The syslog Protocol
draft-ietf-syslog-protocol-14.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 January 13, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

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 spirit of 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. The layered architecture also provides a solid basis that allows code to be written once instead of multiple times, once for each syslog feature.

Table of Contents

1. Introduction ........................................... 4
2. Conventions Used in This Document ...................... 5
3. Definitions ........................................... 6
4. Basic Principles ........................................ 7
   4.1 Example Deployment Scenarios .......................... 7
5. Transport Layer Protocol ................................ 9
   5.1 Minimum Required Transport Mapping .................. 9
6. Required syslog Format .................................. 10
   6.1 Message Length ..................................... 11
   6.2 HEADER ............................................ 12
      6.2.1 VERSION ....................................... 12
      6.2.2 FACILITY ...................................... 12
      6.2.3 SEVERITY ...................................... 12
      6.2.4 TRUNCATE ....................................... 13
      6.2.5 TIMESTAMP ..................................... 14
      6.2.6 HOSTNAME ...................................... 16
      6.2.7 APP-NAME ....................................... 17
      6.2.8 PROCID ......................................... 17
      6.2.9 MSGID ......................................... 17
   6.3 STRUCTURED-DATA .................................... 17
      6.3.1 SD-ELEMENT ..................................... 18
      6.3.2 SD-ID .......................................... 18
      6.3.3 SD-PARAM ....................................... 18
      6.3.4 Change Control .................................. 19
      6.3.5 Examples ....................................... 19
   6.4 MSG ................................................ 20
   6.5 Examples ........................................... 20
7. Structured Data IDs .................................... 23
   7.1 timeQuality ......................................... 23
      7.1.1 tzKnown ......................................... 23
      7.1.2 isSynced ......................................... 23
      7.1.3 syncAccuracy ................................... 23
      7.1.4 Examples ....................................... 24
   7.2 origin ................................................ 24
      7.2.1 ip ................................................ 24
      7.2.2 enterpriseId .................................... 25
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 parsable, structured information and allows for vendor extensions.

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

This document has been written with the spirit of RFC 3164 [14] 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. The layered architecture also provides a solid basis that allows code to be written once instead of multiple times, 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 [5].
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, conceptionally, 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 pass along all or some of the messages that they receive and also pass along messages that they internally generate. 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 [13] and is consistent with the traditional UDP transport.

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 (like signatures) will be broken. Of course, message alteration might occur due to transmission or similar errors. Guarding against such alterations is not within the scope of this requirement.

5.1 Minimum Required Transport Mapping

All syslog implementations MUST support a UDP-based transport as described in [13]. This requirement ensures interoperability between all systems implementing the protocol described in this document.
6. Required syslog Format

The syslog message has the following ABNF [7] definition:

SYSLOG-MSG = HEADER SP STRUCTURED-DATA [SP MSG]
HEADER = VERSION SP FACILITY SP SEVERITY SP
TRUNCATE SP TIMESTAMP SP HOSTNAME
SP APP-NAME SP PROCID SP MSGID
VERSION = NONZERO-DIGIT 0*2DIGIT
FACILITY = "0" / (NONZERO-DIGIT 0*9DIGIT)
; range 0..2147483647
SEVERITY = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7"
TRUNCATE = 1*2DIGIT
HOSTNAME = 1*255PRINTUSASCII
APP-NAME = 1*48PRINTUSASCII
PROCID = "-" / 1*128PRINTUSASCII
MSGID = "-" / 1*32PRINTUSASCII
TIMESTAMP = 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 ; 00-58, 00-59, 00-60 based on leap
; second rules
TIME-SECFRAC = "." 1*6DIGIT
TIME-OFFSET = "Z" / TIME-NUMOFFSET
TIME-NUMOFFSET = ("+" / "-" ) TIME-HOUR ":" TIME-MINUTE
STRUCTURED-DATA = 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*32OCTET ; VALID UTF-8 String
; except ’’, SP, ’}’, %d34 ("
6.1 Message Length

A receiver MUST be able to accept messages up to and including 480 octets in length. For interoperability reasons, all receiver implementations SHOULD be able to accept messages up to and including 2,048 octets in length.

