TP/Diagnostics

BLF Logging Format

Specification

Version 1.1 of 2020-02-07

Status	Completed
Publisher	Vector Informatik GmbH

Document Management

Release

Revision list

Version	Date	Editor	Section	Changes, comments
1.0	2016-01-28	Mar	All	Initial version created
1.0.1	2017-02-02	Mom	all	CI and layout
1.1	2020-02-07	vsn	3	Public API uses standard types, e.g. uint32_t instead of DWORD. Mentioned Linux libbinlog.so

VECTOR

Contents

1 Disclaimer

Severability clause - Restrictions for the usage of Vector logging data formats outside of Vector products

The format specification / access functions for the Vector BLF and ASC logging data formats are made available under the restrictions and conditions cited hereafter.

Please note that Vector Informatik neither gives any guarantee nor assumes any liability beyond compulsory legal regulations for the BLF or ASC logging format respectively as well as for the access functions to the single objects.

Vector Informatik disclaims all liability for errors which might be contained in the access functions or the format specification itself.

Vector Informatik does neither provide support for the integration into your software nor for problems occurring inside your software on the customer side.

Beyond that Vector Informatik reserves the right to change the BLF or ASC data format respectively anytime without prior notification. Therefore, the compatibility of the format is not ensured.

2 Overview

This document specifies the format of transport protocol and diagnostics events in the CANoe/CANalyzer BLF logging. The described structures can be used to read and write BLF logging files using the binlog.dll, libbinlog.so, which can be found in the CANoe/CANalyzer User Data folder:

<UserDataFolder>\Programming\BLF_Logging

3 Format Description

3.1 VBLDiagRequestInterpretation

For diagnostic requests sent by CANoe, the target ECU, the active diagnostic variant and the used diagnostic service are logged.

This information ensures that the request and response are interpreted correctly in CANoe.

Corresponding object type: `BL_OBJ_TYPE_DIAG_REQUEST_INTERPRETATION`

Object available starting from CANoe/CANalyzer version 9.0

Parameter	Type	Description
mHeader	VBLObjectHeader	
mDiagDescriptionHandle	uint32_t	Unique ID identifying the used diagnostic description (ECU)
mDiagVariantHandle	uint32_t	Unique ID identifying the used diagnostic variant
mDiagServiceHandle	uint32_t	Unique ID identifying the used diagnostic service
mEcuQualifierLength	uint32_t	Length of mEcuQualifier without terminating null character.
mVariantQualifierLength	uint32_t	Length of mVariantQualifier without terminating null character.
mServiceQualifierLength	uint32_t	Length of mServiceQualifier without terminating null character.
mEcuQualifier	BL_LPSTR	Qualifier of the ECU the request was sent to
mVariantQualifier	BL_LPSTR	Qualifier of the active diagnostic variant
mServiceQualifier	BL_LPSTR	Qualifier of the diagnostic service

Note: The size of a VBLDiagRequestInterpretation object depends on the length of the strings. To calculate the size correctly you have to add the text lengths (in bytes!) to the object size:

```
VBLDiagRequestInterpretation object;
 object.mHeader.mBase.mObjectSize
= sizeof(VBLDiagRequestInterpretation)
+ object.mExecutingObjectNameLength
+ object.mNameLength
+ object.mTextLength;
```