Document Title	Requirements on Log and Trace
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	350

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R23-11

Document Change History

Date	Release	Changed by	Description
2023-11-23	R23-11	AUTOSAR Release Management	• Fix "AppliesTo: (FO) -> (CP,AP)"
2022-11-25	R22-11	AUTOSAR Release Management	• Support of harmonized logging • Reserved Log Context IDs for the framework
2021-11-25	R21-11	AUTOSAR Release Management	• Fix references to RS_Main
2020-11-30	R20-11	AUTOSAR Release Management	• Update "Applies to" information for some requirements • Fix wording of some requirements to comply with common wording rules
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2019-03-29	1.5.1	AUTOSAR Release Management	• No content changes
2018-10-31	1.5.0	AUTOSAR Release Management	• Requirement to provide Logging Information added • Editorial changes
2018-03-29	1.4.0	AUTOSAR Release Management	• No content changes

▽

△

2017-12-08	1.3.0	AUTOSAR Release Management	• No content changes
2017-10-27	1.2.0	AUTOSAR Release Management	• Migration of document to standard Foundation

Disclaimer

Contents

1 Scope of Document

This following specification defines the functional and non-functional requirements of Log and Trace (LT) for AUTOSAR.

The focus of this document is to specify the requirements for:

- The interface of LT to other CP:BSW modules / AP:Functional Clusters
- The interface to CP:RTE/VFB / AP:ARA Tracing
- The interface to CP:SWCs / AP:Applications
- The transmission and storage format of the log and trace messages
- The internal interface to the LT communication module
- The general configuration of LT

The document does NOT specify the requirements for:

- The transport layer of the communication over the LT communication module

2 Conventions to be used

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see AUTOSAR Standardization Template [1], chapter Support for Traceability.

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see AUTOSAR Standardization Template [1], chapter Support for Traceability.

3 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].

Abbreviation / Acronym:	Description:
Log and trace message	A log and trace message contains all data and options to specify a log and trace event in a software.
User	The user of LT is the programmer of the software, which uses the LT API to generate log and trace messages.
Log	The user generates log messages on demand. Each time the user wants to show some information about state changes or value changes, he adds an API call to LT.
Trace	Trace messages can be generated by instrumentation of the code (e.g. VFB traces). The instrumented code calls the API of LT.
ECU ID	ECU ID is the name of each ECU.
Session ID	Session ID is the identification number of a log or trace session. If an Application is instantiated several times the log sessions get a new Session ID. A Application can have several log or trace sessions. A BSW module uses the module-number as Session ID.
Application ID	Application ID is a short name of the Application/BSW module. It identifies the Application/BSW module in the log and trace message.
Context ID	Context ID is a user defined ID to group log and trace messages produced by an Application/BSW module to distinguish functionality. Each Application ID can own several Context IDs. Context ID's are grouped by Application ID's. Context IDs shall be unique within an Application ID. The identification of the source of a log and trace message is done with a pair of Application ID and Context ID.
Message ID	Messaged ID is the ID to characterize the information, which is transported by the message itself. A Message ID identifies a 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.
Log and trace level	A log level defines a classification for the severity grade of a log message. The trace status provides information if a trace message should be send.
Time	Each log and trace message may contain a time attribute. The time attribute is a free defined time-value. It is the time since the start of the ECU.
External client	An external client is a tool, which can be run on a PC or another ECU, which is connected to LT over DCM or over the LT communication module.

4 Requirements Specification

This chapter describes all requirements driving the work to define Log and Trace in AUTOSAR.

4.1 Functional Overview

LT provides a generic Logging and Tracing functionality for Applications and other modules.

Generally, LT provides the following functionalities:

- Logging
  - Logging of errors, warnings and info messages from AUTOSAR Applications, providing a standardized AUTOSAR interface
  - Gather all LT messages from all AUTOSAR Applications Level and Middleware Level software in a centralized AUTOSAR module (DLT).
  - Log messages from DET[3], DEM[4] (Classic Platform)
- Tracing
  - Trace RTE/VFB ([5]) (Classic Platform)
- Control
  - Enable/Disable individual LT messages
  - Control trace levels individually by back channel
- Generic
  - LT available during debugging and production phase
  - Access over standard diagnosis or platform specific test interface
  - Security mechanisms to prevent misuse in production phase

In addition, the following functionalities are provided for the Adaptive Platform:

- Timestamps
  - In case a timestamp is required, the LT module provides the possibility of time-stamping both the logging as well as the tracing messages.
  - Such timestamping is handled directly by the LT module, without any needed intervention from the application.
- Application communication tracing

– The LT module is capable of handling the tracing of the communication flow between multiple Adaptive Applications without further interaction of the applications being necessary.