If a receiver receives a message with a length larger than 2,048 octets, or larger than it supports, the receiver MAY discard the message or truncate the payload.

Receivers SHOULD follow this order of preference when it comes to truncation:

1) No truncation
2) Truncation by dropping SD-ELEMENTs
3) If 2) not sufficient, truncate MSG

When a receiver or initial sender truncates a message, the TRUNCATE (Section 6.2.4) field MUST be updated. In the case of a receiver, please note that this will break message integrity checking mechanisms such as digital signatures. This is irrelevant, as the truncation itself breaks the signature. So no extra harm is done by updating the TRUNCATE field.

When the MSG part is truncated, the UTF-8 encoding MUST be kept valid.

If the last SD-ELEMENT of a message is deleted, the STRUCTURED-DATA field MUST be changed to "-" to indicate empty STRUCTURED-DATA.

Please note that it is possible that the MSG field is truncated without dropping any SD-PARAMS. This is the case if a message with an empty STRUCTURED-DATA field must be truncated.
6.2 HEADER

The character set used in the HEADER MUST be seven-bit ASCII in an eight-bit field as described in RFC 2234 [7]. 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 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 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 [9].

6.2.2 FACILITY

FACILITY is an integer in the range from 0 to 2147483647. It can be used for filtering by the receiver. It is a category, allowing a coarse grouping of messages. There exist some traditional FACILITY code semantics for the codes in the range from 0 to 23. These semantics are not closely followed by all senders, and practice has shown that common semantics for message categories are hard to establish. Therefore, no specific semantics for FACILITY codes are specified or implied in this document.

There is no relationship between MSGID (Section 6.2.9) and FACILITY, because MSGID identifies a specific message whereas FACILITY specifies a coarse message group and is expected to be operator assigned most-often.

6.2.3 SEVERITY

The SEVERITY field is used to indicate the severity that the sender of a message assigned to it. It contains one of the values shown in table 1 below.
Numerical SEVERITY
Code

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 conditions
6 Informational: informational messages
7 Debug: debug-level messages

Table 1. SEVERITY Values.

6.2.3.1 Relation to Alarm MIB

The Alarm MIB RFC3877 [11] defines ITU perceived severities which are useful to be able to relate to the syslog severities, particularly in the case where alarms are being logged. The ITUPerceivedSeverity corresponds to a syslog SEVERITY as shown in table 2 below.

ITU Perceived Severity          syslog SEVERITY
Critical                        Alert
Major                           Critical
Minor                           Error
Warning                         Warning
Indeterminate                   Notice
Cleared                         Notice

Table 2. ITUPerceivedSeverity to syslog SEVERITY mapping.

6.2.4 TRUNCATE

The TRUNCATE field is used to indicate if the message has been truncated since it was sent or generated by an application. Such a truncation might happen on the initial sender and any receiver, including receivers on interim systems (relays). Values in the TRUNCATE field are made up of bits. Each of this bits has been assigned a specific value so that there is no doubt about bit ordering. The values described in table 3 below MUST be used.
Table 3. TRUNCATE values.

<table>
<thead>
<tr>
<th>VALUE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>all or some SD-ELEMENTs were truncated</td>
</tr>
<tr>
<td>2</td>
<td>all or part of MSG was truncated</td>
</tr>
<tr>
<td>4</td>
<td>truncation occurred at the receiver</td>
</tr>
<tr>
<td>8</td>
<td>truncation occurred at an interim system</td>
</tr>
<tr>
<td>16</td>
<td>truncation occurred at the initial sender</td>
</tr>
</tbody>
</table>

The value in the TRUNCATE field is the ASCII representation of these ORed bits. If the initial sender truncates a message, this MUST be indicated by setting the "truncation occurred at the initial sender" bit (value 16). If the truncation occurs while receiving the message, the "truncation occurred at the receiver" (value 4) bit MUST be set. If the receiver forwards the message to another system, the value of 4 MUST be changed to "truncation occurred at an interim system" (value 8).

