New DLT Requirements

New DLT Requirements

Comments by Simon Brandner are marked as S.B.

```
S.B.: In general, it would be nice to have a concrete porposal in the comment section,
where a requirement should be fulfilled/which component is affected by the change.


This could be:
* protocol/data format
* Dlt-daemon
* dlt-library
* additional application (on ECU/off board)
* DLT viewer
* system requirement
```

Meeting Minutes

Meeting minutes are tracked withing Genivi, asl the telco is also hosted by Genivi:

https://collab.genivi.org/wiki/display/genivi/New+DLT+requirements+-+Meeting+20130606

https://collab.genivi.org/wiki/pages/viewpage.action?pageId=29196820

Multi-Node Logging

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]

SW-DLT-001	In a multi-node system, log information from all nodes are forwarded via a physical port on one of the nodes to the log viewer (DLT-Client Application). *The DLT Daemon on that node provides a gateway for DLT-Daemons on external nodes to send application logs and to receive control request messages.*	In a head-unit with more than one processor connected to each other, the interface for debugging is available on only one processor. A processor can be a GENIVI(Linux) or AUTOSAR system and are interconnected via Ethernet, UART etc. Entire head unit shall be treated as one ECU.	• Dlt-daemon • Additional library	P1	Agreed	S.B. Can't this be implemented by the usage of port forwarding of a firewall? Definition of "log viewer"? -> DLT-Viewer/dlt-client, including data loggers? B.Kebianyor[28.05.2013]:Port forwarding of a firewall will only work with TCP/IP and not possible for non TCP/IP environments; and therefore a limitation. The Port forwarding assumes having Linux on all ECUs, but will not be feasible if a GENIVI Node and AUTOSAR are connected using a different protocol. A.W.: It think you are talking about CPUs or Microcontrollers, not ECU. Entire headunit is ECU. The problem we today have, that the different CPUs have different timing requirements and different logging interfaces. The AUTOSAR CPU is needed quite early and contains also the Supervising functinality of the AUTOSAR ECU, thats why we need today two different logging interfaces. GENIVI is only running on one ECU. For two CPUs running two GENIVI instances in parallel i would also prefer to use two seperated TCP ports. I think both can also have the same ECU Id if needed. If you really need to use a single Interface or TCP Port, you need to take care, to use APP Ids only on one of both CPUs. We need then a seperated multiplexer process, which then connects two bot dlt-daemons, or in the dlt-daemon an interface to the second CPU (but this must be designed as generic interface because we do not know which kind of interface is used between both ECUs). Another solution is to have only one dlt-daemon running and provide e.g. TCP interface to each application running on another CPU. S.B.(2): I think such a "multiplexer process" would not require changes in the standard daemon/library, when it would implement like a proxy: a dlt-client (have a look at dlt-receive) process, which just forwards messages via the user library to the main-daemon. In case there is no tcp, you would have to implement an alternative dlt-client, which can connect to other interface types. And: Please modify the term "ECU" in the Use case into "processors" or "independent operating systems", to make it clear, that they the intention is that they are seen as one ECU from OEM perspective.
SW-DLT-002	In a multi-node system, commands and control messages are sent to applications on different nodes from the log viewer via a physical port on one node.	A head-unit is built into a complex system. Not all of the stimuli that are available in this complex system are available at a developers desk. Therefore a control channel to the head-unit is mandatory. The control channel allows to change the behaviour of the head-unit. The control channel can also be used to access information of components. E.g. the channel can be used to display resource usage information. With a head unit that is partitioned into different nodes, the control of the behavior of applications on each node is needed.		P1	Agreed	S.B. Can't this be implemented by the usage of port forwarding of a firewall? B.Kebianyor[28.05.2013]:See comment above. A.W.: See above we have two different interfaces for each CPU, so we decide the control channel to be used by the interface. With two GENIVI CPUs you should use two different ports. If you really want a single interface for two CPUs you will need a registry in ther Multiplexer, which knows which APP Id runs on which CPU. Then you can route the control messages to the right CPU. You can build the registry by looking at the application registration messages.

SW-DLT-003	In a multi-node system, DLT on each node has its own default configuration file.	When trouble shooting activities during development, and during testing, the behavior of the system can be understood if the behaviour of single components can be understood. This can be achieved by providing special output which lets a user understand what the component is doing. For better analysis of the problem, node specific information included in the output messages is of importance. This can be considered as part of the configuration.		P1	Agreed	A.W.: What must be changed in current implementation? I do not understand the needed change.

