support](#) is set to `TRUE`, the Dlt messages shall be transmitted with the maximum bandwidth per LogChannel as configured using the parameter [DltLogChannel-TrafficShapingBandwidth](#).]

[SWS_Dlt_00759] [If the configuration parameter [DltGeneralTrafficShaping-Support](#) 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] Definition of mandatory interfaces required by module Dlt [

API Function	Header File	Description
PduR_DltTransmit	PduR_Dlt.h	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] Definition of optional interfaces requested by module Dlt

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.
Gpt_EnableNotification	Gpt.h	Enables the interrupt notification for a channel (relevant in normal mode).
Gpt_StartTimer	Gpt.h	Starts a timer channel.
NvM_EraseNvBlock	NvM.h	Service to erase a NV block.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.
StbM_GetCurrentTime	StbM.h	Returns a time tuple (Local time, Global time and Timebase status) and user data details Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).

8.6.3 Configurable interfaces

This section defines all configurable external interfaces.

[SWS_Dlt_00259] Definition of configurable interface Dlt_InjectCall_<SESSION>

Service Name	Dlt_InjectCall_<SESSION>	
Syntax	<pre>void Dlt_InjectCall_<SESSION> (Dlt_ApplicationIDType appId, Dlt_ContextIDType contextId, uint32 serviceId, uint32 dataLength, const uint8* data)</pre>	
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 triggered by this function should depend on the service_id also the interpretation of the user data. Both are specific to the application.	
Available via	Dlt.h	

8.7 Service Interfaces

8.7.1 Client-Server-Interfaces

8.7.1.1 DltControlService

[SWS_Dlt_00772] Definition of ClientServerInterface DltControlService [

Name	DltControlService		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	7	DLT_E_NOT_SUPPORTED	Operation/Command unsupported
	9	DLT_E_ERROR	Operation/Command unsuccessful

Operation	GetDefaultLogLevel		
Comment	–		
Relates to	Dlt_GetDefaultLogLevel		
Variation	–		
Parameters	defaultLogLevel		
	Type	Dlt_MessageLogLevelType	
	Direction	OUT	
	Comment	Returns the stored LogLevel setting	
	Variation	–	
Possible Errors	E_OK DLT_E_ERROR		

Operation	GetDefaultTraceStatus		
Comment	–		
Relates to	Dlt_GetDefaultTraceStatus		
Variation	–		
Parameters	traceStatus		
	Type	boolean	
	Direction	OUT	
	Comment	current trace status (enabled/disabled)	
	Variation	–	
Possible Errors	E_OK DLT_E_ERROR		

Operation	GetLogChannelNames		
Comment	–		
Relates to	Dlt_GetLogChannelNames		
Variation	–		
Parameters	numberOfLogChannels		
	Type	uint8	
	Direction	INOUT	





	Comment	Contains the number of requested LogChannels names. On Return it holds the number of configured LogChannels
	Variation	–
	logChannelNames	
	Type	Dlt_LogChannelNameInfoType
	Direction	OUT
	Comment	Returns a list of configured LogChannel names. The size of the list is limited by MaxNumberOfChannels.
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	GetLogChannelThreshold	
Comment	–	
Relates to	Dlt_GetLogChannelThreshold	
Variation	–	
Parameters	logChannelName	
	Type	Dlt_LogChannelNameType
	Direction	IN
	Comment	Addressed LogChannel name
	Variation	–
	logChannelThreshold	
	Type	Dlt_MessageLogLevelType
	Direction	OUT
	Comment	Current LogChannelThreshold
	Variation	–
	traceStatusPtr	
	Type	boolean
	Direction	OUT
	Comment	Current TraceStatus. TRUE: TraceMessages enabled. FALSE: TraceMessages disabled.
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	GetLogInfo	
Comment	–	
Relates to	Dlt_GetLogInfo	
Variation	–	
Parameters	options	
	Type	uint8
	Direction	IN
	Comment	Used to filter the response in respect to the ApplicationId, ContextId and Trace Status information
	Variation	–
	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN





	Comment	Representation of the ApplicationId
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	Representation of the ContextId
	Variation	–
	status	
	Type	uint8
	Direction	OUT
	Comment	–
	Variation	–
	logInfo	
	Type	Dlt_LogInfoType
	Direction	OUT
	Comment	Details about the returned Application IDs
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	GetTraceStatus	
Comment	–	
Relates to	Dlt_GetTraceStatus	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	ApplicationId
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	ContextId
	Variation	–
	traceStatus	
	Type	boolean
	Direction	OUT
	Comment	current Trace Status of the tuple ApplicationId/ContextId
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	ResetToFactoryDefault	
Comment	–	
Relates to	Dlt_ResetToFactoryDefault	
Variation	–	
Possible Errors	E_OK DLT_E_ERROR	

Operation	SetDefaultLogLevel	
Comment	–	
Relates to	Dlt_SetDefaultLogLevel	
Variation	–	
Parameters	newDefaultLogLevel	
	Type	Dlt_MessageLogLevelType
	Direction	IN
	Comment	sets the new filter value
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	SetDefaultTraceStatus	
Comment	–	
Relates to	Dlt_SetDefaultTraceStatus	
Variation	–	
Parameters	newTraceStatus	
	Type	boolean
	Direction	IN
	Comment	enabling/disabling of Trace messages
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	SetLogChannelAssignment	
Comment	–	
Relates to	Dlt_SetLogChannelAssignment	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	ID of the addressed application / SW-C
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	ID of the addressed context
	Variation	–
	logChannelName	
	Type	Dlt_LogChannelNameType
	Direction	IN
	Comment	Name of the addressed LogChannel
	Variation	–
	addRemoveOp	
	Type	Dlt_AssignmentOperation
	Direction	IN





	Comment	Operation to add/remove the addressed tuple ApplicationId/ContextId to/from the addressed LogChannel
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	SetLogChannelThreshold	
Comment	–	
Relates to	Dlt_SetLogChannelThreshold	
Variation	–	
Parameters	logChannelName	
	Type	Dlt_LogChannelNameType
	Direction	IN
	Comment	Name of the addressed LogChannel
	Variation	–
	newLogLevelThreshold	
	Type	Dlt_MessageLogLevelType
	Direction	IN
	Comment	Threshold for LogMessages
	Variation	–
	newTraceStatus	
	Type	boolean
	Direction	IN
	Comment	TRUE: enable TraceMessages FALSE: disable TraceMessages
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	SetLogLevel	
Comment	–	
Relates to	Dlt_SetLogLevel	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	ID of the SW-C
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	ID of the context
	Variation	–
	newLogLevel	
	Type	Dlt_MessageLogLevelType
	Direction	IN
	Comment	new log level to set
	Variation	–





Possible Errors	E_OK DLT_E_ERROR
------------------------	---------------------

Operation	SetMessageFiltering	
Comment	–	
Relates to	Dlt_SetMessageFiltering	
Variation	–	
Parameters	status	
	Type	boolean
	Direction	IN
	Comment	TRUE: enable message filtering FALSE: disable message filtering
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	SetTraceStatus	
Comment	–	
Relates to	Dlt_SetTraceStatus	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	ID of the SW-C
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	ID of the context
	Variation	–
	newTraceStatus	
	Type	boolean
	Direction	IN
	Comment	New trace status
	Variation	–
Possible Errors	E_OK DLT_E_ERROR	

Operation	StoreConfiguration	
Comment	–	
Relates to	Dlt_StoreConfiguration	
Variation	–	
Possible Errors	E_OK DLT_E_NOT_SUPPORTED DLT_E_ERROR	

]

8.7.1.2 InjectionCallback

[SWS_Dlt_00498] Definition of ClientServerInterface InjectionCallback [

Name	InjectionCallback		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	InjectCall	
Comment	–	
Relates to	Dlt_InjectCall_<SESSION>	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	–
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	–
	Variation	–
	serviceId	
	Type	uint32
	Direction	IN
	Comment	–
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	–
	Variation	–
	data	
	Type	uint8*
	Direction	IN
	Comment	–
Variation	–	
Possible Errors	E_OK E_NOT_OK	

]

8.7.1.3 LogTraceSessionControl

[SWS_Dlt_00496] Definition of ClientServerInterface LogTraceSessionControl [

Name	LogTraceSessionControl		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	LogLevelChangedNotification		
Comment	–		
Relates to	–		
Variation	–		
Parameters	appld		
	Type	Dlt_ApplicationIDType	
	Direction	IN	
	Comment	–	
	Variation	–	
	contextId		
	Type	Dlt_ContextIDType	
	Direction	IN	
	Comment	–	
	Variation	–	
	logLevel		
	Type	Dlt_MessageLogLevelType	
	Direction	IN	
	Comment	–	
	Variation	–	
Possible Errors	E_OK		

Operation	TraceStatusChangedNotification	
Comment	–	
Relates to	–	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	–
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	–
	Variation	–
	newTraceStatus	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK	

]

8.7.1.4 DltSwcMessageService

[SWS_Dlt_00495] Definition of ClientServerInterface DltSwcMessageService [

Name	DltSwcMessageService		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	2	DLT_E_MSG_TOO_LARGE	The message is too big for the internal Dlt buffer.
	3	DLT_E_CONTEXT_ALREADY_REG	The software module context has already registered.
	4	DLT_E_UNKNOWN_SESSION_ID	The provided session id is unknown.
	5	DLT_E_NO_BUFFER	Buffer overflow.
	6	DLT_E_CONTEXT_NOT_YET_REG	The software module context has not registered before.
	7	DLT_E_NOT_SUPPORTED	Operation/Command unsupported
	9	DLT_E_ERROR	Operation/Command unsuccessful

Operation	RegisterContext	
Comment	–	
Relates to	Dlt_RegisterContext	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	–
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	–
	Variation	–
	appDescription	
	Type	uint8[]
	Direction	IN
	Comment	–
	Variation	–
	appDescLen	
	Type	uint8
	Direction	IN
	Comment	–
	Variation	–
	contextDescription	
	Type	uint8[]
	Direction	IN





	Comment	–
	Variation	–
	contextDescLen	
	Type	uint8
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK DLT_E_CONTEXT_ALREADY_REG DLT_E_UNKNOWN_SESSION_ID	

Operation	SendLogMessage	
Comment	–	
Relates to	Dlt_SendLogMessage	
Variation	–	
Parameters	logInfo	
	Type	Dlt_MessageLogInfoType
	Direction	IN
	Comment	–
	Variation	–
	logData	
	Type	uint8[]
	Direction	IN
	Comment	–
	Variation	–
	logDataLength	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK DLT_E_MSG_TOO_LARGE DLT_E_UNKNOWN_SESSION_ID DLT_E_NO_BUFFER	

Operation	SendLogMessageWithAttributes	
Comment	–	
Relates to	Dlt_SendLogMessageWithAttributes	
Variation	–	
Parameters	logInfo	
	Type	Dlt_MessageLogInfoType
	Direction	IN
	Comment	–
	Variation	–
	logData	
	Type	uint8[]
	Direction	IN
	Comment	–
	Variation	–





	Variation	–
	logDataLength	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
	attributes	
	Type	Dlt_MessageAttributesType
	Direction	IN
	Comment	–
Possible Errors	Variation	–
	E_OK DLT_E_MSG_TOO_LARGE DLT_E_UNKNOWN_SESSION_ID DLT_E_NO_BUFFER DLT_E_NOT_SUPPORTED	

Operation	SendTraceMessage	
Comment	–	
Relates to	Dlt_SendTraceMessage	
Variation	–	
Parameters	traceInfo	
	Type	Dlt_MessageTraceInfoType
	Direction	IN
	Comment	–
	Variation	–
	traceData	
	Type	uint8[]
	Direction	IN
	Comment	–
	Variation	–
	traceDataLength	
	Type	uint16
	Direction	IN
	Comment	–
Possible Errors	Variation	–
	E_OK DLT_E_MSG_TOO_LARGE DLT_E_UNKNOWN_SESSION_ID DLT_E_NO_BUFFER	

Operation	SendTraceMessageWithAttributes	
Comment	–	
Relates to	Dlt_SendTraceMessageWithAttributes	
Variation	–	
Parameters	traceInfo	
	Type	Dlt_MessageTraceInfoType
	Direction	IN
	Comment	–





	Variation	–
	traceData	
	Type	uint8[]
	Direction	IN
	Comment	–
	Variation	–
	traceDataLength	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
	attributes	
	Type	Dlt_MessageAttributesType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK DLT_E_MSG_TOO_LARGE DLT_E_UNKNOWN_SESSION_ID DLT_E_NO_BUFFER DLT_E_NOT_SUPPORTED	

Operation	UnregisterContext	
Comment	–	
Relates to	Dlt_UnregisterContext	
Variation	–	
Parameters	appld	
	Type	Dlt_ApplicationIDType
	Direction	IN
	Comment	–
	Variation	–
	contextId	
	Type	Dlt_ContextIDType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK DLT_E_UNKNOWN_SESSION_ID DLT_E_CONTEXT_NOT_YET_REG	

8.7.2 Implementation Data Types

8.7.2.1 Dlt_ApplicationIDType

[SWS_Dlt_00226] Definition of ImplementationDataType Dlt_ApplicationIDType [

Name	Dlt_ApplicationIDType		
Kind	Type		
Derived from	uint32		
Range	0x00000000-0xFFFFFFFF	–	–
Description	This type describes the ApplicationId. 0x00000000 means the so-called wildcard.		
Variation	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.2 Dlt_ContextIDType