Truncation on the initial sender sounds illogical, but may happen. On many systems, a syslog library or subsystem is responsible for actually sending the syslog messages. This library or subsystem is passed the message to send from another application. That application might ask the library or subsystem to send a message that is larger than is supported. One alternative is that a failure status is passed back to the application and the message is not sent. In other cases, it might be advisable to send the message, in which case it must be truncated directly at the initial sender. As the information about this might be helpful for the receiver (e.g. signatures might be valid, which is not the case in other truncations), there is a bit that can be used to reflect this information.

Some examples: If no truncation occurred, TRUNCATE MUST have a value of 0. If SD-ELEMENTs were truncated on the receiver, TRUNCATE MUST have a value of 5. If they were truncated on the initial sender, TRUNCATE MUST have the value of 17. If structured data and MSG were truncated on an interim system, TRUNCATE MUST have the value 11. If only MSG was truncated on the initial sender, TRUNCATE MUST have the value 18. If MSG and structured data were truncated on the sender, an interim system and the receiver, TRUNCATE MUST have the value 31.

Please see Message Length (Section 6.1) for details on truncation. The TRUNCATE field does not specify how much of the STRUCTURED-DATA or MSG was truncated. It just indicates that truncation occurred.

6.2.5 TIMESTAMP

The TIMESTAMP field is a formalized timestamp derived from RFC 3339 [8].
Whereas RFC 3339 [8] makes allowances for multiple syntaxes, this document imposes further restrictions. The TIMESTAMP MUST follow these restrictions:

- The "T" and "Z" characters in this syntax MUST be upper case.
- Usage of the "T" character is REQUIRED.

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

6.2.5.1 Syslog Senders Without Knowledge of Time

A syslog sender incapable of obtaining system time MUST use the following TIMESTAMP:

2000-01-01T00:00:60Z

This TIMESTAMP is in the past and it shows a time that never existed, because 1 January 2000 had no leap second. So it can never occur in a valid syslog message of a time-aware sender. A receiver receiving this TIMESTAMP MUST treat this value as an undefined date and time.

6.2.5.2 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.

6.2.6 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 [3]. 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 protocol 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. "0:0:0:0:0:0:0:0"

If an IPv4 address is used, it MUST be in the format of the dotted decimal notation as used in STD 13 [4]. If an IPv6 address is used, a valid textual representation described in RFC 3513 [10], 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 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.

6.2.7 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.

6.2.8 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 dash ("-") is a reserved PROCID field value that SHOULD be used only to indicate that the PROCID is not provided.

PROCID is primarily meaningful for analysis tools. Properly used, it might 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 syslogd is restarted, the PROCID value MAY change. That enables the analysis script to detect the syslogd restart.

6.2.9 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 dash ("-") is a reserved MSGID field value that MUST be used only to indicate that the message has no specific ID.

6.3 STRUCTURED-DATA

STRUCTURED-DATA transports data in a well defined, easily parsable 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
value dash ("-") MUST be used.

The character set used in STRUCTURED-DATA MUST be UNICODE, encoded
using UTF-8 as specified in RFC 3629 [6]. A sender MAY issue any
valid UTF-8 sequence. A receiver MUST accept any valid UTF-8
sequence. It MUST NOT fail if control characters are present in the
STRUCTURED-DATA part.

If STRUCTURED-DATA is malformed, a diagnostic entry SHOULD be logged. A
receiver MAY ignore malformed STRUCTURED-DATA elements.

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. IANA controls all SD-IDs. SD-IDs with the
prefatory string "x-" are set aside for experimental or vendor-
specific use. There is no standardization for SD-IDs with the
prefatory string "x-". The same SD-ID MUST NOT exist more than once
in a message.

