FAW HS7

IVI System Design

LOG-API User Guide

Released on the: 2016.10.08

Version Number: 0.500

Neusoft Automotives ChangChun

**Version History**

| No. | Version | Section | Brief Description | Date | Rseponsible |
| --- | --- | --- | --- | --- | --- |
| 1 | 0.500 | All | Initial Create | 2016-10-25 | ZhangShuang |
|  | 0.99 | 3.6 3.8 3.10 | 去掉3.6节中关于打log函数参数数量的限制；增加了3.8节中获得格式化字符串函数的std::string版本；增加了3.10节获取函数头的宏的说明 | 2018-04-03 | ZhangShuang |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Content

[1 概述 4](#_Toc510526773)

[2 相关文件 4](#_Toc510526774)

[3 接口说明 4](#_Toc510526775)

[3.1 LOG-API使用约束 4](#_Toc510526776)

[3.2 LOG-API初始化 4](#_Toc510526777)

[3.3 注册ContextID 4](#_Toc510526778)

[3.4 注销ContextID 5](#_Toc510526779)

[3.5 Log级别 5](#_Toc510526780)

[3.6 输出log 5](#_Toc510526781)

[3.7 无符号整形转换为16进制字符串 10](#_Toc510526782)

[3.8 获得格式化后的字符串 10](#_Toc510526783)

[3.9 域log变量 11](#_Toc510526784)

[3.10 获取函数头的宏 11](#_Toc510526785)

[4 Refrences 12](#_Toc510526786)

# 概述

本文档主要描述LOG-API。LOG工具主要负责应用程序的日志记录，所有应用程序必须使用此接口。

查看log的方法包括使用DLT-viewer和使用命令行工具两种方法。

# 相关文件

头文件：/usr/include/appSDK/log.h

库文件：/usr/lib/libama-log.so

# 接口说明

## LOG-API使用约束

约束条件：

* 无

## LOG-API初始化

LOG-API使用之前需要初始化。初始化时使用的字符串为当前进程的application ID。因为log系统接收了系统所有进程的log，所以指定一个application ID是非常有用的。可以在dlt-viewer中利用application ID对log进行过滤。

需要注意的是，APPManager的SDK在初始化时会自动为app/service初始化LOG-API，并分配一个application ID，如果在这之后再次初始化，新的application ID会覆盖掉旧的。这样虽然不会导致严重的后果，但是不建议这样做。

|  |  |  |
| --- | --- | --- |
| *declare* | unsigned int GetSurfaceID(void) | |
| *description* | 获取surfaceID | |
| *parameter* | type void | null |
| *return* | surfaceID的值 | |

## 注册ContextID

为了方便对相同application ID的log进行区分，以划分统一进程中的不同模块，LOG-API支持注册Context ID。在输出log时附加上注册过的Context ID就可以方便的在dlt-viewer中对不同Context ID的log进行筛选。如果输出log时不指定context ID，则系统默认指定一个叫def的context ID。

需要注意的是，不管是application ID还是context ID，它的名字长度不能超过4个字母。

|  |  |  |
| --- | --- | --- |
| *declare* | void RegisterContext(LogContext &handle, const char \*contextid) | |
| *description* | 注册Log Context | |
| *parameter* | LogContext &handle, const char \*contextid | LogContext &  const char \* |
| *return* | AMPID的值 | |

## 注销ContextID

当Context ID不再使用时，需要将注册过的Context ID注销。注意，使用已经注销或者未经注册的Context ID会导致程序崩溃！

|  |  |  |
| --- | --- | --- |
| *declare* | void UnregisterContext(LogContext &handle); | |
| *description* | 注销Log Context | |
| *parameter* | LogContext & | handle |
| *return* | Null | |

## Log级别

Log系统支持对log消息进行分级。有如下几个级别：

|  |  |
| --- | --- |
| Log级别 | 描述 |
| LOG\_ERRO | 不可恢复的严重错误，发生后系统无法继续正常运行。 |
| LOG\_WARN | 预料之中的错误，系统运行不受影响。但是仍然需要引起关注。 |
| LOG\_INFM | 供常规使用。如状态变化，操作完成等。用于判断程序执行了哪些操作。 |
| LOG\_DEBG | 调试用log。在不方便调试的情况下输出一些调试人员感兴趣的信息。 |
| LOG\_VBOS | 通常不需要关注的log。常在函数的出入口用这些log标记函数的执行区间。参见[域log](#_域log变量) |

通过枚举类型LogLevelType在输出log时可以指定log级别。通常来说，不需要显式指定log的级别，因为log输出函数对此已经进行了封装。

## 输出log

Log系列函数提供了类似std::cout的输出功能，但是它不使用“<<”运算符，而是拼接起log函数的所有参数。~~参数的上限是10个（不包括context和logLevel）~~。同时还支持类似printf的格式化字符串输出。没有参数上限。