Log Storage

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]
SW-DLT-004	The log information are saved in the activated sets of files.	1. In production system , since client will not be available the log messages shall be stored into external storage devices for debugging applicaitons 2. Since the devices also have size limitations, its required to store log messages to multiple devices (FLASH memory, USB, SD Card etc). Thereby its possible to dump critical log messages and non critical log messages to different devices		P1	Agreed	A.W.: Please describe the Use Cases for this feature. The purpose of DLT should be used for developer support and debugging. The current log level concept is already quiet complex to be understood. If you have different log levels for different file sets or interfaces, this could be more complex to be understood. DLT should not cost to much CPU performance. Complex filtering in SW could be problem. Filtering should only performed directly in the application, to not produce extra traffic. The overall topic looks quiet complex. Perhaps it can be separated from the DLT daemon implementation into an extra process.
SW-DLT-005	i) The log information are filtered based on the configured application identifier and log level per set.			P1	Agreed	S.B. the actual library already does pre-filtering according to the contexts set log level- B.Kebianyor[28.05.2013]: To be reused.
SW-DLT-005	ii) Log files are stored at configured locations per set.			P1	Agreed	
SW-DLT-006	iii) The size of log files do not exceed the configured maximum file size per set.			P1	Agreed	S.B. Is implemented for one log file type B.Kebianyor[28.05.2013]: To be reused and extended to support more than one file.
SW_DLT-007	iv) The number of files per set do not exceed the configured maximum.			P1	Agreed	S.B. Is implemented for one log file type. B.Kebianyor[28.05.2013]: To be reused and extended to support more than one file.
SW-DLT-008	v) The oldest log file of a set is replaced to save more logs if the maximum size of log files is reached.			P1	Agreed	S.B. Is implemented for one log file type B.Kebianyor[28.05.2013]: Current implemetation has only one file and so there is no oldest log file.Plan to extended for support of more than one file. S.B.(2); Indeed: Is implemented for one log file type! Just set the max log file size to 1MB, max Log size to 1 MB, and it will create 4 log files with 1MB each. The oldest will be deleted, when the latest is "full" and a 5ths is to be created.
SW-DLT-009	The activation and deactivation of a set of log files is triggered by a command.			P1	Agreed	
SW-DLT-010	The activation of a set of log files is possible during system startup.			P1	Agreed	S.B. There is an internal interface in the daemon "dlt_daemon_process_user_message_log_mode" to set that mode. It could be extended to a library interface to enable file logging. It should then just be necessary to write a helper tool which gets an injection as a trigger to call this function. B.Kebianyor[28.05.2013]: Investigate this if it works fine. Plan to reuse existing implementation/interfaces where applicable.
SW-DLT-011	The activation of a set of log files is possible when a medium is inserted.			P1	Agreed	S.B. Such a system specific functionality should definitely be separately developed in a helper function, not in the daemon.
SW-DLT-012	The sets of log files are stored on the target system.			P1	Agreed	

SW-DLT-013	It is possible to transfer the sets of log files from the target system to the host system on request.			P1	Agreed	S.B: Therefore the file system transfer exists yet (have a look into dlt-system). Nice would be to have a dlt-reader, which decodes the payload and sends it as standard messages to avoid overhead. I think this would then modify the timestamps, so it is not universally applicable.
SW-DLT-014	It is possible to view the transferred sets of log files using the log viewer.			P1	Agreed	S.B. this is part of the normal file transfer feature and already implemented.

Log Storage – Configuration

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]
SW-DLT-015	Different configurations files exist for different sets of log files.			P1	Agreed	S.B.: What is the motivation for this? Use case? A.W.: The complexity of DLT should be small? Use Cases?

SW-DLT-016	The configuration for sets of log files is retrieved from a file or several files.			P1	Agreed	B.K: Sample Configuration File
						[OFFLINE LOG]
						LogAppName=IPOD
						ContextName=IPCI,IPCA
						LogLevel=DLT_LOG_WARNING
						File=${MOUNT_LOCATION}/ipod.log
						FileSize=10000
						NOFiles=2
						
						[OFFLINE LOG]
						LogAppName=Test
						# Here .* refers to all contexts
						ContextName=.*
						LogLevel=DLT_LOG_ERROR
						File=${MOUNT_LOCATION}/Test.log
						FileSize=20000
						NOFiles=3

SW-DLT-017	The configuration files for saving sets of log files shall be created on requests.			P1	Agreed	
SW-DLT-018	The configuration files for saving sets of log files shall comprise: Sets of Application Identifiers and Log Levels Location per sets of log files Maximum file size per set of log files Number of files per set of log files Write mode Flag: Sets of files can be overwritten or flushed when the configured maximum file size is reached. Overwrite: Bytes per bytes overwritten in a file Flush: Complete file is flushed before writing new data when maximum file size is reached.			P1	Agreed	

