UDP Application Logging
A Proposal for an ad-hoc Standard

James H. Price

February 18, 2002

Table of Contents

Introduction and Purpose

This proposed standard defines a common message format that can be used by a variety of applications and devices. A display program on a PC can receive the messages and display them or log them to disk. Because the message format identifies both the source of the message (application name), and the severity of the message, the display program can use different colors or other display mechanics to indicate those messages of most interest to the user.

The intent of the messaging mechanism is to allow low-overhead one-to-one, one-to-many, many-to-one and many-to-many communication. That is, a single display program might receive messages from a single application (or device), or multiple applications (or devices), or multiple display programs receive messages from one or more applications.

The goal is to provide moderately reliable message delivery within a local area network. For this reason, UDP messages (sent to either individual IP addresses, or network broadcast addresses) shall be used. Nothing about the protocol should necessarily preclude its use over a larger network, except that the unreliability of UDP may cause some messages to get lost. Developers expecting reliable transport over large networks may look to standards other than this proposal.

The Standard Message

The standard message consists of the following fixed-length fields:

Field Name	Number of Bytes	Notes
Application Name	16	Null-terminated ASCII string. Identifies the application sending the message. Only 15 bytes may be used for the name. Unused trailing bytes should be set to 0
Service Name	16	Null-terminated ASCII string. Identifies the service (within the application) sending the message. (An application may have more than one service.) Only 15 bytes may be used for the name. Unused trailing bytes should be set to 0
OpCode	1	Binary value. Values assigned as follows: 0 – Textual log message (Message body contains text.) 1 – Meter message (Message body contains one or more name-values pairs of the format: ”*fieldname=value*,scalemax=*value*,scalemin=*value*” For example: level=127,scalemax=255,scalemin=0 indicates a field named "level" with a max value of 255 and minimum value of 0, and a current value of 127.
Severity	1	Binary value. A value of 0 indicates lowest severity. A value of 255 indicates the highest. Recommended usage: Messages of different severity may be displayed with different colored icons. 0-9 Black 10-49 Blue 50-99 Green 100-199 Yellow 200-255 Red Items of the display application should continue to display an icon indicating the highest priority message received until either: a) reset by the user or b) a message with priority 0-9 is received
Message Body	200	Null-terminated ASCII string. Unused trailing bytes should be set to 0.

Header File

Assuming that the "C" compiler uses byte-packed structures, the following header file defines the message:

```
// udpapplg.h

#ifndef UDPAPPLGH_INCLUDE
#define UDPAPPLGH_INCLUDE 1

#define UDP_APP_LOG_APP_NAME_SIZE 16
#define UDP_APP_LOG_SERVICE_NAME_SIZE 16
#define UDP_APP_LOG_PRIORITY_SIZE 1
#define UDP_APP_LOG_OPCODE_SIZE 1
#define UDP_APP_LOG_MSG_TXT_SIZE 200

#ifdef __cplusplus
extern "C" {
#endif

typedef struct udp_app_log_msg{
 char appname[UDP_APP_LOG_APP_NAME_SIZE];
 char servicename[UDP_APP_LOG_SERVICE_NAME_SIZE];
 char opcode[UDP_APP_LOG_OPCODE_SIZE];
 char priority[UDP_APP_LOG_PRIORITY_SIZE];
 char msgtxt[UDP_APP_LOG_MSG_TXT_SIZE];
 } UDP_APP_LOG_MSG;

#ifdef __cplusplus
}
#endif

#define UDP_APP_LOG_MAX_BLACK_PRIORITY 9
#define UDP_APP_LOG_MAX_BLUE_PRIORITY 49
#define UDP_APP_LOG_MAX_GREEN_PRIORITY 99
#define UDP_APP_LOG_MAX_YELLOW_PRIORITY 199
#define UDP_APP_LOG_MAX_RED_PRIORITY 255

#endif

// end udpapplg.h
```