

DLT Viewer User Manual

Gernot Wirschal <Gernot.Wirschal@bmw.de >

August 12, 2019

(Alexander Wenzel, Simon Brandner, Lassi Marttala, Sven Hassler)

Changelog:

v1.0.0, July 2019: adjust for 2.19.0 Release

v0.0.7, February 2019: added UDP reception

v0.0.6, September 2018: added loglevels, complete missing descriptions, bugfixing

v0.0.5, June 2018: converted to L^AT_EX, extended documentation, based on version 2.19.0 Release

v0.0.4, February 2016: Updates and improvements for current DLT Viewer state

v0.0.3, April 2013: Added Filetransfer command line description

v0.0.2, December 2012: Converted to asciidoc



Contents

1 Introduction

1.1 Purpose

The DLT Viewer tool can decode, view and store DLT messages generated by the DLT Daemon. The DLT Viewer tool enables the software developer and the tester of the device to view and control the log and trace information. It is not the goal of GENIVI to provide full functional analyzing tools for traces, but to provide a basic set of utilities to visualize, control and test all features of the DLT Daemon component in a simple way. The DLT Daemon component and the DLT Viewer are based on the AUTOSAR 4.0 standard DLT.

We try to keep the GUI simple for an effective and efficient workflow, but also to integrate as much functionality as needed.

1.2 Restrictions and limitations

The DLT Viewer once was created as an example application how to receive and display messages delivered by a DLT Daemon running on a Linux DUT (device under test). Do not regard it as a perfect reference implementation. The only existing requirement when starting the DLT Viewer development was "we need something like a DLT Viewer". So depending on the used hardware (RAM and hard disk size), the amount of messages send by the daemon and the amount of connected devices or the pure size of a logfile limits performance.

1.3 Feature List

- Graphical User Interface
 - Menu and toolbar
 - Project widget
 - DLT Message Table
 - Search results widget
 - Footer with information about the current file etc.
- View DLT files
 - Drag and Drop support
 - Recent files selection
 - Temporary files support
 - Append files
 - Index cache of already opened DLT files
 - Default DLT file loading
 - Open multiple files (v2.10.1)
- Retrieve DLT messages from targets and store them in DLT files
 - Serial connections
 - TCP/IP connections
 - Multiple parallel connections
 - Autoconnect to targets
 - Configure log levels and trace status
 - Store configuration in target
 - Reset the target's configuration
 - Organise connections in projects
- Filter DLT messages for analysing

- Support to show only a defined subset of the messages
- Complex filter configurations
- Save and restore filter configurations
- Multiple default filter configurations
- Markers to highlight specific messages
- Filter index cache of already filtered DLT files
- Regular expressions
- Sorting by receive time (starting with v2.10.1)
- Clipboard support
 - Copy selected DLT messages to clipboard
- Export of DLT files in multiple formats
 - Supported formats:
 - * DLT format with selection
 - * ASCII format
 - * CSV format
 - Select the messages to be exported
 - * All messages
 - * Filtered messages
 - * Marked messages
- Search DLT messages
 - Step by step search
 - Search export view
 - Search by regular expressions
- Project configurations
 - Control log levels
 - Organise configurations in projects
 - Save and restore projects
 - Recent projects selection
 - Select displayed columns
 - Default project loading
 - Automatic timezone synchronisation (v2.10.1)
- Plugin configurations
 - Decoder plugins to decode messages
 - Viewer plugins to show more detailed information and analyse logs. Viewer plugins provide extra widgets to visualize data
 - Control plugins to control applications on the target
 - Available plugins
 - * DLT Viewer Plugin
 - * Non Verbose Mode Plugin
 - * Filetransfer Plugin
 - * DLT Logstorage Config Creator Plugin (outdated)

- * DLT System Viewer Plugin (outdated)
 - * DLT DBus plugin
- Command line support
 - Silent mode
 - Loading project
 - Loading DLT file
 - Using filter configuration
 - Export DLT files to ASCII
 - Execute commands in plugins
- Plugins programming guide

2 DLT Viewer GUI

2.1 Layout

The screenshot in ?? will give you a quick impression about how the Viewer could look. The Viewer is based on Qt, so there are widgets you can move around and resize as you like. Your settings of position and size of the widgets are stored when you close the Viewer so that you have exactly the same window when you launch the Viewer the next time. To get a better understanding about the DLT Viewer parts the screenshot below marks the relevant areas followed by a description.

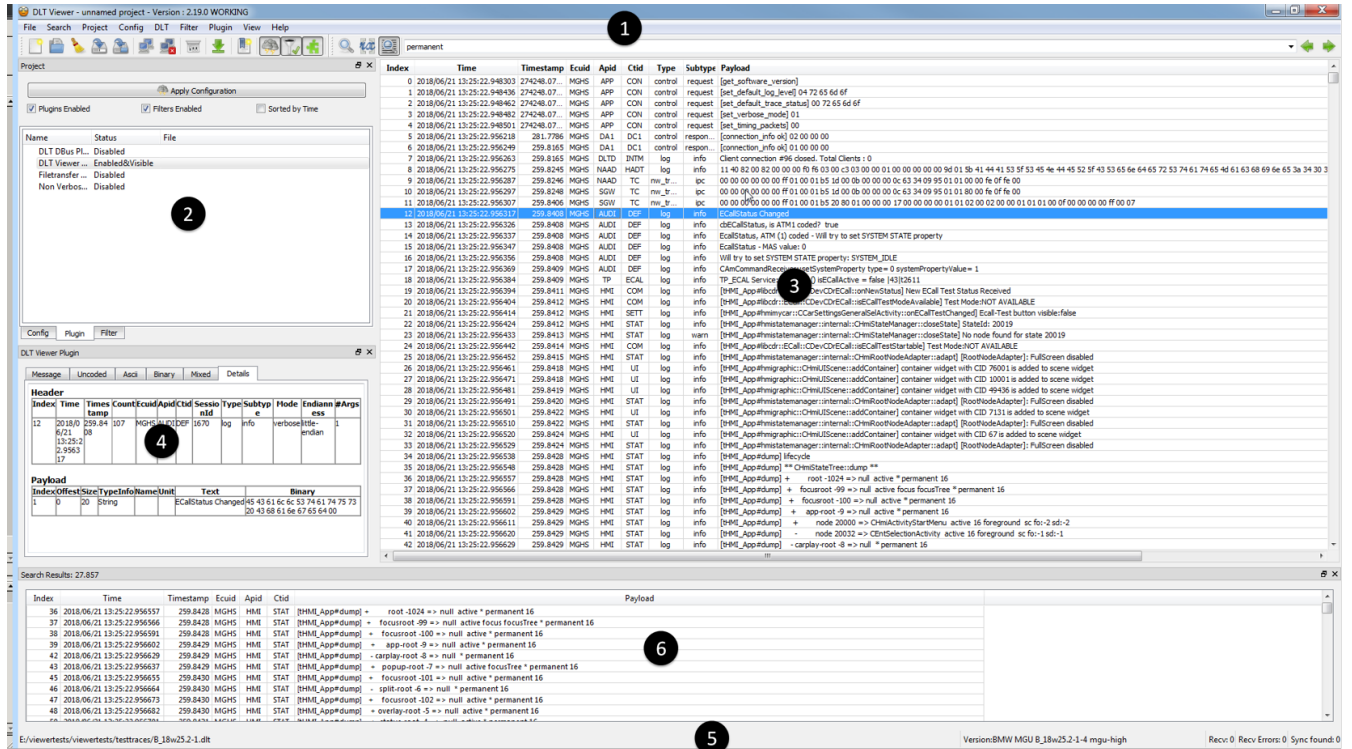


Figure 1: Main DLT Window sections

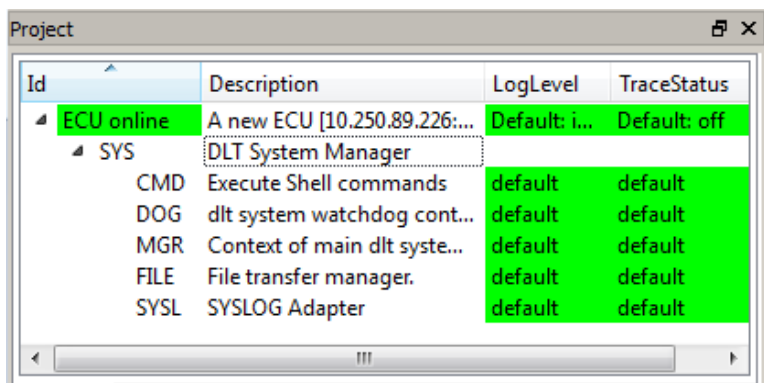
Table 1: GUI section description

Nr	Title	Description
1	Menu and Tool-bar	<p>In the window title you can find the absolute path of your project file and the project name. If you start the DLT Viewer with no default project file it will create an unnamed project.</p> <p>The menu consists of:</p> <ul style="list-style-type: none"> • File Open and save files, access DLT Viewer settings etc. • Search Search for specific messages in the current log. • Project Create, open and save DLT Viewer projects. • Config Add, edit and remove ECUs, applications and contexts. • DLT Additional functions concerning the DLT daemon and the current configuration. • Filter Create, open and save filters for the message log. • Plugin Enable, show and edit plugins. • View Select the visible components of the DLT Viewer. • Help Get information and help for the DLT Viewer. Find here detailed version information, the support email address as well as an overview of the comandline options. <p>The menu provides main functions to the user. Many menu functions are also easily accessible through the toolbar. Hover the cursor over toolbar buttons to see their functions.</p>
2	Project Panel	<p>The project widget allows you to configure and control the project, load an existing project, save a changed configuration to the DLT Daemon and more. The project panel is split into three tabs:</p> <ul style="list-style-type: none"> • The "Config" tab, which contains all connected ECUs/Devices. Each Device contains its connected applications. Each application contains its used contexts. • The "Filter" tab lets you configure filters and markers, which are used to show or mark only a defined subset of DLT messages in the message table. • The "Plugin" tab shows the available DLT Viewer plugins. The development team provides a plugin SDK to you. You are able to write your own DLT Viewer plugins to decode messages, display an extra GUI panel or control the Viewer and target applications. <p>In the settings you can e.g. select a default project, which is loaded during start-up.</p>

3	DLT Message Table	<p>In the DLT Message Table you see all DLT messages in the current DLT log file. Newly received DLT messages are written to the end of the log file. If the AutoScroll option is enabled in the settings, the table will scroll along with new DLT messages while they are received. The columns in the message table are:</p> <ul style="list-style-type: none"> • Index Index of the DLT message in the DLT log file. • Time Time when the message was received. • Timestamp Time when the message was sent, measured since the startup of the target. • Count Cyclical counter, one per context. Can be used to detect lost messages. • EcuId Name of the sending ECU. Defined in the "Projects" tab. • ApId Identifier of the message sender application. Defined by the sending application. • Ctid Context identifier of the message sender. Defined by the sending application. • SessionId Process ID of the target process / application submitting the message. • Type Type of the message (Log / Trace). • Subtype Log level or trace type. • Mode Verbose or Non-Verbose. • Args The number of arguments in the payload of the message. • Payload The payload shows all parameters of the log message as text. <p>If a filter is configured, you will see only DLT messages which match this filter. Remove or disable all filters if you want to see all messages. If there are decoder plugins and a DLT message matches one of these plugins, the decoded payload is displayed instead of the raw payload. DLT log files can be loaded and saved in the file menu.</p>
4	Plugin Widget	Provides one or more widgets to display additional data. You can choose which plugins are to be shown in the "Plugin" tab. See section ?? and chapter ?? for more information.
5	Footer	The footer shows information about the current log file, the version string (see ??) and other options of the DLT Viewer.
6	Search table	The search table shows all resulting lines of a search pattern as well as the amount of search hits

2.2 Project Panel

2.2.1 Config Tab



Id	Description	LogLevel	TraceStatus
ECU online	A new ECU [10.250.89.226:...	Default: i...	Default: off
SYS	DLT System Manager		
CMD	Execute Shell commands	default	default
DOG	dlt system watchdog cont...	default	default
MGR	Context of main dlt syste...	default	default
FILE	File transfer manager.	default	default
SYSL	SYSLOG Adapter	default	default

Figure 2: Config Tab

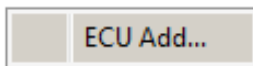
In the config tab you can see a hierarchical list of all configured devices, applications and contexts. The ECU item lists the ID of the ECU, its status (red = offline, yellow = connecting, green = online), a description of the ECU including the connection interface, the default log level and the default trace status.

The application item lists the ID and the description of the application.

The context item lists the ID, the description, the log level and the trace status of the context.

