syslog Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 2, 2007

R. Gerhards
Adiscon GmbH

November 29, 2006

The syslog Protocol
draft-ietf-syslog-protocol-19.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on June 2, 2007.

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 anticipated original design goals for traditional syslog in mind. The reason 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.

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 not necessary for system interoperability.

This document has been written with the anticipated original design goals for traditional syslog in mind. The reason 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. It 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 instead of once for each syslog feature.

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 [4].

3. Definitions

The following definitions are used in this document:

- An application that can generate a syslog message is called a "sender".
- An application that can receive a syslog message is called a "receiver".
- An application that can receive syslog messages and forward them to another receiver is called a "relay".
- An application that receives messages and does not relay them to any other receiver is called a "collector".

A single application can have multiple roles at the same time.

4. Basic Principles

The following principles apply to syslog communication:

- The syslog protocol does not provide for any mechanism of acknowledgement of message delivery. Though some transports may provide status information, conceptually, syslog is a pure simplex communications protocol.

- Senders send messages to receivers with no knowledge of whether they are collectors or relays.

- Senders may be configured to send the same message to multiple receivers.

- Relays may send all or some of the messages that they receive to a subsequent relay or collector. They may also store or otherwise locally process some or all messages without forwarding. In the case where a receiver stores some messages and relays some messages, it is acting as both a collector and a relay.

- Relays may also generate their own messages and send them on to subsequent relays or collectors. In that case they are acting as senders and a relay.

- Sender and receiver may reside on the same or different systems.

4.1. Example Deployment Scenarios

Sample deployment scenarios are shown in Diagram 1. 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.

