Network Working Group R. Gerhards
Request for Comments: 5424 Adiscon GmbH
Obsoletes: 3164 March 2009
Category: Standards Track

The Syslog Protocol

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes the syslog protocol, which is used to convey event notification messages. This protocol utilizes a layered architecture, which allows the use of any number of transport protocols for transmission of syslog messages. It also provides a message format that allows vendor-specific extensions to be provided in a structured way.

This document has been written with the original design goals for traditional syslog in mind. The need for a new layered specification has arisen because standardization efforts for reliable and secure syslog extensions suffer from the lack of a Standards-Track and transport-independent RFC. Without this document, each other standard needs to define its own syslog packet format and transport mechanism, which over time will introduce subtle compatibility issues. This document tries to provide a foundation that syslog extensions can build on. This layered architecture approach also provides a solid basis that allows code to be written once for each syslog feature rather than once for each transport.

This document obsoletes RFC 3164.

Table of Contents

1. Introduction

This document describes a layered architecture for syslog. The goal of this architecture is to separate message content from message transport while enabling easy extensibility for each layer.

This document describes the standard format for syslog messages and outlines the concept of transport mappings. It also describes structured data elements, which can be used to transmit easily parseable, structured information, and allows for vendor extensions.

This document does not describe any storage format for syslog messages. It is beyond of the scope of the syslog protocol and is unnecessary for system interoperability.

This document has been written with the original design goals for traditional syslog in mind. The need for a new layered specification has arisen because standardization efforts for reliable and secure syslog extensions suffer from the lack of a Standards-Track and transport-independent RFC. Without this document, each other standard would need to define its own syslog packet format and transport mechanism, which over time will introduce subtle compatibility issues. This document tries to provide a foundation that syslog extensions can build on. This layered architecture approach also provides a solid basis that allows code to be written once for each syslog feature instead of once for each transport.

This document obsoletes RFC 3164, which is an Informational document describing some implementations found in the field.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Definitions

Syslog utilizes three layers:

- "syslog content" is the management information contained in a syslog message.
- The "syslog application" layer handles generation, interpretation, routing, and storage of syslog messages.
- The "syslog transport" layer puts messages on the wire and takes them off the wire.

Certain types of functions are performed at each conceptual layer:

- o An "originator" generates syslog content to be carried in a message.
- o A "collector" gathers syslog content for further analysis.
- o A "relay" forwards messages, accepting messages from originators or other relays and sending them to collectors or other relays.
- o A "transport sender" passes syslog messages to a specific transport protocol.
- o A "transport receiver" takes syslog messages from a specific transport protocol.

Diagram 1 shows the different entities separated by layer.

```
 +---------------------+    +---------------------+
 |  content            |    |  content            |
 |---------------------|    |---------------------|
 |  syslog application |    |  syslog application | (originator,
 |                     |    |                     |  collector, relay)
 |---------------------|    |---------------------|
 |  syslog transport   |    |  syslog transport   | (transport sender,
 |                     |    |                     | (transport receiver)
 +---------------------+    +---------------------+
           ^                           ^
           |                           |
            --------------------------
```

Diagram 1. Syslog Layers

4. Basic Principles

The following principles apply to syslog communication:

- o The syslog protocol does not provide acknowledgment of message delivery. Though some transports may provide status information, conceptually, syslog is a pure simplex communications protocol.
- o Originators and relays may be configured to send the same message to multiple collectors and relays.
- o Originator, relay, and collector functionality may reside on the same system.

4.1. Example Deployment Scenarios

Sample deployment scenarios are shown in Diagram 2. Other arrangements of these examples are also acceptable. As noted, in the following diagram, relays may send all or some of the messages that they receive and also send messages that they generate internally. The boxes represent syslog-enabled applications.

```
            +----------+         +---------+
            |Originator|---->----|Collector|
            +----------+         +---------+

            +----------+         +-----+         +---------+
            |Originator|---->----|Relay|---->----|Collector|
            +----------+         +-----+         +---------+

            +----------+     +-----+            +-----+     +---------+
            |Originator|-->--|Relay|-->--..-->--|Relay|-->--|Collector|
            +----------+     +-----+            +-----+     +---------+

            +----------+         +-----+         +---------+
            |Originator|---->----|Relay|---->----|Collector|
            |          |-+       +-----+         +---------+
            +----------+  \
                           \     +-----+         +---------+
                            +->--|Relay|---->----|Collector|
                                 +-----+         +---------+

            +----------+         +---------+
            |Originator|---->----|Collector|
            |          |-+       +---------+
            +----------+  \
                           \     +-----+         +---------+
                            +->--|Relay|---->----|Collector|
                                 +-----+         +---------+

            +----------+         +-----+            +---------+
            |Originator|---->----|Relay|---->-------|Collector|
            |          |-+       +-----+        +---|         |
            +----------+  \                    /    +---------+
                           \     +-----+      /
                            +->--|Relay|-->--/
                                 +-----+
```

```
            +----------+         +-----+                   +---------+
            |Originator|---->----|Relay|---->--------------|Collector|
            |          |-+       +-----+                +--|         |
            +----------+  \                            /   +---------+
                           \     +------------+       /
                            \    |+----------+|      /
                             +->-||Relay     ||->---/
                                 |+----------+|    /
                                 ||Originator||->-/
                                 |+----------+|
                                 +------------+
```

Diagram 2. Some Possible Syslog Deployment Scenarios

5. Transport Layer Protocol

This document does not specify any transport layer protocol. Instead, it describes the format of a syslog message in a transport layer independent way. Syslog transports are defined in other documents. One such transport is defined in [RFC5426] and is consistent with the traditional UDP transport. This transport is needed to maintain interoperability as the UDP transport has historically been used for the transmission of syslog messages.

Any syslog transport protocol MUST NOT deliberately alter the syslog message. If the transport protocol needs to perform temporary transformations at the transport sender, these transformations MUST be reversed by the transport protocol at the transport receiver so that the relay or collector will see an exact copy of the message generated by the originator or relay. Otherwise, end-to-end cryptographic verifiers (such as signatures) will be broken. Of course, message alteration might occur due to transmission errors or other problems. Guarding against such alterations is not within the scope of this document.

5.1. Minimum Required Transport Mapping

All implementations of this specification MUST support a TLS-based transport as described in [RFC5425].

All implementations of this specification SHOULD also support a UDP-based transport as described in [RFC5426].

It is RECOMMENDED that deployments of this specification use the TLS-based transport.