4.2 Functional Requirements

4.2.1 Log and trace interfaces

4.2.1.1 Generic

4.2.1.1.1 Logging shall support initialization and registration.

[RS_LT_00047] ⌈

Description:	Logging shall support to initialize the logging framework and to register the source of logging information.
Rationale:	To be able to filter and associate logging information with the origin, it is necessary that applications register themselves at the logging framework.
AppliesTo:	CP,AP
Use Case:	Associate logging information with the origin, apply filter settings and provide additional information.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00400*)

4.2.1.1.2 Logging shall enable applications to provide meta information.

[RS_LT_00048] ⌈

Description:	Logging shall enable applications to provide additional information about themselves.
Rationale:	Being able to identify the origin of generated logging information.
AppliesTo:	CP,AP
Use Case:	Sort, filter and associate received logging information.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00400*)

4.2.1.1.3 Logging shall enable applications to provide Logging Information.

[RS_LT_00049] ⌈

Description:	Logging shall enable applications to provide logging information. The logging information shall include information about its severity.
Rationale:	While developing applications, it is important to get additional information what is going on internally of an application.
AppliesTo:	CP,AP
Use Case:	Getting internal information of an application, e.g.: Variable values, the current internal state of a state machine and other information.
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00400)*

4.2.1.1.4 Logging shall enable tracing of communication between applications.

[RS_LT_00057] ⌈

Description:	It shall be possible to trace the communication between applications.
Rationale:	More and more applications are integrated on one ECU. As a consequence, the communication between applications is done locally and not over an external traceable bus like CAN.
AppliesTo:	CP,AP
Use Case:	Trace the communication between applications.
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00400)*

4.2.1.1.5 Logging shall support grouping of Logging Information.

[RS_LT_00050] ⌈

Description:	Logging shall support to logically group logging information.
Rationale:	Cluster logging information which logically belongs together.
AppliesTo:	CP,AP
Use Case:	Associate or filter all logging information which belongs together.
Dependencies:	–

▽

△

Supporting Material:	–

⌋(*RS_Main_00400*)

4.2.1.1.6 Logging shall allow to select the destination of logging information.

[RS_LT_00051] ⌈

Description:	Logging shall allow to select the destination of the provided logging information.
Rationale:	There are different possibilities where the logging information can be stored.
AppliesTo:	CP,AP
Use Case:	Forward logging information to the console, to the file system or to send it via the communication bus.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00060*)

4.2.1.1.7 Logging shall provide early logging capabilities.

[RS_LT_00052] ⌈

Description:	Logging framework shall provide early logging possibility. The API shall be able to handle log message attempts as well as creating new log contexts instances before the main initialization phase of the Logging back-end was accomplished.
Rationale:	Calling the initialization procedure at earliest is possible when the program code is executed within the main event loop. Before that, applications might create global/static objects which constructors might already log useful information. Since global/static object are instantiated before the main event loop enters, the Logging framework must support calls to all its interfaces without crashing. Ideally it shall buffer log attempts (an unspecified amount) and statically created log context instances and process them ASAP after the initialization phase.
AppliesTo:	CP,AP
Use Case:	Create log contexts and/or initiate log messages before initialization phase of the Logging back-end was accomplished.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00060*)

4.2.1.1.8 The LT shall transmit log and trace messages from several sources over a communication interface to a receiving external client.

[RS_LT_00001] ⌈

Description:	The LT module shall be a BSW module. It shall receive log and trace messages from several sources, like Applications, some BSW modules and the RTE. These messages may be transmitted over a communication interface to an external client. Log and trace is a debugging mechanism needed by a lot of ECU's. A log and trace module shall provide a mechanism to gather log and trace messages from several sources. The log and trace module buffers the log and trace entries if necessary and shall have a connection to an external client.
Rationale:	Each Tier1 uses its own mechanisms to provide such a logging interface, using some internal or external debugging interfaces. The format of the logging content also differs from ECU to ECU. When testing several ECU's, many different tools and parsers are needed to get the right information out of the logs. A standard Diagnostic Logging Component with standardized logging content may help to reduce the testing efforts and enable new automated testing mechanisms. Also the number of tools could be reduced by a standard logging content and protocol.
AppliesTo:	CP,AP
Use Case:	• Development support • Functional Testing • Test Automation • Test against models • Driver intensive tests • Advanced Diagnostic Tracing, optional over telematic services
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.1.9 All log and trace messages sent by an ECU shall have a standardized transmission format and a standardized storage format.

[RS_LT_00002] ⌈

Description:	A specified format shall be defined, which covers all requirements of log and trace. Some examples of the information stored in a log or trace message are the source, the context and the timestamp of the message to be able to filter the log and trace messages.