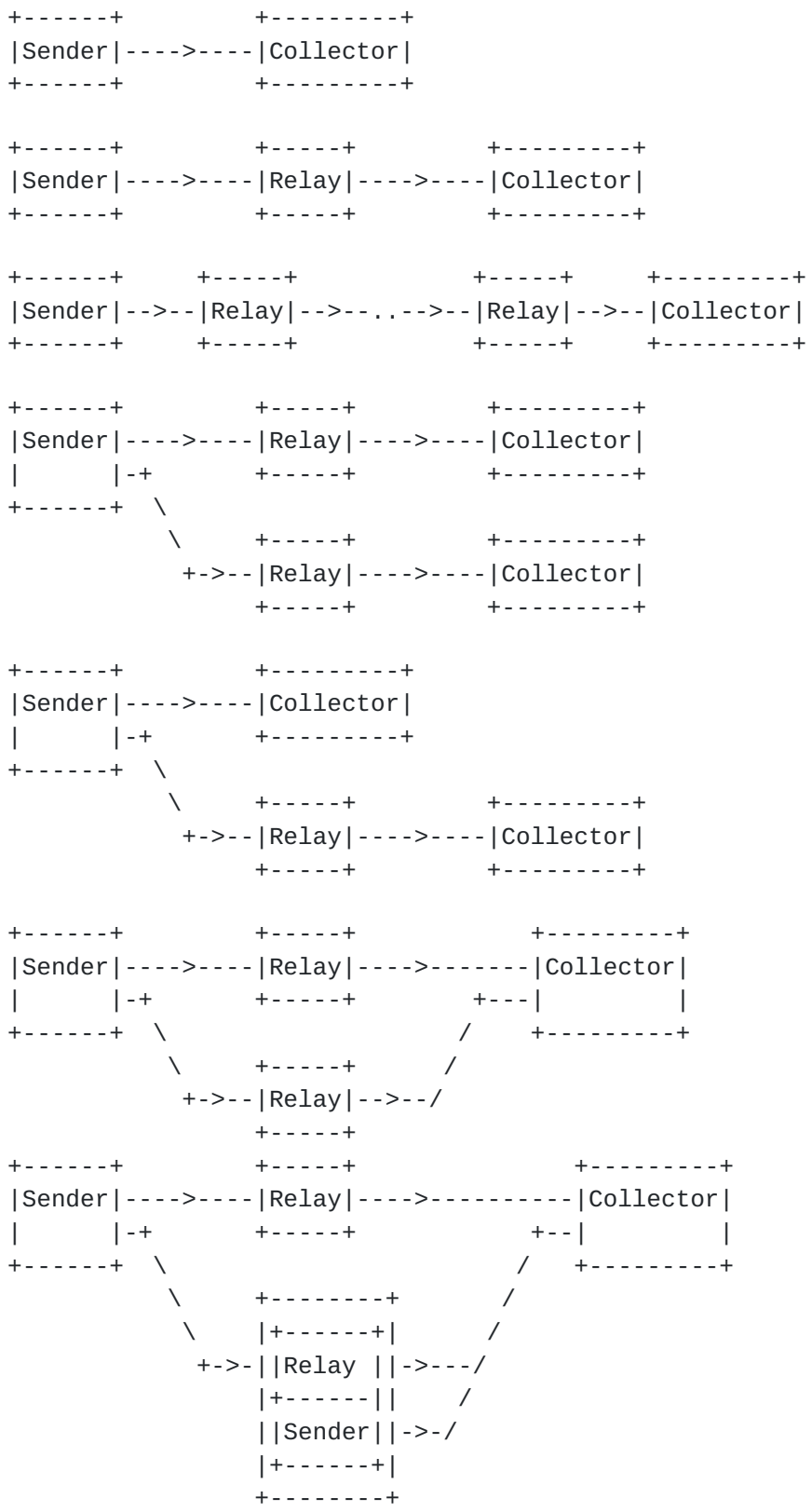

Diagram 1. Some possible syslog deployment scenarios.

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. This requires that syslog transports be defined in other documents. The first transport is defined in RFCXXXX [14] and is consistent with the traditional UDP transport. This transport is mandatory to implement for compliance to the standard to support 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, these transformations MUST be reversed by the transport protocol at the receiver, so that the upper layer will see an exact copy of the message sent from the originator. Otherwise 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 effort.

5.1. Minimum Required Transport Mapping

All implementations of this specification MUST support a UDP-based transport as described in RFCXXXX [14]. This is to provide a non-disruptive transition path from devices that have historically supported syslog over IPv4 UDP.

All implementations of this specification MUST also support a TLS-based transport as described in RFCZZZZ [15].

6. Syslog Message Format

The syslog message has the following ABNF [6] 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 ; Any VALID UTF-8 String
                  ; "shortest form" MUST be used

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 MUST define the minimum maximum required message length support. Any syslog transport mapping MUST support messages of up to and including 480 octets in length.

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

If a 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 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 RFC 4234 [6]. These are the ASCII codes as defined in "USA Standard Code for Information Interchange" ANSI.X3-4.1968 [1].

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.

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.

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 RFC 2434 [8].

6.2.3. TIMESTAMP

The TIMESTAMP field is a formalized timestamp derived from RFC 3339 [7].

Whereas RFC 3339 [7] 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.

o Leap seconds MUST NOT be used.

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

A syslog sender incapable of obtaining system time MUST use the NILVALUE as TIMESTAMP.

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 the Eastern US time zone (daylight savings time being observed).

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 by providing just three digits for the fractional part of a second, it 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 host name and the domain name of the originator in the format specified in STD 13 [2]. This format is called a Fully Qualified Domain Name (FQDN) in this document.

In practice, not all senders are able to provide a FQDN. As such, other values MAY also be present in HOSTNAME. This document makes provisions for using other values in such situations. A sender 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 [3]. If an IPv6 address is used, a valid textual representation as described in RFC 4291 [9], Section 2.2, MUST be used.

Senders SHOULD consistently use the same value in the HOSTNAME field for as long as possible. If the sender uses IP addresses inside hostname, the following rules apply: If the sender is multihomed, this value SHOULD be one of its actual IP addresses. If a sender is running on a machine that has both statically and dynamically assigned addresses, then that value SHOULD be from the statically assigned addresses. As an alternative, the sender MAY use the IP address of the interface that is used to send the message.

The NILVALUE SHOULD only be used when the sender 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 generated the message. It is a string without further semantics. It is intended for filtering messages on the receiver.

The NILVALUE MAY be used when the sender has no idea of its APP-NAME or cannot provide that information. It may be that a device may not be able 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

The PROCID field SHOULD be used to provide the sender's process name or process ID. The field does not have any specific syntax.