6. Syslog Message Format

The syslog message has the following ABNF [RFC5234] definition:

```
SYSLOG-MSG      = HEADER SP STRUCTURED-DATA [SP MSG]

HEADER          = PRI VERSION SP TIMESTAMP SP HOSTNAME
                  SP APP-NAME SP PROCID SP MSGID
PRI             = "<" PRIVAL ">"
PRIVAL          = 1*3DIGIT ; range 0 .. 191
VERSION         = NONZERO-DIGIT 0*2DIGIT
HOSTNAME        = NILVALUE / 1*255PRINTUSASCII

APP-NAME        = NILVALUE / 1*48PRINTUSASCII
PROCID          = NILVALUE / 1*128PRINTUSASCII
MSGID           = NILVALUE / 1*32PRINTUSASCII

TIMESTAMP       = NILVALUE / FULL-DATE "T" FULL-TIME
FULL-DATE       = DATE-FULLYEAR "-" DATE-MONTH "-" DATE-MDAY
DATE-FULLYEAR   = 4DIGIT
DATE-MONTH      = 2DIGIT  ; 01-12
DATE-MDAY       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                          ; month/year
FULL-TIME       = PARTIAL-TIME TIME-OFFSET
PARTIAL-TIME    = TIME-HOUR ":" TIME-MINUTE ":" TIME-SECOND
                  [TIME-SECFRAC]
TIME-HOUR       = 2DIGIT  ; 00-23
TIME-MINUTE     = 2DIGIT  ; 00-59
TIME-SECOND     = 2DIGIT  ; 00-59
TIME-SECFRAC    = "." 1*6DIGIT
TIME-OFFSET     = "Z" / TIME-NUMOFFSET
TIME-NUMOFFSET  = ("+" / "-") TIME-HOUR ":" TIME-MINUTE


STRUCTURED-DATA = NILVALUE / 1*SD-ELEMENT
SD-ELEMENT      = "[" SD-ID *(SP SD-PARAM) "]"
SD-PARAM        = PARAM-NAME "=" %d34 PARAM-VALUE %d34
SD-ID           = SD-NAME
PARAM-NAME      = SD-NAME
PARAM-VALUE     = UTF-8-STRING ; characters '"', '\' and
                               ; ']' MUST be escaped.
SD-NAME         = 1*32PRINTUSASCII
                  ; except '=', SP, ']', %d34 (")

MSG             = MSG-ANY / MSG-UTF8
MSG-ANY         = *OCTET ; not starting with BOM
MSG-UTF8        = BOM UTF-8-STRING
BOM             = %xEF.BB.BF
```

```
UTF-8-STRING    = *OCTET ; UTF-8 string as specified
                  ; in RFC 3629

OCTET           = %d00-255
SP              = %d32
PRINTUSASCII    = %d33-126
NONZERO-DIGIT   = %d49-57
DIGIT           = %d48 / NONZERO-DIGIT
NILVALUE        = "-"
```

6.1. Message Length

Syslog message size limits are dictated by the syslog transport mapping in use. There is no upper limit per se. Each transport mapping defines the minimum maximum required message length support, and the minimum maximum MUST be at least 480 octets in length.

Any transport receiver MUST be able to accept messages of up to and including 480 octets in length. All transport receiver implementations SHOULD be able to accept messages of up to and including 2048 octets in length. Transport receivers MAY receive messages larger than 2048 octets in length. If a transport receiver receives a message with a length larger than it supports, the transport receiver SHOULD truncate the payload. Alternatively, it MAY discard the message.

If a transport receiver truncates messages, the truncation MUST occur at the end of the message. After truncation, the message MAY contain invalid UTF-8 encoding or invalid STRUCTURED-DATA. The transport receiver MAY discard the message or MAY try to process as much as possible in this case.

6.2. HEADER

The character set used in the HEADER MUST be seven-bit ASCII in an eight-bit field as described in [RFC5234]. These are the ASCII codes as defined in "USA Standard Code for Information Interchange" [ANSI.X3-4.1968].

The header format is designed to provide some interoperability with older BSD-based syslog. For details on this, see Appendix A.1.

6.2.1. PRI

The PRI part MUST have three, four, or five characters and will be bound with angle brackets as the first and last characters. The PRI part starts with a leading "<" (’less-than’ character, %d60), followed by a number, which is followed by a ">" (’greater-than’ character, %d62). The number contained within these angle brackets is known as the Priority value (PRIVAL) and represents both the Facility and Severity. The Priority value consists of one, two, or three decimal integers (ABNF DIGITS) using values of %d48 (for "0") through %d57 (for "9").

Facility and Severity values are not normative but often used. They are described in the following tables for purely informational purposes. Facility values MUST be in the range of 0 to 23 inclusive.

Numerical Code	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon (note 2)
16	local use 0 (local0)
17	local use 1 (local1)
18	local use 2 (local2)
19	local use 3 (local3)
20	local use 4 (local4)
21	local use 5 (local5)
22	local use 6 (local6)
23	local use 7 (local7)

Table 1. Syslog Message Facilities

Each message Priority also has a decimal Severity level indicator. These are described in the following table along with their numerical values. Severity values MUST be in the range of 0 to 7 inclusive.

Numerical Code	Severity
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

Table 2. Syslog Message Severities

The Priority value is calculated by first multiplying the Facility number by 8 and then adding the numerical value of the Severity. For example, a kernel message (Facility=0) with a Severity of Emergency (Severity=0) would have a Priority value of 0. Also, a "local use 4" message (Facility=20) with a Severity of Notice (Severity=5) would have a Priority value of 165. In the PRI of a syslog message, these values would be placed between the angle brackets as <0> and <165> respectively. The only time a value of "0" follows the "<" is for the Priority value of "0". Otherwise, leading "0"s MUST NOT be used.

6.2.2. VERSION

The VERSION field denotes the version of the syslog protocol specification. The version number MUST be incremented for any new syslog protocol specification that changes any part of the HEADER format. Changes include the addition or removal of fields, or a change of syntax or semantics of existing fields. This document uses a VERSION value of "1". The VERSION values are IANA-assigned (Section 9.1) via the Standards Action method as described in [RFC5226].

6.2.3. TIMESTAMP

The TIMESTAMP field is a formalized timestamp derived from [RFC3339].

Whereas [RFC3339] makes allowances for multiple syntaxes, this document imposes further restrictions. The TIMESTAMP value MUST follow these restrictions:

- The "T" and "Z" characters in this syntax MUST be upper case.

- Usage of the "T" character is REQUIRED.

- Leap seconds MUST NOT be used.

The originator SHOULD include TIME-SECFRAC if its clock accuracy and performance permit. The "timeQuality" SD-ID described in Section 7.1 allows the originator to specify the accuracy and trustworthiness of the timestamp.

A syslog application MUST use the NILVALUE as TIMESTAMP if the syslog application is incapable of obtaining system time.

6.2.3.1. Examples

Example 1

```
1985-04-12T23:20:50.52Z
```

This represents 20 minutes and 50.52 seconds after the 23rd hour of 12 April 1985 in UTC.