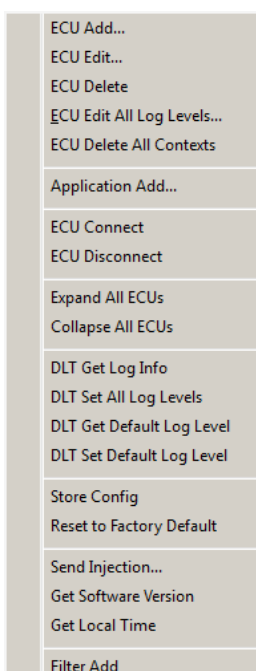
Double clicking an item lets you configure it. Right clicking an item gets you a context menu, depending on the type of the item:

Right click on empty space to add a new ECU connection

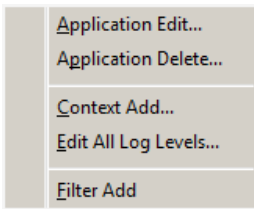


See ?? on how to connect to an ECU.

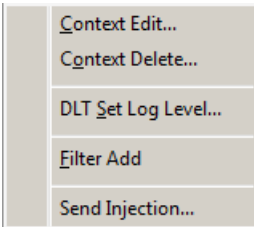
Right click on an ECU (ECU Edit) to change or configure ECU settings



Right click on an application to configure an application



Right click on a context to configure a context



2.2.2 Filter Tab

The filter list gives you an overview of all configured filters. Only DLT messages which match the enabled filters are displayed in the message table view. Double clicking a filter item lets you change the filter. Each filter can be customized to include or exclude messages, to check for specific ECUs, applications or contexts, for header and payload content and more. It is also possible to create marker filters, which don't include or exclude any messages but highlight them in a selected color.

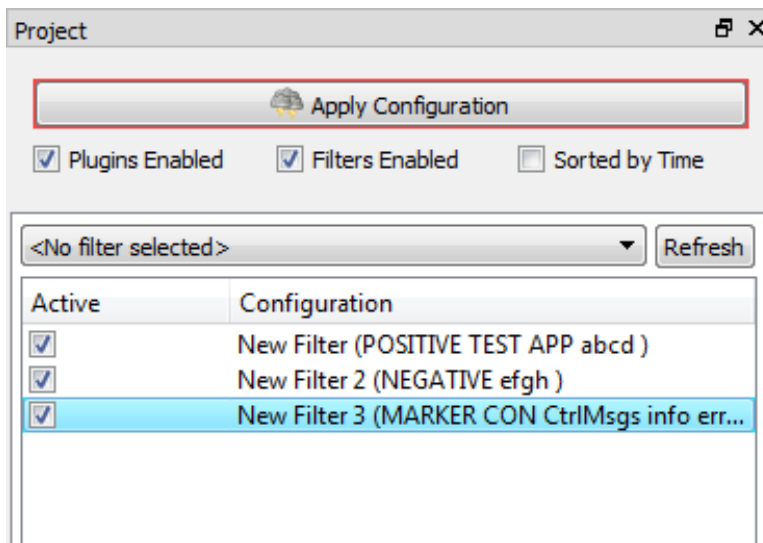
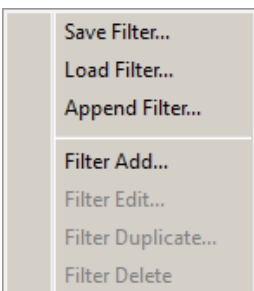


Figure 3: Filter Tab

By right clicking (or select Menu bar → Filter) you get the following menu to add or change a filter:



"Save Filter" lets you store your filter configuration to a file with extension *.dlf. Both the "Load Filter" and the "Append Filter" option will load a saved filter file, but "Load Filter" will reset your current filter list while appending keeps it.

The filters you set will apply to all incoming or loaded DLT messages. If you also want to apply the filters to previously logged messages, hit the "Apply Configuration" button on the top of the "Filter" tab.

See also ?? for more information on how working with filters.

2.2.2.1 Filter Quickselect

You can use the quick select bar to choose and enable your favourite filters placed in the default filter path. The filters can be selected from the drop down list. If you place a new filter there, it will be recognized at start up or after pressing "Refresh". To set the default filter path see also ?? Viewer Configuration Settings .

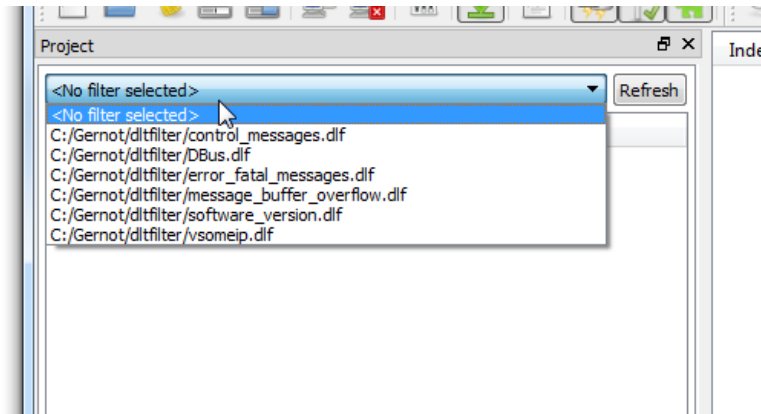


Figure 4: Filter quickselect

2.2.3 Plugin Tab

The plugin list shows all loaded plugins. Plugins e.g. can be used to:

- decode DLT messages
- provide additional graphical widgets
- control the viewer
- send injection messages to target applications
- control any attached measurement equipment like USB or IEC devices

Double clicking on a plugin item creates a popup window which lets you enable/disable the plugin and select a configuration file for it. On more information for the plugins see chapter ??.

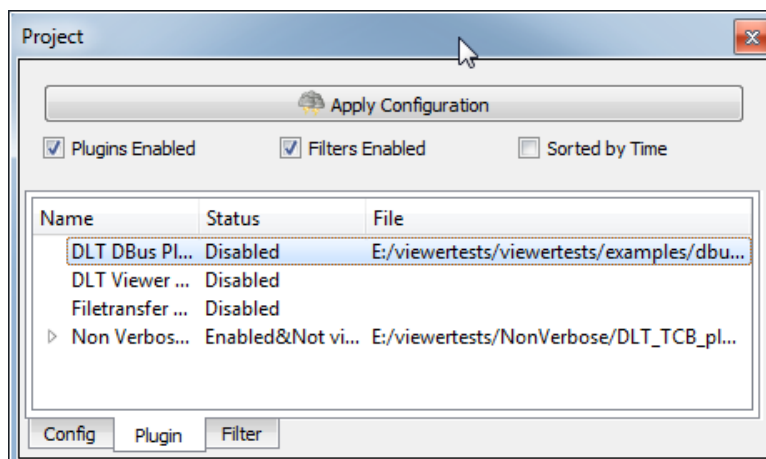


Figure 5: Plugin Tab

- The "Name" column shows the name of each plugin. This name is provided by the plugin itself.
- The "Status" column shows which plugins are enabled and visible. Only plugins that implement the "View" interface can have the status "Visible".
- The "File" column tells you the path of the configuration file or files for plugins that have such a file or files selected. Double click on plugin items to choose configuration files for them.

2.3 Viewer Configuration Settings

Access via "File → Settings Menu".

Set general behaviour of the viewer like temp file locations.

2.3.1 Global Settings

Here the "file behaviour" like e.g. locations and usage of log files or index files can be configured.

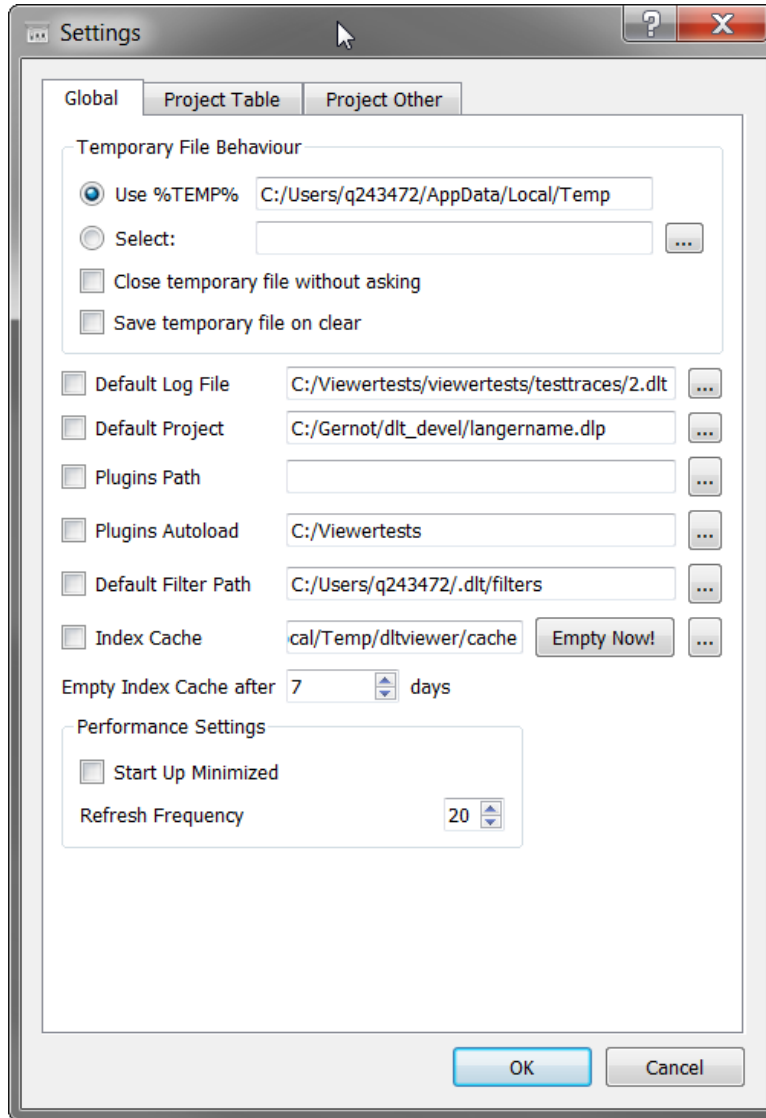


Figure 6: Global Settings Tab

Here find some selected descriptions:

1. Temporary File Behaviour

In "Temporary File Behaviour" you can set some parameters concerning the temporary file which is always created during live tracing. You can just use the system default location or define a custom folder of your choice.

CAUTION: *Keep in mind that in case "Save temporary file on clear" is selected the amount of files in your tmp path increases and your hard disk will easily run out of space. So take care about this temp path !*

2. Default log File

If enabled, the given log file will always be opened instead of creating a new tmp file at Viewer startup. Live tracing messages are appended to this file. The changes to this setting take place the next time the viewer is started. In case the file does not exist it is created.

CAUTION: In case you do not remove/change the logfile e.g. in an automated setup, it may easily increase and cause the viewer to take a long time loading the file at next start !

3. Default Project

If enabled, the given project file will always be opened at viewer startup. The settings in the project file rules the default viewer configuration settings. The changes to this setting take place the next time the viewer is started. On project files see ??.

4. Plugins Path

If enabled the viewer will look for plugins in this location additionally to the default locations. See also ??.

5. Plugins Autoload

If the plugins autoload path is enabled and set accordingly you can make use of the plugin autoload feature.

On details how to use plugins autoload see ??

6. Default Filter Path

If the default filter path is enabled and set accordingly you can use the "Filter quick select bar" in the Filter Tab. Also see ?? Filter Quickselect .

7. Index Cache

When the index cache is enabled, reloading of previous loaded files is much faster because the index is cached in additional files. To reduce disk space consumption these files can be automatically deleted after the specified amount of days of last access. The deleting takes place the next time the viewer is started. The index files have the extension "*.dix" and are located in the specified folder.

CAUTION: If this option is enabled, additional disk space is used. Take care to observe the size of the index file directory, especially in case you perform frequent comandline conversions e.g. in an automated setup. So it is recommended to disable index caching for commandline usage.

8. Performance Settings

When "Start up Minimized" is enabled the viewer window is not shown diretly. Most propably makes sense when using logging only mode.

The Refresh Frequency specifies the amount of viewer GUI refreshes per second.