The NILVALUE MAY be used when the sender can not obtain its PROCID or cannot provide it.

PROCID is primarily meaningful for analysis tools. Properly used, it can enable log analyzers to detect which messages were generated by the same sender process. For example, on a UNIX system the syslog daemon (syslogd) might emit messages to the log. All messages logged by the same syslogd process will bear the same PROCID. When the syslog sender is restarted, the PROCID value MAY change. That may enable the analysis script to detect the syslogd restart.

This field MAY be operator-assigned. Some non-normative additional information about PROCID values can be found in Appendix A.8.

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 the receiver.

The NILVALUE SHOULD be used when the sender does not intend to provide a real MSGID.

This field MAY be operator-assigned.

6.3. STRUCTURED-DATA

STRUCTURED-DATA transports data in a well defined, easily parseable and interpretable format. There are multiple usage scenarios. For example, it may transport 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 RFC 4234 [6]. These are the ASCII codes as defined in "USA Standard Code for Information Interchange" ANSI.X3-4.1968 [1]. An exception is the PARAM-VALUE field (see Section 6.3.3), in which UTF-8 encoding MUST be used.

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

6.3.1. SD-ELEMENT

A 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 CONSENSUS as described in BCP26 (RFC2434) [8]. 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), or whitespace, or control characters (ASCII code 127 and codes 32 or less).

- Anyone can define additional SD-IDs using names in the format name@enterpriseId, e.g., "ourSDID@0". 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 the equal-sign ('=', ABNF %d61), a closing brace (']', ABNF %d93), a quote-character ('"', ABNF %d34), or whitespace, or control characters. The part following the at-sign MUST be an enterpriseId as specified in Section 7.2.2.

6.3.3. SD-PARAM

Each SD-PARAM consist 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 sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence in the "shortest form". It MUST NOT fail if control characters are present in PARAM-VALUE. It 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 [12], section 3.1, a sender MUST encode messages in the "shortest form" and a receiver 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 parser 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.

A SD-PARAM MAY be repeated multiple times inside a 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 only. A description is given after each example.

Example 1 - Valid

```
[exampleSDID@0 iut="3" eventSource="Application"
eventID="1011"]
```

This example is a structured data element with a non-IANA controlled SD-ID of type "exampleSDID@0" which has three parameters.

Example 2 - Valid

```
[exampleSDID@0 iut="3" eventSource="Application"
eventID="1011"][examplePriority@0 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@0 iut="3" eventSource="Application"
eventID="1011"] [examplePriority@0 class="high"]
```

This is nearly the same example as 2, but it has a subtle error. Please note that there is a 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@0 iut="3" eventSource="Application"
eventID="1011"][examplePriority@0 class="high"]
```

This example again is nearly the same as 2. It has another subtle error. Please note the SP character after the initial bracket. A structured data element SD-ID MUST immediately follow the beginning bracket, so the SP character invalidates the STRUCTURED-DATA. Thus,

the receiver 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. Please note that the ellipses denote missing content, which has been left out of this document 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 RFC 3629 [5]. If the sender can not encode the MSG in Unicode, it MAY use any other encoding.

The sender SHOULD avoid octet values below 32 (the traditional US-ASCII control character range except DEL). These values are legal, but a receiver 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 receiver SHOULD NOT modify any other octet values.

If a sender 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 sender MUST encode in the "shortest form" and MAY use any valid UTF-8 sequence.

If a receiver receives a MSG starting with a BOM, then it MUST be interpreted as being encoded in UTF-8 for the reasons outlined in UNICODE TR36 [12], section 3.1. If a sender does not encode MSG in UTF-8, the string MUST NOT start with the Unicode BOM. Guidance about this is given in Appendix A.9.

Also, according to UNICODE TR36 [12], a receiver 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 RFC 3164 [16] and may be familiar to readers. The otherwise-unprintable Unicode BOM is represented as "BOM" in the examples.

Example 1

```
<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. The MSG is "'su root' failed for lonvick...".