[SWS_Dlt_00227] Definition of ImplementationDataType Dlt_ContextIDType [

Name	Dlt_ContextIDType		
Kind	Type		
Derived from	uint32		
Range	0x00000000-0xFFFFFFFF	–	–
Description	This type describes the ContextId. 0x00000000 means the so-called wildcard.		
Variation	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.3 Dlt_SessionIDType

[SWS_Dlt_00225] Definition of ImplementationDataType Dlt_SessionIDType [

Name	Dlt_SessionIDType		
Kind	Type		
Derived from	uint32		
Description	This type identifies the session.		
Variation	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.4 Dlt_LogInfoType

[SWS_Dlt_91002] Definition of ImplementationDataType Dlt_LogInfoType [

Name	Dlt_LogInfoType	
Kind	Structure	
Elements	appldCount	
	Type	uint16
	Comment	Number of Applds
	appldInfo	
	Type	Array of Dlt_ApplicationIdInfoType
	Size	
	Comment	Details of Application
Description	–	
Variation	–	
Available via	Rte_Dlt_Type.h	

]

8.7.2.5 Dlt_ContextIdInfoType

[SWS_Dlt_91003] Definition of ImplementationDataType Dlt_ContextIdInfoType [

Name	Dlt_ContextIdInfoType	
Kind	Structure	
Elements	contextId	
	Type	Dlt_ContextIDType
	Comment	the ContextId
	logLevel	
	Type	Dlt_MessageLogLevelType
	Comment	the log message filter level
	traceStatus	
	Type	uint8
	Comment	0: off 1: on
	contextDescLen	
	Type	uint8
	Comment	Length of Context Description
	contextDesc	
	Type	Array of uint8
	Size	
	Comment	Context Description
Description	Context Information	
Variation	–	
Available via	Rte_Dlt_Type.h	

]

8.7.2.6 Dlt_ApplicationIdInfoType

[SWS_Dlt_91004] Definition of ImplementationDataType Dlt_ApplicationIdInfoType

Name	Dlt_ApplicationIdInfoType		
Kind	Structure		
Elements	appld		
	Type	Dlt_ApplicationIDType	
	Comment	Application ID	
	contextIdCount		
	Type	uint16	
	Comment	Length of contextInfoList	
	contextInfoList		
	Type	Array of Dlt_ContextIdInfoType	
	Size		
	Comment	List of Context information	
	appDescLen		
	Type	uint8	
	Comment	Length of Application Description	
	appDesc		
	Type	Array of uint8	
	Size		
	Comment	Application Description	
Description	Information about Applications		
Variation	—		
Available via	Rte_Dlt_Type.h		

8.7.2.7 Dlt_MessageOptionsType

[SWS_Dlt_00229] Definition of ImplementationDataType Dlt_MessageOptionsType

Name	Dlt_MessageOptionsType		
Kind	Type		
Derived from	uint8		
Range	verbose_mode	–	Bit 0: If set Verbose mode is used
	message_type	–	Bit 1-3 Dlt_MessageTypeType: determines type of msg (log,trace,...)
Description	Bitfield		
Variation	–		
Available via	Rte_Dlt_Type.h		

8.7.2.8 Dlt_MessageLogInfoType

[SWS_Dlt_00236] Definition of ImplementationDataType Dlt_MessageLogInfoType

Name	Dlt_MessageLogInfoType	
Kind	Structure	
Elements	argCount	
	Type	Dlt_MessageArgumentCount
	Comment	—
	logLevel	
	Type	Dlt_MessageLogLevelType
	Comment	—
	options	
	Type	Dlt_MessageOptionsType
	Comment	—
	contextId	
	Type	Dlt_ContextIDType
	Comment	—
	appld	
	Type	Dlt_ApplicationIDType
	Comment	—
Description	—	
Variation	—	
Available via	Rte_Dlt_Type.h	

8.7.2.9 Dlt_MessageLogLevelType

[SWS_Dlt_00230] Definition of ImplementationDataType Dlt_MessageLogLevelType

Name	Dlt_MessageLogLevelType		
Kind	Type		
Derived from	uint8		
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	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.10 Dlt_MessageTraceType

[SWS_Dlt_00231] Definition of ImplementationDataType Dlt_MessageTraceType

[

Name	Dlt_MessageTraceType		
Kind	Type		
Derived from	uint8		
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	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.11 Dlt_MessageArgumentCount

[SWS_Dlt_00235] Definition of ImplementationDataType Dlt_MessageArgumentCount

[

Name	Dlt_MessageArgumentCount		
Kind	Type		
Derived from	uint16		
Description	The implementation shall mask out the upper 8 bits of the value, and use only the lower 8 bits.		
Variation	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.12 Dlt_MessageTraceInfoType

[SWS_Dlt_00237] Definition of ImplementationDataType Dlt_MessageTraceInfoType

Name	Dlt_MessageTraceInfoType	
Kind	Structure	
Elements	traceInfo	
	Type	Dlt_MessageTraceType
	Comment	—
	options	
	Type	Dlt_MessageOptionsType
	Comment	—
	contextId	
	Type	Dlt_ContextIDType
	Comment	—
	appld	
	Type	Dlt_ApplicationIDType
	Comment	—
Description	—	
Variation	—	
Available via	Rte_Dlt_Type.h	

8.7.2.13 Dlt_LogChannelNameInfoType

[SWS_Dlt_91013] Definition of ImplementationDataType Dlt_LogChannelNameInfoType

Name	Dlt_LogChannelNameInfoType		
Kind	Array	Element type	Dlt_LogChannelNameType
Size	MaxNumberOfChannels Elements		
Description	This type describes a list of LogChannel names.		
Variation	–		
Available via	Rte_Dlt_Type.h		

[SWS_Dlt_00232] Definition of ImplementationDataType Dlt_LogChannelNameType

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





Available via	Rte_Dlt_Type.h
----------------------	----------------

]

8.7.2.14 Dlt_AssignmentOperation

[SWS_Dlt_00730] Definition of ImplementationDataType Dlt_AssignmentOperation [

Name	Dlt_AssignmentOperation		
Kind	Type		
Derived from	uint8		
Range	DLT_ASSIGN_REMOVE	0x00	Removing a LogChannel assignment
	DLT_ASSIGN_ADD	0x01	Adding a LogChannel assignment
Description	Adding or removing a LogChannel assignment for the given tuple of ApplicationId/ContextId.		
Variation	–		
Available via	Rte_Dlt_Type.h		

]

8.7.2.15 Dlt_MessageAttributesType

[SWS_Dlt_91010] Definition of ImplementationDataType Dlt_MessageAttributesType [

Name	Dlt_MessageAttributesType	
Kind	Structure	
Elements	withPrivacyLevel	
	Type	boolean
	Comment	—
	privacyLevel	
	Type	uint8
	Comment	—
	messageTags	
	Type	const char*
	Comment	—
Description	—	
Variation	—	
Available via	Rte_Dlt_Type.h	

]

8.7.3 Ports

8.7.3.1 Dlt_ControlService

[SWS_Dlt_00499] Definition of Port ControlService provided by module Dlt [

Name	ControlService		
Kind	ProvidedPort	Interface	DltControlService
Description	Through this port SW-Cs can control log settings and other configuration items of DLT.		
Variation	–		

]

8.7.3.2 Dlt_InjectCallback_{SW-C}

[SWS_Dlt_00778] Definition of Port InjectCallback_{SW-C} required by module Dlt [

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)}		

]

8.7.3.3 Dlt_SessionControlCallback_{SW-C}

[SWS_Dlt_00779] Definition of Port SessionControlCallback_{SW-C} required by module Dlt [

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)}		

]

8.7.3.4 Dlt_SwcMessageService_{SW-C}

[SWS_Dlt_91001] Definition of Port SwcMessageService_{SW-C} provided by module Dlt [