▽

△

Rationale:	Since logging and tracing is an important mechanism for testability and proofing product quality, it is necessary to standardize the transmitted and stored data format. This is important for archiving, comparing and analyzing of log or trace messages. Also it may be possible to build common tools to interpret the incoming data.
AppliesTo:	CP,AP
Use Case:	• Applications sends a log message • LT sends the message over an interface to a data storing external client • The stored data of different ECU's are interpreted by the external client • Log and trace messages from different ECU's can be merged to understand relationship of behavior from distributed functionality
Dependencies:	–
Supporting Material:	

⌋(*RS_Main_00260*, *RS_Main_00420*)

4.2.1.2 Interface for Applications

4.2.1.2.1 Applications shall have the possibility to send log or trace messages to the LT module.

[RS_LT_00003] ⌈

Description:	LT shall offer a generic interface for Applications independent from the type of log and trace message.
Rationale:	To reduce the amount of interfaces a generic and message independent interface may be implemented.
AppliesTo:	AP, CP
Use Case:	• Interfacing Application with LT module • Tracing or logging messages from Applications
Dependencies:	–
Supporting Material:	

⌋(*RS_Main_00260*, *RS_Main_00420*, *RS_Main_00011*)

4.2.1.2.2 Logging shall provide an interface for logging information.

[RS_LT_00043] ⌈

Description:	Logging shall provide interfaces for applications to forward its logging information, including the associated severity level.
Rationale:	Forward logging information to the logging framework.
AppliesTo:	CP,AP
Use Case:	Forward logging information to a logging framework for further processing (e.g. store it locally or forward it to the communication bus).
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00060*)

4.2.1.2.3 Logging shall be able to handle raw buffer content as logging information.

[RS_LT_00044] ⌈

Description:	Logging shall support to provide the content of a raw buffer as logging information.
Rationale:	The original content of a buffer might be of interest for developing purpose.
AppliesTo:	CP,AP
Use Case:	Get the exact raw data which is located in the memory.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00060*)

4.2.1.2.4 Logging shall enable applications to check the current severity level.

[RS_LT_00045] ⌈

Description:	Logging shall provide the possibility for applications to check the current active severity level.
Rationale:	To avoid unnecessary CPU and/or memory consumption, which is needed for the generation of logging information, applications shall have the possibility to first check whether or not its created logging information will be filtered out anyway by the underlying logging framework.

▽

△

AppliesTo:	CP,AP
Use Case:	Avoidance of unnecessary CPU and/or memory consumption.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00060*)

4.2.1.2.5 Logging shall provide conversion functions for hexadecimal and binary values.

[RS_LT_00046] ⌈

Description:	Logging shall provide the possibility for applications to convert decimal values into the hexadecimal or binary system.
Rationale:	Provide some debug information as a hexadecimal or as a binary value, instead of a decimal value.
AppliesTo:	CP,AP
Use Case:	Providing logging information as hexadecimal value or as binary value.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00060*)

4.2.1.2.6 The LT shall provide the actual set of log levels and the trace status to an Application.

[RS_LT_00004] ⌈

Description:	Application shall be aware of its log levels and trace status in order not to generate unnecessary log and trace messages. Log level and trace status should be checked by the Applications before sending the messages.
Rationale:	The generation of unnecessary load on the communication interface and for the system has to be avoided. The LT shall provide information about the actual configuration.
AppliesTo:	CP,AP
Use Case:	• Control of the log levels and of the trace status
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.1.2.7 For each Application the interface to LT shall be configured.

[RS_LT_00005] ⌈

Description:	The generation of the LT interface of each Application shall be enabled by the configuration.
Rationale:	Only the configured interfaces to the Applications, which are using log and trace, shall be generated.
AppliesTo:	CP,AP
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.3 DET trace interface

4.2.1.3.1 Trace events from errors generated by BSW and Applications shall be forwarded to the LT module.

[RS_LT_00006] ⌈

Description:	The DET receives trace events from errors generated by BSW and Applications during debugging time. These events may be forwarded to the LT module.
Rationale:	To have an overview of all log, trace and error messages and to set all of them in the correct context, it is important to have all these messages and events in one list. Also it is not practicable to use more than one mechanism to report errors, logs and traces to an external client. So all these sources may be forwarded to the LT module.
AppliesTo:	CP
Use Case:	In a debugging scenario, an Application or BSW Module uses the DET interface to trace an error. This error is forwarded by the DET module to the LT module. The LT turns these events into a log message and sends it to the external client.
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.4 DEM trace interface

4.2.1.4.1 The DEM shall forward error events to the LT module.

[RS_LT_00007] ⌈

Description:	The DEM may forward error events to the LT module.
Rationale:	It may be possible to get an overview of all error messages in an ECU. It may be possible to set them in the correct context with the error events reported by DEM. This makes an analysis of the reported errors more efficient and gives a correct picture of a failure situation.
AppliesTo:	CP
Use Case:	A Application or BSW module sets an DTC in the DEM. The DEM forwards this event to the LT. The LT turns these events in the LT format and sends it over a communication interface to a LT external client.
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.5 RTE/VFB trace interface

4.2.1.5.1 RTE shall provide an interface for LT to trace RTE/VFB calls.

[RS_LT_00008] ⌈

Description:	RTE shall provide the possibility to trace the VFB.
Rationale:	In the future more and more Applications will be integrated in one ECU. As a consequence the communication between Application is done locally and not over an external traceable bus like CAN or Flexray. It is important to trace the internal communication over RTE/VFB.
AppliesTo:	CP
Use Case:	• Trace of VFB interface • Access to VFB for advanced diagnostic services
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.5.2 The LT shall implement an interface to trace the RTE/VFB.