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 experimental SD-IDs or those with the
prefatory string "x-" (without the quotes). PARAM-NAMEs with the
prefatory string x- are set aside for experimental purposes. They
can be specified in any SD-ID. The PARAM-NAME scope is within a
specific SD-ID. Thus, an equally-named PARAM-NAME contained in two
different SD-IDs is not the same.

Inside PARAM-VALUE, the characters '"', '\', and ']' 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.

A backslash (‘\’) followed by none of the three described characters is considered an invalid escape sequence. Upon reception of such an invalid escape sequence, the receiver MAY replace the two-character sequence with only the second character received. Alternatively, it MAY drop the message. It is RECOMMENDED that the receiver log a diagnostic message on receipt of a message with an invalid escape sequence.

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.

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 for readability purposes only. A description is given after each example.

Example 1 - Valid

[x-exampleSDID iut="3" eventSource="Application" eventID="1011"]

This example is a structured data element with an experimental SD-ID of type "x-exampleSDID" which has three parameters.

Example 2 - Valid

[x-exampleSDID iut="3" eventSource="Application" eventID="1011"] [x-examplePriority 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

[x-exampleSDID iut="3" eventSource="Application"
  eventID="1011"] [x-examplePriority 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

[x-exampleSDID iut="3" eventSource="Application"
  eventID="1011"] [x-examplePriority 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 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 MUST be UNICODE, encoded using UTF-8 as specified in RFC 3629 [6]. A sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence. It MUST NOT fail if control characters are present in the MSG part.

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 [14] and may be familiar to readers.
Example 1

```
1 888 4 0 2003-10-11T22:14:15.003Z mymachine.example.com
su - ID47 - 'su root' failed for lonvick on /dev/pts/8
```

In this example, the VERSION is 1 and the FACILITY has the value of 888. The severity is 4 ("Warning" semantics). The message was not truncated (0). It 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". 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

```
1 20 6 0 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 within the legacy syslog range (20). The severity is 6 ("Notice" semantics). It 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 specific MSGID and this is indicated by the "-" in the MSGID field. The message is "%% It’s time to make the do-nuts.".

Example 3 - with STRUCTURED-DATA

```
1 888 4 0 2003-10-11T22:14:15.003Z mymachine.example.com
evntslog - ID47 [x-exampleSDID iut="3" eventSource="Application" eventID="1011"] An application event log entry...
```

This example is modeled after example 1. However, this time it contains STRUCTURED-DATA, a single element with the value "[x-exampleSDID iut="3" eventSource="Application" eventID="1011"]". The MSG itself is "An application event log entry...".

Example 4 - STRUCTURED-DATA Only