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

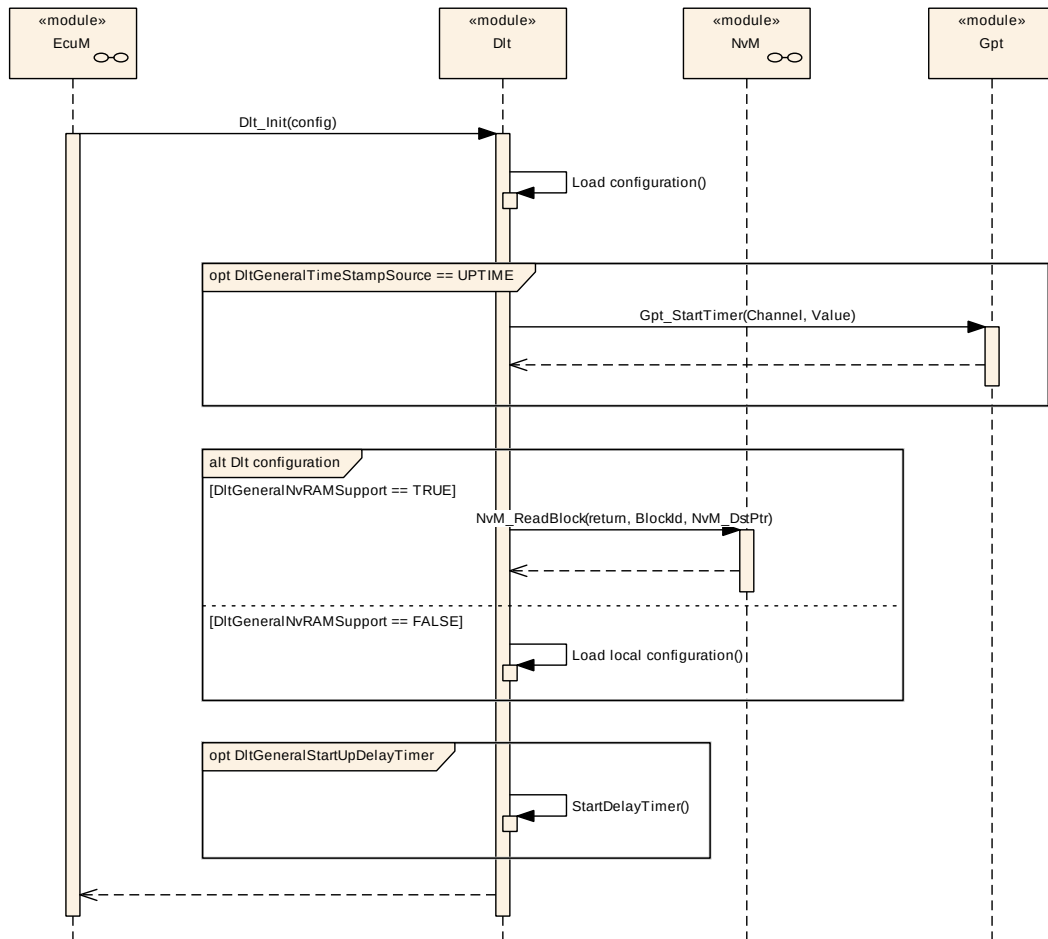


Figure 9.1: Dlt initialization

9.2 Overview of Dlt message transmission on one LogChannel

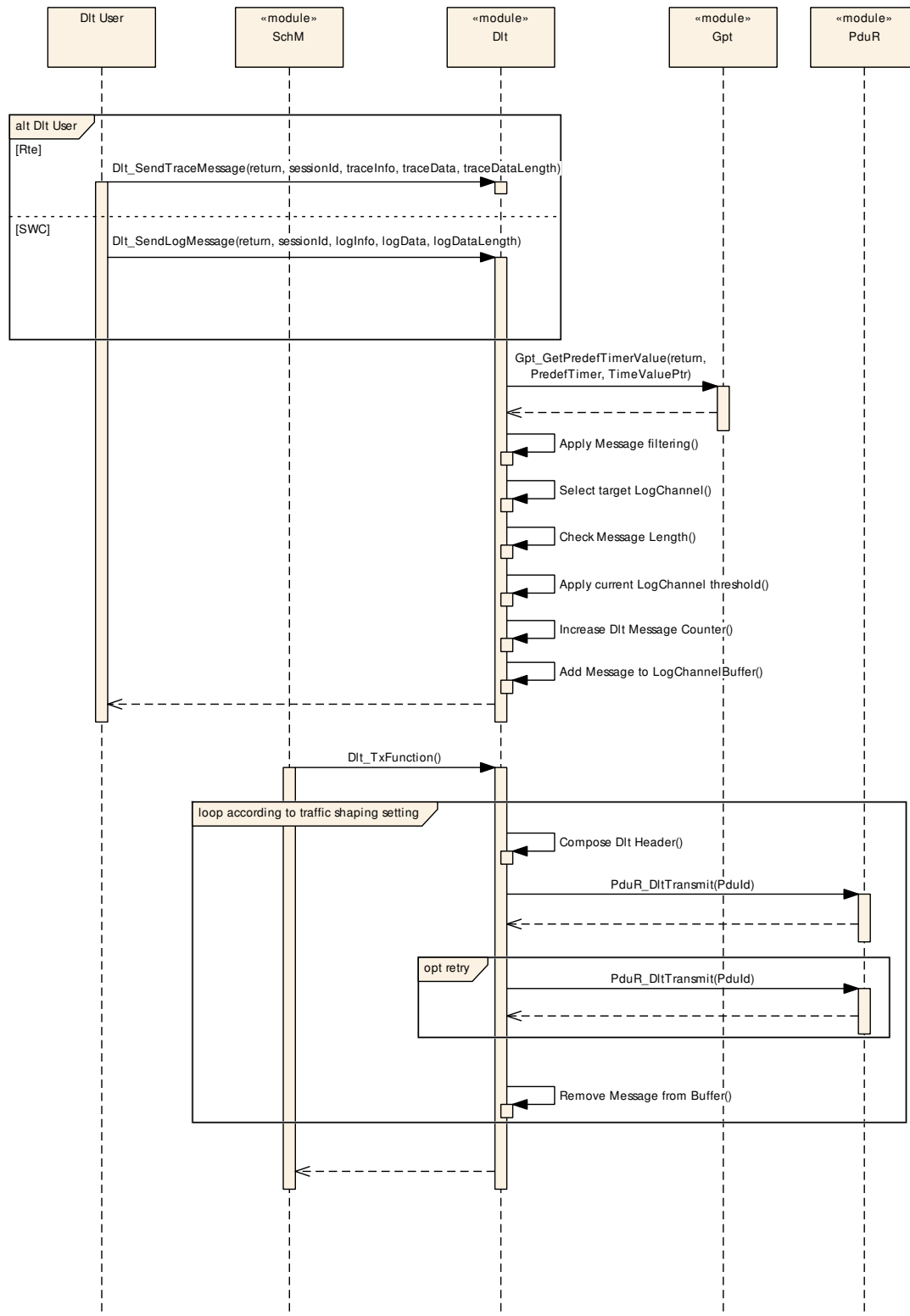


Figure 9.2: Overview of Dlt message transmission on one LogChannel

9.3 SetLogLevelFilter

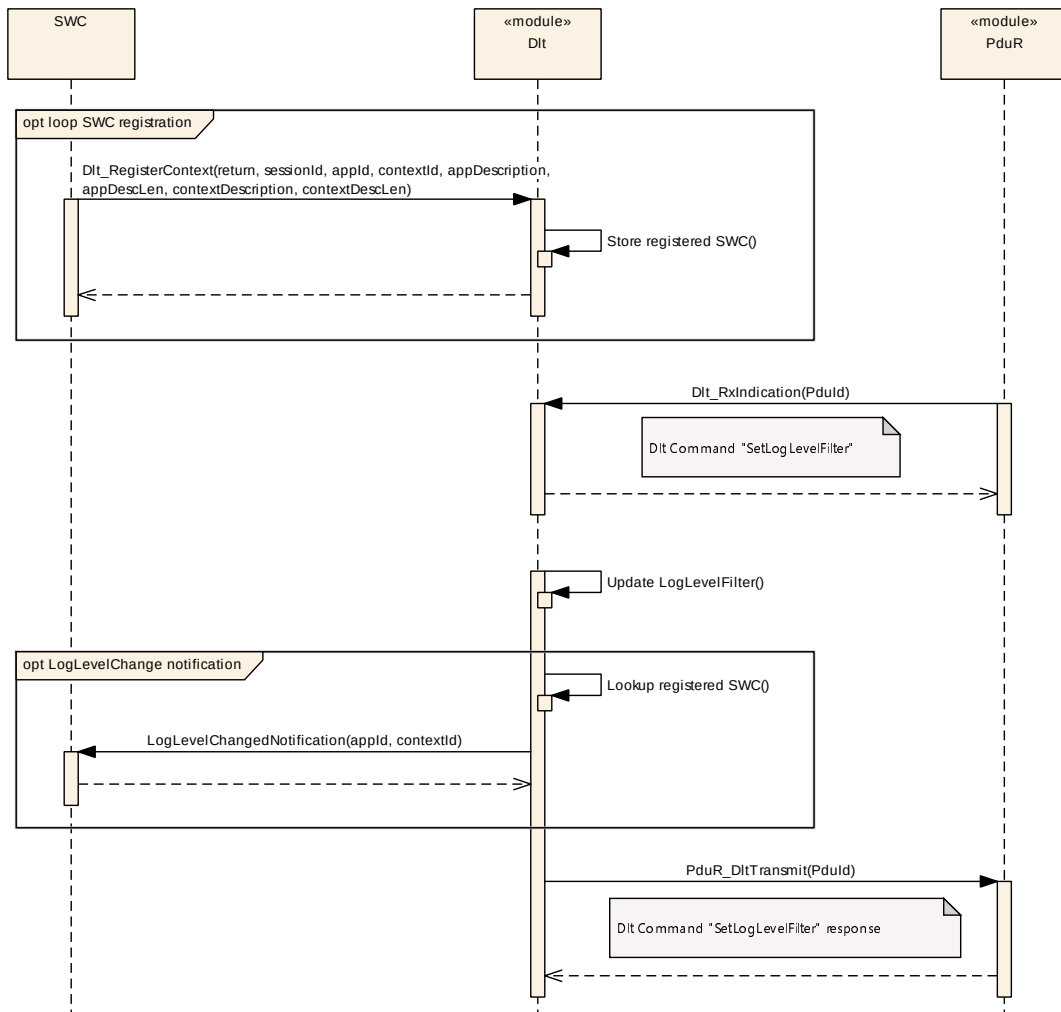


Figure 9.3: Set Log Level Filter

9.4 Buffer overflow indication

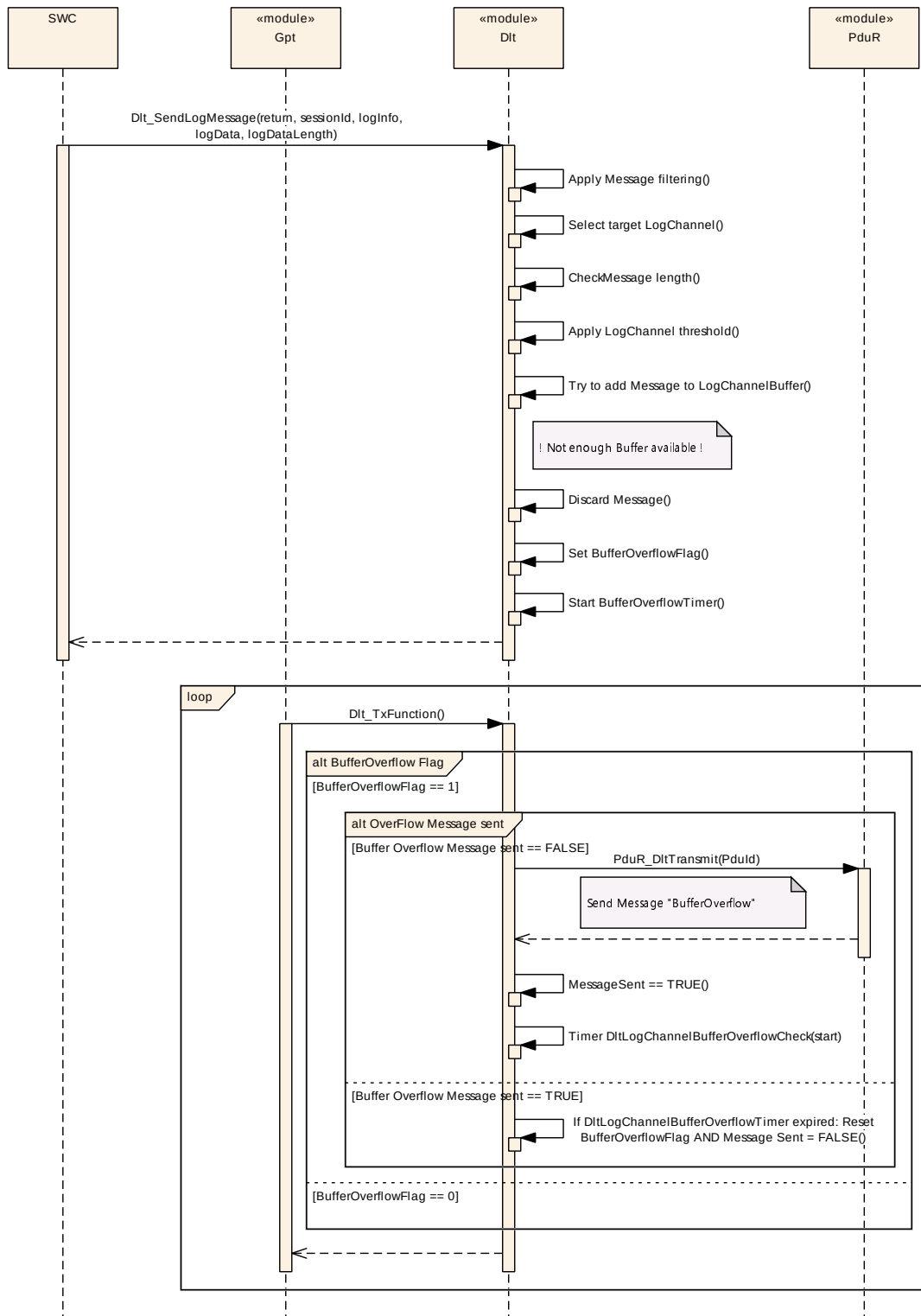


Figure 9.4: Buffer overflow indication

10 Configuration specification

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

Chapter 10.2 specifies 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 Chapter 7 and Chapter 8.

10.1.1 Dlt

[ECUC_Dlt_00800] Definition of EcucModuleDef Dlt [

Module Name	Dlt
Description	Configuration of the Dlt (Log&Trace) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	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.

]

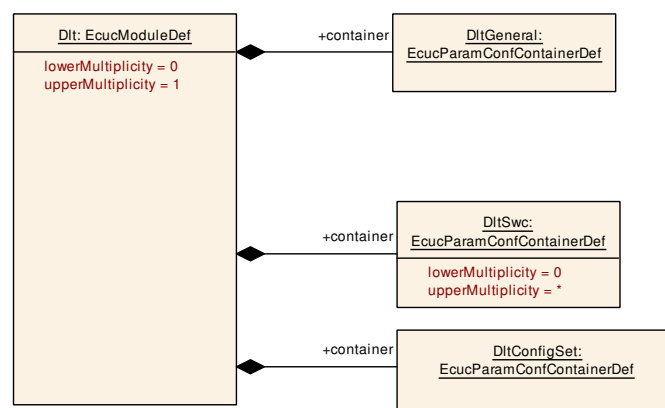


Figure 10.1: Dlt

10.1.2 DltGeneral

[ECUC_Dlt_00809] Definition of EcucParamConfContainerDef DltGeneral [

Container Name	DltGeneral
Parent Container	Dlt
Description	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltGeneralDevErrorDetect	1	[ECUC_Dlt_00840]
DltGeneralInjectionSupport	1	[ECUC_Dlt_00847]
DltGeneralNvRAMSupport	1	[ECUC_Dlt_00915]
DltGeneralRegisterContextNotification	1	[ECUC_Dlt_00846]
DltGeneralRxDataPathSupport	1	[ECUC_Dlt_00848]
DltGeneralStartUpDelayTimer	0..1	[ECUC_Dlt_00897]
DltGeneralTimeStampSupport	1	[ECUC_Dlt_00850]
DltGeneralTrafficShapingSupport	1	[ECUC_Dlt_00849]
DltGeneralVersionInfoApi	0..1	[ECUC_Dlt_00844]
DltMaxNumberOfChannels	1	[ECUC_Dlt_00918]
DltProtocolVersion	0..1	[ECUC_Dlt_00917]
DltGeneralGptChannelRef	0..1	[ECUC_Dlt_00905]
DltGeneralNvRamRef	0..1	[ECUC_Dlt_00845]
DltGeneralStbmTimeBaseRef	0..1	[ECUC_Dlt_00914]

No Included Containers

]

[ECUC_Dlt_00840] Definition of EcucBooleanParamDef DltGeneralDevErrorDetect [

Parameter Name	DltGeneralDevErrorDetect		
Parent Container	DltGeneral		
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	–	
Dependency			

]

[ECUC_Dlt_00847] Definition of EcucBooleanParamDef DltGeneralInjectionSupport

Parameter Name	DltGeneralInjectionSupport		
Parent Container	DltGeneral		
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	–	
Dependency			

[ECUC_Dlt_00915] Definition of EcucBooleanParamDef DltGeneralNvRAMSupport

Parameter Name	DltGeneralNvRAMSupport		
Parent Container	DltGeneral		
Description	Enables or disables the Dlt NvRAM Support 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	–	
Dependency			

[ECUC_Dlt_00846] Definition of EcucBooleanParamDef DltGeneralRegisterContextNotification

Parameter Name	DltGeneralRegisterContextNotification		
Parent Container	DltGeneral		
Description	If this parameter is set to TRUE, a Dlt Control Message is sent every time a SWC registers 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	–	
Dependency			

[ECUC_Dlt_00848] Definition of EcucBooleanParamDef DltGeneralRxDataPathSupport

Parameter Name	DltGeneralRxDataPathSupport		
Parent Container	DltGeneral		
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	–	
Dependency	At least one RxPdu needs to be configured if DltGeneralRxDataPathSupport = TRUE		

[ECUC_Dlt_00897] Definition of EcucFloatParamDef DltGeneralStartUpDelayTimer

Parameter Name	DltGeneralStartUpDelayTimer		
Parent Container	DltGeneral		
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	–	
Dependency			

[ECUC_Dlt_00850] Definition of EcucBooleanParamDef DltGeneralTimeStampSupport

Parameter Name	DltGeneralTimeStampSupport		
Parent Container	DltGeneral		
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	–	
Dependency			

]

[ECUC_Dlt_00849] Definition of EcucBooleanParamDef DltGeneralTrafficShapingSupport [

Parameter Name	DltGeneralTrafficShapingSupport		
Parent Container	DltGeneral		
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	–	
Dependency			

]

[ECUC_Dlt_00844] Definition of EcucBooleanParamDef DltGeneralVersionInfo Api [

Parameter Name	DltGeneralVersionInfoApi		
Parent Container	DltGeneral		
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	–	
Dependency			

]

[ECUC_Dlt_00918] Definition of EcucIntegerParamDef DltMaxNumberOfChannels [

Parameter Name	DltMaxNumberOfChannels		
Parent Container	DltGeneral		
Description	Maximum number of log channels. This value is used to determine the size of arrays of log channel names in the DLT API.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Dlt_00917] Definition of EcucIntegerParamDef DltProtocolVersion [

Parameter Name	DltProtocolVersion		
Parent Container	DltGeneral		
Description	Selects the DLT protocol version to be used by Dlt module. Currently the versions 1 and 2 are supported.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	1		
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	–	
Dependency			

]

[ECUC_Dlt_00905] Definition of EcucReferenceDef DltGeneralGptChannelRef [

Parameter Name	DltGeneralGptChannelRef
Parent Container	DltGeneral
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 Gpt ChannelConfiguration 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	–	
Dependency	DltGeneralTimeStampSupport is set to TRUE and DltGeneralStbMTimeBaseRef is not configured.		