[RS_LT_00009] ⌈

Description:	VFB-Tracing shall be provided to the RTE generation system. The LT module shall implement the handling of the RTE/VFB trace events.
Rationale:	To understand the communication between the Applications it is important to trace all function calls to the Applications except the RTE system function calls.
AppliesTo:	CP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.5.3 A global switch shall be defined to switch on and off the RTE tracing.

[RS_LT_00010] ⌈

Description:	There shall be a global switch to turn on and off the RTE tracing of LT at configuration time and at run time.
Rationale:	To have a high level activation for all RTE tracing messages a global switch is needed.
AppliesTo:	CP
Use Case:	Turning all tracing events for LT on/off
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.1.5.4 The LT shall implement the handling of the RTE/VFB trace events.

[RS_LT_00011] ⌈

Description:	The LT may implement the handling of the RTE/VFB trace events. The groups may be configured at configuration time and the events may be assigned to one group. At run time the log level of each group can be set individually.
Rationale:	To understand the communication between the Applications it is important to trace the function calls of an Application.
AppliesTo:	CP

▽

△

Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.1.5.5 LT shall provide a solution to trace events linked to implicit communication mechanism.

[RS_LT_00012] ⌈

Description:	LT may provide a solution to trace events linked to implicit communication mechanism. The implicit communication may be traced at runnable invocation and termination.
Rationale:	From the point of view of an Application no extra routine for receiving or sending signals (Sender Receiver Communication) with implicit sending/ receiving is called. The RTE manages the copying of the data before invocation and after exiting.
AppliesTo:	CP
Use Case:	Tracing implicit communication of Applications
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.2 Format of log and trace message

4.2.2.1 The transmitted data shall be packetized.

[RS_LT_00013] ⌈

Description:	The log and trace messages shall be packet oriented. Each packet shall have one or several headers and a payload. The headers may be used for identifying, filtering and classifying the log and trace message. The payload shall contain the parameters given from the Application or BSW.
Rationale:	LT shall not care about the under laying transport protocol. LT shall define the log or trace message format.
AppliesTo:	CP,AP
Use Case:	–

▽

△

Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*, *RS_Main_00100*, *RS_Main_00010*)

4.2.2.2 The transport format shall be binary.

[RS_LT_00014] ⌈

Description:	The log and trace messages shall be in binary format. This is necessary for reducing the bandwidth and the CPU consumption.
Rationale:	Scope of the LT component is to log and trace events without perturbing the ECU behavior. CPU consumption and bandwidth occupation may be reduced by using a binary format for the protocol.
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*, *RS_Main_00100*, *RS_Main_00010*)

4.2.2.3 The format shall deal with Big and Little Endianess.

[RS_LT_00016] ⌈

Description:	The Endian format shall be considered in the message specification. The message shall signal which Endianess mode is actually used.
Rationale:	Big and Little Endianess of data representation has mixed occurrences in automotive ECUs. The message format shall be able to deal with both. To reduce CPU load it shall be possible to transport the data in the corresponding format of the ECU.
AppliesTo:	CP,AP
Use Case:	Marker for format
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*, *RS_Main_00430*)

4.2.2.4 Each log and trace message shall contain a timestamp, which will be added to the message during reception of the message in the LT module.

[RS_LT_00017] ⌈

Description:	Each log and trace message may contain a timestamp, which shall be added to the message during reception of the message in the LT module.
Rationale:	For comparing and analyzing complex systems a timestamp is important. The time shall be in a defined granularity.
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.2.5 A global message counter shall be implemented, to detect messages loss.

[RS_LT_00018] ⌈

Description:	Every time a message from an Application is received by the LT module or other software module, it shall increment its global messages counter and set the counter in the message.
Rationale:	It is important to know if a message is lost. A receiving external client may then detect if messages are lost (e.g.because of a buffer overflow in LT or network loss).
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.2.6 For each log message, a log level shall be provided.

[RS_LT_00019] ⌈

Description:	Log levels shall be for example error, fatal, debug, information or verbose.
Rationale:	Standard for logging tools
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260, RS_Main_00011*)

4.2.2.7 The log and trace message shall contain a parameter, which represents the source of the log and trace message.

[RS_LT_00020] ⌈

Description:	The log and trace message may contain a parameter, which represents the source of the log and trace message.
Rationale:	The source can be identified by the Session ID of the port interface (port defined argument value) which was used to send the message from Application to LT.
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260, RS_Main_00011*)

4.2.2.8 There shall be a logical grouping for log messages by using different identifiers.

[RS_LT_00021] ⌈

Description:	There shall be additional parameters describing the group of the log and trace message.
Rationale:	Identifying some logically groups of log and trace to define some filtering action on the external client.
AppliesTo:	CP,AP
Use Case:	–

▽

△

Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.2.9 Each ECU shall have its unique ECU ID.