```
1 888 4 0 2003-10-11T22:14:15.003Z mymachine.example.com
evntslog - ID47 [x-exampleSDID iut="3" eventSource="Application" eventID="1011"] [x-examplePriority class="high"]
```

This example shows a message with only STRUCTURED-DATA and no MSG
Example 5 - with truncated STRUCTURED-DATA

1 888 4 5 2003-10-11T22:14:15.003Z mymachine.example.com
   evntslog - ID47 - An application event log entry...

This example is modeled after example 1. However, this time it originally contained STRUCTURED-DATA, which has been truncated by a receiver. This is indicated by the TRUNCATE field, which is now set to "5".
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 are OPTIONAL.

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 6.2.5. 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 knows that it knows neither 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. So if the sender reports it is 9:00:00, it is no earlier than 8:59:00 and no later than 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 6.2.6.

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/>. Only that number and any-enterprise assigned ID below it MUST be specified in the "enterpriseId" parameter. If sub-identifiers are used, they MUST be separated by periods and be represented as decimal numbers ("9.1.30" and "11.2.3.7.5.12"). 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 always contains 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 allows to track 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 [12].

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".
8. Security Considerations

8.1 Diagnostic Logging

This document recommends that an implementation writes a diagnostic message to indicate unusual situations or other things noteworthy. Diagnostic messages are a useful tool in discovering configuration issues as well as instances of system penetration.

Unfortunately, diagnostic logging can cause issues by itself, for example, if an attacker tries to create a denial of service condition by willingly sending malformed messages that will lead to the creation of diagnostic log entries. Due to sheer volume, the resulting diagnostic log entries may exhaust system resources, e.g. processing power, I/O capability, or simply storage space. For example, an attacker could flood a system with messages generating diagnostic log entries after he has compromised a system. If the log entries are stored in a circular buffer, the flood of diagnostic log entries would eventually overwrite useful previous diagnostics.

Besides this risk, too verbose diagnostic logging can cause the administrator to turn logging off.

8.2 Control Characters

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

In some programming languages (most notably C and C++), the NUL (0x00) character 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 unencoded, 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.

8.3 More than Maximum Message Length

The message length MAY exceed the RECOMMENDED maximum value specified in Section 6. Various problems may result if a sender sends messages with a greater length. Also, an attacker might deliberately introduce very large messages. As such, it is vital that each receiver performs the necessary sanity checks to ensure that it will gracefully discard or truncate messages of larger sizes than it supports.

8.4 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 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.

8.5 Replaying

Messages may be recorded and replayed at a later time. 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 collector. Even with a 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.

8.6 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 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.

8.7 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. 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.

8.8 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. Neither the syslog protocol nor the syslog application 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 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.

8.9 Misconfiguration

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 these problems.

8.10 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.

8.11 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.

8.12 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. Implementors 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 must maintain a registry of VERSION values as described in Section 6.2.1. 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 [9]. IANA must register the VERSIONs shown in table 4 below.

<table>
<thead>
<tr>
<th>VERSION</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>according to this document</td>
</tr>
</tbody>
</table>

Table 4. IANA-registered VERSIONs.

9.2 SD-IDs

IANA must maintain a registry 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 Standards Action method as described in RFC 2434 [9].

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.

SD-IDs and PARAM-NAMEs with the prefatory string "x-" (without the quotes) are reserved for experimental use. IANA must not register any SD-ID or PARAM-NAME with the prefatory string "x-" for any other use.

IANA must register the SD-IDs and PARAM-NAMEs shown in table 5 below.
Table 5. IANA-registered SD-IDs and their PARAM-NAMEs.

<table>
<thead>
<tr>
<th>SD-ID</th>
<th>PARAM-NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeQuality</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>tzKnown</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>isSynced</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>syncAccuracy</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>origin</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>ip</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>enterpriseId</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>software</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>swVersion</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>meta</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>sequenceId</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>sysUpTime</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>x- (prefatory string)</td>
<td>RESERVED for experimental SD-IDs</td>
</tr>
<tr>
<td>(any)</td>
<td>x- (prefatory string)</td>
</tr>
</tbody>
</table>