┌

[ECUC_Dlt_00845] Definition of EcucReferenceDef DltGeneralNvRamRef ┌

Parameter Name	DltGeneralNvRamRef		
Parent Container	DltGeneral		
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	–	
Dependency			

┌

[ECUC_Dlt_00914] Definition of EcucReferenceDef DltGeneralStbMTimeBaseRef ┌

Parameter Name	DltGeneralStbMTimeBaseRef		
Parent Container	DltGeneral		
Description	If TimeStampSupport is used the Dlt module shall fetch the time from the StbM module by calling StbM_GetCurrentTime with the here referenced StbMSynchronizedTime Base.		
Multiplicity	0..1		
Type	Symbolic name reference to StbMSynchronizedTimeBase		
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	–	
Dependency	DltGeneralTimeStampSupport is set to TRUE and DltGeneralGptChannelRef is not configured		

└

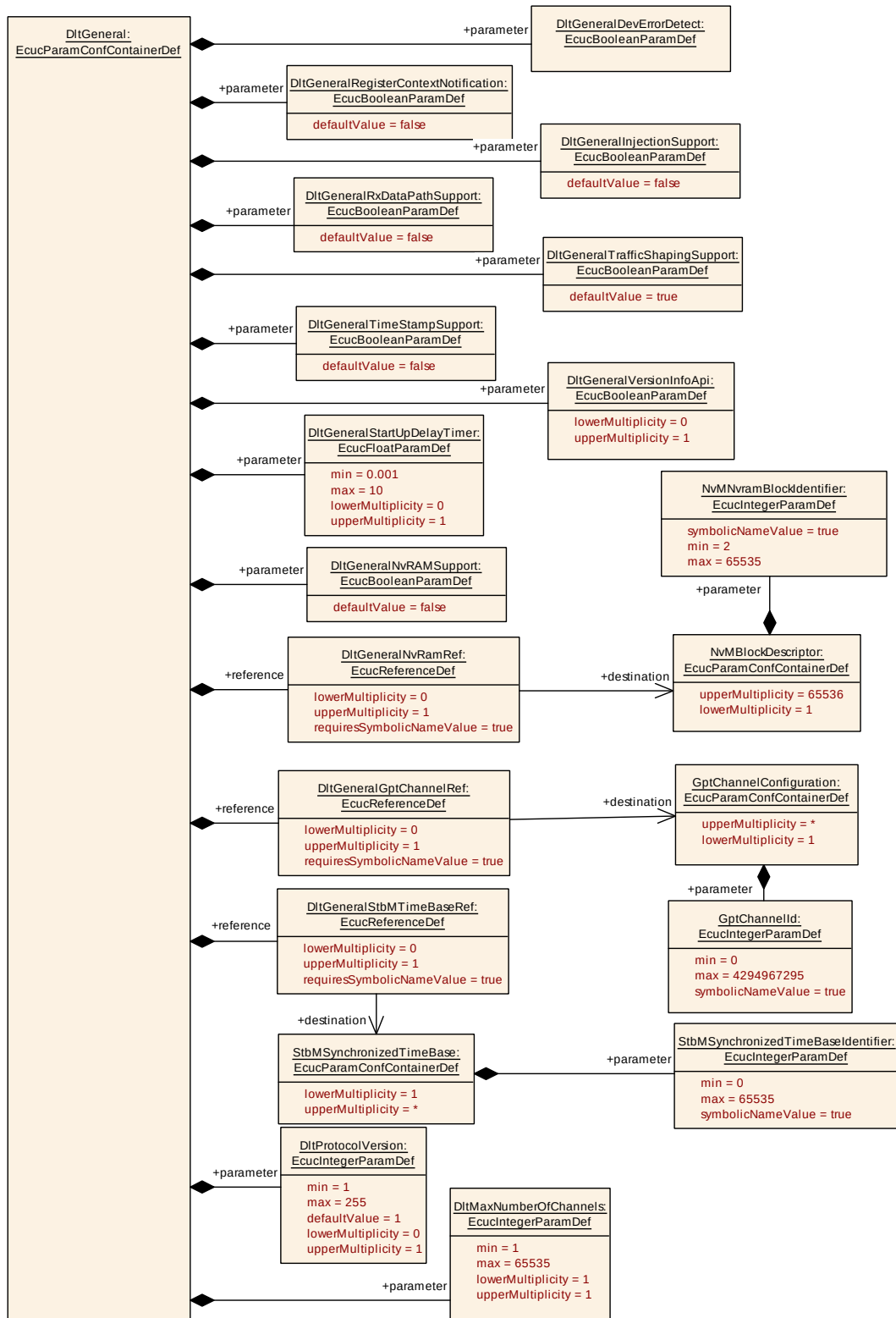


Figure 10.2: DltGeneral

10.1.3 DltSwc

[ECUC_Dlt_00856] Definition of EcucParamConfContainerDef DltSwc [

Container Name	DltSwc		
Parent Container	Dlt		
Description	Contains necessary configuration parameters of the AUTOSAR Dlt module to interact with SWCs.		
Multiplicity	0..*		
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			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltSwcSessionId	1	[ECUC_Dlt_00852]
DltSwcSupportLogLevelAndTraceStatusChange Notification	1	[ECUC_Dlt_00853]
MaxSwcLogMessageLength	1	[ECUC_Dlt_00909]
MaxSwcTraceMessageLength	1	[ECUC_Dlt_00910]

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

[ECUC_Dlt_00852] Definition of EcucIntegerParamDef DltSwcSessionId [

Parameter Name	DltSwcSessionId		
Parent Container	DltSwc		
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	–	
Dependency			

[ECUC_Dlt_00853] Definition of EcucBooleanParamDef DltSwcSupportLogLevelAndTraceStatusChangeNotification

Parameter Name	DltSwcSupportLogLevelAndTraceStatusChangeNotification		
Parent Container	DltSwc		
Description	Flag indicating, whether Dlt has to provide a R-Port for the notification of the SWC about LogLevel or TraceStatus 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	–	
Dependency			

[ECUC_Dlt_00909] Definition of EcucIntegerParamDef MaxSwcLogMessageLength

Parameter Name	MaxSwcLogMessageLength		
Parent Container	DltSwc		
Description	Defines the maximum allowed length (uint16) for LogMessages. The upper limit for the range of this parameter is currently defined by the range of the data type. The actual upper limit for the range of this parameter is identical to the maximum length of all configured Dlt log or trace messages, which is known when all log or trace messages are configured.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 65535		
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	–	
Dependency			

[ECUC_Dlt_00910] Definition of EcucIntegerParamDef MaxSwcTraceMessageLength

Parameter Name	MaxSwcTraceMessageLength		
Parent Container	DltSwc		
Description	Defines the maximum allowed length (uint16) for TraceMessages. The upper limit for the range of this parameter is currently defined by the range of the data type. The actual upper limit for the range of this parameter is identical to the maximum length of all configured Dlt log or trace messages, which is known when all log or trace messages are configured.		
Multiplicity	1		





Type	EcucIntegerParamDef		
Range	8 .. 65535		
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	–	
Dependency			

└

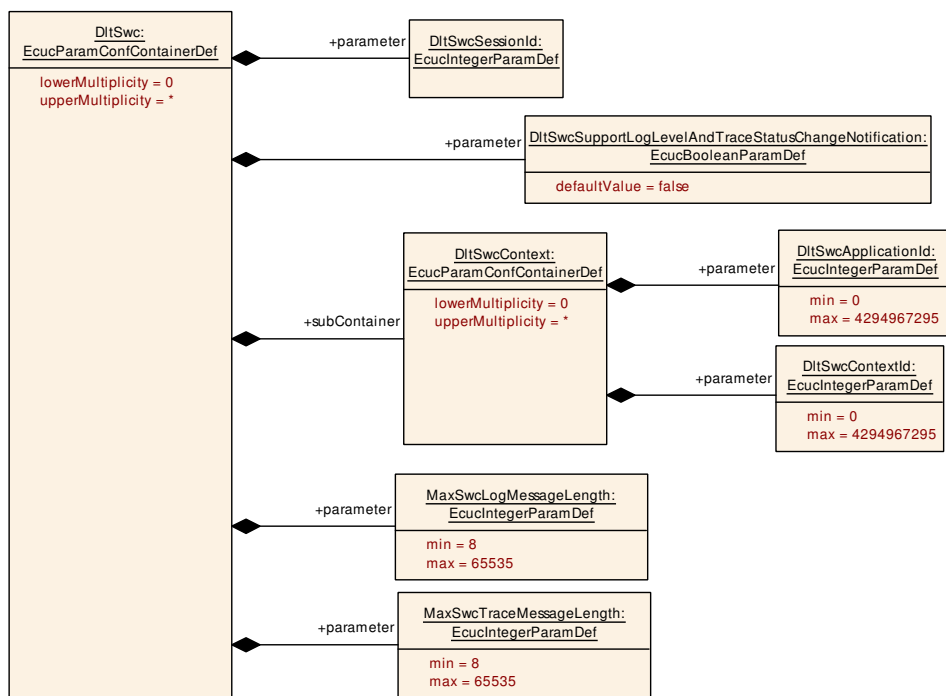


Figure 10.3: DltSwc

10.1.4 DltSwcContext

[ECUC_Dlt_00854] Definition of EcucParamConfContainerDef DltSwcContext [

Container Name	DltSwcContext		
Parent Container	DltSwc		
Description	This container contains the configuration of ApplicationId / ContextId pairs which are supported by this SWC.		
Multiplicity	0..*		
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			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltSwcApplicationId	1	[ECUC_Dlt_00858]
DltSwcContextId	1	[ECUC_Dlt_00859]

No Included Containers

]

[ECUC_Dlt_00858] Definition of EcucIntegerParamDef DltSwcApplicationId [

Parameter Name	DltSwcApplicationId		
Parent Container	DltSwcContext		
Description	Abbreviation for the SWC (4 characters)		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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
Dependency			

]

[ECUC_Dlt_00859] Definition of EcucIntegerParamDef DltSwcContextId [

Parameter Name	DltSwcContextId		
Parent Container	DltSwcContext		
Description	Abbreviation for the ContextId (4 characters)		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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
Dependency			

]

10.1.5 DltConfigSet

[ECUC_Dlt_00842] Definition of EcucParamConfContainerDef DltConfigSet [

Container Name	DltConfigSet
Parent Container	Dlt
Description	This container lists all the global Dlt functionalities that can be enabled or disabled at pre-compile time to optimize resource consumption.
Multiplicity	1
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	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

]

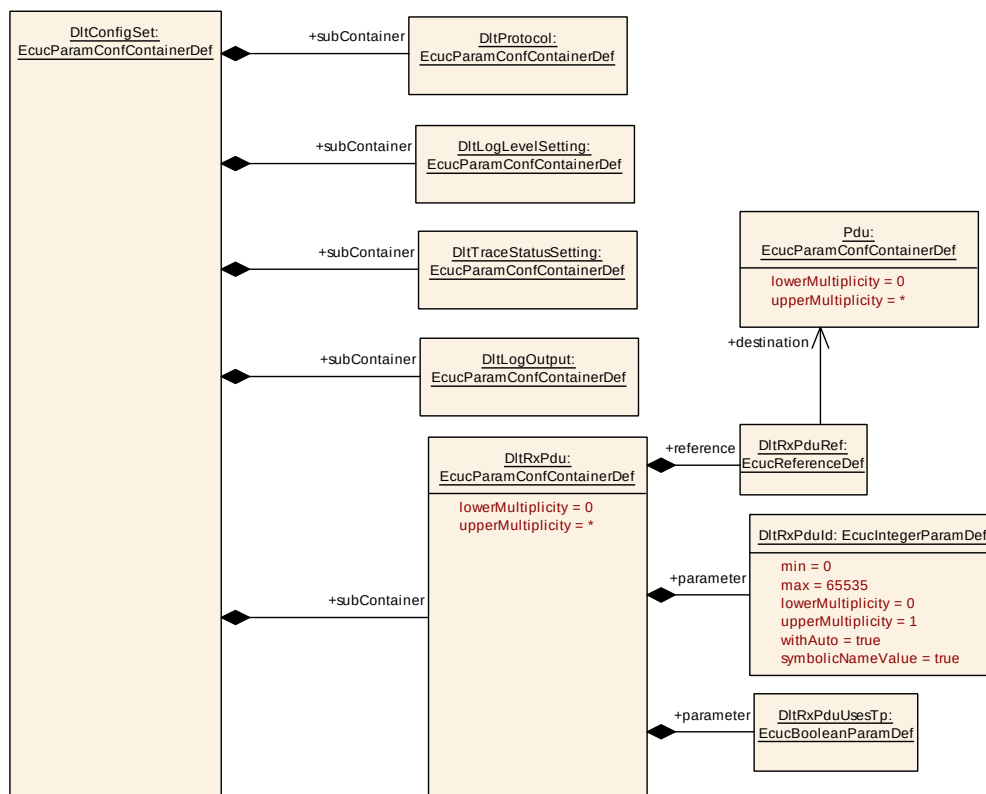


Figure 10.4: DltConfigSet

10.1.6 DltProtocol

[ECUC_Dlt_00832] Definition of EcucParamConfContainerDef DltProtocol [

Container Name	DltProtocol
Parent Container	DltConfigSet
Description	Configuration parameters for handling the specific protocol variants.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltHeaderUseEculd	1	[ECUC_Dlt_00811]
DltHeaderUseSessionID	1	[ECUC_Dlt_00813]
DltHeaderUseTimestamp	1	[ECUC_Dlt_00814]
DltUseExtHeaderInNonVerbMode	1	[ECUC_Dlt_00812]
DltUseVerboseMode	1	[ECUC_Dlt_00911]

Included Containers		
Container Name	Multiplicity	Dependency
DltEculd	1	This is a choice container to choose between a Eculd value or a callout to get the Eculd.