[RS_LT_00022] ⌈

Description:	Each ECU may have its unique ECU ID, which may be a parameter of each log and trace message.
Rationale:	An external client receives tracing from different ECU's so it's important to know the source.
AppliesTo:	CP,AP
Use Case:	Distinguish tracing from different ECU's
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00500*)

4.2.2.10 The payload shall transport the parameters of a log and trace message.

[RS_LT_00023] ⌈

Description:	In the payload the parameters of each log and trace message from the Application shall be transported. It may be possible to transport more than one parameter.
Rationale:	–
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.2.11 It shall be possible to transmit the parameters in a raw format.

[RS_LT_00024] ⌈

Description:	In most cases the parameter shall be transmitted in the raw format. This means that a separate file shall be provided containing the information about the transmitted parameters.
Rationale:	The non-verbose mode is for saving memory and bandwidth, because on the ECU only the ID's of the log and trace messages shall be stored.
AppliesTo:	CP,AP
Use Case:	Interpreting binary data
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.2.12 There shall be the possibility to transmit the parameters with additional information about themselves (self-description).

[RS_LT_00056] ⌈

Description:	The log and trace messages shall contain a description of the parameters in verbose mode. No additional description file needs to be provided in this case.
Rationale:	In the verbose mode all parameters of the payload contain a self-description of their own type (e.g. integer/float, bit length) and some additional information (e.g. variable name, unit, etc).
AppliesTo:	CP,AP
Use Case:	Interpreting binary data
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.2.13 It shall be possible to transmit ASCII text in log or trace messages.

[RS_LT_00025] ⌈

Description:	Because log and trace messages often consists of texts, there shall be the possibility to transmit text strings. It may be possible to replace the given text by an ID which may be replaced by the text in the external client (Compare verbose - non-verbose mode).

▽

△

Rationale:	In small ECU strings can not be stored because of their memory usage. For this reasons these strings may be replaced by a message ID.
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.2.14 The data in non-verbose mode shall be described by an extra file

[RS_LT_00026] ⌈

Description:	The data in non-verbose mode need to be described by an extra file. This file may be in a ARXML format.
Rationale:	In small ECUs memory is very limited, so temporarily buffered data shall be very small.
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.2.15 Each message in non-verbose mode shall have a unique Message ID significant for identifying the source of the tracing.

[RS_LT_00027] ⌈

Description:	Each message may have a unique identifier significant for identifying the source of the tracing. The ID of a specific message shall be permanent over the developing phase and shall be unique for a given ECU.
Rationale:	Easy identifying the right information in the log and trace message is very important. Unique message IDs helps doing this.
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00500*)

4.2.3 Transport interfaces

4.2.3.1 Generic

4.2.3.1.1 A control message shall be implemented to permit the external client to evaluate the round trip time.

[RS_LT_00028] ⌈

Description:	A control message shall be implemented to permit the external client to evaluate the round trip time. The external client shall use this message to synchronize the individual log and trace messages from the different ECUs.
Rationale:	The log and trace time information may enable a user to understand what happens if there is a correlation among the recorded events. It is important to be able to refer all the events logged to an absolute time scale.
AppliesTo:	CP,AP
Use Case:	• An external client is able to synchronize packet reception from different sources • A round trip time calculation for each ECU is done • In each message the ECUs local time is transmitted • For every message from several ECUs a global time line can be calculated
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260)*

4.2.3.1.2 A protection against unauthorized access in production phase shall be provided.

[RS_LT_00029] ⌈

Description:	In production phase, the communication module (see RS_LT_00034) shall be disabled by default. In production phase LT shall use the security mechanisms provided by DCM to handle the access to the log and trace messages. A mechanism shall be implemented to enable the communication module for testing phase permanently.
Rationale:	Only authorized accesses to LT shall be allowed during production phase to avoid misuse.
AppliesTo:	CP,AP
Use Case:	Avoid LT misuse
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00170, RS_Main_00260)*

4.2.3.1.3 Logging shall be able to monitor and shape the amount of LT log and trace events.

[RS_LT_00030] ⌈

Description:	LT may shape the traffic bandwidth depending on which transport interface is used for communication. The traffic shaping shall be done at runtime.
Rationale:	Since LT shall be used also in production phase, it is important that the traffic produced by the LT component will not cause an overload of the network and the ECU resources.
AppliesTo:	CP,AP
Use Case:	• Bandwidth occupation has to be controlled • Bandwidth configuration has to be permitted
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.2.3.1.4 The LT shall be configurable at runtime.

[RS_LT_00031] ⌈

Description:	LT shall be configurable at runtime. LT shall have for this propose runtime configurable variables which are initialized with the corresponding configuration parameters. It shall be possible to query the LT runtime configuration. The configuration variables shall be writeable and readable over with some control .mechanism triggered remotely.
Rationale:	To adapt LT behavior to ECU and network resources it is important to have the capability of configuring it.
AppliesTo:	CP,AP
Use Case:	• Changing default levels with storing in NVRAM • Query LT state
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260)*

4.2.3.1.5 A protocol shall be implemented to be able to set and query the trace status and log levels of log and trace sources of each ECU.