2.3.2 Project Table Settings

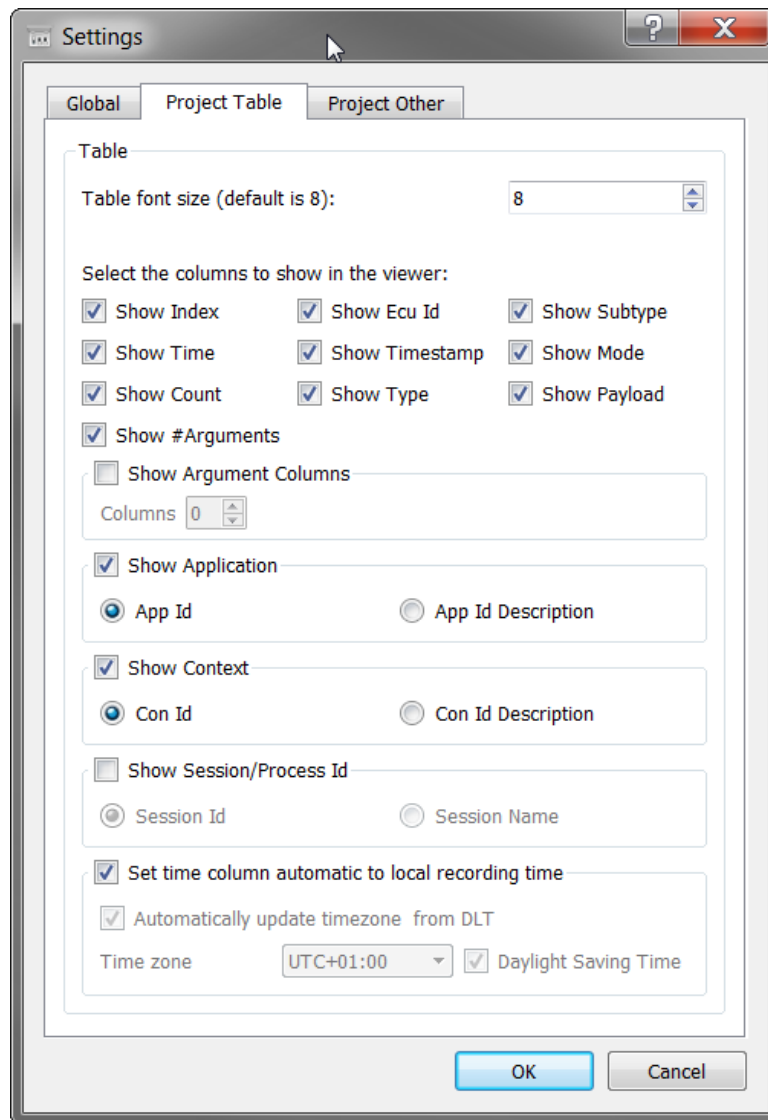


Figure 7: Project Table Tab

1. Table font size

Here you can set the desired font size to be used in the message table view, the default font size is 8.

2. Select columns to show in the viewer

In the this settings tab you mainly can select which columns you want to have displayed in the table view. See also ?? on information about the columns.

3. Show Arguments

This setting may be only interesting for dlt application developers. The column "Args" shows the number of payload arguments contained in the DLT message.

The check box "Show Argument Columns" is not functional.

4. Show Application

Select if the Apid shall be shown, if so you can choose the Apid itself or the description string which comes with the application registration message.

5. Show Context

Select if the Ctid shall be shown, if so you can choose the Ctid itself or the description string which comes with the context registration message.

6. Show Session/Process Id

Process ID of the target process submitting the message. "Session Name" only is available if transmitted by the daemon.

7. Set time column automatic to local recording time

Here you can choose which time for the receive time column should be used. You can just take local time of the system or use the DLT time settings by choosing the time zone and the mode of daylight saving time.

2.3.3 Project Other Settings

General logging behaviour configurations

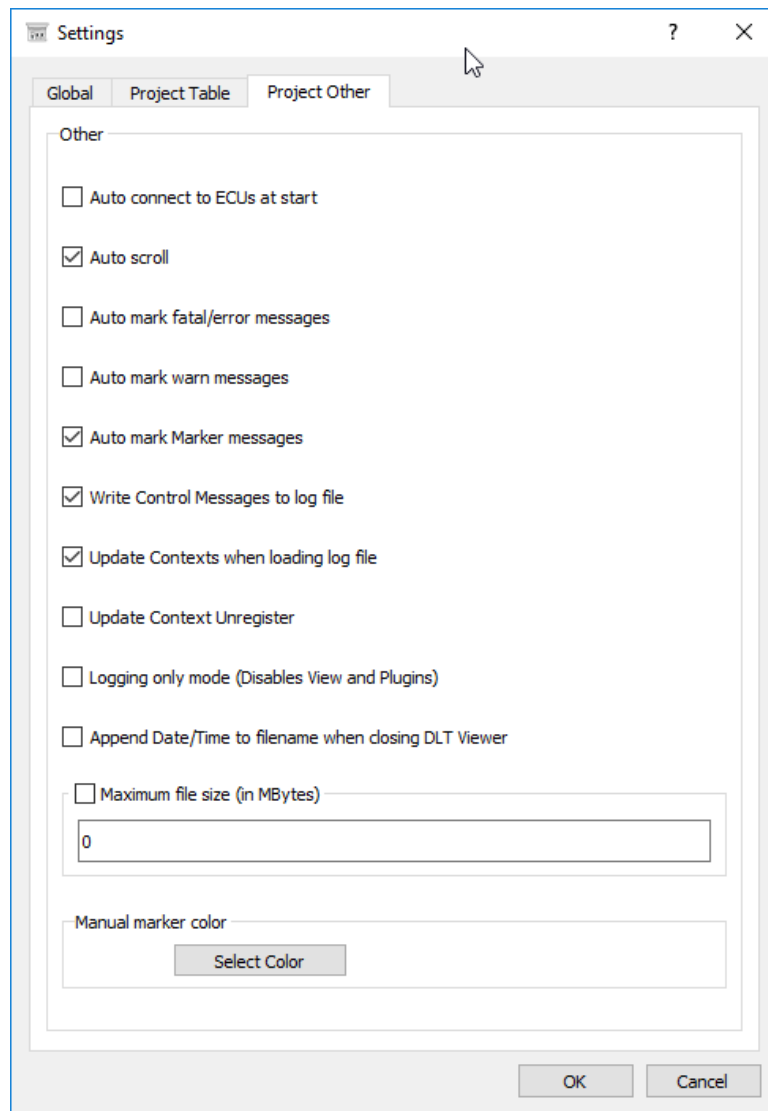


Figure 8: Project Other tab

Some chosen settings:

1. Auto connect to ECUs at start

If this checkbox is enabled the last ECU connected is tried to be connected automatically. In case you are using a project file this setting may be overruled !

2. Auto scroll

3. Auto mark fatal/error messages

Automatically mark all messages of subtype (loglevel) "fatal" and "error" with red color.

Note: in case you use a filter using a marker the filter overcolor rules this setting.

4. Auto mark warn messages Automatically mark all messages of subtype (loglevel) "warn" with yellow color.

Note: in case you use a filter using a marker, the filter color overrules this setting.

5. Auto mark marker messages

Automatically mark all marker messages with green color.

Note: in case you use a filter using a marker, the filter color overrules this setting.

Example:

1.303	2018/07/03 08:38:11.000000	2926.4850	DLTV	DLTV	DLTV	control	response	MARKER
-------	----------------------------	-----------	------	------	------	---------	----------	--------

6. Write Control Messages to Log File When this checkbox is enabled, control messages like register or unregister of applications or contexts, SW version and get_log_info from the daemon are displayed in the message table and stored in the logfile.

7. Update Contexts when loading Log file

The list of contexts is updated automatically when a log file is loaded.

8. Update Contexts Unregister

The list of contexts is updated automatically when a context is deleted.

9. Logging only mode When enabling this check box you activate the "logging only" mode, see ??:

10. Append Date/Time to filename when closing DLT Viewer

The time when the DLT Viewer is closed is appended to the current logfile name. Of course this implies that you did save the temporary log file to a custom file before as the tmp file usually is not saved ... Here also the log file files gets a name appendix of type

__YYYYMMDD_hhmmss____YYYYMMDD_hhmmss

11. Maximum File Size

The logfile is split up in several files of given maximum size in [MB]. The files get a name appendix of type

__YYYYMMDD_hhmmss____YYYYMMDD_hhmmss

and the end receive time stamp of the log. So later you can also concatenate the files to a single one in the right order.

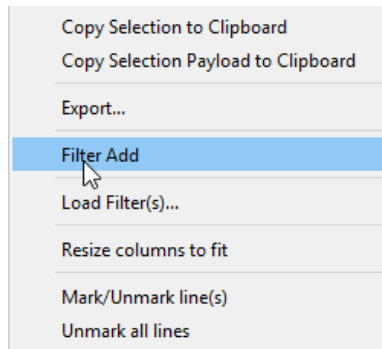
E.g.:

```
cat *.dlt >> combine.dlt
```

CAUTION: Depending on the message traffic and the choosen split file size there may occur a message loss in case the split file size is to small (< 10 kb)

12. Manual marker color

Select your favourite color for marking selected lines in the table view. Right mouse click on the table view to highlight the selected lines. Marking only has a visual effect, there will be no additional information stored in a file. The coloring refers to the line index, not to the message index. In case you change the order of the messages e.g. by filtering or loading another logfile, the highlighting will remain on this line index anyway. To completely remove the highlighting again also use "File clear".



2.3.4 Viewer Configuration File

The basic viewer configuration file is "config.ini". It e.g. stores the Viewer Settings, plugin states as well as size and windows position. Its default location is:

`~/.dlt/config` (on Linux)

and

`C:\Users\<username>\.dlt\config` (on Windows)

This file is not overwritten in case you reinstall the Viewer. In case you install a new release of the viewer it may be better to delete the file to avoid inconsistency. A new instance of the file containing default values will be created when the viewer starts.

3 Plugins

The feature set of the DLT Viewer can be easily extended with DLT Viewer plugins. Plugins have full read access to trace messages. They can decode messages, provide additional representations inside the DLT Viewer GUI, control the Viewer or send injection messages to target applications, depending on which DLT interfaces the respective plugin implements. It is easy to write your own DLT Viewer plugin, examples are provided in the DLT Viewer source code.

Here is an overview of the some useful plugins:

3.1 Non Verbose Mode Plugin

Interface:

- QDLTPluginInterface
- QDLTPluginDecoderInterface

Description: Decodes the DLT non-verbose messages after a catalog file has been loaded.

If you receive DLT messages in non verbose mode, you need to decode them to see their plain text. To do this, go to the "Plugin" tab in the "Project" panel. Double click on the plugin "Non Verbose Mode Plugin". In the dialog that appears, you can open a configuration file to load. Use this to load a non verbose XML description file, which corresponds to your software on the ECU. Hit "OK" and the Plugin shows a list of all available non verbose messages. As soon as you select a DLT message you can view the decoded information. Incoming messages that fit the description file will now get decoded. To also decode the already received messages, hit "Apply Configuration" on the top of the "Plugin" tab. This plugin has no visible widget.

Raw nonverbose messages example:

[illegible]

Decoded nonverbose messages example:

3	2018/03/15 13:37:08.336371	13016.2090	TCB	USB	TCM	CMD: AT+CLCC
4	2018/03/15 13:37:08.336460	13016.2110	TCB	USB	TCM	RSP: ..OK....
5	2018/03/15 13:37:08.336494	13016.3130	TCB	PCHN	STAT	+GPS: 2189135000,2320970466,65535,0,0,34
6	2018/03/15 13:37:08.336526	13016.3140	TCB	DTSV	MAIN	received Local date/time info outdated
7	2018/03/15 13:37:08.700651	13016.3520	TCB	DEAD	GEN	SlowTimer: IFastCAN=0, ICAN=0, IGPS= 130104, EXTRAPOL= 13011, MOVED=0, ADJUST=0, REBASE=0
8	2018/03/15 13:37:08.700759	13016.5470	TCB	LAT	CTRL	[ok]
9	2018/03/15 13:37:08.701002	13016.5520	TCB	LAT	CTRL	[ok]
10	2018/03/15 13:37:08.701025	13016.8570	TCB	USB	ECM	CMD: AT
11	2018/03/15 13:37:09.308365	13016.8570	TCB	USB	ECM	RSP: ..OK..
12	2018/03/15 13:37:09.308384	13017.2660	TCB	USB	TCM	CMD: AT+CLCC

3.1.1 DLT Parser

To create an input file for this plugin based on your target DLT Daemon configuration in nonverbose mode, have a look on the tool "DLT Parser". Also see the DLT Daemon documentation on the non verbose mode.

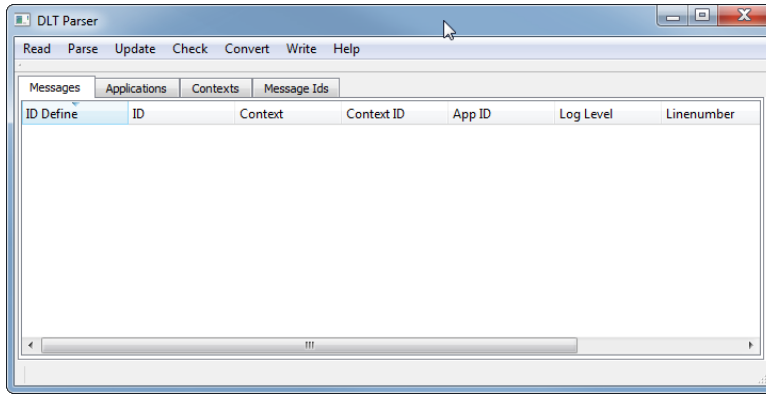


Figure 9: DLT Parser

3.2 DLT Viewer Plugin

Interface:

- QDLTPuginInterface
- QDLTPuginViewerInterface

Description: Advanced visual representation for selected log messages.