]

[ECUC_Dlt_00811] Definition of EcucBooleanParamDef DltHeaderUseEculd [

Parameter Name	DltHeaderUseEculd		
Parent Container	DltProtocol		
Description	Corresponds to field WEID (With ECU ID). If set ECU ID shall be placed in the header, else not. If DltGeneralNvRAMSupport is enabled the value of the parameter defined here is also 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
Dependency			

]

[ECUC_Dlt_00813] Definition of EcucBooleanParamDef DltHeaderUseSessionID

[

Parameter Name	DltHeaderUseSessionID		
Parent Container	DltProtocol		
Description	Corresponds to field WSID (with Session ID). If set the Session ID shall be placed in the header, else not. If DltGeneralNvRAMSupport is enabled the value of the parameter defined here is also 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
Dependency			

]

[ECUC_Dlt_00814] Definition of EcucBooleanParamDef DltHeaderUseTimestamp

[

Parameter Name	DltHeaderUseTimestamp		
Parent Container	DltProtocol		
Description	Corresponds to field WTMS (With Timestamp). If set the timestamp shall be placed in the header, else not. If DltGeneralNvRAMSupport is enabled the value of the parameter defined here is also 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
Dependency	Can only be true if DltGeneralTimeStampSupport is true.		

]

[ECUC_Dlt_00812] Definition of EcucBooleanParamDef DltUseExtHeaderInNonVerbMode

[

Parameter Name	DltUseExtHeaderInNonVerbMode		
Parent Container	DltProtocol		
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 the value of the parameter defined here is also 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
Dependency			

[ECUC_Dlt_00911] Definition of EcucBooleanParamDef DltUseVerboseMode [

Parameter Name	DltUseVerboseMode		
Parent Container	DltProtocol		
Description	If this flag is set to TRUE, the payload shall be transmitted in verbose mode, else the payload shall be transmitted in none-verbose mode.		
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
Dependency			

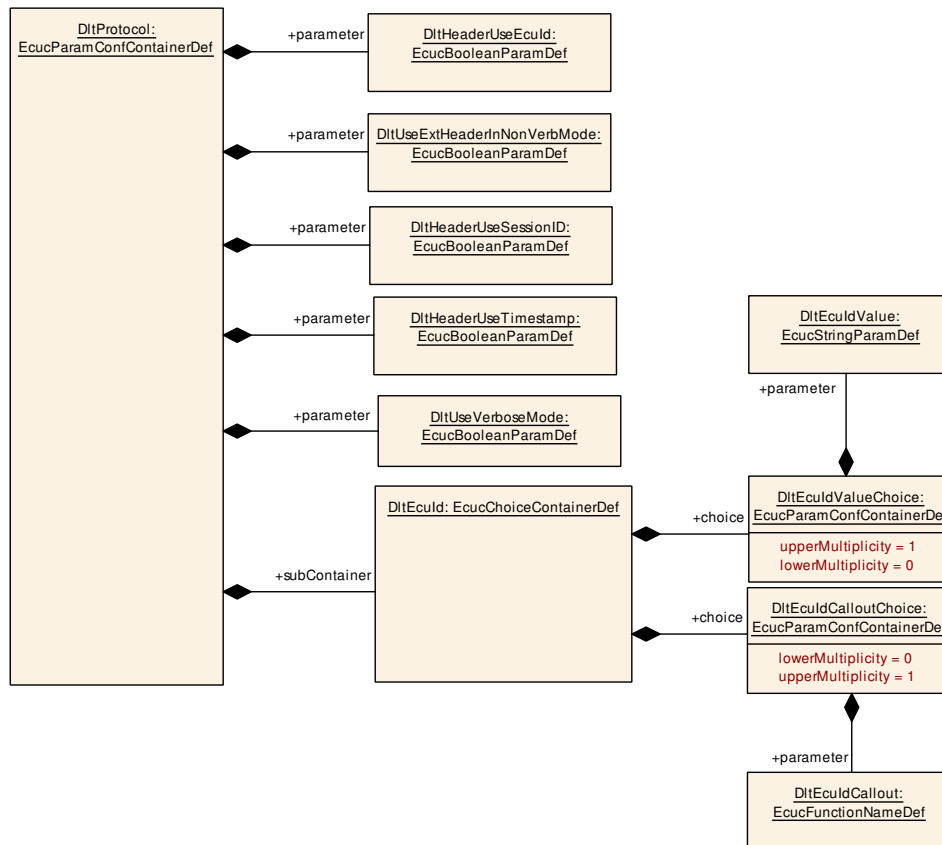


Figure 10.5: DltProtocol

10.1.7 DltEcud

[ECUC_Dlt_00860] Definition of EcucChoiceContainerDef DltEcud [

Choice Container Name	DltEcud
Parent Container	DltProtocol
Description	This is a choice container to choose between a Ecud value or a callout to get the Ecu Id.
Multiplicity	1

No Included Parameters

Container Choices		
Container Name	Multiplicity	Dependency
DltEcudCalloutChoice	0..1	Ecud via user defined callout.
DltEcudValueChoice	0..1	Ecud value configuration

]

10.1.8 DltEcudCalloutChoice

[ECUC_Dlt_00902] Definition of EcucParamConfContainerDef DltEcudCallout Choice [

Container Name	DltEcudCalloutChoice
Parent Container	DltEcud
Description	Ecud via user defined callout.
Multiplicity	0..1
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltEcudCallout	1	[ECUC_Dlt_00862]

No Included Containers

]

[ECUC_Dlt_00862] Definition of EcucFunctionNameDef DltEcudCallout [

Parameter Name	DltEcudCallout
Parent Container	DltEcudCalloutChoice
Description	If this choice is used the Ecud shall be fetched by calling the here configured callout function.
Multiplicity	1
Type	EcucFunctionNameDef





Default value	–		
Regular Expression	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

10.1.9 DltEcudValueChoice

[ECUC_Dlt_00901] Definition of EcucParamConfContainerDef DltEcudValueChoice [

Container Name	DltEcudValueChoice
Parent Container	DltEcud
Description	Ecud value configuration
Multiplicity	0..1
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltEcudValue	1	[ECUC_Dlt_00861]

No Included Containers

[ECUC_Dlt_00861] Definition of EcucStringParamDef DltEcudValue [

Parameter Name	DltEcudValue		
Parent Container	DltEcudValueChoice		
Description	If this choice is used the Ecud 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	–		
Regular Expression	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

10.1.10 DltLogLevelSetting

[ECUC_Dlt_00863] Definition of EcucParamConfContainerDef DltLogLevelSetting

Container Name	DltLogLevelSetting
Parent Container	DltConfigSet
Description	Contains settings for thresholds.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltDefaultLogLevel	1	[ECUC_Dlt_00864]

Included Containers		
Container Name	Multiplicity	Dependency
DltLogLevelThreshold	0..*	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned LogLevel threshold.

[ECUC_Dlt_00864] Definition of EcucEnumerationParamDef DltDefaultLogLevel

Parameter Name	DltDefaultLogLevel		
Parent Container	DltLogLevelSetting		
Description	This is the effective log level 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	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
Dependency			

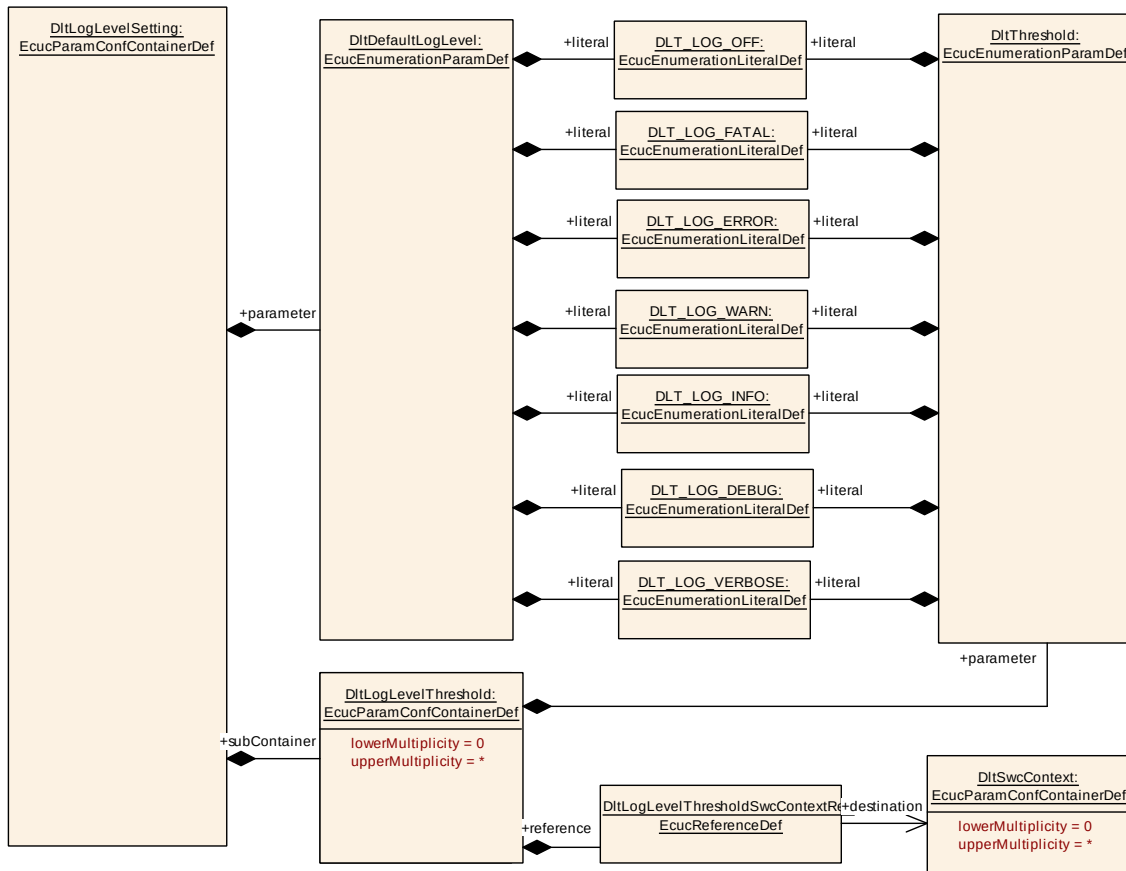


Figure 10.6: DltLogLevelSetting

10.1.11 DltLogLevelThreshold

[ECUC_Dlt_00865] Definition of EcucParamConfContainerDef DltLogLevelThreshold

Container Name	DltLogLevelThreshold		
Parent Container	DltLogLevelSetting		
Description	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned LogLevel threshold.		
Multiplicity	0..*		
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			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltThreshold	1	[ECUC_Dlt_00866]
DltLogLevelThresholdSwcContextRef	1	[ECUC_Dlt_00894]

No Included Containers

[ECUC_Dlt_00866] Definition of EcucEnumerationParamDef DltThreshold

Parameter Name	DltThreshold		
Parent Container	DltLogLevelThreshold		
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
Dependency			

[ECUC_Dlt_00894] Definition of EcucReferenceDef DltLogLevelThresholdSwcContextRef

Parameter Name	DltLogLevelThresholdSwcContextRef		
Parent Container	DltLogLevelThreshold		
Description	Reference to an ApplicationId/ContextId pair to which a LogLevel threshold 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
Dependency			

10.1.12 DltLogChannelAssignment

[ECUC_Dlt_00887] Definition of EcucParamConfContainerDef DltLogChannelAssignment

Container Name	DltLogChannelAssignment		
Parent Container	DltLogOutput		
Description	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned log channel.		
Multiplicity	0..*		
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			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltLogChannelAssignmentSwcContextRef	1	[ECUC_Dlt_00896]
DltLogChannelRef	1	[ECUC_Dlt_00888]

No Included Containers

[ECUC_Dlt_00896] Definition of EcucReferenceDef DltLogChannelAssignmentSwcContextRef

Parameter Name	DltLogChannelAssignmentSwcContextRef		
Parent Container	DltLogChannelAssignment		
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
Dependency			

[ECUC_Dlt_00888] Definition of EcucReferenceDef DltLogChannelRef

Parameter Name	DltLogChannelRef		
Parent Container	DltLogChannelAssignment		
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
Dependency			

10.1.13 DltTraceStatusSetting

[ECUC_Dlt_00869] Definition of EcucParamConfContainerDef DltTraceStatusSetting

Container Name	DltTraceStatusSetting
Parent Container	DltConfigSet
Description	Contains settings for trace status
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltDefaultTraceStatus	1	[ECUC_Dlt_00870]