Example 2

```
1985-04-12T19:20:50.52-04:00
```

This represents the same time as in example 1, but expressed in US Eastern Standard Time (observing daylight savings time).

Example 3

```
2003-10-11T22:14:15.003Z
```

This represents 11 October 2003 at 10:14:15pm, 3 milliseconds into the next second. The timestamp is in UTC. The timestamp provides millisecond resolution. The creator may have actually had a better resolution, but providing just three digits for the fractional part of a second does not tell us.

Example 4

```
2003-08-24T05:14:15.000003-07:00
```

This represents 24 August 2003 at 05:14:15am, 3 microseconds into the next second. The microsecond resolution is indicated by the additional digits in TIME-SECFRAC. The timestamp indicates that its local time is -7 hours from UTC. This timestamp might be created in the US Pacific time zone during daylight savings time.

Example 5 - An Invalid TIMESTAMP

    2003-08-24T05:14:15.000000003-07:00

This example is nearly the same as Example 4, but it is specifying TIME-SECFRAC in nanoseconds. This results in TIME-SECFRAC being longer than the allowed 6 digits, which invalidates it.

6.2.4. HOSTNAME

The HOSTNAME field identifies the machine that originally sent the syslog message.

The HOSTNAME field SHOULD contain the hostname and the domain name of the originator in the format specified in STD 13 [RFC1034]. This format is called a Fully Qualified Domain Name (FQDN) in this document.

In practice, not all syslog applications are able to provide an FQDN. As such, other values MAY also be present in HOSTNAME. This document makes provisions for using other values in such situations. A syslog application SHOULD provide the most specific available value first. The order of preference for the contents of the HOSTNAME field is as follows:

1. FQDN

2. Static IP address

3. hostname

4. Dynamic IP address

5. the NILVALUE

If an IPv4 address is used, it MUST be in the format of the dotted decimal notation as used in STD 13 [RFC1035]. If an IPv6 address is used, a valid textual representation as described in [RFC4291], Section 2.2, MUST be used.

Syslog applications SHOULD consistently use the same value in the HOSTNAME field for as long as possible.

The NILVALUE SHOULD only be used when the syslog application has no way to obtain its real hostname. This situation is considered highly unlikely.

6.2.5. APP-NAME

The APP-NAME field SHOULD identify the device or application that originated the message. It is a string without further semantics. It is intended for filtering messages on a relay or collector.

The NILVALUE MAY be used when the syslog application has no idea of its APP-NAME or cannot provide that information. It may be that a device is unable to provide that information either because of a local policy decision, or because the information is not available, or not applicable, on the device.

This field MAY be operator-assigned.

6.2.6. PROCID

PROCID is a value that is included in the message, having no interoperable meaning, except that a change in the value indicates there has been a discontinuity in syslog reporting. The field does not have any specific syntax or semantics; the value is implementation-dependent and/or operator-assigned. The NILVALUE MAY be used when no value is provided.

The PROCID field is often used to provide the process name or process ID associated with a syslog system. The NILVALUE might be used when a process ID is not available. On an embedded system without any operating system process ID, PROCID might be a reboot ID.

PROCID can enable log analyzers to detect discontinuities in syslog reporting by detecting a change in the syslog process ID. However, PROCID is not a reliable identification of a restarted process since the restarted syslog process might be assigned the same process ID as the previous syslog process.

PROCID can also be used to identify which messages belong to a group of messages. For example, an SMTP mail transfer agent might put its SMTP transaction ID into PROCID, which would allow the collector or relay to group messages based on the SMTP transaction.

6.2.7. MSGID

The MSGID SHOULD identify the type of message. For example, a firewall might use the MSGID "TCPIN" for incoming TCP traffic and the MSGID "TCPOUT" for outgoing TCP traffic. Messages with the same MSGID should reflect events of the same semantics. The MSGID itself is a string without further semantics. It is intended for filtering messages on a relay or collector.

The NILVALUE SHOULD be used when the syslog application does not, or cannot, provide any value.

This field MAY be operator-assigned.

6.3. STRUCTURED-DATA

STRUCTURED-DATA provides a mechanism to express information in a well defined, easily parseable and interpretable data format. There are multiple usage scenarios. For example, it may express meta-information about the syslog message or application-specific information such as traffic counters or IP addresses.

STRUCTURED-DATA can contain zero, one, or multiple structured data elements, which are referred to as "SD-ELEMENT" in this document.

In case of zero structured data elements, the STRUCTURED-DATA field MUST contain the NILVALUE.

The character set used in STRUCTURED-DATA MUST be seven-bit ASCII in an eight-bit field as described in [RFC5234]. These are the ASCII codes as defined in "USA Standard Code for Information Interchange" [ANSI.X3-4.1968]. An exception is the PARAM-VALUE field (see Section 6.3.3), in which UTF-8 encoding MUST be used.

A collector MAY ignore malformed STRUCTURED-DATA elements. A relay MUST forward malformed STRUCTURED-DATA without any alteration.

6.3.1. SD-ELEMENT

An SD-ELEMENT consists of a name and parameter name-value pairs. The name is referred to as SD-ID. The name-value pairs are referred to as "SD-PARAM".

6.3.2. SD-ID

SD-IDs are case-sensitive and uniquely identify the type and purpose of the SD-ELEMENT. The same SD-ID MUST NOT exist more than once in a message.

There are two formats for SD-ID names:

- Names that do not contain an at-sign ("@", ABNF %d64) are reserved to be assigned by IETF Review as described in BCP26 [RFC5226]. Currently, these are the names defined in Section 7. Names of this format are only valid if they are first registered with the IANA. Registered names MUST NOT contain an at-sign ('@', ABNF

%d64), an equal-sign ('=', ABNF %d61), a closing brace (']', ABNF %d93), a quote-character ('"', ABNF %d34), whitespace, or control characters (ASCII code 127 and codes 32 or less).

- Anyone can define additional SD-IDs using names in the format name@<private enterprise number>, e.g., "ourSDID@32473". The format of the part preceding the at-sign is not specified; however, these names MUST be printable US-ASCII strings, and MUST NOT contain an at-sign ('@', ABNF %d64), an equal-sign ('=', ABNF %d61), a closing brace (']', ABNF %d93), a quote-character ('"', ABNF %d34), whitespace, or control characters. The part following the at-sign MUST be a private enterprise number as specified in Section 7.2.2. Please note that throughout this document the value of 32473 is used for all private enterprise numbers. This value has been reserved by IANA to be used as an example number in documentation. Implementors will need to use their own private enterprise number for the enterpriseId parameter, and when creating locally extensible SD-ID names.

6.3.3. SD-PARAM

Each SD-PARAM consists of a name, referred to as PARAM-NAME, and a value, referred to as PARAM-VALUE.