[RS_LT_00032] ⌈

Description:	LT shall support control messages to set the trace status and log level of each source of each ECU.
Rationale:	–
AppliesTo:	CP,AP
Use Case:	• Enabling/disabling trace and log messages at different levels. • Set the level of the log messages generated by different sources of each ECU.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.3.1.6 A list of all log and trace sources of an ECU shall be accessible from the external client.

[RS_LT_00033] ⌈

Description:	It shall be possible to get a list of all Application IDs and context IDs with corresponding trace status and log levels which are registered to the LT module.Each Application or BSW module which wants to produce trace or log messages shall register to the LT module.
Rationale:	Users need to know what information can be traced or logged in order to monitor ECU activity.
AppliesTo:	CP,AP
Use Case:	• Log messages from different ECU's can be merged to understand relationship of behavior from distributed Applications. • Set the level of the tracing generated by different sources of each ECU.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.3.2 Communication interface

4.2.3.2.1 LT shall support a generic API for communicating over a LT communication module.

[RS_LT_00034] ⌈

Description:	At debugging phase a communication interface with a high bandwidth is needed. This communication interface is out of scope of the AUTOSAR specification. To use this communication interface with LT a generic API shall be defined. The user shall be able to implement a board specific communication module to communicate over a board specific communication interface.
Rationale:	–
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260, RS_Main_00011*)

4.2.3.3 DCM transport interface

4.2.3.3.1 The DCM shall provide an interface for LT to transport log and trace messages over a diagnostic session.

[RS_LT_00035] ⌈

Description:	DCM shall provide an interface for LT to send and receive data over the diagnostic service. Log and trace messages are sent over this service and control messages for LT are received. Because log and trace messages are event triggered and the storage on the ECU is limited, these messages shall be sent when they occur. As diagnostic services can only send a small amount of data, the amount of data shall be limited, see RS_LT_00030.
Rationale:	LT needs an interface to send log and trace message out of the ECU. DCM provides a bus independent access to the ECU over standardized diagnostics. This is available during production phase and provides a secured session control. As diagnostic services can only send a small amount of data, the amount of data shall be limited, see RS_LT_00030. For higher bandwidth the communication module shall be used, see RS_LT_00034.
AppliesTo:	CP

▽

△

Use Case:	• Transmitting log and trace message during a diagnostic session • Advanced Diagnostic Tracing, optional over telematic services
Dependencies:	RS_LT_00030, RS_LT_00034
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00420*, *RS_Main_00011*)

4.2.4 Operational function

4.2.4.1 Initialization and shutdown

4.2.4.1.1 The LT shall provide a buffer for storing log and trace messages before initialization

[RS_LT_00036] ⌈

Description:	The LT may provide a buffer for storing log and trace messages. This may be a standard C-initialized buffer which shall be read out after init of LT.
Rationale:	This requirement shall guarantee that the LT module can provide its services to other BSW modules also before the complete initialization of all BSW modules.
AppliesTo:	CP,AP
Use Case:	Logging is available at startup phase
Dependencies:	[RS_LT_00052]
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.4.2 Normal operation

4.2.4.2.1 There shall be a buffer to store log and trace message locally.

[RS_LT_00037] ⌈

Description:	If the communication module is not enabled or no external client for dumping log and trace message is connected, log and trace messages shall be stored locally in a memory buffer. The size of the buffer shall be set at configuration time.
Rationale:	To prevent a lack of communication e.g. at startup or if no external client is connected, the LT shall store some messages to be dumped as soon as communication channel is available.

▽

△

AppliesTo:	CP,AP
Use Case:	• ECU startup • No communication available
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*)

4.2.4.2.2 A mechanism shall be implemented to be able to set the trace status and log levels of registered Application IDs and context IDs of each Application.

[RS_LT_00038] ⌈

Description:	LT shall permit to turn off or on traces or to set log levels for registered Application IDs and context IDs at runtime.
Rationale:	To gain efficiency in bandwidth and ECU resource usage a mechanism shall be provided to tune the transmission from Applications to LT. The LT module shall provide some control variables which represent the actual log levels and trace status of the Applications and contexts.
AppliesTo:	CP,AP
Use Case:	Enabling/disabling trace status and log levels.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.4.2.3 The LT shall provide the possibility to store configuration data in a persistent way.

[RS_LT_00039] ⌈

Description:	Log and Trace configuration shall be stored persistent.
Rationale:	The LT can be configured at runtime. L&T configuration data has to be stored to guarantee that at every startup the configuration would be the same.
AppliesTo:	CP,AP
Use Case:	—
Dependencies:	–

▽

△

Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*, *RS_Main_00440*)

4.2.4.2.4 The LT component shall be able to filter log and trace messages.

[RS_LT_00040] ⌈

Description:	In order to adapt network traffic to bandwidth availability, the LT component may be able to filter log and trace messages. A policy shall be defined to distinguish messages by different log and trace levels. The configuration of the filter level may be different for storing log and trace messages temporarily and communication over DCM or the LT communication module.
Rationale:	–
AppliesTo:	CP,AP
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.2.4.2.5 AUTOSAR Log and Trace shall support harmonized logging.