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

[ECUC_Dlt_00870] Definition of EcucBooleanParamDef DltDefaultTraceStatus

Parameter Name	DltDefaultTraceStatus		
Parent Container	DltTraceStatusSetting		
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
Dependency			

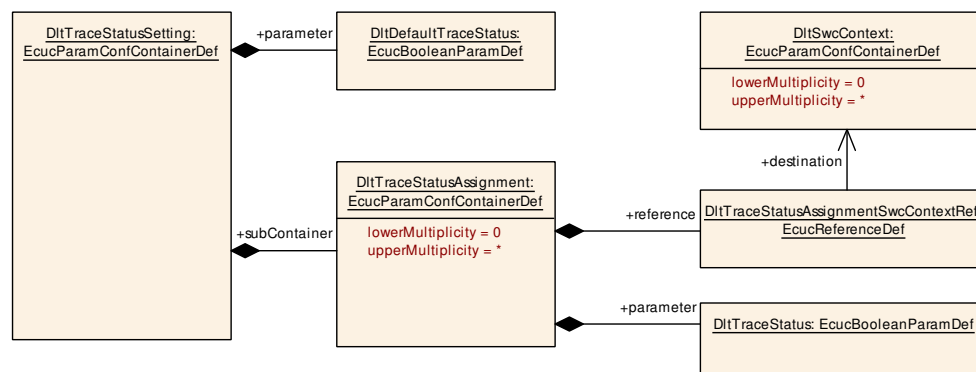


Figure 10.7: DltTraceStatusSetting

10.1.14 DltTraceStatusAssignment

[ECUC_Dlt_00871] Definition of EcucParamConfContainerDef DltTraceStatusAssignment

Container Name	DltTraceStatusAssignment		
Parent Container	DltTraceStatusSetting		
Description	This container contains a preconfiguration of ApplicationId / ContextId pairs and their assigned trace status.		
Multiplicity	0..*		
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			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltTraceStatus	1	[ECUC_Dlt_00874]
DltTraceStatusAssignmentSwcContextRef	1	[ECUC_Dlt_00895]

No Included Containers

[ECUC_Dlt_00874] Definition of EcucBooleanParamDef DltTraceStatus

Parameter Name	DltTraceStatus		
Parent Container	DltTraceStatusAssignment		
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
Dependency			

[ECUC_Dlt_00895] Definition of EcucReferenceDef DltTraceStatusAssignmentSwcContextRef

Parameter Name	DltTraceStatusAssignmentSwcContextRef		
Parent Container	DltTraceStatusAssignment		
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
Dependency			

10.1.15 DltLogOutput

[ECUC_Dlt_00875] Definition of EcucParamConfContainerDef DltLogOutput [

Container Name	DltLogOutput		
Parent Container	DltConfigSet		
Description	Contains settings for log/trace message output		
Multiplicity	1		
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltDefaultLogChannelRef	1	[ECUC_Dlt_00889]

Included Containers		
Container Name	Multiplicity	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.

[ECUC_Dlt_00889] Definition of EcucReferenceDef DltDefaultLogChannelRef [

Parameter Name	DltDefaultLogChannelRef		
Parent Container	DltLogOutput		
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
Dependency			

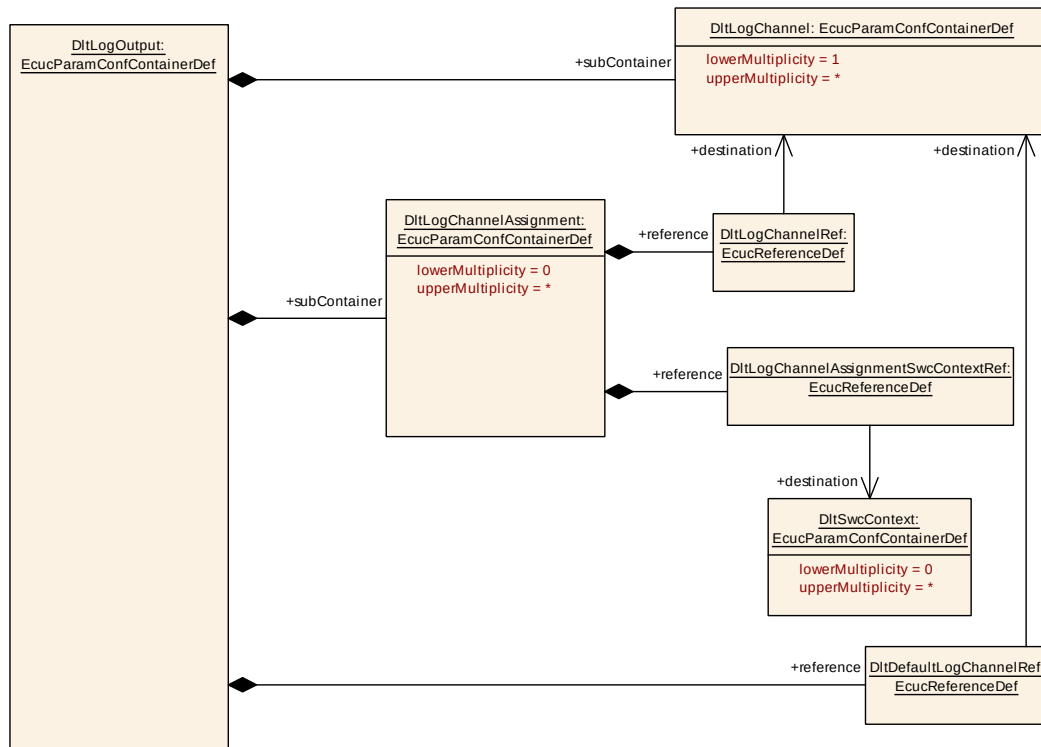


Figure 10.8: DltLogOutput

10.1.16 DltLogChannel

[ECUC_Dlt_00876] Definition of EcucParamConfContainerDef DltLogChannel [

Container Name	DltLogChannel
Parent Container	DltLogOutput
Description	Contains settings for log/trace message output
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltLogChannelBufferOverflowTimer	1	[ECUC_Dlt_00886]
DltLogChannelBufferSize	1	[ECUC_Dlt_00881]
DltLogChannelId	1	[ECUC_Dlt_00877]
DltLogChannelMaxMessageLength	1	[ECUC_Dlt_00882]
DltLogChannelMaxNumOfRetries	1	[ECUC_Dlt_00884]
DltLogChannelSegmentationSupported	1	[ECUC_Dlt_00916]
DltLogChannelThreshold	1	[ECUC_Dlt_00878]
DltLogChannelTrafficShapingBandwidth	0..1	[ECUC_Dlt_00883]
DltLogChannelTransmitCycle	1	[ECUC_Dlt_00885]
DltLogTraceStatusFlag	1	[ECUC_Dlt_00879]

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

[ECUC_Dlt_00886] Definition of EcucFloatParamDef DltLogChannelBufferOverflowTimer

Parameter Name	DltLogChannelBufferOverflowTimer		
Parent Container	DltLogChannel		
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
Dependency			

[ECUC_Dlt_00881] Definition of EcucIntegerParamDef DltLogChannelBufferSize

Parameter Name	DltLogChannelBufferSize		
Parent Container	DltLogChannel		
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	–	
Dependency			

[ECUC_Dlt_00877] Definition of EcucStringParamDef DltLogChannelId

Parameter Name	DltLogChannelId		
Parent Container	DltLogChannel		
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	–		
Regular Expression	–		
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
Dependency			

[ECUC_Dlt_00882] Definition of EcucIntegerParamDef DltLogChannelMaxMessageLength

Parameter Name	DltLogChannelMaxMessageLength		
Parent Container	DltLogChannel		
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	–	
Dependency			

[ECUC_Dlt_00884] Definition of EcucIntegerParamDef DltLogChannelMaxNumOfRetries

Parameter Name	DltLogChannelMaxNumOfRetries		
Parent Container	DltLogChannel		
Description	The maximum amount of retries for sending 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
Dependency			

[ECUC_Dlt_00916] Definition of EcucBooleanParamDef DltLogChannelSegmentationSupported

Parameter Name	DltLogChannelSegmentationSupported		
Parent Container	DltLogChannel		
Description	Segmentation will be used if a DLT message is larger than Pdu length.		
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
Dependency			

[ECUC_Dlt_00878] Definition of EcucEnumerationParamDef DltLogChannelThreshold

Parameter Name	DltLogChannelThreshold		
Parent Container	DltLogChannel		
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
Dependency			

[ECUC_Dlt_00883] Definition of EcucIntegerParamDef DltLogChannelTrafficShapingBandwidth

Parameter Name	DltLogChannelTrafficShapingBandwidth		
Parent Container	DltLogChannel		
Description	Set the maximum possible bandwidth 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
Dependency	DltGeneralTrafficShapingSupport enabled		

[ECUC_Dlt_00885] Definition of EcucFloatParamDef DltLogChannelTransmitCycle

Parameter Name	DltLogChannelTransmitCycle		
Parent Container	DltLogChannel		
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
Dependency			

[ECUC_Dlt_00879] Definition of EcucBooleanParamDef DltLogTraceStatusFlag

Parameter Name	DltLogTraceStatusFlag		
Parent Container	DltLogChannel		
Description	Parameter to turn on/off tracing 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
Dependency			

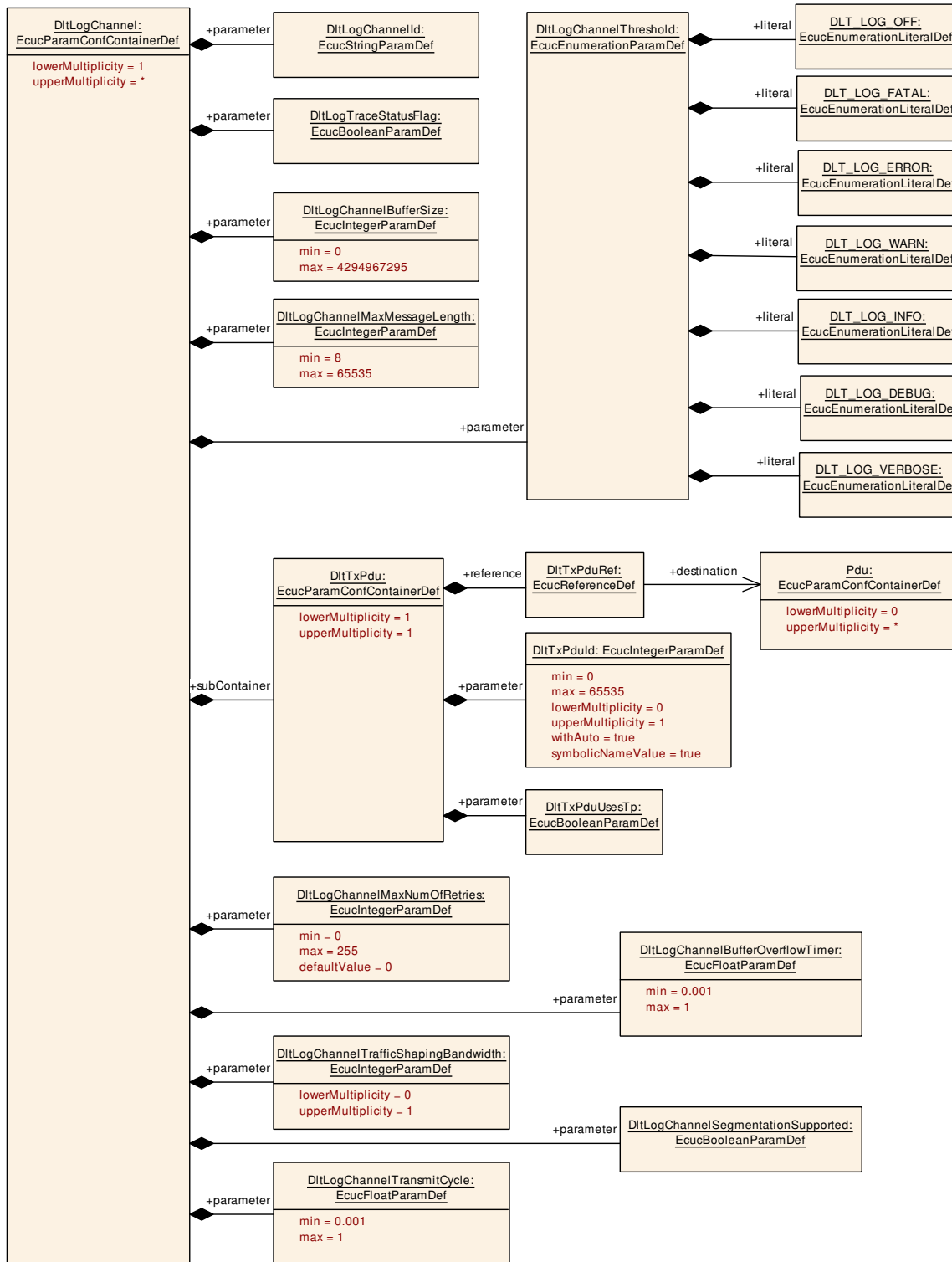


Figure 10.9: DltLogChannel

10.1.17 DltTxPdu

[ECUC_Dlt_00907] Definition of EcucParamConfContainerDef DltTxPdu [

Container Name	DltTxPdu
Parent Container	DltLogChannel
Description	Contains the configuration parameters of the AUTOSAR Dlt module's Tx Pdus.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltTxPduId	0..1	[ECUC_Dlt_00893]
DltTxPduUsesTp	1	[ECUC_Dlt_00913]
DltTxPduRef	1	[ECUC_Dlt_00892]