PARAM-NAME is case-sensitive. IANA controls all PARAM-NAMEs, with the exception of those in SD-IDs whose names contain an at-sign. The PARAM-NAME scope is within a specific SD-ID. Thus, equally named PARAM-NAME values contained in two different SD-IDs are not the same.

To support international characters, the PARAM-VALUE field MUST be encoded using UTF-8. A syslog application MAY issue any valid UTF-8 sequence. A syslog application MUST accept any valid UTF-8 sequence in the "shortest form". It MUST NOT fail if control characters are present in PARAM-VALUE. The syslog application MAY modify messages containing control characters (e.g., by changing an octet with value 0 (USASCII NUL) to the four characters "#000"). For the reasons outlined in UNICODE TR36 [UNICODE-TR36], section 3.1, an originator MUST encode messages in the "shortest form" and a collector or relay MUST NOT interpret messages in the "non-shortest form".

Inside PARAM-VALUE, the characters '"' (ABNF %d34), '\' (ABNF %d92), and ']' (ABNF %d93) MUST be escaped. This is necessary to avoid parsing errors. Escaping ']' would not strictly be necessary but is REQUIRED by this specification to avoid syslog application implementation errors. Each of these three characters MUST be escaped as '\"', '\\', and '\]' respectively. The backslash is used for control character escaping for consistency with its use for escaping in other parts of the syslog message as well as in traditional syslog.

A backslash ('\') followed by none of the three described characters is considered an invalid escape sequence. In this case, the backslash MUST be treated as a regular backslash and the following character as a regular character. Thus, the invalid sequence MUST not be altered.

An SD-PARAM MAY be repeated multiple times inside an SD-ELEMENT.

6.3.4. Change Control

Once SD-IDs and PARAM-NAMEs are defined, syntax and semantics of these objects MUST NOT be altered. Should a change to an existing object be desired, a new SD-ID or PARAM-NAME MUST be created and the old one remain unchanged. OPTIONAL PARAM-NAMEs MAY be added to an existing SD-ID.

6.3.5. Examples

All examples in this section show only the structured data part of the message. Examples should be considered to be on one line. They are wrapped on multiple lines in this document for readability purposes. A description is given after each example.

Example 1 - Valid

```
[exampleSDID@32473 iut="3" eventSource="Application"
eventID="1011"]
```

This example is a structured data element with a non-IANA controlled SD-ID of type "exampleSDID@32473", which has three parameters.

Example 2 - Valid

```
[exampleSDID@32473 iut="3" eventSource="Application"
eventID="1011"][examplePriority@32473 class="high"]
```

This is the same example as in 1, but with a second structured data element. Please note that the structured data element immediately follows the first one (there is no SP between them).

Example 3 - Invalid

```
[exampleSDID@32473 iut="3" eventSource="Application"
eventID="1011"] [examplePriority@32473 class="high"]
```

This is nearly the same example as 2, but it has a subtle error -- there is an SP character between the two structured data elements ("]SP["). This is invalid. It will cause the STRUCTURED-DATA field to end after the first element. The second element will be interpreted as part of the MSG field.

Example 4 - Invalid

```
[ exampleSDID@32473 iut="3" eventSource="Application"
eventID="1011"][examplePriority@32473 class="high"]
```

This example is nearly the same as 2. It has another subtle error -- the SP character occurs after the initial bracket. A structured data element SD-ID MUST immediately follow the beginning bracket, so the SP character invalidates the STRUCTURED-DATA. A syslog application MAY discard this message.

Example 5 - Valid

```
[sigSig ver="1" rsID="1234" ... signature="..."]
```

Example 5 is a valid example. It shows a hypothetical IANA-assigned SD-ID. The ellipses denote missing content, which has been left out of this example for brevity.

6.4. MSG

The MSG part contains a free-form message that provides information about the event.

The character set used in MSG SHOULD be UNICODE, encoded using UTF-8 as specified in [RFC3629]. If the syslog application cannot encode the MSG in Unicode, it MAY use any other encoding.

The syslog application SHOULD avoid octet values below 32 (the traditional US-ASCII control character range except DEL). These values are legal, but a syslog application MAY modify these characters upon reception. For example, it might change them into an escape sequence (e.g., value 0 may be changed to "\0"). A syslog application SHOULD NOT modify any other octet values.

If a syslog application encodes MSG in UTF-8, the string MUST start with the Unicode byte order mask (BOM), which for UTF-8 is ABNF %xEF.BB.BF. The syslog application MUST encode in the "shortest form" and MAY use any valid UTF-8 sequence.

If a syslog application is processing an MSG starting with a BOM and the MSG contains UTF-8 that is not shortest form, the MSG MUST NOT be interpreted as being encoded in UTF-8, for the reasons outlined in [UNICODE-TR36], Section 3.1. Guidance about this is given in Appendix A.8.

Also, according to UNICODE TR36 [UNICODE-TR36], a syslog application MUST NOT interpret messages in the "non-shortest form". It MUST NOT interpret invalid UTF-8 sequences.

6.5. Examples

The following are examples of valid syslog messages. A description of each example can be found below it. The examples are based on similar examples from [RFC3164] and may be familiar to readers. The otherwise-unprintable Unicode BOM is represented as "BOM" in the examples.

Example 1 - with no STRUCTURED-DATA

```
<34>1 2003-10-11T22:14:15.003Z mymachine.example.com su - ID47
- BOM'su root' failed for lonvick on /dev/pts/8
```

In this example, the VERSION is 1 and the Facility has the value of 4. The Severity is 2. The message was created on 11 October 2003 at 10:14:15pm UTC, 3 milliseconds into the next second. The message originated from a host that identifies itself as "mymachine.example.com". The APP-NAME is "su" and the PROCID is unknown. The MSGID is "ID47". The MSG is "'su root' failed for lonvick...", encoded in UTF-8. The encoding is defined by the BOM. There is no STRUCTURED-DATA present in the message; this is indicated by "-" in the STRUCTURED-DATA field.

Example 2 - with no STRUCTURED-DATA

```
<165>1 2003-08-24T05:14:15.000003-07:00 192.0.2.1
myproc 8710 - - %% It's time to make the do-nuts.
```

In this example, the VERSION is again 1. The Facility is 20, the Severity 5. The message was created on 24 August 2003 at 5:14:15am, with a -7 hour offset from UTC, 3 microseconds into the next second. The HOSTNAME is "192.0.2.1", so the syslog application did not know its FQDN and used one of its IPv4 addresses instead. The APP-NAME is "myproc" and the PROCID is "8710" (for example, this could be the UNIX PID). There is no STRUCTURED-DATA present in the message; this is indicated by "-" in the STRUCTURED-DATA field. There is no specific MSGID and this is indicated by the "-" in the MSGID field.

The message is "%% It's time to make the do-nuts.". As the Unicode BOM is missing, the syslog application does not know the encoding of the MSG part.