Example 2

```
<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 sender 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 receiver 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@0 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@0 iut="3" eventSource="Application" eventID="1011"]". The MSG itself is "An application event log entry..." Please note that 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@0 iut="3" eventSource=
"Application" eventID="1011"][examplePriority@0
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 receiver 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 receiver be prepared to receive up to six octets per character.

7.1. timeQuality

The SD-ID "timeQuality" MAY be used by the original sender to describe its notion of system time. This SD-ID SHOULD be written if the sender is not properly synchronized with a reliable external time source or if it does not know whether or not 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 original sender knows its time zone. If it does so, the value "1" MUST be used. If the time zone information is in doubt, the value "0" MUST be used. If the sender 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 original sender is synchronized to a reliable external time source, e.g., via NTP. If the original sender 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 original sender 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 receiver MUST assume that the

time information provided is accurate enough to be considered correct. The "syncAccuracy" parameter MUST be written only if the original sender actually has knowledge of the reliability of the external time source. In practice, in most cases, it will gain this in-depth knowledge through operator configuration.

7.1.4. Examples

The following is an example of a system that does not know its time zone nor whether it is being synchronized:

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

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

The following is an example of a system 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 a system 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 sender expects that its clock will be kept within 60 seconds of the official time. Thus if the sender 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 sender knows it had at the time of sending 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 additional identifying information 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 a sender 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 is unique and may be registered by an on-line form at <http://www.iana.org/cgi-bin/mod_ent.pl>. An enterprise is only authorized to assign values within the iso.org.dod.internet.private.enterprise.<enterprise ID> subtree assigned by IANA to that enterprise. The enterpriseId MUST contain only a value from the iso.org.dod.internet.private.enterprise.<enterprise ID> subtree. In general, only the IANA-assigned enterpriseID is needed (a single number). An enterprise might decide to use sub-identifiers below its enterpriseID. If sub-identifiers are used, they MUST be separated by periods and be represented as decimal numbers. An example for that would be "0.1.2". Please note that the id "0.1.2" is just an example and MUST NOT be used. The complete up-to-date list of Enterprise Numbers is maintained by IANA at <http://www.iana.org/assignments/enterprise-numbers>.

By specifying an enterpriseId, the vendor allows more specific parsing 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 sender 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 sender sent the messages. 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 RFC 3418 [11].

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 RFC3418 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 if the sender intends to convey information about the natural language used inside MSG. If it is specified, it MUST contain a two letter language identifier as defined in ISO 639 [13].

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 [12] (UTR36) to learn about these issues. This document guards against the technical issues outlined in UTR36 by REQUIRING "shortest form" encoding both for senders and receivers. However, the visual spoofing due to character confusability 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, if not all, 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. Please note that 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 potentially become unreadable.

The same is true for other control characters. 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 receiver to reformat control characters received. Among others, the security risks associated with control characters were an important driving force behind this restriction. Senders are advised to not send control characters so that the message text is not altered by any process on the receiver.

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 receiver or interim systems. 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 receiver.

A sender 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. Replaying

There is no mechanism in the syslog protocol to detect message reply. 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 receiver. 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), and 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 either not be noticed, or

it 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 be desirable to use a transport with guaranteed delivery to mitigate congestion.

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

8.6. 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. Additionally, if an attacker is positioned between the sender and collector of syslog messages, they may be able to intercept and modify those messages while in-transit to hide unauthorized activities.

8.7. 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 off of 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.

8.8. 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 senders were

inadvertently configured to send syslog messages to the wrong receivers. In many cases, the inadvertent receivers may not be configured to receive syslog messages and it 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 shall be sent to a system that is not configured to receive messages. It can not identify sending messages to a wrong machine that is accepting messages.

8.9. Forwarding Loop

As shown in Diagram 1, 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.

8.10. Load Considerations

Network administrators must take the time to estimate the appropriate capacity of the syslog receivers. 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 that has the consequence of not ensuring that an administrator will be able to review the records in the future. Along this line, a receiver or collector must have a network interface capable of receiving all messages sent to it.