Lock and Seed Support

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]
SW-DLT-***	DLT communication can be locked and unlocked based on the trigger from Diagnosis component during production phase.	In the production phase, the software is locked from using DLT communication for security reasons. It is possible to unlock the target using a DLT commands with unlock keys which will be known only to Project Integrators. Diagnosis component will trigger the lock/unlock states to DLT.		P2	Rejected (Use provided DLT API and handle in a different application)	Why is it necessary to be a DLT standard functionality? Wouldn't DLT just be the channel, and the actual locking be handled by an external application, which uses the user library to receive the trigger? I think we should avoid blowing the DLT (daemon) up with features which are not logging related. To call such an application, the injection interface is provided. Another requirement cares about handling of large messages, which may contain an authentication key. I think most of that functionality can be implemented in an external process, which provides a command interface via injection, if DLT is the interface of choice. A.W.: API already implemented in DLT library. Better disable TCP Port in firewall. Requirement must be solved outside DLT implementation.

DLT supports logging in Block-Mode and Non-Block Mode

ID	Description	Use Case	Affected DLT Components	Priority	Status [Agreed, In Discussion, Rejected}	Comments [Your Name and Date: Your Comment]
SW-DLT-019	Block Mode: DLT blocks an application call to log information when the DLT-FIFO is full, until space is available to log data again.	For trouble shooting during developer and production, user or developers need all logged and traced data to analyze and reproduce error scenario in detail.		P1	In Discussion	S.B.: Blocking an application by a function call was a main point which is tried to be avoided in DLT. In general, it sounds very dangerous. I doubt anyone would like to have such a system in the field. In general, a use case description would simplify it to understand the requirement and show a maybe simpler/less invasive solution. BTW: Such a switching could be implemented in a controller application - be it a process communicating over the dlt library with the daemon to avoid adding too system specific logic into the daemon. I think it is better to inform logging applications, if a message can be guaranteed to be sent out without loss. E.g. by checking the library buffer state. With little modifications in the daemon, the daemon could be implemented blocking (not throwing away messages when buffer is full), but the libraries are not meant to work blocking in general. Anyway: if a block mode is necessary, additional interface should be marked with a "_blocking" indication. The default function calls (like the DLT_LOG macro) should never become blocking. Instead, you could add a DLT_LOG_BLOCKING macro. To the Rationale: "User or developers need all logged and traced data to analyze and reproduce error scenario in detail." In general, DLT messages should not become lost, when a logger is active. Hence, then the blocking mode is not necessary. And if it is necessary, the blocking would change the system behavior so far, that it would be more likely to help less in analysis then compared to some lost messages. Instead, you could dynamically raise buffer sizes of indiviual applications, which are likely to loose data (this is not implemented as such, as too high buffer sizes of dlt library processes could lead to significant amount of memory usage.) A.W.: Blocking mode is against current philosophy of DLT. This would change strongly the runtime behaviour of teh applications. Nevertheless this can be implemented as an option.
SW-DLT-020	Non-Block Mode: DLT does not block an application call to log information. Some log data are overwritten or discarded when buffer is full.				In Discussion	
SW-DLT-021	Non-Block Mode: In this mode, DLT sends a message once to clients when buffer is full and keeps track of total number of bytes of discarded messages. Once data can be written to buffer again, DLT sends message to clients with total number bytes for messages discarded.	Information is useful for analyzing specific sequences from logged data, if some data has been lost in between or not. Otherwise analyzing is difficult.	* DLT-viewer * DLT-daemon	P1	Agreed	S.B. This is the actual implementation state. Or is it still too blocking because of semaphores? This can be improved in current implementation. Currently only buffer overflows are shown.
SW-DLT-022	Switching between Block and Non-Block: DLT clients can enable or disable the Block Mode Support in DLT. Disabling the Block Mode means activating Non-Block Mode.	Synchronize logging state between the Clients and DLT	* DLT-viewer * DLT-daemon	P1	In Discussion	
SW-DLT-023	Block Mode Status: DLT users can request for the status of the logging mode from the DLT (if block mode is enable or disabled)	Synchronize settings between clients and DLT.	* DLT-viewer * DLT-daemon	P1	In Discussion	
SW-DLT-024	Automatic Switch to Non-Block Mode: DLT switches automatically to Non-Block mode if no client is connected.	Block mode should be set only when host application is available. Otherwise applications will block for write which results in complete system freeze.	* DLT-daemon	P1	In Discussion	

DLT supports appending Timestamp to log information.