Example 3 - with STRUCTURED-DATA

```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com
evntslog - ID47 [exampleSDID@32473 iut="3" eventSource=
"Application" eventID="1011"] BOMAn application
event log entry...
```

This example is modeled after Example 1. However, this time it contains STRUCTURED-DATA, a single element with the value "[exampleSDID@32473 iut="3" eventSource="Application" eventID="1011"]". The MSG itself is "An application event log entry..." The BOM at the beginning of MSG indicates UTF-8 encoding.

Example 4 - STRUCTURED-DATA Only

```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com
evntslog - ID47 [exampleSDID@32473 iut="3" eventSource=
"Application" eventID="1011"][examplePriority@32473
class="high"]
```

This example shows a message with only STRUCTURED-DATA and no MSG part. This is a valid message.

7. Structured Data IDs

This section defines the initial IANA-registered SD-IDs. See Section 6.3 for a definition of structured data elements. All SD-IDs defined here are OPTIONAL.

In some of the following, a maximum length is quantified for the parameter values. In each of those cases, the syslog application MUST be prepared to receive the number of defined characters in any valid UTF-8 code point. Since each character may be up to 6 octets, it is RECOMMENDED that each syslog application be prepared to receive up to 6 octets per character.

7.1. timeQuality

The SD-ID "timeQuality" MAY be used by the originator to describe its notion of system time. This SD-ID SHOULD be written if the originator is not properly synchronized with a reliable external time source or if it does not know whether its time zone information is correct. The main use of this structured data element is to provide some information on the level of trust it has in the TIMESTAMP described in Section 6.2.3. All parameters are OPTIONAL.

7.1.1. tzKnown

The "tzKnown" parameter indicates whether the originator knows its time zone. If it does, the value "1" MUST be used. If the time zone information is in doubt, the value "0" MUST be used. If the originator knows its time zone but decides to emit time in UTC, the value "1" MUST be used (because the time zone is known).

7.1.2. isSynced

The "isSynced" parameter indicates whether the originator is synchronized to a reliable external time source, e.g., via NTP. If the originator is time synchronized, the value "1" MUST be used. If not, the value "0" MUST be used.

7.1.3. syncAccuracy

The "syncAccuracy" parameter indicates how accurate the originator thinks its time synchronization is. It is an integer describing the maximum number of microseconds that its clock may be off between synchronization intervals.

If the value "0" is used for "isSynced", this parameter MUST NOT be specified. If the value "1" is used for "isSynced" but the "syncAccuracy" parameter is absent, a collector or relay can assume that the time information provided is accurate enough to be considered correct. The "syncAccuracy" parameter MUST be written only if the originator actually has knowledge of the reliability of the external time source. In most cases, it will gain this in-depth knowledge through operator configuration.

7.1.4. Examples

The following is an example of an originator that does not know its time zone or whether it is being synchronized:

```
[timeQuality tzKnown="0" isSynced="0"]
```

With this information, the originator indicates that its time information is unreliable. This may be a hint for the collector or relay to use its local time instead of the message-provided TIMESTAMP for correlation of multiple messages from different originators.

The following is an example of an originator that knows its time zone and knows that it is properly synchronized to a reliable external source:

[timeQuality tzKnown="1" isSynced="1"]

The following is an example of an originator that knows both its time zone and that it is externally synchronized. It also knows the accuracy of the external synchronization:

[timeQuality tzKnown="1" isSynced="1" syncAccuracy="60000000"]

The difference between this and the previous example is that the originator expects that its clock will be kept within 60 seconds of the official time. Thus, if the originator reports it is 9:00:00, it is no earlier than 8:59:00 and no later then 9:01:00.

7.2. origin

The SD-ID "origin" MAY be used to indicate the origin of a syslog message. The following parameters can be used. All parameters are OPTIONAL.

Specifying any of these parameters is primarily an aid to log analyzers and similar applications.

7.2.1. ip

The "ip" parameter denotes an IP address that the originator knows it had at the time of originating the message. It MUST contain the textual representation of an IP address as outlined in Section 6.2.4.

This parameter can be used to provide identifying information in addition to what is present in the HOSTNAME field. It might be especially useful if the host's IP address is included in the message while the HOSTNAME field still contains the FQDN. It is also useful for describing all IP addresses of a multihomed host.

If an originator has multiple IP addresses, it MAY either list one of its IP addresses in the "ip" parameter or it MAY include multiple "ip" parameters in a single "origin" structured data element.

7.2.2. enterpriseId

The "enterpriseId" parameter MUST be a 'SMI Network Management Private Enterprise Code', maintained by IANA, whose prefix is iso.org.dod.internet.private.enterprise (1.3.6.1.4.1). The number that follows MUST be unique and MUST be registered with IANA as per

RFC 2578 [RFC2578]. An enterprise is only authorized to assign values within the iso.org.dod.internet.private.enterprise.<private enterprise number> subtree assigned by IANA to that enterprise. The enterpriseId MUST contain only a value from the iso.org.dod.internet.private.enterprise.<private enterprise number> subtree. In general, only the IANA-assigned private enterprise number is needed (a single number). An enterprise might decide to use sub-identifiers below its private enterprise number. If sub-identifiers are used, they MUST be separated by periods and be represented as decimal numbers. An example for that would be "32473.1.2". Please note that the ID "32473.1.2" is just an example and MUST NOT be used. The complete up-to-date list of Private Enterprise Numbers (PEN) is maintained by IANA.

By specifying a private enterprise number, the vendor allows more specific processing of the message.

7.2.3. software

The "software" parameter uniquely identifies the software that generated the message. If it is used, "enterpriseId" SHOULD also be specified, so that a specific vendor's software can be identified. The "software" parameter is not the same as the APP-NAME header field. It MUST always contain the name of the generating software, whereas APP-NAME can contain anything else, including an operator-configured value.

The "software" parameter is a string. It MUST NOT be longer than 48 characters.

7.2.4. swVersion

The "swVersion" parameter uniquely identifies the version of the software that generated the message. If it is used, the "software" and "enterpriseId" parameters SHOULD be provided, too.

The "swVersion" parameter is a string. It MUST NOT be longer than 32 characters.

7.2.5. Example

The following is an example with multiple IP addresses:

```
[origin ip="192.0.2.1" ip="192.0.2.129"]
```

In this example, the originator indicates that it has two IP addresses, one being 192.0.2.1 and the other one being 192.0.2.129.

7.3. meta

The SD-ID "meta" MAY be used to provide meta-information about the message. The following parameters can be used. All parameters are OPTIONAL. If the "meta" SD-ID is used, at least one parameter SHOULD be specified.