No Included Containers

[ECUC_Dlt_00893] Definition of EcucIntegerParamDef DltTxPduId [

Parameter Name	DltTxPduId		
Parent Container	DltTxPdu		
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.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
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	—	
Dependency	withAuto = true		

[ECUC_Dlt_00913] Definition of EcucBooleanParamDef DltTxPduUsesTp [

Parameter Name	DltTxPduUsesTp
Parent Container	DltTxPdu
Description	If set to TRUE, the PDU is transmitted using the TP API. If FALSE, the IF API is 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
Dependency			

[ECUC_Dlt_00892] Definition of EcucReferenceDef DltTxPduRef [

Parameter Name	DltTxPduRef		
Parent Container	DltTxPdu		
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
Dependency			

10.1.18 DltRxPdu

[ECUC_Dlt_00900] Definition of EcucParamConfContainerDef DltRxPdu [

Container Name	DltRxPdu		
Parent Container	DltConfigSet		
Description	Contains the Pdu IDs to be used for Dlt control messages reception.		
Multiplicity	0..*		
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			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DltRxPduId	0..1	[ECUC_Dlt_00899]
DltRxPduUsesTp	1	[ECUC_Dlt_00912]
DltRxPduRef	1	[ECUC_Dlt_00898]

No Included Containers

[ECUC_Dlt_00899] Definition of EcucIntegerParamDef DltRxPduId [

Parameter Name	DltRxPduId		
Parent Container	DltRxPdu		
Description	The numerical value used as the ID of this I-PDU. The DltRxPduId is required by the API calls Dlt_RxIndication, Dlt_TpRxIndication, Dlt_StartOfReception and Dlt_CopyRxData to receive I-PDUs from the PduR.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
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	–	
Dependency	withAuto = true		

]

[ECUC_Dlt_00912] Definition of EcucBooleanParamDef DltRxPduUsesTp [

Parameter Name	DltRxPduUsesTp		
Parent Container	DltRxPdu		
Description	If set to TRUE, the PDU is received using the TP API. If FALSE, the IF API is 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
Dependency			

]

[ECUC_Dlt_00898] Definition of EcucReferenceDef DltRxPduRef [

Parameter Name	DltRxPduRef		
Parent Container	DltRxPdu		
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





Dependency	
------------	--

」

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.

A Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	PPortPrototype			
Note	Component port providing a certain port interface.			
Base	<i>ARObject</i> , <i>AbstractProvidedPortPrototype</i> , <i>AtpBlueprintable</i> , <i>AtpFeature</i> , <i>AtpPrototype</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PortPrototype</i> , <i>Referrable</i>			
Aggregated by	<i>AtpClassifier.atpFeature</i> , <i>SwComponentType.port</i>			
Attribute	Type	Mult.	Kind	Note
provided Interface	PortInterface	0..1	tref	The interface that this port provides. Stereotypes: isOfType

Table A.1: PPortPrototype

Class	RPortPrototype			
Note	Component port requiring a certain port interface.			
Base	<i>ARObject</i> , <i>AbstractRequiredPortPrototype</i> , <i>AtpBlueprintable</i> , <i>AtpFeature</i> , <i>AtpPrototype</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PortPrototype</i> , <i>Referrable</i>			
Aggregated by	<i>AtpClassifier.atpFeature</i> , <i>SwComponentType.port</i>			
Attribute	Type	Mult.	Kind	Note
mayBe Unconnected	Boolean	0..1	attr	If set to true, this attribute indicates that the enclosing <i>RPortPrototype</i> may be left unconnected and that this aspect has explicitly been considered in the software-component's design. This Attribute is only used by the AUTOSAR Classic Platform.
required Interface	PortInterface	0..1	tref	The interface that this port requires. Stereotypes: isOfType

Table A.2: RPortPrototype

Class	Referrable (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	<i>ARObject</i>			
Subclasses	<i>AtpDefinition</i> , <i>BswDistinguishedPartition</i> , <i>BswModuleCallPoint</i> , <i>BswModuleClientServerEntry</i> , <i>BswVariableAccess</i> , <i>CouplingPortTrafficClassAssignment</i> , <i>DiagnosticEnvModeElement</i> , <i>EthernetPriorityRegeneration</i> , <i>ExclusiveAreaNestingOrder</i> , <i>HwDescriptionEntity</i> , <i>ImplementationProps</i> , <i>LinSlaveConfigIdent</i> , <i>ModeTransition</i> , <i>MultilanguageReferrable</i> , <i>PncMappingIdent</i> , <i>SingleLanguageReferrable</i> , <i>SoConIPdulIdentifier</i> , <i>TpConnectionIdent</i>			
Attribute	Type	Mult.	Kind	Note
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. Stereotypes: atpIdentityContributor Tags: xml.enforceMinMultiplicity=true xml.sequenceOffset=-100





Class	Referrable (abstract)			
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. Tags: xml.sequenceOffset=-90

Table A.3: Referrable

B Change History

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These specification items do not appear as hyperlinks in the document.

B.1 Change History of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

Number	Heading
[SWS_DIt_00003]	
[SWS_DIt_00005]	
[SWS_DIt_00021]	
[SWS_DIt_00022]	
[SWS_DIt_00023]	
[SWS_DIt_00027]	
[SWS_DIt_00031]	
[SWS_DIt_00078]	
[SWS_DIt_00224]	Definition of datatype Dlt_MessageType
[SWS_DIt_00225]	Definition of ImplementationDataType Dlt_SessionIDType
[SWS_DIt_00226]	Definition of ImplementationDataType Dlt_ApplicationIDType
[SWS_DIt_00227]	Definition of ImplementationDataType Dlt_ContextIDType
[SWS_DIt_00228]	Definition of datatype Dlt_MessageIDType
[SWS_DIt_00229]	Definition of ImplementationDataType Dlt_MessageOptionsType
[SWS_DIt_00230]	Definition of ImplementationDataType Dlt_MessageLogLevelType
[SWS_DIt_00231]	Definition of ImplementationDataType Dlt_MessageTraceType
[SWS_DIt_00232]	Definition of ImplementationDataType Dlt_LogChannelNameType
[SWS_DIt_00233]	Definition of datatype Dlt_MessageNetworkTraceInfoType
[SWS_DIt_00235]	Definition of ImplementationDataType Dlt_MessageArgumentCount
[SWS_DIt_00236]	Definition of ImplementationDataType Dlt_MessageLogInfoType
[SWS_DIt_00237]	Definition of ImplementationDataType Dlt_MessageTraceInfoType
[SWS_DIt_00239]	Definition of API function Dlt_Init
[SWS_DIt_00241]	Definition of API function Dlt_SendLogMessage
[SWS_DIt_00243]	Definition of API function Dlt_SendTraceMessage
[SWS_DIt_00245]	Definition of API function Dlt_RegisterContext
[SWS_DIt_00252]	Definition of API function Dlt_SetLogLevel
[SWS_DIt_00254]	Definition of API function Dlt_SetTraceStatus





Number	Heading
[SWS_Dlt_00259]	Definition of configurable interface Dlt_InjectCall_<SESSION>
[SWS_Dlt_00271]	Definition of API function Dlt_GetVersionInfo
[SWS_Dlt_00272]	Definition of callback function Dlt_RxIndication
[SWS_Dlt_00273]	Definition of callback function Dlt_TxConfirmation
[SWS_Dlt_00276]	
[SWS_Dlt_00277]	
[SWS_Dlt_00278]	
[SWS_Dlt_00279]	
[SWS_Dlt_00280]	
[SWS_Dlt_00281]	
[SWS_Dlt_00282]	
[SWS_Dlt_00283]	
[SWS_Dlt_00284]	
[SWS_Dlt_00285]	
[SWS_Dlt_00332]	
[SWS_Dlt_00335]	
[SWS_Dlt_00337]	
[SWS_Dlt_00350]	
[SWS_Dlt_00376]	
[SWS_Dlt_00377]	
[SWS_Dlt_00430]	
[SWS_Dlt_00432]	Definition of API function Dlt_DetForwardErrorTrace
[SWS_Dlt_00437]	Definition of datatype Dlt_ConfigType
[SWS_Dlt_00449]	
[SWS_Dlt_00451]	
[SWS_Dlt_00453]	
[SWS_Dlt_00484]	
[SWS_Dlt_00495]	Definition of ClientServerInterface DltSwcMessageService
[SWS_Dlt_00496]	Definition of ClientServerInterface LogTraceSessionControl
[SWS_Dlt_00498]	Definition of ClientServerInterface InjectionCallback
[SWS_Dlt_00499]	Definition of Port ControlService provided by module Dlt
[SWS_Dlt_00516]	Definition of callback function Dlt_CopyTxData
[SWS_Dlt_00632]	
[SWS_Dlt_00643]	Supported Service ID to Dlt Command Name mapping
[SWS_Dlt_00644]	
[SWS_Dlt_00645]	
[SWS_Dlt_00646]	
[SWS_Dlt_00647]	





Number	Heading
[SWS_DIt_00648]	
[SWS_DIt_00649]	
[SWS_DIt_00650]	
[SWS_DIt_00651]	
[SWS_DIt_00652]	
[SWS_DIt_00653]	
[SWS_DIt_00654]	
[SWS_DIt_00655]	
[SWS_DIt_00656]	
[SWS_DIt_00657]	
[SWS_DIt_00658]	
[SWS_DIt_00659]	
[SWS_DIt_00660]	
[SWS_DIt_00661]	
[SWS_DIt_00662]	
[SWS_DIt_00663]	
[SWS_DIt_00664]	
[SWS_DIt_00665]	
[SWS_DIt_00666]	
[SWS_DIt_00667]	
[SWS_DIt_00668]	
[SWS_DIt_00669]	
[SWS_DIt_00670]	
[SWS_DIt_00671]	
[SWS_DIt_00672]	
[SWS_DIt_00673]	
[SWS_DIt_00674]	
[SWS_DIt_00675]	
[SWS_DIt_00676]	
[SWS_DIt_00677]	
[SWS_DIt_00678]	
[SWS_DIt_00679]	
[SWS_DIt_00680]	
[SWS_DIt_00681]	
[SWS_DIt_00682]	
[SWS_DIt_00683]	
[SWS_DIt_00684]	
[SWS_DIt_00685]	





Number	Heading
[SWS_DIt_00686]	
[SWS_DIt_00687]	
[SWS_DIt_00688]	
[SWS_DIt_00689]	
[SWS_DIt_00690]	
[SWS_DIt_00691]	
[SWS_DIt_00692]	
[SWS_DIt_00693]	
[SWS_DIt_00694]	
[SWS_DIt_00695]	
[SWS_DIt_00696]	
[SWS_DIt_00697]	
[SWS_DIt_00698]	
[SWS_DIt_00699]	
[SWS_DIt_00700]	
[SWS_DIt_00701]	
[SWS_DIt_00702]	
[SWS_DIt_00703]	
[SWS_DIt_00704]	
[SWS_DIt_00705]	
[SWS_DIt_00706]	
[SWS_DIt_00708]	
[SWS_DIt_00709]	
[SWS_DIt_00710]	
[SWS_DIt_00711]	
[SWS_DIt_00712]	
[SWS_DIt_00713]	
[SWS_DIt_00714]	
[SWS_DIt_00715]	
[SWS_DIt_00716]	
[SWS_DIt_00717]	
[SWS_DIt_00718]	
[SWS_DIt_00719]	
[SWS_DIt_00720]	
[SWS_DIt_00721]	
[SWS_DIt_00722]	
[SWS_DIt_00723]	
[SWS_DIt_00724]	





Number	Heading
[SWS_Dlt_00725]	
[SWS_Dlt_00726]	
[SWS_Dlt_00727]	Definiton of development errors in module Dlt
[SWS_Dlt_00728]	Definiton of runtime errors in module Dlt
[SWS_Dlt_00729]	
[SWS_Dlt_00730]	Definition of ImplementationDataType Dlt_AssignmentOperation
[SWS_Dlt_00732]	Definition of API function Dlt_GetLogInfo
[SWS_Dlt_00733]	Definition of API function Dlt_GetDefaultLogLevel
[SWS_Dlt_00734]	
[SWS_Dlt_00735]	
[SWS_Dlt_00736]	Definition of API function Dlt_StoreConfiguration
[SWS_Dlt_00737]	
[SWS_Dlt_00738]	
[SWS_Dlt_00739]	Definition of API function Dlt_ResetToFactoryDefault
[SWS_Dlt_00740]	Definition of API function Dlt_SetDefaultLogLevel
[SWS_Dlt_00741]	
[SWS_Dlt_00742]	
[SWS_Dlt_00743]	Definition of API function Dlt_SetDefaultTraceStatus
[SWS_Dlt_00744]	
[SWS_Dlt_00745]	
[SWS_Dlt_00746]	Definition of API function Dlt_GetDefaultTraceStatus
[SWS_Dlt_00747]	
[SWS_Dlt_00748]	
[SWS_Dlt_00749]	Definition of API function Dlt_GetLogChannelNames
[SWS_Dlt_00750]	Definition of API function Dlt_GetTraceStatus
[SWS_Dlt_00751]	Definition of API function Dlt_SetLogChannelAssignment
[SWS_Dlt_00752]	Definition of API function Dlt_SetLogChannelThreshold
[SWS_Dlt_00753]	Definition of API function Dlt_GetLogChannelThreshold
[SWS_Dlt_00754]	Definition of callback function Dlt_TriggerTransmit
[SWS_Dlt_00755]	
[SWS_Dlt_00756]	Definition of callback function Dlt_TpTxConfirmation
[SWS_Dlt_00758]	
[SWS_Dlt_00759]	
[SWS_Dlt_00760]	
[SWS_Dlt_00761]	
[SWS_Dlt_00762]	Definition of mandatory interfaces in module Dlt
[SWS_Dlt_00763]	Definition of optional interfaces in module Dlt
[SWS_Dlt_00765]	