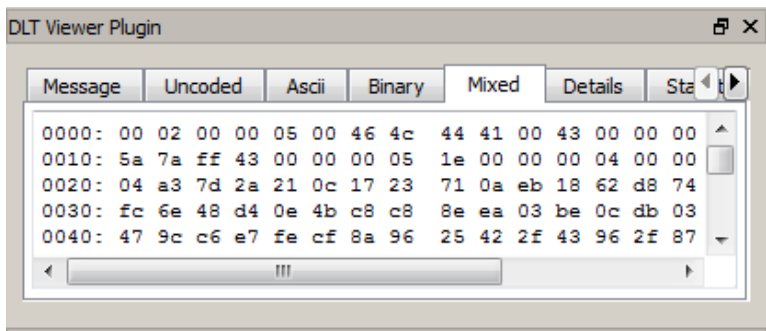


Figure 10: DLT Viewer Plugin widget

3.3 DLT DBus Plugin

Interface:

- QDLTPuginInterface
- QDLTPuginViewerInterface
- QDLTPuginControlInterface
- QDLTPuginDecoderInterface

Description:

Decode and show DBus messages.

This is an example of raw DBus messages in the table view:

158.663	2018/05/04 14:50:21.867478	624.2262	MGHS	DBSY	DBSY	6c 04 01 01 a1 02 00 00 83 0f 00 00 a5 00 00 00 01 01 6f 00 34 00 00 00 2f 6f 72 67 2f 66
158.664	2018/05/04 14:50:21.867545	624.2263	MGHS	DBSY	DBSY	6c 04 01 01 1c 03 00 00 84 0f 00 00 a5 00 00 00 01 01 6f 00 34 00 00 00 2f 6f 72 67 2f 66
159.843	2018/05/04 14:50:24.870014	627.2287	MGHS	DBSY	DBSY	6c 04 01 01 a1 02 00 00 85 0f 00 00 a5 00 00 00 01 01 6f 00 34 00 00 00 2f 6f 72 67 2f 66
159.844	2018/05/04 14:50:24.870080	627.2287	MGHS	DBSY	DBSY	6c 04 01 01 1c 03 00 00 86 0f 00 00 a5 00 00 00 01 01 6f 00 34 00 00 00 2f 6f 72 67 2f 66
160.884	2018/05/04 14:50:27.872831	630.2313	MGHS	DBSY	DBSY	6c 04 01 01 a1 02 00 00 87 0f 00 00 a5 00 00 00 01 01 6f 00 34 00 00 00 2f 6f 72 67 2f 66
160.885	2018/05/04 14:50:27.873076	630.2314	MGHS	DBSY	DBSY	6c 04 01 01 1c 03 00 00 88 0f 00 00 a5 00 00 00 01 01 6f 00 34 00 00 00 2f 6f 72 67 2f 66

This is how it looks like when the messages are decoded:

158.663	2018/05/04 14:50:21.867478	624.2262	MGHS	DBSY	DBSY	S [:1.0] /org/freedesktop/systemd1/unit/hdlic_5fipc4_2eservice org.freedesktop.DBus.Properties
158.664	2018/05/04 14:50:21.867545	624.2263	MGHS	DBSY	DBSY	S [:1.0] /org/freedesktop/systemd1/unit/hdlic_5fipc4_2eservice org.freedesktop.DBus.Properties
159.843	2018/05/04 14:50:24.870014	627.2287	MGHS	DBSY	DBSY	S [:1.0] /org/freedesktop/systemd1/unit/hdlic_5fipc4_2eservice org.freedesktop.DBus.Properties
159.844	2018/05/04 14:50:24.870080	627.2287	MGHS	DBSY	DBSY	S [:1.0] /org/freedesktop/systemd1/unit/hdlic_5fipc4_2eservice org.freedesktop.DBus.Properties
160.884	2018/05/04 14:50:27.872831	630.2313	MGHS	DBSY	DBSY	S [:1.0] /org/freedesktop/systemd1/unit/hdlic_5fipc4_2eservice org.freedesktop.DBus.Properties
160.885	2018/05/04 14:50:27.873076	630.2314	MGHS	DBSY	DBSY	S [:1.0] /org/freedesktop/systemd1/unit/hdlic_5fipc4_2eservice org.freedesktop.DBus.Properties

In the plugin widget you get more detailed information about the DBus message. And a more structured depiction in the Header tab.

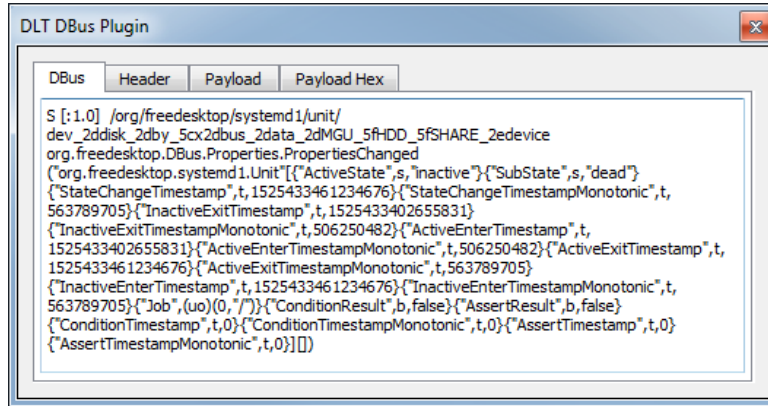


Figure 11: DLT DBus Plugin widget

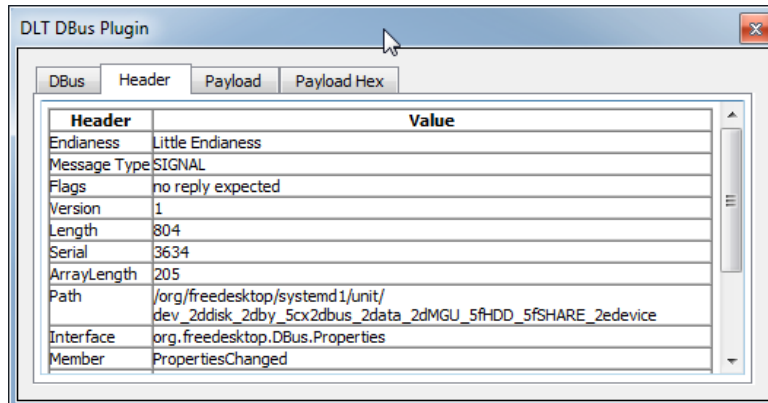


Figure 12: DLT DBus Plugin widget, Header tab

DBUS messages are identified by a special combination of Apid/Ctid: the default is DBUS/ALL in case you do not load a special configuration file.

Example: in case you need to use the two combinations of Apid/Ctid:

DBSY/DBSY

DBCN/DBCN

To enable these two combinations use the delivered example file

"dbusplugin_configuration.xml"

which is located in the plugins/examples folder (Windows) or /usr/share/doc/genivi-dlt-viewer (Ubuntu) of your viewer installation. If the Apid/Ctids changes you can edit the file and change or add items for your need. The total amount of Apid/Ctid combinations is limited to 10 due to performance reasons.

The configuration file is loaded e.g. in the plugin configuration widget:

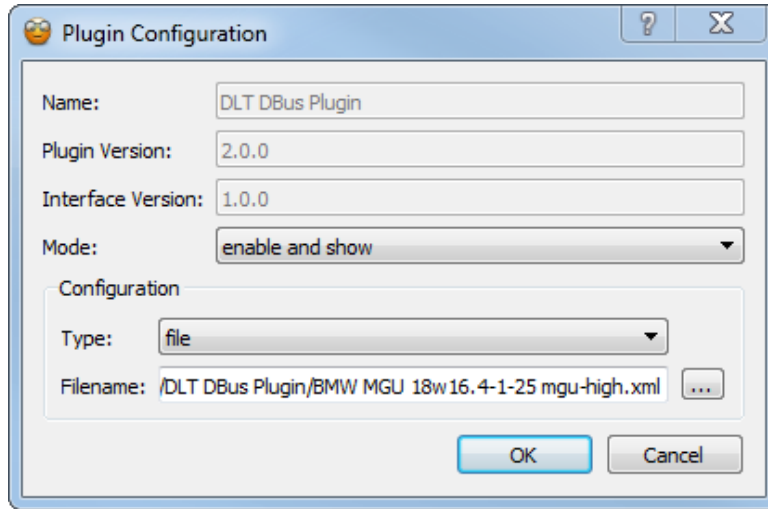


Figure 13: DLT DBus Plugin configuration widget

3.4 Filetransfer Plugin

Interface:

- QDLTPluginInterface
- QDLTPluginViewerInterface
- QDLTPluginControlInterface

Description: In case your log contains files like screenshots, core dumps etc. logged by a DLT Filetransfer application the objects can be viewed and saved with this plugin.

Filetransfer Plugin

Select all complete files

Deselect all complete files

Save all selected files

Id	Filename	Creation Date	Status	Save	Size in Bytes	Packages (Size/	Received Packag	Buffersize
2488241784	core.dlt-system.13789.gz	Tue Sep 15 1...	Complete	<input type="checkbox"/>	209891	205	205	1024
930053386	screenshot_20150915-140257_.png	Tue Sep 15 1...	Complete	<input type="checkbox"/>	65479	64	64	1024
612464907	screenshot_20150915-140256_.png	Tue Sep 15 1...	Complete	<input type="checkbox"/>	71279	70	70	1024
3400885468	screenshot_20150915-140254_.png	Tue Sep 15 1...	Complete	<input type="checkbox"/>	72954	72	72	1024

DLT Statistic Plugin

DLT Viewer Plugin

DLT Graphical System Viewer Plugin

Filetransfer Plugin

Search Results

Figure 14: File transfer widget

By double clicking the file name, you can see the file contents as preview and a text or image viewer is opened.

After enabling the filetransfer plugin when a dlt file already was loaded you will have to reload the logfile !

You can also use the Filetransfer Plugin command line mode to extract files out of a stored dlt file e.g. in an automated test setup. On more information about the command line mode see ??.

3.5 Plugin SDK

More information about the Plugin SDK can be found in the separate document `dlt_viewer_plugins_programming_guide.pdf`. In the source code you also find four dummy plugins, ready to build. You can just use it as a template.

There are

- dummycontrolplugin

- dummyviewerplugin
- dummycommandplugin
- dummydecoderplugin

3.6 Using the plugins autoload feature

For the nonverbose mode or the DBus plugin, the used configuration file may e.g. depend on the SW version of your target. In case the DLT Daemon is configured to send out the target software version instead of the DLT Daemon version (see DLT Daemon documentation) you can automatically load the according configuration file needed by the plugin.

This is especially helpful if you work with traces of different ECUs or software versions.

See also ?? for information about the version string.

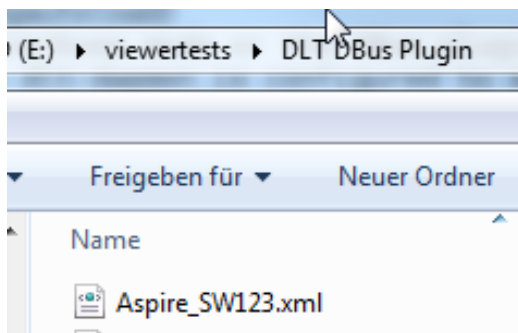
To use the autoloaod feature you need to create a subdirectory whichs name equals exactly the name of the choosen plugin, e.g. "DLT DBus Plugin". In this directory you can place plugin configuration files whichs names exactly equal the version string send by your target. So if you process logs with different target software versions the correct configuration is automatically loaded accordingly.

3.6.1 Plugin autoload example

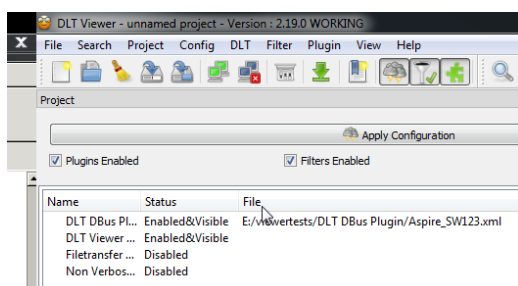
1. E.g. the version string is "Aspire_SW123":

Version:Aspire_SW123

2. Create a directory named "E:\viewertests\DLT DBus Plugin" and place a configuration file namend "Aspire_SW123.xml", (or so named folder which contains this file) there.



3. Set the plugins autoload path accordingly to "E:\viewertests" and set the checkbox, see ??
4. Enable the plugin itself and load the dlt file or connect to the target
5. The configuration file is automatically processed as soon as the version string is received:



3.7 Install, remove, enable and disable plugins

3.7.1 Windows

The DLT Viewer will look for plugins in two places. The first place is the "plugins" folder inside the main application path. After that, if a plugins path is defined in the DLT Viewer settings (see ??), the application will also look in this specified location for plugins.