7.3.1. sequenceId

The "sequenceId" parameter tracks the sequence in which the originator submits messages to the syslog transport for sending. It is an integer that MUST be set to 1 when the syslog function is started and MUST be increased with every message up to a maximum value of 2147483647. If that value is reached, the next message MUST be sent with a sequenceId of 1.

7.3.2. sysUpTime

The "sysUpTime" parameter MAY be used to include the SNMP "sysUpTime" parameter in the message. Its syntax and semantics are as defined in [RFC3418].

As syslog does not support the SNMP "INTEGER" syntax directly, the value MUST be represented as a decimal integer (no decimal point) using only the characters "0", "1", "2", "3", "4", "5", "6", "7", "8", and "9".

Note that the semantics in RFC 3418 are "The time (in hundredths of a second) since the network management portion of the system was last re-initialized." This of course relates to the SNMP-related management portion of the system, which MAY be different than the syslog-related management portion of the system.

7.3.3. language

The "language" parameter MAY be specified by the originator to convey information about the natural language used inside MSG. If it is specified, it MUST contain a language identifier as defined in BCP 47 [RFC4646].

8. Security Considerations

8.1. UNICODE

This document uses UTF-8 encoding for the PARAM-VALUE and MSG fields. There are a number of security issues with UNICODE. Any implementer and operator is advised to review UNICODE TR36 [UNICODE-TR36] (UTR36) to learn about these issues. This document guards against the technical issues outlined in UTR36 by REQUIRING "shortest form" encoding for syslog applications. However, the visual spoofing due to character confusion still persists. This document tries to minimize the effects of visual spoofing by allowing UNICODE only where local script is expected and needed. In all other fields, US-ASCII is REQUIRED. Also, the PARAM-VALUE and MSG fields should not be the primary source for identifying information, further reducing the risks associated with visual spoofing.

8.2. Control Characters

This document does not impose any mandatory restrictions on the MSG or PARAM-VALUE content. As such, they MAY contain control characters, including the NUL character.

In some programming languages (most notably C and C++), the NUL character (ABNF %d00) traditionally has a special significance as string terminator. Most implementations of these languages assume that a string will not extend beyond the first NUL character. This is primarily a restriction of the supporting run-time libraries. This restriction is often carried over to programs and script languages written in those languages. As such, NUL characters must be considered with great care and be properly handled. An attacker may deliberately include NUL characters to hide information after them. Incorrect handling of the NUL character may also invalidate cryptographic checksums that are transmitted inside the message.

Many popular text editors are also written in languages with this restriction. Encoding NUL characters when writing to text files is advisable. If they are stored without encoding, the file can become unreadable.

Other control characters may also be problematic. For example, an attacker may deliberately include backspace characters to render parts of the log message unreadable. Similar issues exist for almost all control characters.

Finally, invalid UTF-8 sequences may be used by an attacker to inject ASCII control characters.

This specification permits a syslog application to reformat control characters received. Among others, the security risks associated with control characters were an important driving force behind this restriction. Originators are advised that if any encoding other than ASCII and UTF8 are used, the receiver may corrupt the message in an attempt to filter ASCII control characters.

8.3. Message Truncation

Message truncation can be misused by an attacker to hide vital log information. Messages over the minimum supported size may be discarded or truncated by the transport receiver. As such, vital log information may be lost.

In order to prevent information loss, messages should not be longer than the minimum maximum size required by Section 6.1. For best performance and reliability, messages should be as small as possible. Important information should be placed as early in the message as possible because information at the beginning of the message is less likely to be discarded by a size-limited transport receiver.

An originator should limit the size of any user-supplied data within a syslog message. If it does not, an attacker may provide large data in hopes of exploiting a potential weakness.

8.4. Replay

There is no mechanism in the syslog protocol to detect message replay. An attacker may record a set of messages that indicate normal activity of a machine. At a later time, that attacker may remove that machine from the network and replay the syslog messages to the relay or collector. Even with the TIMESTAMP field in the HEADER part, an attacker may record the packets and could simply modify them to reflect the current time before retransmitting them. The administrators may find nothing unusual in the received messages, and their receipt would falsely indicate normal activity of the machine.

Cryptographically signing messages could prevent the alteration of TIMESTAMPs and thus the replay attack.

8.5. Reliable Delivery

Because there is no mechanism described within this document to ensure delivery, and the underlying transport may be unreliable (e.g., UDP), some messages may be lost. They may either be dropped through network congestion, or they may be maliciously intercepted and discarded. The consequences of dropping one or more syslog messages cannot be determined. If the messages are simple status updates, then their non-receipt may not be noticed or may cause an annoyance for the system operators. On the other hand, if the messages are more critical, then the administrators may not become aware of a developing and potentially serious problem. Messages may also be intercepted and discarded by an attacker as a way to hide unauthorized activities.

It may also be desirable to include rate-limiting features in syslog originators and relays. This can reduce potential congestion problems when message bursts happen.

Reliable delivery may not always be desirable. Reliable delivery means that the syslog originator or relay must block when the relay or collector is not able to accept any more messages. In some operating systems, namely Unix/Linux, the syslog originator or relay runs inside a high-priority system process (syslogd). If that process blocks, the system at large comes to a stand-still. The same occurs if there is a deadlock situation between syslogd and e.g., the DNS server.

To prevent these problems, reliable delivery can be implemented in a way that intentionally discards messages when the syslog application would otherwise block. The advantage of reliable delivery in this case is that the syslog originator or relay knowingly discards the message and is able to notify the relay or collector about that fact. So the relay or collector receives the information that something is lost. With unreliable delivery, the message would simply be lost without any indication that loss occurred.

8.6. Congestion Control

Because syslog can generate unlimited amounts of data, transferring this data over UDP is generally problematic, because UDP lacks congestion control mechanisms. Congestion control mechanisms that respond to congestion by reducing traffic rates and establish a degree of fairness between flows that share the same path are vital to the stable operation of the Internet [RFC2914]. This is why the syslog TLS transport is REQUIRED to implement and RECOMMENDED for general use.

The only environments where the syslog UDP transport MAY be used as an alternative to the TLS transport are managed networks, where the network path has been explicitly provisioned for UDP syslog traffic through traffic engineering mechanisms, such as rate limiting or capacity reservations. In all other environments, the TLS transport SHOULD be used.

In any implementation, the situation may arise in which an originator or relay would need to block sending messages. A common case is when an internal queue is full. This might happen due to rate-limiting or slow performance of the syslog application. In any event, it is highly RECOMMENDED that no messages be dropped but that they should be temporarily stored until they can be transmitted. However, if they must be dropped, it is RECOMMENDED that the originator or relay drop messages of lower severity in favor of higher severity messages.

Messages with a lower numerical SEVERITY value have a higher practical severity than those with a numerically higher value. In that situation, the messages that are to be dropped SHOULD simply be discarded. The syslog application may notify a collector or relay about the fact that it has dropped messages.