Administrators and network planners must also critically review the network paths between the devices, 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.

8.11. Denial of Service

As with any system, an attacker may just overwhelm a 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 is requested to create 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 RFC 2434 [8]. IANA is requested to register the VERSIONs shown in table 4 below.

VERSION	FORMAT
1	Defined in RFCYYYY

Table 4. IANA-registered VERSIONs.

9.2. SD-IDs

IANA is requested to create 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 CONSENSUS method as described in RFC 2434 [8].

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 is requested to register the SD-IDs and PARAM-NAMEs shown in table 5 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 5. 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 other people who commented on various versions of this proposal.

12. Notes to the RFC Editor

This is a note to the RFC editor. This ID is submitted along with draft-ietf-syslog-transport-udp and draft-ietf-syslog-transport-tls. These documents cross-reference each other. When RFC numbers are determined for each of these IDs, replace XXXX with the proper RFC number for draft-ietf-syslog-transport-udp and replace ZZZZ with the proper RFC number for draft-ietf-syslog-transport-tls and remove this note.

This document uses the term "RFCYYYY" for self-references. When a RFC number is assigned to it, replace YYYY with the proper RFC number and remove this note.

13. References

13.1. Normative

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

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

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

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

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

[6] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

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

[8] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

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

[10] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, September 2004.

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

[12] Davis, M. and M. Suignard, "UNICODE Security Considerations", July 2005, <http://www.unicode.org/reports/tr36/tr36-3.html>.

[13] International Organization for Standardization, "Code for the representation of names of languages", ISO Standard 639-1:2002, July 2002.

[14] Okmianski, A., "Transmission of syslog messages over UDP", RFC XXXX, November 2006.

[15] Miao, F. and M. Yuzhi, "TLS Transport Mapping for SYSLOG", RFC ZZZZ, August 2006.

13.2. Informative

[16] 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. RFC 3164 [16] 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 implementation to put messages generated by senders compliant to this specification into the right bins.

Most existing implementations support UDP as the transport protocol for syslog. This specification REQUIRES UDP support in compliant implementations, and allows additional transport protocols to 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 sender's local time zone be used, and the time zone information and the year be dropped. If a RFC 3164 formatted message is received and must be transformed to be compliant to this document, the current year should be added and the receiver's time zone be assumed.

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 parts early in the message (within the minimum guaranteed length). This ensures they will be seen by the receiver even if it (or a relay on the message path) truncates the message.

The reason syslog receivers must 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. As such, 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 then 480 octets. This behaviour 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 senders and receivers 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 enforced.

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 receiver should not assume that all senders 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 word starts with an upper case letter, but 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 Senders Without Knowledge of Time

In Section 6.2.3, the NILVALUE has been allowed for usage by senders without knowledge of time. This is done to support a special case when a sender is not aware of time at all. It can be argued whether such a sender 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, but doable, 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 explicitly acknowledge 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). A sender should only indicate a given accuracy if it actually knows it is within these bounds. It is generally assumed that the sender gains this in-depth knowledge through operator configuration. As such, by default, an accuracy should not be provided.

A.8. Additional Information on PROCID

The objective behind PROCID (Section 6.2.6) is to provide a quick way to detect a new instance of the sender's syslog process. It must be noted that this is not a reliable identification as a second sender process may actually be assigned the same process ID as a previous one. Properly used, PROCID can be helpful for analysis purposes.

While PROCID is defined to contain the sender's process ID, it is up to the sender to decide what this ID is. For example, on a general purpose OS, it might actually be the operating system process ID of the syslog sender's process. Other syslog senders might decide that it is more appropriate to put an internal identification into PROCID.

For example, a SMTP MTA might not put the operating system process ID into PROCID but might prefer to put its SMTP transaction ID into PROCID. This might be very useful, because it allows the receiver to group messages based on the SMTP transaction, which could also be called the SMTP "process" in this case. On an embedded system without any operating system process ID, PROCID might actually be a reboot ID, which might be the closest thing to a process ID on this hypothetical embedded system.

A.9. 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 process conforming to this specification may not be able to ascertain that the information given to it from a process is encoded in UTF-8. If it cannot determine that with certainty, the syslog process may choose to not incorporate the BOM in the MSG. If the syslog process 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 process may be relaying a message from a device that does not comform 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

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).