[RS_LT_00058] ⌈

Description:	AUTOSAR shall define globally usable log messages with a unique ID that can be used by all software parts (system software, AUTOSAR functional clusters and other basic software, application). This shall be enabled by using a mandatory Extension Header with pre-defined message IDs and context IDs that are reserved for each AUTOSAR functional cluster.
Rationale:	To ensure that similar things are logged in a similar way.
AppliesTo:	CP,AP
Use Case:	The application and e.g. UCM might both report a communication problem using ara::com.
Dependencies:	–
Supporting Material:	–

⌋(*RS_Main_00260*, *RS_Main_00011*)

4.3 Non-Functional Requirements

4.3.1 [RS_LT_00041] LT shall be a central software component for the log and trace functionality.

[RS_LT_00041] LT shall be a central software component for the log and trace functionality. ⌈

Description:	Only one SW module per ECU shall gather the log and trace messages centrally.
Rationale:	-
AppliesTo:	CP,AP
Use Case:	-
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00260, RS_Main_00011)*

4.3.2 [RS_LT_00042] The Log and trace SW component shall be part of the system during production phase.

[RS_LT_00042] The Log and trace SW component shall be part of the system during production phase. ⌈

Description:	The Log and trace SW component shall be part of the system during production phase. It is needed for use cases like test drives and remote diagnosis.
Rationale:	-
AppliesTo:	CP,AP
Use Case:	• Test drives • Advanced remote diagnosis
Dependencies:	–
Supporting Material:	–

⌋*(RS_Main_00491)*

4.3.3 [RS_LT_00059] LT shall provide an interface for trace points.

[RS_LT_00059]{DRAFT} **LT shall provide an interface for trace points** ⌈

Description:	Logging shall provide an interface for trace points in the source code on application software and system software level.
Rationale:	In SW development and integration it is a necessary and important to know the timing behaviour. The possibility to define and collect events emitted by trace points is an essential feature to determine this timing behaviour.
AppliesTo:	AP
Use Case:	• Trace the dynamic status of System-Level Software and of Application-Level Software • Have API calls for trace points in source code
Dependencies:	-
Supporting Material:	-

⌋(*RS_Main_01026*)

4.3.4 [RS_LT_00060] LT shall send modeled trace messages

[RS_LT_00060]{DRAFT} **LT shall send modeled trace messages** ⌈

Description:	Trace messages shall be sent via modeled messages.
Rationale:	In distributed systems it is often necessary to determine the timing behaviour of a distributed function over several ECUs. Additionally it is important to be able to compare different implementations. With modeled messages, a comparable format can be used and trace information can be sent with less overhead (no static, descriptive information).
AppliesTo:	AP
Use Case:	Efficient and standardized tracing format
Dependencies:	RS_LT_00002
Supporting Material:	-

⌋(*RS_Main_01026*)

4.3.5 [RS_LT_00061] Tracing shall be configurable at compile time

[RS_LT_00061]{DRAFT} **Tracing shall be configurable at compile time** ⌈

Description:	It shall be possible to enable and disable tracing for the whole system and for selected applications per configuration.
Rationale:	Tracing is usually only needed during development and integration. So it is necessary to enable the disabling of it. During development, only several applications might be interesting.
AppliesTo:	AP
Use Case:	Select which System Level and Application to trace
Dependencies:	RS_LT_00010, RS_LT_00005
Supporting Material:	-

⌋*(RS_Main_01026)*

4.3.6 [RS_LT_00062] LT shall be configurable to use ARTI to process the tracing information

[RS_LT_00062]{DRAFT} **LT shall be configurable to use ARTI to process the tracing information** ⌈

Description:	For modeled tracing messages, it shall be configurable to define ARTI to process the tracing information.
Rationale:	By using the ARTI Trace API, the overhead of the tracing can be minimized. Since OS is using ARTI as well for tracing, it is possible to trace OS, System and Application Levels at the same time.
AppliesTo:	AP
Use Case:	Rely on the timing of the whole platform
Dependencies:	RS_LT_00051
Supporting Material:	-