You can install plugins by putting them into one of the directories mentioned, and uninstall them by removing them from these directories.

It also is possible to enable or disable plugins in the DLT Viewer itself. To do this, double-click or right-click on a plugin in the "Plugins" tab. You will get a drop-down menu for enabling and disabling the plugin. When enabled, messages are passed to the plugin. When disabled, the plugin will be ignored in the message processing chain. For plugins that implement the "viewer" interface, there is also a third selection, "Enable and Show", which brings up the UI widget of this plugin.

3.7.2 Linux

On Linux, organizing plugins works in a similar fashion. In addition to the previously mentioned folders, the Linux version will also look into `"/usr/share/dlt-Viewer/plugins"` for any plugins. You will need to have root permissions to modify the plugin directories that are installed system-wide. To organize plugins without root permissions, you can set the user defined plugin directory in the DLT Viewer settings to some other path, e.g. somewhere in your home directory.

4 Best Practices

4.1 Connect to ECU

Press the connect button  to connect to the target:

If no ECU is configured, the ECU configuration dialog appears. Select the "Interface Type" you want to use (TCP, UDP or serial). The ECU configuration dialog also can be evoked by selecting right mouse button "ECU add" in the Project Config Panel.

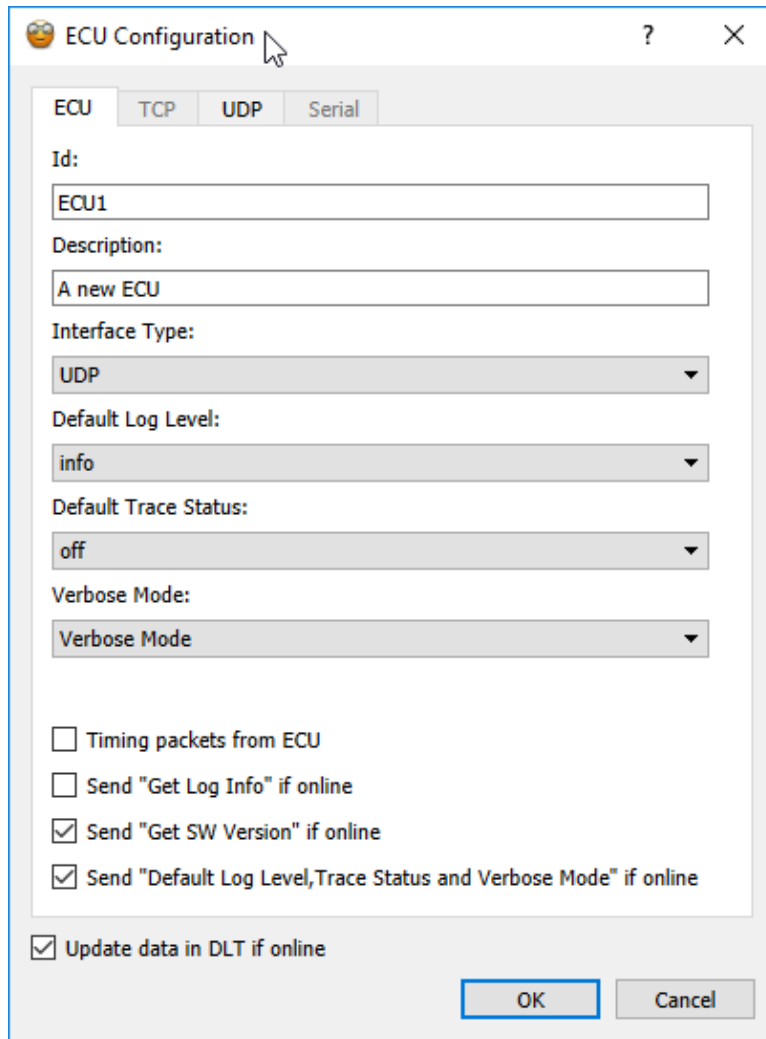


Figure 15: ECU dialog

Here you can set some basic parameters like

- Id
Identifier of the "Electronic Control Unit" used in the config tab. If it is left at its default value (ECU1) it will automatically be set by the ECU messages which come with the connection.
- Description
Description of the ECU Id used in the config tab
- Interface Type
You can choose TCP, UDP or Serial
- Default Log Level (equals "Subtype" in the message table), see also ??
- Default Trace Status (on / off)
- Verbose Mode (yes / no)
See also ??
- Timing packets from ECU
Request the ECU to send additional timing messages (see daemon documentation)
- Send "Get Log Info" if online
See ??
- Send "Get SW Version" if online
Request SW version from daemon, see also ??

- Send "Default Log Level, Trace Status and Verbose Mode" if online
- Update data in DLT if online
Send the above selected request to the ECU always when connecting to it

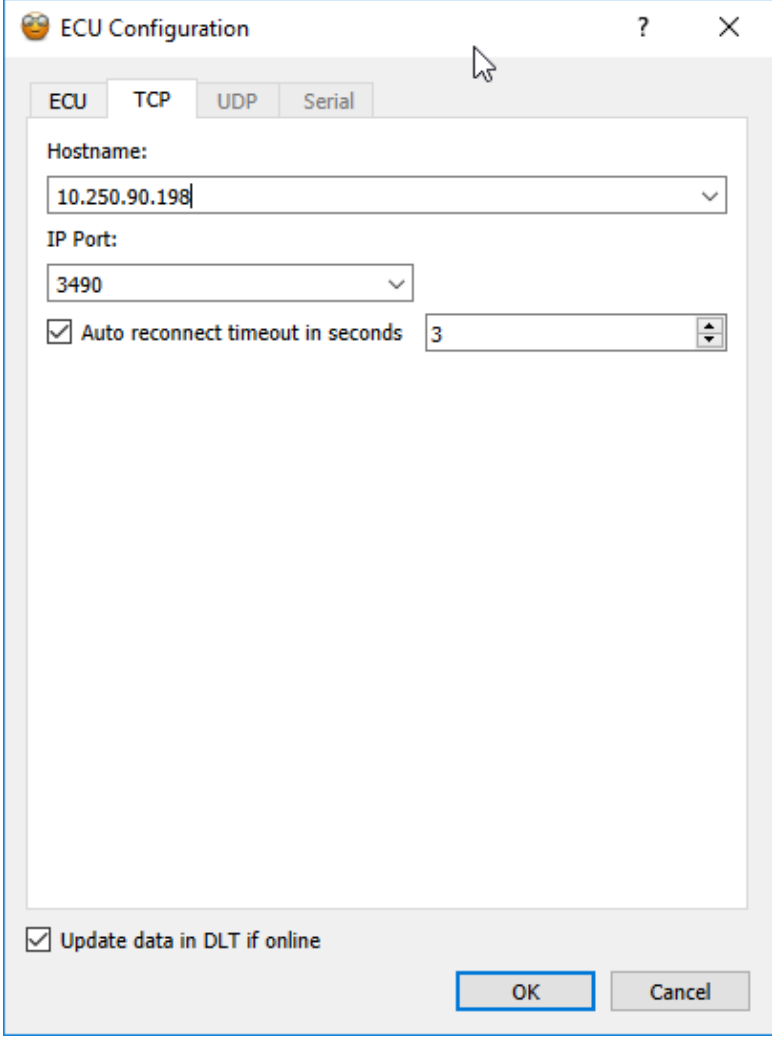
4.1.1 TCP

If you choose a TCP connection enter the the hostname or the IP address.

Format is e.g. 192.168.0.5 of the target ECU.

In case of an IPv6 address just use the format (e.g.) fe80::xxxx:xxxx:xxxx:%eth1

IPv4 example:



The screenshot shows a window titled "ECU Configuration" with a mouse cursor over the "TCP" tab. The "TCP" tab is selected, and the "Hostname" field contains "10.250.90.198". The "IP Port" field contains "3490". The "Auto reconnect timeout in seconds" checkbox is checked, and the value is "3". At the bottom, the "Update data in DLT if online" checkbox is also checked. The "OK" and "Cancel" buttons are visible at the bottom right.

Figure 16: IP configuration tab

4.1.2 UDP

In case of UDP reception you are able to specify the interface and the IP port where you expect the UDP datagrams to be received. The selection box "Receiving interface" will offer you all available ethernet interfaces of your machine. In case of AnyIP the operating system will choose an interface for you. To receive multicast messages you can activate "Multicast on network interface" and specify the UDP multicast address to join the multicast group accordingly.

CAUTION: *On Windows machines it was observed that the selection of the specific IF instead of AnyIP is essential*

IPv4 example:

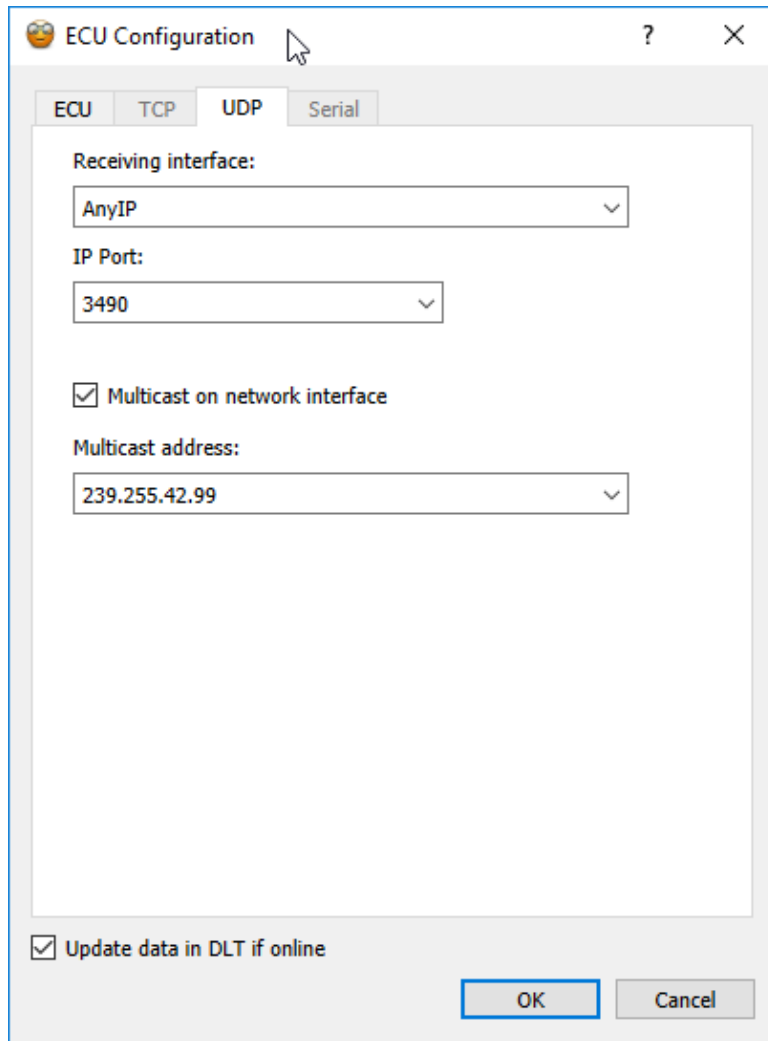


Figure 17: UDP configuration tab

4.1.3 Serial

If you choose a serial connection, select the serial port here. Check the COM port where your serial device is connected to in your Windows configuration. A drop down list provides a list of examples for the port configuration.