Number	Heading
[SWS_Dlt_00766]	
[SWS_Dlt_00768]	
[SWS_Dlt_00769]	Definition of API function Dlt_UnregisterContext
[SWS_Dlt_00770]	Definition of API function Dlt_SetMessageFiltering
[SWS_Dlt_00772]	Definition of ClientServerInterface DltControlService
[SWS_Dlt_00773]	
[SWS_Dlt_00774]	
[SWS_Dlt_00775]	
[SWS_Dlt_00776]	
[SWS_Dlt_00777]	
[SWS_Dlt_00778]	Definition of Port InjectCallback_{SW-C} required by module Dlt
[SWS_Dlt_00779]	Definition of Port SessionControlCallback_{SW-C} required by module Dlt
[SWS_Dlt_00780]	
[SWS_Dlt_00782]	
[SWS_Dlt_00783]	
[SWS_Dlt_00784]	
[SWS_Dlt_00785]	
[SWS_Dlt_00787]	
[SWS_Dlt_91001]	Definition of Port SwcMessageService_{SW-C} provided by module Dlt
[SWS_Dlt_91002]	Definition of ImplementationDataType Dlt_LogInfoType
[SWS_Dlt_91003]	Definition of ImplementationDataType Dlt_ContextIdInfoType
[SWS_Dlt_91004]	Definition of ImplementationDataType Dlt_ApplicationIdInfoType
[SWS_Dlt_91005]	Definition of scheduled function Dlt_TxFunction
[SWS_Dlt_91006]	Definition of callback function Dlt_StartOfReception
[SWS_Dlt_91007]	Definition of callback function Dlt_TpRxIndication
[SWS_Dlt_91008]	Definition of callback function Dlt_CopyRxData
[SWS_Dlt_91009]	Definition of imported datatypes of module Dlt
[SWS_Dlt_91010]	Definition of ImplementationDataType Dlt_MessageAttributesType
[SWS_Dlt_91011]	Definition of API function Dlt_SendLogMessageWithAttributes
[SWS_Dlt_91012]	Definition of API function Dlt_SendTraceMessageWithAttributes
[SWS_Dlt_91013]	Definition of ImplementationDataType Dlt_LogChannelNameInfoType

Table B.1: Added Specification Items in R23-11

B.1.2 Changed Specification Items in R23-11

none

B.1.3 Deleted Specification Items in R23-11

none

B.2 Change History of this document according to AUTOSAR Release R24-11

B.2.1 Added Specification Items in R24-11

none

B.2.2 Changed Specification Items in R24-11

Number	Heading
[SWS_Dlt_00229]	Definition of ImplementationDataType Dlt_MessageOptionsType
[SWS_Dlt_00495]	Definition of ClientServerInterface DltSwcMessageService
[SWS_Dlt_00736]	Definition of API function Dlt_StoreConfiguration
[SWS_Dlt_00737]	
[SWS_Dlt_00738]	
[SWS_Dlt_00763]	Definition of optional interfaces requested by module Dlt
[SWS_Dlt_00772]	Definition of ClientServerInterface DltControlService
[SWS_Dlt_91009]	Definition of imported datatypes of module Dlt
[SWS_Dlt_91011]	Definition of API function Dlt_SendLogMessageWithAttributes
[SWS_Dlt_91012]	Definition of API function Dlt_SendTraceMessageWithAttributes

Table B.2: Changed Specification Items in R24-11

B.2.3 Deleted Specification Items in R24-11

none

B.3 Change History of this document according to AUTOSAR Release R25-11

B.3.1 Added Specification Items in R25-11

none

B.3.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_Dlt_00800]	Definition of EcucModuleDef Dlt
[ECUC_Dlt_00809]	Definition of EcucParamConfContainerDef DltGeneral
[ECUC_Dlt_00811]	Definition of EcucBooleanParamDef DltHeaderUseEculd
[ECUC_Dlt_00812]	Definition of EcucBooleanParamDef DltUseExtHeaderInNonVerbMode
[ECUC_Dlt_00813]	Definition of EcucBooleanParamDef DltHeaderUseSessionID
[ECUC_Dlt_00814]	Definition of EcucBooleanParamDef DltHeaderUseTimestamp
[ECUC_Dlt_00832]	Definition of EcucParamConfContainerDef DltProtocol
[ECUC_Dlt_00840]	Definition of EcucBooleanParamDef DltGeneralDevErrorDetect
[ECUC_Dlt_00842]	Definition of EcucParamConfContainerDef DltConfigSet
[ECUC_Dlt_00844]	Definition of EcucBooleanParamDef DltGeneralVersionInfoApi
[ECUC_Dlt_00845]	Definition of EcucReferenceDef DltGeneralNvRamRef
[ECUC_Dlt_00846]	Definition of EcucBooleanParamDef DltGeneralRegisterContextNotification
[ECUC_Dlt_00847]	Definition of EcucBooleanParamDef DltGeneralInjectionSupport
[ECUC_Dlt_00848]	Definition of EcucBooleanParamDef DltGeneralRxDataPathSupport
[ECUC_Dlt_00849]	Definition of EcucBooleanParamDef DltGeneralTrafficShapingSupport
[ECUC_Dlt_00850]	Definition of EcucBooleanParamDef DltGeneralTimeStampSupport
[ECUC_Dlt_00852]	Definition of EcucIntegerParamDef DltSwcSessionId
[ECUC_Dlt_00853]	Definition of EcucBooleanParamDef DltSwcSupportLogLevelAndTrace StatusChangeNotification
[ECUC_Dlt_00854]	Definition of EcucParamConfContainerDef DltSwcContext
[ECUC_Dlt_00856]	Definition of EcucParamConfContainerDef DltSwc
[ECUC_Dlt_00858]	Definition of EcucIntegerParamDef DltSwcApplicationId
[ECUC_Dlt_00859]	Definition of EcucIntegerParamDef DltSwcContextId
[ECUC_Dlt_00860]	Definition of EcucChoiceContainerDef DltEculd
[ECUC_Dlt_00861]	Definition of EcucStringParamDef DltEculdValue
[ECUC_Dlt_00862]	Definition of EcucFunctionNameDef DltEculdCallout
[ECUC_Dlt_00863]	Definition of EcucParamConfContainerDef DltLogLevelSetting
[ECUC_Dlt_00864]	Definition of EcucEnumerationParamDef DltDefaultLogLevel
[ECUC_Dlt_00865]	Definition of EcucParamConfContainerDef DltLogLevelThreshold
[ECUC_Dlt_00866]	Definition of EcucEnumerationParamDef DltThreshold
[ECUC_Dlt_00869]	Definition of EcucParamConfContainerDef DltTraceStatusSetting
[ECUC_Dlt_00870]	Definition of EcucBooleanParamDef DltDefaultTraceStatus
[ECUC_Dlt_00871]	Definition of EcucParamConfContainerDef DltTraceStatusAssignment
[ECUC_Dlt_00874]	Definition of EcucBooleanParamDef DltTraceStatus
[ECUC_Dlt_00875]	Definition of EcucParamConfContainerDef DltLogOutput
[ECUC_Dlt_00876]	Definition of EcucParamConfContainerDef DltLogChannel





Number	Heading
[ECUC_Dlt_00877]	Definition of EcucStringParamDef DltLogChannelId
[ECUC_Dlt_00878]	Definition of EcucEnumerationParamDef DltLogChannelThreshold
[ECUC_Dlt_00879]	Definition of EcucBooleanParamDef DltLogTraceStatusFlag
[ECUC_Dlt_00881]	Definition of EcucIntegerParamDef DltLogChannelBufferSize
[ECUC_Dlt_00882]	Definition of EcucIntegerParamDef DltLogChannelMaxMessageLength
[ECUC_Dlt_00883]	Definition of EcucIntegerParamDef DltLogChannelTrafficShapingBandwidth
[ECUC_Dlt_00884]	Definition of EcucIntegerParamDef DltLogChannelMaxNumOfRetries
[ECUC_Dlt_00885]	Definition of EcucFloatParamDef DltLogChannelTransmitCycle
[ECUC_Dlt_00886]	Definition of EcucFloatParamDef DltLogChannelBufferOverflowTimer
[ECUC_Dlt_00887]	Definition of EcucParamConfContainerDef DltLogChannelAssignment
[ECUC_Dlt_00888]	Definition of EcucReferenceDef DltLogChannelRef
[ECUC_Dlt_00889]	Definition of EcucReferenceDef DltDefaultLogChannelRef
[ECUC_Dlt_00892]	Definition of EcucReferenceDef DltTxPduRef
[ECUC_Dlt_00893]	Definition of EcucIntegerParamDef DltTxPduld
[ECUC_Dlt_00894]	Definition of EcucReferenceDef DltLogLevelThresholdSwcContextRef
[ECUC_Dlt_00895]	Definition of EcucReferenceDef DltTraceStatusAssignmentSwcContextRef
[ECUC_Dlt_00896]	Definition of EcucReferenceDef DltLogChannelAssignmentSwcContextRef
[ECUC_Dlt_00897]	Definition of EcucFloatParamDef DltGeneralStartUpDelayTimer
[ECUC_Dlt_00898]	Definition of EcucReferenceDef DltRxPduRef
[ECUC_Dlt_00899]	Definition of EcucIntegerParamDef DltRxPduld
[ECUC_Dlt_00900]	Definition of EcucParamConfContainerDef DltRxPdu
[ECUC_Dlt_00901]	Definition of EcucParamConfContainerDef DltEculdValueChoice
[ECUC_Dlt_00902]	Definition of EcucParamConfContainerDef DltEculdCalloutChoice
[ECUC_Dlt_00905]	Definition of EcucReferenceDef DltGeneralGptChannelRef
[ECUC_Dlt_00907]	Definition of EcucParamConfContainerDef DltTxPdu
[ECUC_Dlt_00909]	Definition of EcucIntegerParamDef MaxSwcLogMessageLength
[ECUC_Dlt_00910]	Definition of EcucIntegerParamDef MaxSwcTraceMessageLength
[ECUC_Dlt_00911]	Definition of EcucBooleanParamDef DltUseVerboseMode
[ECUC_Dlt_00912]	Definition of EcucBooleanParamDef DltRxPduUsesTp
[ECUC_Dlt_00913]	Definition of EcucBooleanParamDef DltTxPduUsesTp
[ECUC_Dlt_00914]	Definition of EcucReferenceDef DltGeneralStbMTimeBaseRef
[ECUC_Dlt_00915]	Definition of EcucBooleanParamDef DltGeneralNvRAMSupport
[ECUC_Dlt_00916]	Definition of EcucBooleanParamDef DltLogChannelSegmentationSupported
[ECUC_Dlt_00917]	Definition of EcucIntegerParamDef DltProtocolVersion
[ECUC_Dlt_00918]	Definition of EcucIntegerParamDef DltMaxNumberOfChannels
[SWS_Dlt_00225]	Definition of ImplementationDataType Dlt_SessionIDType
[SWS_Dlt_00226]	Definition of ImplementationDataType Dlt_ApplicationIDType
[SWS_Dlt_00227]	Definition of ImplementationDataType Dlt_ContextIDType





Number	Heading
[SWS_Dlt_00229]	Definition of ImplementationDataType Dlt_MessageOptionsType
[SWS_Dlt_00230]	Definition of ImplementationDataType Dlt_MessageLogLevelType
[SWS_Dlt_00231]	Definition of ImplementationDataType Dlt_MessageTraceType
[SWS_Dlt_00232]	Definition of ImplementationDataType Dlt_LogChannelNameType
[SWS_Dlt_00235]	Definition of ImplementationDataType Dlt_MessageArgumentCount
[SWS_Dlt_00236]	Definition of ImplementationDataType Dlt_MessageLogInfoType
[SWS_Dlt_00237]	Definition of ImplementationDataType Dlt_MessageTraceInfoType
[SWS_Dlt_00437]	Definition of datatype Dlt_ConfigType
[SWS_Dlt_00495]	Definition of ClientServerInterface DltSwcMessageService
[SWS_Dlt_00496]	Definition of ClientServerInterface LogTraceSessionControl
[SWS_Dlt_00498]	Definition of ClientServerInterface InjectionCallback
[SWS_Dlt_00499]	Definition of Port ControlService provided by module Dlt
[SWS_Dlt_00730]	Definition of ImplementationDataType Dlt_AssignmentOperation
[SWS_Dlt_00732]	Definition of API function Dlt_GetLogInfo
[SWS_Dlt_00772]	Definition of ClientServerInterface DltControlService
[SWS_Dlt_91002]	Definition of ImplementationDataType Dlt_LogInfoType
[SWS_Dlt_91003]	Definition of ImplementationDataType Dlt_ContextIdInfoType
[SWS_Dlt_91004]	Definition of ImplementationDataType Dlt_ApplicationIdInfoType
[SWS_Dlt_91005]	Definition of scheduled function Dlt_TxFunction
[SWS_Dlt_91010]	Definition of ImplementationDataType Dlt_MessageAttributesType
[SWS_Dlt_91013]	Definition of ImplementationDataType Dlt_LogChannelNameInfoType

Table B.3: Changed Specification Items in R25-11

B.3.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_Dlt_00228]	Definition of datatype Dlt_MessageIDType
[SWS_Dlt_00376]	
[SWS_Dlt_00430]	
[SWS_Dlt_00432]	Definition of API function Dlt_DetForwardErrorTrace

Table B.4: Deleted Specification Items in R25-11