下面列出了所有的log函数：

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void log(LogContext\* const context, LogLevelType loglevel, ...) | |
| *description* | 输出指定级别，指定context的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logUI(LogLevelType loglevel, ...) | |
| *description* | 输出指定级别，context ID为UI的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logError(...) | |
| *description* | 输出Error级别，context ID为def的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logError(LogContext context,...) | |
| *description* | 输出Error级别，指定context ID的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logErrorUI(...) | |
| *description* | 输出Error级别， context ID为UI的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logWarn(...) | |
| *description* | 输出Warn级别， context ID为def的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logWarn(LogContext context,...) | |
| *description* | 输出Warn级别，指定context ID的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | ttemplate<typename T> void logWarnUI(..) | |
| *description* | 输出Warn级别， context ID为UI的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logInfo(...) | |
| *description* | 输出Info级别， context ID为def的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logInfo(LogContext context,...) | |
| *description* | 输出Info级别，指定 context ID的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logInfoUI(...) | |
| *description* | 输出Info级别， context ID为UI的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logDebug(...) | |
| *description* | 输出Debug级别， context ID为def的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logDebug(LogContext context,...) | |
| *description* | 输出Debug级别， 指定context ID的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logDebugUI(...) | |
| *description* | 输出Debug级别， context ID为UI的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logVerbose(...) | |
| *description* | 输出Verbose级别， context ID为def的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logVerbose(LogContext context,...) | |
| *description* | 输出Verbose级别， 指定context ID的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | template<typename T> void logVerboseUI(...) | |
| *description* | 输出Verbose级别， context ID为UI的log | |
| *parameter* | 任意基本类型~~，不超过10个~~ | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logF(LogContext\* const context, LogLevelType loglevel, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出指定log级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logUIF(LogLevelType loglevel, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出指定log级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logErrorF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Error级别， context ID为def的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logErrorF(LogContext context, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Error级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logErrorUIF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Error级别，context ID为UI的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logWarnF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Warn级别，context ID为def的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logWarnF(LogContext context, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Warn级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logWarnUIF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Warn级别，context ID为UI的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logInfoF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Info级别，context ID为def的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logInfoF(LogContext context, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Info级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logInfoUIF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Info级别，context ID为UI的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logDebugF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Debug级别，context ID为def的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logDebugF(LogContext context, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Debug级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logDebugUIF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Debug级别，context ID为UI的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logVerboseF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Verbose级别，context ID为def的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logVerboseF(LogContext context, const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Verbose级别，指定context ID的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

|  |  |  |
| --- | --- | --- |
| *declare* | static void logVerboseUIF(const char \*format, ...) | |
| *description* | 以格式化字符串的形式，输出Verbose级别，context ID为UI的log。 | |
| *parameter* | 任意基本类型 | T |
| *return* | Null | |

## 无符号整形转换为16进制字符串

该函数支持将最多64位整形变量转换成16进制字符串。

|  |  |  |
| --- | --- | --- |
| *declare* | static std::string ToHEX(unsigned long long ULL) | |
| *description* | 将最多64位整形变量转化为16进制字符串 | |
| *parameter* | unsigned long long | ULL |
| *return* | 参数对应的16进制字符串。 | |

## 获得格式化后的字符串

该函数用于将格式化字符串及其参数转换成字符串。

|  |  |  |
| --- | --- | --- |
| *declare* | static std::string formatText(const char \*format, ...) | |
| *description* | 将格式化字符串及其参数转换成字符串 | |
| *parameter* | const char \* | format |
| *return* | 转换后得到的字符串 | |

还有std::string版本：

|  |  |  |
| --- | --- | --- |
| *declare* | static std::string formatText(const std::string &format, ...) | |
| *description* | 将格式化字符串及其参数转换成字符串 | |
| *parameter* | const std::string & | format |
| *return* | 转换后得到的字符串 | |

## 域log变量

因为当前的工作环境调试不便，很多时候为了便于定位问题，需要在函数的出入口打标记log。但是在多个出口的函数中这样做很麻烦。一旦标记log的内容发生变化，维护就更加麻烦了。域log变量正是为了解决这一问题引入的。只要在函数入口处声明该变量，并传入标记log的内容，在函数退出时，变量自动析构，输出离开函数的log，非常方便。但是由于它的构造函数只支持输入一个字符串，所以有时需要跟[formatText](#_获得格式化后的字符串)函数配合使用才能支持复杂的log。

由于标记函数出入口的log被认为是Verbose级log，所以目前域log变量只支持Verbose log级别。

|  |  |  |
| --- | --- | --- |
| *declare* | class DomainVerboseLog | |
| *description* | 在进入/离开所在定义域时，输出log。 | |
| *parameter* | const LogContext& context, const std::string& name | const LogContext&  const std::string& |
| *return* | Null | |

## 获取函数头的宏

打log时往往需要在log开始添加函数头。以下宏可以实现该功能：

|  |  |  |
| --- | --- | --- |
| *declare* | \_\_STR\_FUNCTION\_\_ | |
| *description* | 返回一个临时std::string对象，包含了形如“virtual void ClassName::FunctionName”的函数头。 | |
| *parameter* | 无 |  |
| *return* | std::string | |

|  |  |  |
| --- | --- | --- |
| *declare* | \_\_STR\_FUNCTIONP\_\_ | |
| *description* | 返回一个临时std::string对象，包含了形如“virtual void ClassName::FunctionName()”的函数头。(P = parentheses) | |
| *parameter* | 无 |  |
| *return* | std::string | |

|  |  |  |
| --- | --- | --- |
| *declare* | \_\_STR\_FUNCTIONPC\_\_ | |
| *description* | 返回一个临时std::string对象，包含了形如“virtual void ClassName::FunctionName():”的函数头。(C = colon) | |
| *parameter* | 无 |  |
| *return* | std::string | |

|  |  |  |
| --- | --- | --- |
| *declare* | LOG\_HEAD | |
| *description* | \_\_STR\_FUNCTIONPC\_\_的别名。但是这个名字突出了其作为log头的属性。 | |
| *parameter* | 无 |  |
| *return* | std::string | |

# Refrences

无