10. Authors and Working Group Chair

The working group can be contacted via the mailing list:

syslog-sec@employees.org

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

Chris Lonvick
Cisco Systems
Email: clonvick@cisco.com

The author of this draft is:

Rainer Gerhards
Email: rgerhards@adiscon.com

Phone: +49-9349-92880
Fax: +49-9349-928820

Adiscon GmbH
Mozartstrasse 21
97950 Grossrinderfeld
Germany
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 ID draft-ietf-syslog-transport-udp and they cross-reference each other. When RFC numbers are determined for each of these IDs, replace XXXX with RFC number and remove this note.
13. References

13.1 Normative


13.2 Informative


Author’s Address

Rainer Gerhards
Adiscon GmbH
Mozartstrasse 21
Grossrinderfeld, BW 97950
Germany

Email: rgerhards@adiscon.com
Appendix A. Implementor Guidelines

Information in this section is given as an aid to implementors. 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. In RFC 3164 [14] observed formats were specified. However, RFC 3164 is an informal document, and practice shows that there are many different implementations.

Consequently, RFC 3164 mandates 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. However, in almost all cases observed in practice, a BSD syslog message starts with a priority value, which is a number between brackets. An example is "<133>". This document uses that known convention to provide some minimal version detection. It has deliberately changed the syslog message header so that it will never contain a less-than sign as the first character of the message. This has two advantages:

If an older receiver receives a message that does not start with a less-than sign, it still assumes this is a valid syslog message. However, it does not try to parse any header fields, at least if it obeys to the rule outlined in RFC 3164. This prevents the receiver from parsing the message invalidly. It should be noted, however, that at least some of the older implementations will experience problems if the message received is larger than 1024 octets. Most of the implementations will truncate a message after the first 1024 octets. So it is wise not to send messages larger than 1024 octets to receivers known to be older.

If a receiver compliant with this document receives a message generated by a non-compliant, older sender, it notices that the message does not have a proper header and thus is not formatted according to this document. This enables the receiver to take appropriate action. Please also see the description on header parsing in Appendix A.4 for more information on this scenario.

RFC 3164 mandates UDP as transport protocol for syslog. This document places no restrictions on the transport.

RFC 3164 specifies relay behavior. This document does not specify relay behavior. This might be done in a separate document.
The PRI part in RFC 3164 is split into two fields -- FACILITY and SEVERITY -- in this document. These new fields support the RFC 3164 values but also allow additional values.

The TIMESTAMP in RFC 3164 offers less precision and lacks the year and timezone information. If a message formatted according to this document needs to be reformatted to be RFC 3164 compliant, 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 defined 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 defined. If a message compliant with this document contains STRUCTURED-DATA and must be reformatted to be compliant with 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 parsable header with clear field separations whereas traditional BSD syslog suffers from some historically developed, hard to parse field separation rules.

A.2 Message Length

Implementors 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 and have this 480 restriction to avoid session overhead and message fragmentation. In a network being troubleshooted, the likelihood of getting one single-packet message...
delivered successfully is higher than getting two message fragments delivered successfully. So using a larger size may prevent the operator from getting some critical information about the problem, whereas keeping within that limit 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 than 480 octets.

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.

Implementors 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 environment only might thus assume this as a larger minimum size.

A.3 Message Truncation

As outlined in Section 6.1, messages may be subject to truncation. In this case, the TRUNCATE field must be updated as specified in Section 6.2.4. Implementors must keep in mind that the TRUNCATE field is a variable-sized entity. Thus, its size requirement (in octets) may grow during truncation. For example, if a relay truncates the MSG part of a message previously not truncated, the TRUNCATE field value changes from 0 to 10, taking up one additional octet of space. As such, it is necessary to truncate one additional octet from the MSG to make room for the now-expanded HEADER.

Similar reformatting may be necessary if a relay is operator-instructed to perform header modifications, e.g. to change the FACILITY or SEVERITY of a message before forwarding it.

A.4 HEADER Parsing

This section recommends a message header parsing method based on the VERSION field described in Section 6.2.1.

The receiver should check the VERSION. If the VERSION is within the
set of versions supported by the receiver, it should parse the message according to the correct syslog protocol specification.

If the receiver does not support the specified VERSION, it should log a diagnostic message. It should not parse beyond the VERSION field. This is because the header format may have changed in a newer version. The receiver should not try to process the message, but it may try this if the administrator has configured the receiver to do so. In the latter case, the results may be undefined. If the administrator has configured the receiver to parse a non-supported version, it should assume that these messages are legacy syslog messages and parse and process them with respect to RFC 3164 [14].

To be precise, a receiver receiving an unknown VERSION number, or a message without a valid VERSION, should discard the message by default. However, the administrator may configure it to not discard these messages. If that happens, the receiver may parse it according to RFC 3164 [14]. The administrator may again override this setting and configure the receiver to parse the messages in any way.