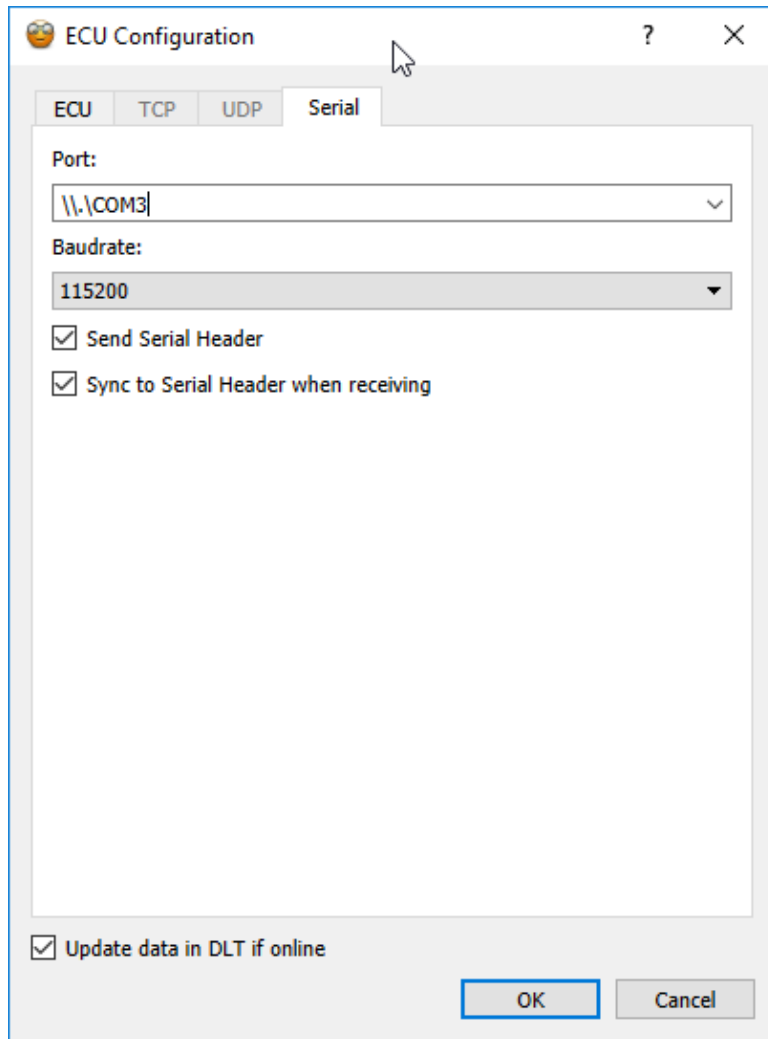


Figure 18: Serial configuration tab

If you select "OK" the DLT Viewer will try to connect to the ECU. The ECU Id shows "online" (green) as soon as the DLT Viewer is connected to the specified device. All received DLT messages should then be displayed in DLT message table.

4.2 Available List of Applications and Contexts

4.2.1 Hierarchy

Applications and contexts are organized in a hierarchy. Each ECU connection can have several applications. Each application can have several contexts. Contexts are the instance where the logging application configures log levels and trace status.

In the Project/Config tab you will see a tree with all used applications and contexts in a sub tree of an ECU.

4.2.2 Get all used contexts from ECU

In case you are connected to the ECU you can get all used contexts by double click on the ECU item in the Project/Config tab and select menu item "DLT Get log info". The DLT Viewer then sends a request to the target and receives a response message with all available contexts and their log levels. The list of available contexts is added to your "Config" list.

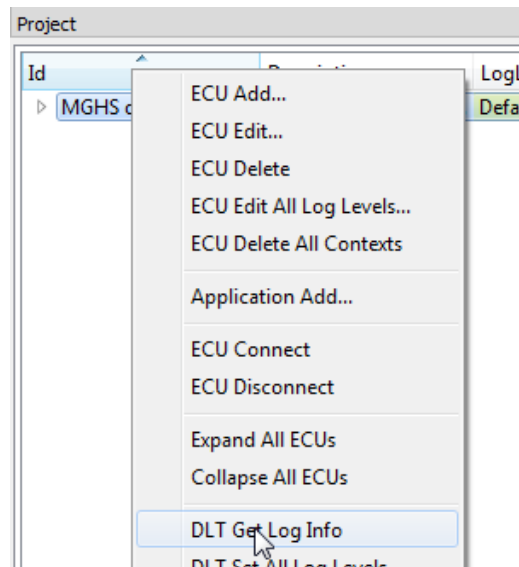


Figure 19: Get log info

4.2.3 Automatically update used context

There are several possibilities how to automatically update the list of contexts of a ECU.

4.2.3.1 Get context list when connected to ECU

Each time you connect to the ECU you can get a list of all used context. You can activate this feature in the ECU configuration by activating "Send Get Log Info if online". See ??.

4.2.3.2 ECU sends automatically new registered contexts

This feature must be activated in the DLT daemon configuration of the ECU.

4.3 Log levels and trace status configuration

Each DLT log message provides a log level. The following table shows the different available log levels and describes in which use cases they should be used.

Table 2: GUI section description

Log Level	Usage
DLT_LOG_FATAL	Fatal errors are errors which prevents the software to continue working.
DLT_LOG_ERROR	Normal errors are errors which are detected by the software, but do not prevent the software to continue working.
DLT_LOG_WARN	Warnings are used when some minor problems are detected.
DLT_LOG_INFO	Info is the standard log level. It is used to display all kind of information, so that the device tester knows that the functionality of the software is correctly working. This log contains e.g. version information or state change information of SW modules.
DLT_LOG_DEBUG	This log level is normally turned off. It should be used to display some more deep information about the functionality of a software component.
DLT_LOG_VERBOSE	This log level is normally turned off. This log level provides huge amount of logs from a SW component and should only be enabled for individual SW components.

In case the default log level is set to DLT_LOG_INFO for the ECU, all log messages from DLT_LOG_FATAL to DLT_LOG_INFO are send out by the daemon and displayed. DLT_LOG_DEBUG and DLT_LOG_VERBOSE messages are not generated in this use case.

4.3.1 Get default log level

Request the current default log level directly from the ECU.

4.3.2 Set ECU default log level

The default log level can be set for each ECU connection separately. Just open a current ECU configuration by double click on the ECU item in the Project/Config tab. You can change the value of "Default Log Level" and "Default Trace Status" and press OK. If you are connected the new default values are send to the ECU immediately. All Contexts which are set to "default" get the new default values. The option "Update data in DLT if online" must be activated. When you connect to the ECU the configured default values are send immediately. See ??.

4.3.3 Edit all log levels

Here you can edit and set the loglevels for all contexts at once. If the checkbox "Update data in DLT if online" then the log level for all contexts is set. E.g. you can switch off tracing for all log levels at once. In this case the DLT daemen does not send out any messages except DLT control messages.

4.3.4 Set individual log levels

To set individual log levels and trace status for a dedicated context you can to double click on the context item in the "Project/Config" tab. A Context configuration dialog is opened where you can set individual log level and trace status. Press Ok to send the changed log level to the ECU.

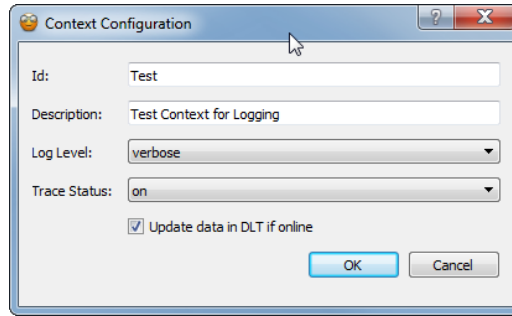


Figure 20: Context configuration

4.3.5 Set all configured log levels of all contexts

You can send all configured log levels, e.g. when you stored the log levels in the project file. Right click on a ECU item in the "Project/Config" tab and press the menu item "DLT Set All Log levels":

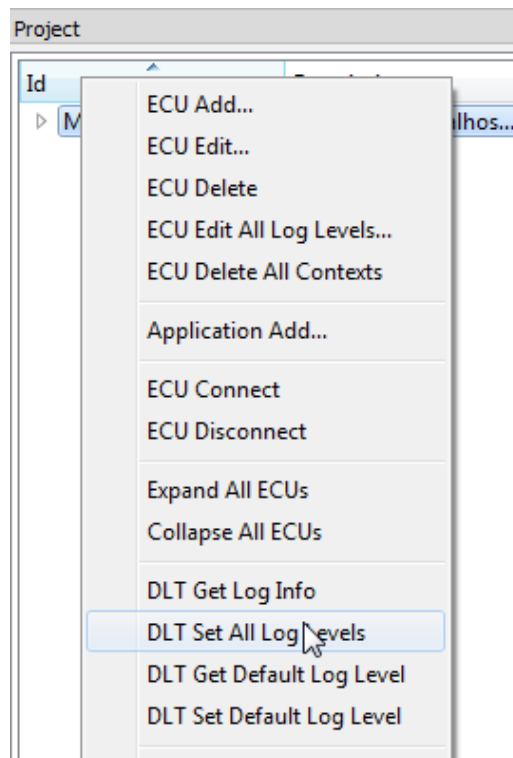


Figure 21: DLT Set all loglevels

4.3.6 Store all log levels and trace status in ECU

You can store the current configuration of all log levels on the ECU. After restart of the ECU the current set log levels and trace status are available again. Right click on a ECU item in the "Project/Config" tab and press the menu item "Store Config":

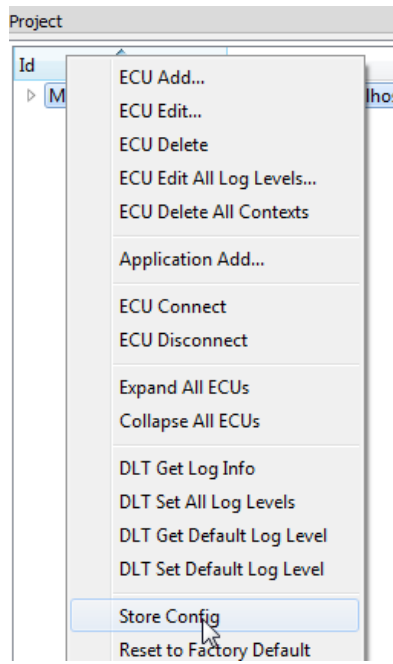


Figure 22: Store all loglevels

4.4 The version string

Depending on the setting "ECU Software version info" in dlt.conf configuration of the DLT Daemon the daemon sends either the daemon version or the content of a given file on the target periodically. The version is transmitted by a special control message, for example:

35	2018/06/29 16:02:00.346752	1620.0177	MGHS	DA1	DC1	control	response	[get_software_version ok] BMW MGU B_18w26.4-1-21 mgu-high
----	----------------------------	-----------	------	-----	-----	---------	----------	---

You also can request the version string manually:

Right click on a ECU item in the "Project/Config" tab and press the menu item "Get software version".

The version string is shown in the footer (see ??) of the dlt viewer. Assumend this method is used to identify target and software version you can also use it for plugins autoloader feature, see also ??.

Version string example 1 (daemon version)

Version:DLT Package Version: 2.17.0 UNSTABLE, Package Revision: v2.15.0_65_ga961dba, build on Mar 23 2018 15:26:03 -SYSTEMD -SYSTEMD_WATCHDOG -TEST -SHM	Recv: 4
--	---------

Version string example 2 (daemon version with tooltip)


In case the version string is too long to be displayed in the footer you can see the full name in the tooltip by hovering above the version string.

Version:DLT Package Version: 2.17.0 UNSTABLE, Package Revision: v2.15.0_65_ga961dba, build on Mar 23 2018 15:26:03 -SYSTEMD -SYSTEMD_WATCHDOG -TEST -SHM	Version:DLT Package Version: 2.17.0 UNSTABLE, Package Revision: v2.15.0_65_ga961dba, build on Mar 23 2018 15:26:03 -SYSTEMD -SYSTEMD_WATCHDOG -TEST -SHM	Recv: 300.17
--	--	--------------

4.5 Working with DLT Log Files

When you start the DLT Viewer, a temporary log file is created. The default location of these logfiles is configurable (see ??) in the Global Settings Tab. All received DLT messages get stored in this temporary log file and shown in the DLT message table. A file containing messages usually is associated with a target / ECU connection. In case of livetracing the messages received from all connected ECUs are stored in the same file.

4.5.1 Save as

Select the menu option "File"→"Save as" or  to save a file. Saving a file just renames the current tmp file and stores it to the selected destination, the received messages are still added to this file and the file is processed / accessed by the DLT Viewer. A saved file also contains information about ECU configuration, Apids, Ctids and log levels.

CAUTION: In case you did open multiple files to concatenate them, e.g. by Drag&Drop then only the last opened file will be saved. If you want to save all files in one please use "Export ..."

4.5.2 Export

Select the menu option "File"→"Export... ". There are several options for exporting messages:

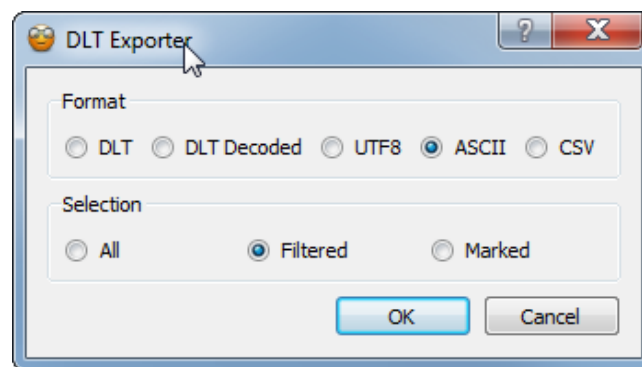


Figure 23: Exporter dialog

1. Format: file format of the exported messages
 - (a) DLT: DLT binary format, the file can be loaded and processed by DLT Viewer.
 - (b) DLT Decoded: store the selection in DLT binary format, the resulting file can be loaded and processed by DLT Viewer. In case a decoder plugin was active, the decoded message payload is stored in human readable way.
 - (c) UTF8: stores the selection as UTF8 text file.
 - (d) ASCII: stores the selection as ASCII text file.
 - (e) CSV: like ASCII but columns are separated by semicolon.
2. Selection: choose which messages to be exported
 - (a) All: all messages in the current logfile.
 - (b) Filtered: only messages which apply to the active filters.
 - (c) Marked: all marked lines are exported. Marked lines are either manually marked in the table view or lines marked by a filter. "Automarked" lines like warn/error or marker messages are not taken into account.