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]
SW-DLT-025	Log information or data shall be appended with Timestamps. Different Timestamp modes to be supported include: i) System Ticks (Ticks of System Clock - Time since System Boot) The timestamp value is a system Tick, at the instant when the log information is logged into DLT buffer. ii) RTC Timestamp (Real Time Clock) The timestamp value corresponds to a RTC value, at the instant when the log information is logged into DLT buffer. iii) PC Timestamp The timestamp value corresponds to a timestamp value, when the log message arrives the DLT-Client. ***Rationale: To analyze timing related issues timestamp should be activated along with the application context and log levels. Based on the timestamp it will be possible to identify the time of execution of the specific code section.***		DLT-user library <br\>DLt-daemon	P1	Agreed (i & iii) <br\> In Discussion (ii)	S.B. Does DLT not fulfill this requirement yet? Please clarify the difference between the current Timestamps and the requirement. What is the difference between PC and RTC timestamp? Where can I find the definition of "system tick"? A.W.: i) and iii) already part of DLT standard. ii) can be implemented by extra DLT message containing the current RTC of ECU. We must fullfill the AUTOSAR DLT standard.
SW-DLT-026	DLT Clients can select a preferred timestamp mode from the different supported timestamp modes.		DLT-daemon, <br\> DLT-user library <br\> DLT-viewer	P1	Agreed	

Application Context Groups Support

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]
SW-DLT-027	DLT supports enabling groups of application contexts using a single command from a client. ***Rationale: Used from Viewer to create configuration for log storage.***	For creating configuration file for offline logging by simply selecting groups of application contexts and log levels	* DLT-Viewer	P2	Agreed	A.W.: This is a DLT Viewer issue. How looks the concept for grouping, by which criteria. Do not understand whats a benefit for this. Why not creating search functions etc.?

Large Message Support

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]

SW-DLT-028	DLT supports sending data of size between 241 bytes and 128Kbytes from DLT-clients to DLT-applications.	Large message injection is used in simulation clients based on proprietary MST/CCA protocol communication (project specific). This allows user to exchange larger data between Log Viewer & Log Application. Eg. In our project a Remote client in NAVI appliaction exchanges data with NAVI application on target, Message Sequence Tester used in testing speech, Screen-dump. More and more development tools are being developed based on large message communication.		P2		S.B. What means: "Used for simulating client."? In general a more flexible interface could be an improvement over the existing interface. Regards backward compatibility in a new implementation, as well as used resource! A.W.: Segmentation needed and special flow control needed, to do not block smaller messages. Look at Filetransfer where this is already implemented. We developed similar thing for big MOST messages trace.

Dynamic Switching between Supported Physical devices (I/O)

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]
SW-DLT-029	DLT supports switching its connection to a client between different I/O devices at runtime of DLT-Daemon.	Flexibility to change the active IO channel is required without restarting the dlt-daemon.	DLT-daemon <br\> DLT-viewer	P2	Agreed	S.B. Define Runtime: runtime of ECU, or runtime of DLT-daemon? I think the daemon would require rework, to handle that scenario without restart. Should not be impossible to handle. If the clients are always connected via TCP/IP, the interface change should already be possible - a disconnection of one client and connection of the other is no big problem. Only requirement: dlt-daemon is listening to the corresponding interfaces of the underlying I/O devices /network interfaces. A.W.: Use Cases for this? Which I/O devices should be supported?
SW-DLT-030	Switching to a selected device shall be triggered from the DLT-client. Switching is possible while application logs are sent to the DLT-client and also when no application logs are sent to the DLT-client.		DLT-daemon <br\> DLT-viewer	P2	Agreed	S.B. Use case required. Can this be implemented in an external tool?

System Monitoring

ID	Description	Use Case	Affected DLT Component	Priority	Status [Agreed, In Discussion, Rejected]	Comments [Your Name and Date: Your Comment]

SW-DLT-031	DLT supports capturing and displaying the Memory Usage, CPU Load, Tasks and Threads information for the system.<br\>	These features are useful for analysing performance related issues. In the current platforms this is part of Performance Analyser or System Information tools implemented for projects.	DLT-system	P2	Agreed	S.B. This feature is definitely meant to be implemented in an helper application. If it only collects and logs only standard Linux/Genivi information, then it could become part of the dlt distribution as an extra process. Have a look at dlt-system as an example Bewoayia Kebianyor [30.05.2013]: Agreed. This could be a separate process. A.W.: Already implemented in dlt-system and dlt system Plugin of DLT Viewer.