The spirit behind these guidelines is that the administrator may sometimes need the power to allow overriding of version-specific parsing, but this should be done in the most secure and reliable way. Therefore, the receiver should use the appropriate defaults specified above. This document is specific on this point because it is common experience that parsing unknown formats often leads to security issues.

A.5 SEVERITY Values

This section describes guidelines for using SEVERITY as outlined in Section 6.2.3.

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.6 TIME-SECFRAC Precision

The TIMESTAMP described in Section 6.2.5 supports fractional seconds. This provides ground for a very common coding error, where leading zeros are removed from the fractional seconds. For example, the

A.7 Case Convention for Names

Names are used at various places in this document, for example for SD-IDs and PARAM-NAMEs. This document uses "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.

There is one exception from this convention in this document itself. That is that experimental SD-IDs start with "x-", so they are hyphenated. While this looks like an inconsistency, it was done because it is common practice (e.g., in SMTP) to prefix experimental headers with "x-".

A.8 Leap Seconds

The TIMESTAMP described in Section 6.2.5 permits leap seconds, as described in RFC 3339 [8].

The value "60" in the TIME-SECOND field is used to indicate a leap second. This must not be misinterpreted. Implementors are advised to replace the value "60" if seen in the header, with the value "59" if it otherwise can not be processed, e.g., stored in a database. It should not be converted to the first second of the next minute. Please note that such a conversion, if done on the message text itself, will cause cryptographic signatures to become invalid. As such, it is suggested that the adjustment is not performed when the plain message text is to be stored (e.g., for later verification of signatures).

A.9 Syslog Senders Without Knowledge of Time

In Section 6.2.5.1, a specific TIMESTAMP for usage by senders without knowledge of time is defined. 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 TIMESTAMP described in Section 6.2.5.1 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.

If a receiver receives that special TIMESTAMP, it should know that the sender has no idea of what the time actually is and act accordingly.

A.10 Additional Information on PROCID

The objective behind PROCID (Section 6.2.8) 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.11 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.12 Recommendation for Diagnostic Logging

In Section 8.1, this document describes the need for as well as potential problems with diagnostic logging. In this section, a real-world approach to useful diagnostic logging is recommended.

While this document recommends writing meaningful diagnostic logs, it also recommends allowing an operator to limit the amount of diagnostic logging. At least, an implementation should differentiate between critical, informational, and debugging or diagnostic message. Critical messages should be issued only in real critical states, e.g., expected or occurring malfunction of the application or parts of it. A strong indication of an ongoing attack may also be considered critical. As a guideline, there should be very few critical messages. Informational messages should be used to indicate that all conditions are not fully correct, but still within the bounds of normal processing. A diagnostic message logging the fact that a malformed message has been received is a good example of this category. A debug diagnostic message should not be needed during normal operation, but merely as a tool for setting up or testing a system (which includes the process of an operator configuring multiple syslog applications in a complex environment). An application may decide not to provide any debugging diagnostic messages.

An administrator should be able to configure the level for which diagnostic messages will be written. Non-configured diagnostic messages should not be written but discarded. An implementor may create as many different levels of diagnostic messages as useful—the above recommendation is just based on real-world experience of what is considered useful. Please note that experience shows that too many levels of diagnostics typically do no good, because the typical administrator may no longer be able to understand what each level means.

Even with this categorization, a single diagnostic (or a set of them) may frequently be generated when a specific condition exists (or a system is being attacked). It will lead to the security issues outlined at the beginning of Section 8.1. To solve this, it is recommended that an implementation be allowed to set a limit of how many duplicate diagnostic messages will be generated within a limited amount of time. For example, an administrator should be able to configure that groups of 50 identical messages are logged within a specified time period with only a single diagnostic message. All subsequent identical messages will be discarded until the next time
interval. It is usually considered good form to generate a subsequent message identifying the number of duplicate messages that were discarded. While this causes some information loss, it is considered a good compromise between avoiding overruns and providing the most in-depth diagnostic information. An implementation offering this feature should allow the administrator to configure the number of duplicate messages as well as the time interval to whatever the administrator thinks is reasonable. It is up to the implementor what the term "duplicate" means. Some may decide that only totally identical (in octet-to-octet comparison) messages are actually duplicates, whereas others may say that a message that is of identical type but with just some changed parameter (e.g., changed remote host address) is also considered to be a duplicate. Both approaches have their advantages and disadvantages. Probably, it is best to also leave this configurable and allow the administrator to set the parameters.
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.