4.5.3 New file or open file

Select the menu option "File"→"New/Open" to create a new or open an existing log file.

4.5.4 Import DLT Stream

Select the menu option "File"→"Import DLT Stream" to import a an existing log file which does not contain DLT file storage headers. The logfile messages are imported into the currently opened dlt file.


4.5.5 Import DLT Stream Serial

Select the menu option "File"→"Import DLT Stream" to import an exiting serial log file. Just the logfile messages are imported without an ECU configuration or Apid / Ctid informations.

4.5.6 Append DLT File

Select the menu option "File"→"Append DLT File" to append the message content of the selected file to a current file. The message index is automatically increased but the messages may not be in the correct order regarding reception time stamp. This can be achieved by "Filters Enabled" with "Sorted by Time". See ??.

4.5.7 File clear

Select "File"→"Clear" or  to clear the current log file. The message table view and the indexes as well as the search result windows are cleared. If configured, a new tmp file is automatically opened.

4.5.8 Open multiple files



Data logging devices usually create logfiles of a determined size. So for analyzing a trace you may need to concatenate these files to one single file. On a Linux machine you e.g. could do that with:
"cat *.dlt » resulting.dlt"
cat *.dlt accesses the files sorted by the file name, that means the correct order of the file must be determined by the file name.

4.5.9 Searching and navigating in the logfile


4.5.9.1 Jump to line / index

Select Menu "Search"→"Jump to" or keypress combination "STRG + G" to directly navigate to the desired line / index number in the table view.

4.5.9.2 Quicksearch

Using the input field in the menu bar you can perform a quicksearch, the string entered is used a search pattern to find matches in the payload of all messages as configured in the Search Dialog. Use the green arrows   in the menu bar to navigate in single step mode or fill the search results window.

4.5.9.3 Advanced search

If you want to specify more details and other search parameters you can use the Menu "Search"→"Find" or search icon  in the toolbar to open the search dialog.

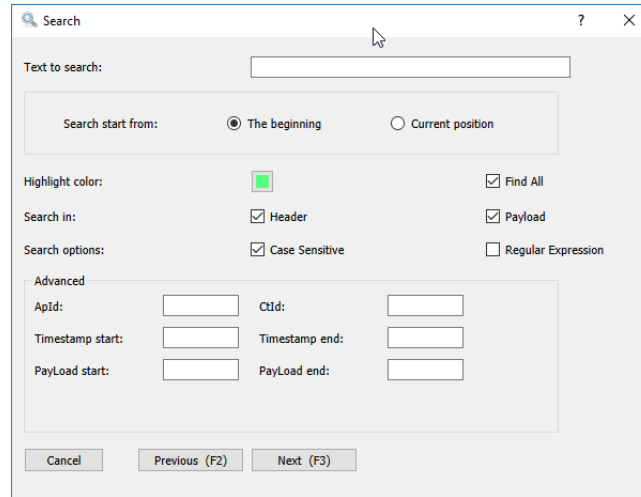




Figure 24: Search dialog

- Text to search
The search string you are looking for in the trace.
- Search from:
Search whole log or starting from the current selected line.
- Highlight color:
Select the color to highlight the hit line in single step mode.
- Find all:
If enabled then all search hits are listed in the search results window, else the search runs in single step mode: jump to the results with Previous (F2) or Next (F3) or using the green arrows   in the menu bar.
- Search in Header
Search hits in the message header will be shown too. E.g. search for a certain receive or target time stamp.
- Search in Payload
Search hits in the message payload will be displayed. This is most probably the standard option.
- Search Options: Case Sensitive


CAUTION: *If enabled the search time may increase significantly !*

- Search Options: Regular Expression
Use regular expression search.
- ApId
Only take care of messages with this ApId.
- CtId
Only take care of messages with this CtId.
- Timestamp start / end
All search hits will have target time stamps within the given time range will be shown.
- Payload start / end
All search hits will be in the range of a payload which contains the given start string and a payload which contains the given stop string will be shown.

4.5.9.4 Search history

You can use the menu "Search"→"History" to select the last used search strings. There also is a drop - down menu in the search input field of the toolbar to select the last used search strings.

4.5.10 Setting a Marker

When you set a marker in the logfile using DLT → Marker, or just click on the icon: , a marker line is added to the bottom of the current trace, this is especially meaningfull when live tracing to mark a certain point in the log.

The marker line looks like this:

1.303	2018/07/03 08:38:11.000000	2926.4850	DLTV	DLTV	DLTV	control	response	MARKER
-------	----------------------------	-----------	------	------	------	---------	----------	--------

These marker messages can be automatically highlighted in green color, see in ?? in Settings / Project Other.

But keep in mind: Marker messages are "control messages". So they are not displayed in case "Write Control Messages to log file" is not enabled. See ?? in Settings / Project Other.

4.5.11 Logging only mode

In case you only want to record traces for later analysis without viewing them in the DLT message table, it is recommended to enable the "Logging only mode". All plugins and the main table view are disabled. So you reduce performance impact and the probability of a viewer crash e.g. due to a faulty plugin.

See ??, "Viewer Configuration Settings/Project Other", how to enable it.

Basically you also can use the commandline tool dlt-receive to receive dlt messages and write them to a file. It is part of the dlt-daemon and so only is available on Linux systems.

Example:

```
dlt-receive -o ./logit.dlt 160.48.199.99
```

4.6 Working with Filters

You can find some example filters in your viewer installation directories and in the source code in directory "filters":

```
* control_messages.dlf
* error_fatal_messages.dlf
* message_buffer_overflow.dlf
* software_version.dlf
```

In case you define filters please take into account that message filtering has performance impact. It is recommended to combine several search parameters in one filter definition instead of cascading multiple filters. When searching for payload content you can improve performance if you also limit the filter using a certain Ctid or Apid where you expect your message in.

CAUTION: *More and complex filters slow down the viewer !*

4.6.1 How to create a filter

There are several ways to add a filter:

1. Menu "Filter"→"Filter Add"
2. On a context or application item in the Config tab of the Project Panel: selecting right mouse button "Filter Add". The available EcuId, Apid, Ctid will be filled in already.
3. On a message in the message table view select a line and right mouse button "Filter Add". All message elements except log levels are filled in already.

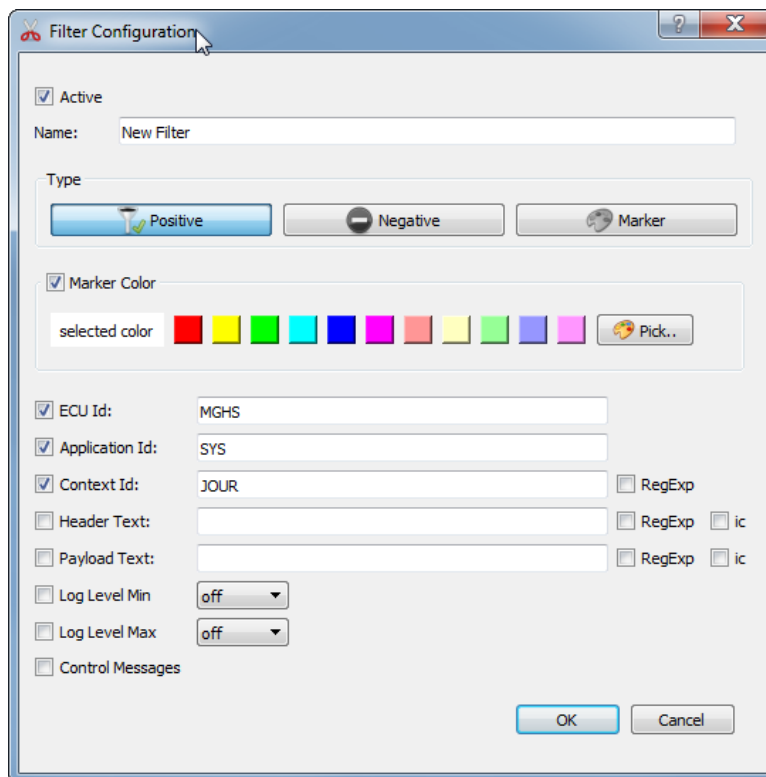


Figure 25: Filter configuration

1. Name: any name describing your filter function
2. Type:
Positive (default): all hits which match the filter will be shown in the message table

Negative: messages which fit the filter will NOT be shown in the message table
Marker: messages which fit the filter will just be highlighted with the chosen color

3. Marker color: choose a proposed color or create one from the color palette
4. ECU Id
5. Application Id
6. Context Id
7. Header Text: the string consists of all the message header data like Time, Timestamp, EcuId ...
(regardless if displayed in the table view)
8. Payload Text
9. Log Level Min
10. Log Level Max
11. Control Messages

4.6.2 Sort by Time

If the filters are enabled, even if there is no active filter applied, you can select "Sorted by Time" in the Filter tab of the Project Panel. In this case all messages will be sorted by received time. E.g. in case you load several logfiles which contain messages from different ECUs you will get the messages sorted by the receive time, not by the message index.

CAUTION: Take care of the (target) Timestamp - it may happen that messages are not received in the order like they are emitted by the specific application on the target but like they are sent out by the target system

4.6.3 Using regular expressions

You can create more complex filters by making use of the Qt regular expressions, e.g. using wildcards and so on. To enable regular expressions select the checkboxes "regex" available for Ctid, Header Text and Payload Text.

CAUTION: Regular expressions allow complex and conditional search criteria but can slow down the filter performance tremendously

4.6.4 Recommendations and examples

4.6.4.1 Filtering for multiple context IDs

For filtering multiple Ctids use one filter with "TC|HADT|PRER|GNRL" as a regular expression in "Context Id" instead of defining four separate filters.

4.6.4.2 Filtering for message payload containing multiple strings

E.g. in case you want to filter out all lines where the payload contains the strings "getBatterieData" or "message start" use:

`(getBatterieData|message start)`


as "Payload Text" and enable RegExp in the filter configuration

4.6.4.3 Filtering for certain loglevel only

If you only want to see messages between loglevels fatal and warn you should set Log Level Min to warn and Log Level Max to fatal

4.7 Project Files

You can save project specific settings like the status of plugins (including used configuration files), active filters, ECU autoconnect and so on for later reuse in a project file. The project file extension is "*.dlp". The project file rules the standard configuration settings of config.ini. You also can specify a project file as default to be automatically loaded at viewer start, see ?? in ??.

Projectfiles can also be specified in a command line call and are especially helpful here. Select Menu "Project" to manage project files. To save the current settings you also can press the "save project" icon .

4.8 Command line mode

If you call the DLT Viewer on command line you get the following output:

4.8.1 Commandline help

dlt_viewer -h (Linux) or dlt_viewer.exe -h (Windows)

Usage: dlt_viewer [OPTIONS]

Options:

```
-h Print usage
-p projectfile          Loading project file on startup (must end with .dlp)
-l logfile              Loading logfile on startup (must end with .dlt)
-f filterfile           Loading filterfile on startup (must end with .dlf)
-s or --silent          Enable silent mode without warning message boxes.
-v or --version         Only show version and buildtime information
-c logfile textfile     Convert logfile file to textfile (logfile must end with .dlt)
-u Conversion will be done in UTF8 instead of ASCII
-csv Conversion will be done in CSV format
-d Conversion will NOT be done, save in dlt file format again instead
-dd Conversion will NOT be done, save as decoded messages in dlt format
-e "plugin|command|param1|...|param<n>" Execute a plugin command with <n> parameters.
```

Examples:

```
dlt_viewer -c ./traces/trace.dlt ./trace.txt
dlt_viewer -s -c -u ./trace/trace.dlt ./trace.txt
dlt_viewer -s -d -c ./trace/trace.dlt ./trace.dlt
dlt_viewer -s -p ./proj/decodeded.dlp -dd -c ./trace/trace.dlt ./trace.dlt
dlt_viewer -s -csv -c ./trace/trace.dlt ./trace.csv
dlt_viewer -s -d -f ./filter/filter.dlf -c ./trace/trace.dlt ./filteredtrace.dlt
dlt_viewer -p ./proj/export.dlp -l ./trace/trace.dlt -e "Filetransfer Plugin|export|./ftransferdir"
```

4.8.2 Command line run

You can just start the viewer on command line to get more detailed output on console. As on windows the console currently is closed when giving no parameter you have to use -s or -l as a workaround. At the beginning you get information about DLT Viewer version, build date and plugin status. Also you get some debug output on what the viewer actual is doing.