8.7. Message Integrity

Besides being discarded, syslog messages may be damaged in transit, or an attacker may maliciously modify them. In such cases, the original contents of the message will not be delivered to the collector or relay. Additionally, if an attacker is positioned between the transport sender and transport receiver of syslog messages, they may be able to intercept and modify those messages while in-transit to hide unauthorized activities.

8.8. Message Observation

While there are no strict guidelines pertaining to the MSG format, most syslog messages are generated in human-readable form with the assumption that capable administrators should be able to read them and understand their meaning. The syslog protocol does not have mechanisms to provide confidentiality for the messages in transit. In most cases, passing clear-text messages is a benefit to the operations staff if they are sniffing the packets from the wire. The operations staff may be able to read the messages and associate them with other events seen from other packets crossing the wire to track down and correct problems. Unfortunately, an attacker may also be able to observe the human-readable contents of syslog messages. The attacker may then use the knowledge gained from those messages to compromise a machine or do other damage.

Operators are advised to use a secure transport mapping to avoid this problem.

8.9. Inappropriate Configuration

Because there is no control information distributed about any messages or configurations, it is wholly the responsibility of the network administrator to ensure that the messages are actually going to the intended recipients. Cases have been noted where syslog applications were inadvertently configured to send syslog messages to the wrong relays or collectors. In many cases, the inadvertent relays or collectors may not be configured to receive syslog messages and will probably discard them. In certain other cases, the receipt of syslog messages has been known to cause problems for the unintended recipient. If messages are not going to the intended recipient, then they cannot be reviewed or processed.

Using a reliable transport mapping can help identify some of these problems. For example, it can identify a problem where a message is being sent to a system that is not configured to receive messages. It cannot identify sending messages to a wrong machine that is accepting messages.

8.10. Forwarding Loop

As shown in Diagram 2, machines may be configured to relay syslog messages to subsequent relays before reaching a collector. In one particular case, an administrator found that he had mistakenly configured two relays to forward messages with certain SEVERITY values to each other. When either of these machines either received or generated that type of message, it would forward it to the other relay. That relay would, in turn, forward it back. This cycle did cause degradation to the intervening network as well as to the processing availability on the two devices. Network administrators must take care not to cause such a death spiral.

8.11. Load Considerations

Network administrators must take the time to estimate the appropriate capacity of the syslog collector. An attacker may perform a Denial of Service attack by filling the disk of the collector with false messages. Placing the records in a circular file may alleviate this but has the consequence of not ensuring that an administrator will be able to review the records in the future. Along this line, a transport receiver must have a network interface capable of receiving the messages sent to it.

Administrators and network planners must also critically review the network paths between the originators, the relays, and the collectors. Generated syslog messages should not overwhelm any of the network links.

In order to reduce the impact of this issue, using transports with guaranteed delivery is recommended.

8.12. Denial of Service

As with any system, an attacker may just overwhelm a transport receiver by sending more messages to it than can be handled by the infrastructure or the device itself. Implementers should attempt to provide features that minimize this threat, such as only accepting syslog messages from known IP addresses.

9. IANA Considerations

9.1. VERSION

IANA has created a registry entitled "syslog Version Values" of VERSION values as described in Section 6.2.2. Version numbers MUST be incremented for any new syslog protocol specification that changes any part of the HEADER. Changes include addition or removal of fields or a change of syntax or semantics of existing fields.

VERSION numbers must be registered via the Standards Action method as described in [RFC5226]. IANA has registered the VERSIONs shown in Table 3 below.

VERSION	FORMAT
1	Defined in [RFC5424]

Table 3. IANA-Registered VERSIONs

9.2. SD-IDs

IANA has created a registry entitled "syslog Structured Data ID Values" of Structured Data ID (SD-ID) values together with their associated PARAM-NAME values as described in Section 7.

New SD-ID and new PARAM-NAME values must be registered through the IETF Review method as described in [RFC5226].

Once SD-IDs and SD-PARAMs are defined, syntax and semantics of these objects MUST NOT be altered. Should a change to an existing object be desired, a new SD-ID or SD-PARAM MUST be created and the old one remain unchanged.

A provision is made here for locally extensible names. The IANA will not register, and will not control names with the at-sign (ABNF %d64) in them.

IANA has registered the SD-IDs and PARAM-NAMEs shown in Table 4 below.

SD-ID	PARAM-NAME	
timeQuality		OPTIONAL
	tzKnown	OPTIONAL
	isSynced	OPTIONAL
	syncAccuracy	OPTIONAL

origin		OPTIONAL
	ip	OPTIONAL
	enterpriseId	OPTIONAL
	software	OPTIONAL
	swVersion	OPTIONAL
meta		OPTIONAL
	sequenceId	OPTIONAL
	sysUpTime	OPTIONAL
	language	OPTIONAL

Table 4. IANA-Registered SD-IDs and their PARAM-NAMEs

10. Working Group

The working group can be contacted via the mailing list:

syslog@ietf.org

The current Chairs of the Working Group may be contacted at:

Chris Lonvick
Cisco Systems
EMail: clonvick@cisco.com

David Harrington
Huawei Technologies USA
EMail: dbharrington@comcast.net

11. Acknowledgments

The authors wish to thank Chris Lonvick, Jon Callas, Andrew Ross, Albert Mietus, Anton Okmianski, Tina Bird, Devin Kowatch, David Harrington, Sharon Chisholm, Richard Graveman, Tom Petch, Dado Colussi, Clement Mathieu, Didier Dalmasso, and all the other people who commented on various versions of this proposal.

12. References

12.1. Normative References

[ANSI.X3-4.1968] American National Standards Institute, "USA Code for Information Interchange", ANSI X3.4, 1968.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5425] Fuyou, M., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for Syslog", RFC 5425, March 2009.

[RFC5426] Okmianski, A., "Transmission of Syslog Messages over UDP", RFC 5426, March 2009.

[UNICODE-TR36] Davis, M. and M. Suignard, "UNICODE Security Considerations", July 2005.

12.2. Informative References

[RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, August 2001.

Appendix A. Implementer Guidelines

Information in this section is given as an aid to implementers. While this information is considered to be helpful, it is not normative. As such, an implementation is NOT REQUIRED to follow it in order to claim compliance to this specification.

A.1. Relationship with BSD Syslog

While BSD syslog is in widespread use, its format has never been formally standardized. [RFC3164] describes observed formats. It is an Informational RFC, and practice shows that there are many different implementations. Research during creation of this document showed that there is very little in common between different syslog implementations on different platforms. The only thing that all of them agree upon is that messages start with "<" PRIVAL ">". Other than that, legacy syslog messages are not formatted in a consistent way. Consequently, RFC 3164 describes no specific elements inside a syslog message. It states that any message destined to the syslog UDP port must be treated as a syslog message, no matter what its format or content is.