⌋*(RS_Main_01026)*

5 Requirements Tracing

The following table references the features specified in [6] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_Main_00010]	Safety Mechanisms	[RS_LT_00013] [RS_LT_00014]
[RS_Main_00011]	Mechanisms for Reliable Systems	[RS_LT_00001] [RS_LT_00003] [RS_LT_00004] [RS_LT_00005] [RS_LT_00006] [RS_LT_00007] [RS_LT_00008] [RS_LT_00009] [RS_LT_00010] [RS_LT_00011] [RS_LT_00012] [RS_LT_00013] [RS_LT_00014] [RS_LT_00016] [RS_LT_00017] [RS_LT_00018] [RS_LT_00019] [RS_LT_00020] [RS_LT_00021] [RS_LT_00023] [RS_LT_00024] [RS_LT_00025] [RS_LT_00026] [RS_LT_00030] [RS_LT_00032] [RS_LT_00033] [RS_LT_00034] [RS_LT_00035] [RS_LT_00036] [RS_LT_00038] [RS_LT_00039] [RS_LT_00040] [RS_LT_00041] [RS_LT_00056] [RS_LT_00058]
[RS_Main_00060]	Standardized Application Communication Interface	[RS_LT_00043] [RS_LT_00044] [RS_LT_00045] [RS_LT_00046] [RS_LT_00051] [RS_LT_00052]
[RS_Main_00100]	AUTOSAR shall provide standardized Basic Software	[RS_LT_00013] [RS_LT_00014]
[RS_Main_00170]	AUTOSAR shall provide secure access to ECU data and services	[RS_LT_00029]
[RS_Main_00260]	Runtime Diagnostics Means	[RS_LT_00001] [RS_LT_00002] [RS_LT_00003] [RS_LT_00004] [RS_LT_00005] [RS_LT_00006] [RS_LT_00007] [RS_LT_00008] [RS_LT_00009] [RS_LT_00010] [RS_LT_00011] [RS_LT_00012] [RS_LT_00013] [RS_LT_00014] [RS_LT_00016] [RS_LT_00017] [RS_LT_00018] [RS_LT_00019] [RS_LT_00020] [RS_LT_00021] [RS_LT_00023] [RS_LT_00024] [RS_LT_00025] [RS_LT_00026] [RS_LT_00028] [RS_LT_00029] [RS_LT_00030] [RS_LT_00031] [RS_LT_00032] [RS_LT_00033] [RS_LT_00034] [RS_LT_00035] [RS_LT_00036] [RS_LT_00037] [RS_LT_00038] [RS_LT_00039] [RS_LT_00040] [RS_LT_00041] [RS_LT_00056] [RS_LT_00058]
[RS_Main_00400]	AUTOSAR shall provide a layered software architecture	[RS_LT_00047] [RS_LT_00048] [RS_LT_00049] [RS_LT_00050] [RS_LT_00057]
[RS_Main_00420]	AUTOSAR shall use established software standards and consolidate de-facto standards for basic software functionality	[RS_LT_00002] [RS_LT_00003] [RS_LT_00035]
[RS_Main_00430]	AUTOSAR shall support established automotive communication standards	[RS_LT_00016]
[RS_Main_00440]	AUTOSAR shall standardize access to non-volatile memory	[RS_LT_00039]
[RS_Main_00491]	Function Monitoring	[RS_LT_00042]
[RS_Main_00500]	AUTOSAR shall provide naming conventions	[RS_LT_00022] [RS_LT_00027]
[RS_Main_01026]	AUTOSAR shall support tracing and profiling on the target and onboard	[RS_LT_00059] [RS_LT_00060] [RS_LT_00061] [RS_LT_00062]

Table 5.1: RequirementsTracing

6 References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[4] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

[5] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[6] Main Requirements
AUTOSAR_FO_RS_Main

A 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 constraints and specification items do not appear as hyperlinks in the document.

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

A.1.1 Added Requirements in R23-11

Number	Heading
[RS_LT_00001]	
[RS_LT_00002]	
[RS_LT_00003]	
[RS_LT_00004]	
[RS_LT_00005]	
[RS_LT_00006]	
[RS_LT_00007]	
[RS_LT_00008]	
[RS_LT_00009]	
[RS_LT_00010]	
[RS_LT_00011]	
[RS_LT_00012]	
[RS_LT_00013]	
[RS_LT_00014]	
[RS_LT_00016]	
[RS_LT_00017]	
[RS_LT_00018]	
[RS_LT_00019]	
[RS_LT_00020]	
[RS_LT_00021]	
[RS_LT_00022]	
[RS_LT_00023]	
[RS_LT_00024]	
[RS_LT_00025]	
[RS_LT_00026]	
[RS_LT_00027]	
[RS_LT_00028]	

△

Number	Heading
[RS_LT_00029]	
[RS_LT_00030]	
[RS_LT_00031]	
[RS_LT_00032]	
[RS_LT_00033]	
[RS_LT_00034]	
[RS_LT_00035]	
[RS_LT_00036]	
[RS_LT_00037]	
[RS_LT_00038]	
[RS_LT_00039]	
[RS_LT_00040]	
[RS_LT_00041]	LT shall be a central software component for the log and trace functionality.
[RS_LT_00042]	The Log and trace SW component shall be part of the system during production phase.
[RS_LT_00043]	
[RS_LT_00044]	
[RS_LT_00045]	
[RS_LT_00046]	
[RS_LT_00047]	
[RS_LT_00048]	
[RS_LT_00049]	
[RS_LT_00050]	
[RS_LT_00051]	
[RS_LT_00052]	
[RS_LT_00056]	
[RS_LT_00057]	
[RS_LT_00058]	
[RS_LT_00059]	LT shall provide an interface for trace points
[RS_LT_00060]	LT shall send modeled trace messages
[RS_LT_00061]	Tracing shall be configurable at compile time
[RS_LT_00062]	LT shall be configurable to use ARTI to process the tracing information

Table A.1: Added Requirements in R23-11

A.1.2 Changed Requirements in R23-11

none

A.1.3 Deleted Requirements in R23-11

none