Example output:

```
C:\01-viewer\r690>dlt_viewer.exe -s
Enable silent mode
Start "dlt_viewer.exe"
Build time Jun 15 2018 09:24:40
Version 2.19.0 WORKING
*****
Loading plugin "DLT Viewer Plugin" "1.0.1"
Loading plugin "Filetransfer Plugin" "1.2.1"
Loading plugin "Lifecycle Plugin" "1.3.1"
Loading plugin "Non Verbose Mode Plugin" "1.0.0"
Activate plugin "DLT Viewer Plugin" 1.0.1
Try to connect to ECU "160.48.199.99" "12:55:43"
Reconnect timeout for "160.48.199.99"
...
```

4.8.3 Examples

Command line usage examples:

4.8.3.1 Extract files from logfile using the fileransfer plugin

You can use the Filetransfer Plugin to extract files out of a stored dlt file e.g. in an automated test setup. To do this, call the Viewer with this command and parameters:

```
dlt_viewer.exe -p x.dlp -l y.dlt -e "Filetransfer Plugin|export|[path_to_extract]"
```

- **dlt_viewer.exe:**
The executable of the DLT Viewer. E.g. dlt_viewer.exe in Windows, dlt_viewer in Linux
- **x.dlp:**
A project settings file which has the filetransfer properly configured, e.g. by setting a valid plugin xml file (if needed)
- **y.dlt:**
Path to a dlt trace file containing embedded files which shall be extracted
- **-e:**
triggers the plugin internal command (depending on plugin implementation)
- **"Filetransfer Plugin":**
Filetransfer Plugin will be called for the command interface
- **export:**
the plugin internal function "export" is addressed
- **[path_to_extract]:**
Path to the folder where you want to extract all file dumps. If you apply this for a lot of files, think about using individual paths, to avoid overwriting files with identical names.

4.8.3.2 Command line start of GUI with dedicated logfile

```
dlt_viewer -l ./example.dlt
```

4.8.3.3 Command line start of GUI with dedicated logfile and filterfile

```
dlt_viewer -l ./example.dlt -f ./filter.dlf
```

4.8.3.4 Convert to ASCII file

```
dlt_viewer -c ./example.dlt ./example.txt
```

4.8.3.5 Convert to UTF8 file

```
dlt_viewer -u -c ./example.dlt ./example.txt
```

4.8.3.6 Convert to ASCII file in silent mode

```
dlt_viewer -s -c ./example.dlt ./example.txt
```

4.8.3.7 Filter and convert to ASCII file

```
dlt_viewer -f /viewertests/filterfiles/export.dlf -c ./example.dlt ./example.txt
```

4.8.3.8 Filter, decode and convert to ASCII file

```
dlt_viewer -p /viewertests/filterfiles/example.dlp -c ./example.dlt ./example.txt
```

4.8.3.9 Filter and save as DLT file

```
dlt_viewer -s -d -f /filterfiles/filter.dlf -c ./example.dlt ./example_filtered.dlt
```

4.8.3.10 Filter, decode and save as DLT file

```
dlt_viewer -s -dd -f /project/example.dlp -c ./example.dlt ./example_decoded.dlt
```

4.8.3.11 Decode and save as DLT file

```
dlt_viewer -p ./export.dlp -l ./filetransfer.dlt -e "Filetransfer Plugin|export|./ft_dir"
```

4.8.4 Shell script examples of command line calls

You also can place dlt viewer command line calls to be used in shell scripts for use in automated logfile processing, e.g. in a Jenkins environment and e.g. make use of Linux commands like grep, wc , awk, sed on converted dlt logs.

Example:

```
#!/bin/bash
for files in `ls *.dlt`
do
    dlt_viewer -s -f ./dltfilters/Filter_v5.dlf -c $files ${files}_filtered.txt
done
```

4.9 Send DLT injection

If implemented on the target you can send triggers and data to a specific context of an application. So you can change variable values in a running application or just trigger any kind of functions. For more details see the DLT Daemon documentation.

To open the injection dialog select the choosen Ctid the Control tab of the Project Panel and select right mouse button "Send injection".

E.g. your application ID is "myid", context ID is "myct" you can send some data by opening the injection dialog.

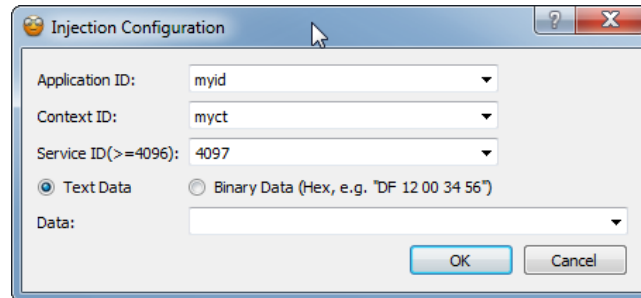


Figure 26: Injection dialog

1. Application ID
2. Context ID
3. Service ID - valid service IDs start from 4096
4. Text Data / Binary Data - any data you want to be processed in your application

You can also evaluate the example implementation dlt-speed-app to see how the message injection works.

4.9.1 DLT injection via command line

Although this subject is not part of the DLT Viewer I would like to emphasize that DLT injections also can be issued by aid of the command line tool dlt-control coming with the dlt daemon. You can use it e.g. in a test script. So the restriction is: it only is available for Linux.

Example:

```
dlt-control localhost -a SPEE -c MISC -s 4097 -x "01 02 03"
Send injection message:
AppId: SPEE
ConId: MISC
ServiceId: 4097
Message: 01 02 03
Size: 3
```

4.9.2 Send injection from a plugin

In case you want to send an injection from within a plugin you can make use of the function

```
QDltControl:: sendInjection(int index,QString applicationId,QString contextId,int serviceId,QByt
```

It is part of the Controlplugin interface. You can find an example in the dlt-speed-plugin source code also.

5 Building the DLT Viewer

This is not a comprehensive instruction how to build the DLT Viewer and Parser. It just is an example how the viewer can be build. On more information also see file INSTALL.txt.

5.1 Windows

If you check out (clone) the DLT Viewer you find some *.bat file which can be used to build the Viewer on command line. Building also was verified and performed using the QT Creator using Qt 5.5.1 + MSVC 2013, Qt5.6.1 + MSVC 2015, Qt5.8 + MSVC 2015 as well as Qt5.12.4 + MSVC 2017

5.1.1 Windows build example command line

5.1.1.1 Preconditions

Download and install:

qt-opensource-windows-x86-msvc2015_64-5.8.0.exe and
wdexpress_full_2015.exe

5.1.1.1.1 Viewer Windows build In the root folder of the viewer source you find some windows bat files. To build the viewer you have to set some parameters regarding your local installation. Example:

```
set ARCHITECTURE=x86_amd64
set MSVC_DIR=C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC
set QTDIR=c:\Qt\Qt5.8.0\5.8\msvc2015_64
set DLT_VIEWER_SDK_DIR=c:\DltViewer
.\build_sdk_windows_qt5_MSVC.bat
```

You find the artefacts in the folder

c:\DltViewer.

5.1.1.1.2 Parser Windows build

```
set ARCHITECTURE=x86_amd64
set MSVC_DIR=C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC
set QTDIR=c:\Qt\Qt5.8.0\5.8\msvc2015_64
.\build_parser_windows_qt5_MSVC.bat
```

You find the artefacts in the folder c:

DltParser

5.2 Linux

The example was build on Ubuntu 16.04 with QT 5.5.1 and 18.04 with QT 5.9.5

5.2.1 Ubuntu 18.04 build example command line

5.2.1.1 Preconditions

Get these packages:

```
sudo apt-get install g++ qt5-default libqt5serialport5-dev
```

5.2.1.2 Using cmake

```
sudo apt-get install cmake
In the root directory do:
mkdir build
cd build
cmake ..
make
or
make install
```

You find the dlt_viewer and the dlt_parser in subdirectory "bin" ...

5.2.1.3 Using qmake

In the Viewer root directory call

```
cmake
make
```

Find the resulting binaries in the subdirectory "release"

5.2.1.4 Using qmake building deb packages

This example additionally creates debian packages for you.

```
sudo apt-get install git devscripts debhelper
```

```
In the root directory call:
./build_viewer_debs.sh
```

Find the resulting debian packages in ./debtemp

For installation:

```
sudo apt-get install ./genivi-dlt-viewer
```

6 Building the documentation

6.1 DLT Viewer user manual

The documentation was changed from asciidoc to L^AT_EX, in September 2018.
and verified on an Ubuntu 18.04 installation.

```
sudo apt-get install texlive texlive-latex-extra
```

To create a pdf format output use:

```
pdflatex dlt-viewer_user_manual.tex
```

6.2 DLT Viewer Plugins Programming Guide

On Linux call convert.sh in the doc folder to get a resulting pdf.

6.3 Doxygen documentation

On Linux:

```
Install doxygen and graphviz  
Change into project directory  
doxygen sdk/doxygen_dlt_viewer_plugininterface.cfg  
(Optional) doxygen sdk/doxygen_dlt_viewer.cfg  
(Optional) doxygen sdk/doxygen_dlt_viewer_qdlt.cfg
```

You will find the documentation in the subdirectory DLT-Viewer in the doc directory.

7 Create DLT Viewer Plugins

In order to create new custom plugins it may be the fastest way to take one of the examples provided. So here is an instruction how to add the example dlt-speed-plugin to the active build configuration. First copy the folder examples/dlt-speed-plugin to the folder plugins of the Viewer. This example uses qwt and so you will have to take care that qwt is available on your system.

7.1 Linux

Install QWT: `sudo apt-get install libqwt-qt5-dev`

7.1.1 qmake

Now add the plugin to plugins/plugin.pro

```
SUBDIRS += dlt-speed-plugin
```

Build the Viewer:

```
qmake BuildDltViewer.pro
make
```

You will find the Viewer in release/dlt_viewer and the plugin in release/plugins/libspeedplugin.so

7.1.2 cmake

Add this line to plugins/CMakeList.txt:

```
add_subdirectory(dlt-speed-plugin)
```

and build the Viewer:

```
mkdir -p build
cd build
cmake ..
make
```

You will find the Viewer in build/dlt-viewer and the plugin in build/plugins/libspeedplugin.so

7.2 Windows

7.2.1 Build qwt

For Windows download "qwt-6.1.3.zip", build and install it using the bat file provided:

`build_qwt_windows_qt5_MSVC.bat`

Example:

```
set ARCHITECTURE=x86_amd64
set QTDIR=c:\Qt\Qt5.8.0\5.8\msvc2015_64
set MSVC_DIR=C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC
set DLT_VIEWER_SDK_DIR=c:\DltViewerSDK
set QWT_DIR=C:\Qwt-6.1.3_2015_5.8_64bit
set QMAKEFEATURES=%QWT_DIR%\features
call .\build_qwt_windows_qt5_MSVC.bat
```

In this example QWT will be installed in

`C:\Qwt-6.1.3_2015_5.8_64bit`

7.2.2 Build Viewer using qmake

Now add the plugin to plugins/plugin.pro

```
SUBDIRS += dlt-speed-plugin
```

and add these lines to: plugins/plugins.prj

```
CONFIG += qwt
```

```
# Directories
QWT_DIR = $$QWT_DIR
!isEmpty(QWT_DIR) {
    QWT_INSTALL_PREFIX = $$QWT_DIR
} else {
    QWT_INSTALL_PREFIX = C:\\Qwt-6.1.3
}

...

# Include path
win32:INCLUDEPATH += $$QWT_INSTALL_PREFIX\\include

...

### Additional Library path ###
win32:QMAKE_LIBDIR += $$QWT_INSTALL_PREFIX\\lib
```

You also have to remove the comment from the line

```
rem copy %BUILD_DIR%\plugins\speedplugin.dll %DLT_VIEWER_SDK_DIR%\plugins
```

in build_sdk_windows_qt5_MSVC.bat

Now call the build bat file:

```
set ARCHITECTURE=x86_amd64
set MSVC_DIR=C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC
set QTDIR=c:\Qt\Qt5.8.0\5.8\msvc2015_64
set DLT_VIEWER_SDK_DIR=c:\DltViewer
.\build_sdk_windows_qt5_MSVC.bat
```

Viewer build example (installation paths may differ) :

The directory DLT_VIEWER_SDK_DIR will contain the Viewer binary as before, the additional plugin can be found in the plugin subdirectory.

7.3 Demo application on Linux target

To see the functional basics of the plugin you will need a Linux target with a running DLT Daemon and an application sending messages to the daemon. About how building and installing the DLT Daemon please have a look at the DLT Daemon documentation. Once the daemon is installed, building the demo application is quite easy.

E.g.

```
cd examples/ dlt-speed-app
cmake .
make
sudo ./dlt-speed-app
```

Now you just need to set up a connection to your target in the DLT Viewer and activate the speed plugin.

8 Known issues

Limitations:

maximum log file size basically limited by the file system, but it is recommended to use < 10 GB
performance is reduced Crashes at about 21 GB on a 64 bit Ubuntu machine with 64GB RAM.