This document retains the PRI value syntax and semantics. This will allow legacy syslog implementations to put messages generated by syslog applications compliant to this specification into the right bins.

Most existing implementations support UDP as the transport protocol for syslog. This specification supports UDP transport, but does not recommend it. Deployment of the required TLS support is recommended. Additional transport protocols may be used.

RFC 3164 describes relay behavior. This document does not specify relay behavior. This might be done in a separate document.

The TIMESTAMP described in RFC 3164 offers less precision than the timestamp specified in this document. It also lacks the year and time zone information. If a message formatted according to this document needs to be reformatted to be in RFC 3164 format, it is suggested that the originator's local time zone be used, and the time zone information and the year be dropped. If an RFC 3164 formatted message is received and must be transformed to be compliant to this document, the current year should be added and the time zone of the relay or collector MAY be used.

The HOSTNAME in RFC 3164 is less specific, but this format is still supported in this document as one of the alternate HOSTNAME representations.

The MSG part of the message is described as TAG and CONTENT in RFC 3164. In this document, MSG is what was called CONTENT in RFC 3164. The TAG is now part of the header, but not as a single field. The TAG has been split into APP-NAME, PROCID, and MSGID. This does not totally resemble the usage of TAG, but provides the same functionality for most of the cases.

In RFC 3164, STRUCTURED-DATA was not described. If a message compliant with this document contains STRUCTURED-DATA and must be reformatted according to RFC 3164, the STRUCTURED-DATA simply becomes part of the RFC 3164 CONTENT free-form text.

In general, this document tries to provide an easily parseable header with clear field separations, whereas traditional BSD syslog suffers from some historically developed, hard to parse field separation rules.

A.2. Message Length

Implementers should note the message size limitations outlined in Section 6.1 and try to keep the most important data early in the message (within the minimum guaranteed length). This ensures the data will be seen by the collector or relay even if a transport receiver at a relay on the message path truncates the message.

The reason syslog transport receivers need only support receiving up to and including 480 octets has, among other things, to do with difficult delivery problems in a broken network. Syslog messages may use a UDP transport mapping with this 480 octet restriction to avoid session overhead and message fragmentation. In a network with problems, the likelihood of getting one single-packet message delivered successfully is higher than getting two message fragments delivered successfully. Therefore, using a larger size may prevent the operator from getting some critical information about the problem, whereas using small messages might get that information to the operator. It is recommended that messages intended for troubleshooting purposes should not be larger than 480 octets. To further strengthen this point, it has also been observed that some UDP implementations generally do not support message sizes of more than 480 octets. This behavior is very rare and may no longer be an issue.

There are other use cases where syslog messages are used to transmit inherently lengthy information, e.g., audit data. By not enforcing any upper limit on the message size, syslog applications can be implemented with any size needed and still be compliant with this document. In such cases, it is the operator’s responsibility to ensure that all components in a syslog infrastructure support the required message sizes. Transport mappings may recommend specific message size limits that must be implemented to be compliant.

Implementers are reminded that the message length is specified in octets. There is a potentially large difference between the length in characters and the length in octets for UTF-8 strings.

It must be noted that the IPv6 MTU is about 2.5 times 480. An implementation targeted towards an IPv6-only environment might thus assume this as a larger minimum size.

A.3. Severity Values

This section describes guidelines for using Severity as outlined in Section 6.2.1.

All implementations should try to assign the most appropriate severity to their message. Most importantly, messages designed to enable debugging or testing of software should be assigned Severity 7. Severity 0 should be reserved for messages of very high importance (like serious hardware failures or imminent power failure). An implementation may use Severities 0 and 7 for other purposes if this is configured by the administrator.

Because severities are very subjective, a relay or collector should not assume that all originators have the same definition of severity.

A.4. TIME-SECFRAC Precision

The TIMESTAMP described in Section 6.2.3 supports fractional seconds. This provides grounds for a very common coding error, where leading zeros are removed from the fractional seconds. For example, the TIMESTAMP "2003-10-11T22:13:14.003" may be erroneously written as "2003-10-11T22:13:14.3". This would indicate 300 milliseconds instead of the 3 milliseconds actually meant.

A.5. Case Convention for Names

Names are used at various places in this document, for example for SD-IDs and PARAM-NAMEs. This document uses "lower camel case" consistently. With that, each name begins with a lower case letter and each new embedded word starts with an upper case letter, with no hyphen or other delimiter. An example of this is "timeQuality".

While an implementation is free to use any other case convention for experimental names, it is suggested that the case convention outlined above is followed.

A.6. Syslog Applications Without Knowledge of Time

In Section 6.2.3, the NILVALUE has been allowed for usage by originators without knowledge of time. This is done to support a special case when a syslog application is not aware of time at all. It can be argued whether such a syslog application can actually be found in today's IT infrastructure. However, discussion has indicated that those things may exist in practice and as such there should be a guideline established for this case.

However, an implementation SHOULD emit a valid TIMESTAMP if the underlying operating system, programming system, and hardware supports a clock function. A proper TIMESTAMP should be emitted even if it is difficult to obtain the system time. The NILVALUE should only be used when it is actually impossible to obtain time information. This rule should not be used as an excuse for lazy implementations.

A.7. Notes on the timeQuality SD-ID

It is recommended that the value of "0" be the default for the "tzKnown" (Section 7.1.1) parameter. It should only be changed to "1" after the administrator has specifically configured the time zone. The value "1" may be used as the default if the underlying operating system provides accurate time zone information. It is still advised that the administrator consider the correctness of the time zone information.

It is important not to create a false impression of accuracy with the timeQuality SD-ID (Section 7.1). An originator should only indicate a given accuracy if it actually knows it is within these bounds. It is generally assumed that the originator gains this in-depth knowledge through operator configuration. By default, an accuracy should not be provided.

A.8. UTF-8 Encoding and the BOM

This document specifies that SD-PARAMS must always be encoded in UTF-8. Other encodings of the message in the MSG portion, including ASCIIPRINT, are not permitted by a device conforming to this specification. There are two cases that need to be addressed here. First, a syslog application conforming to this specification may not be able to ascertain that the information given to it from an originator is encoded in UTF-8. If it cannot determine that with certainty, the syslog application may choose to not incorporate the BOM in the MSG. If the syslog application has a good indication that the content of the message is encoded in UTF-8, then it should include the BOM. In the second case, a syslog relay may be forwarding a message from a device that does not conform to this specification. In that case, the device would likely not include the BOM unless it has ascertained that the received message was encoded in UTF-8.

Author's Address

Rainer Gerhards
Adiscon GmbH
Mozartstrasse 21
Grossrinderfeld, BW 97950
Germany

EMail: rgerhards@adiscon.com