

syslog Working Group
Internet-Draft
Expires: January 13, 2005

R. Gerhards
Adiscon GmbH
July 15, 2004

The syslog Protocol
draft-ietf-syslog-protocol-05.txt

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. 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 become aware will be disclosed, in accordance with RFC 3668.

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, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes the syslog protocol. The syslog protocol is used to convey event notifications. This documents describes a layered architecture for an easily extensible syslog protocol. It also describes the basic message format and structured elements used to provide meta-information about the message.

Gerhards Expires January 13, 2005 [Page 1]

Internet-Draft The syslog Protocol July 2004

Table of Contents

1. Introduction	4
2. Definitions and Architecture	5
3. Transport Layer Protocol	8
3.1 Minimum Required Transport Mapping	8
4. Required syslog Format	9
4.1 Message Length	10
4.2 HEADER	11
4.2.1 VERSION	11
4.2.2 FACILITY	12
4.2.3 SEVERITY	13
4.2.4 TIMESTAMP	13
4.2.5 HOSTNAME	15
4.2.6 TAG	16
4.3 STRUCTURED-DATA	17
4.3.1 Format	18
4.3.2 Examples	19
4.4 Examples	21
5. Structured Data IDs	24
5.1 time	24
5.1.1 tzknown	24
5.1.2 issynced	24
5.1.3 syncaccuracy	25
5.1.4 Examples	25
5.2 origin	26
5.2.1 format	26

- 5.2.2 ip 26
- 5.2.3 enterpriseID 26
- 5.2.4 software 27
- 5.2.5 version 27
- 5.2.6 Example 27
- 6. Security Considerations 28
 - 6.1 Diagnostic Logging 28
 - 6.2 Control Characters 29
 - 6.3 Packet Parameters 30
 - 6.4 Single Source to a Destination 30
 - 6.5 Multiple Sources to a Destination 31
 - 6.6 Multiple Sources to Multiple Destinations 31
 - 6.7 Replaying 31
 - 6.8 Reliable Delivery 32
 - 6.9 Message Integrity 32
 - 6.10 Large Size Messages 32
 - 6.11 Message Observation 33
 - 6.12 Misconfiguration 33
 - 6.13 Forwarding Loop 33
 - 6.14 Load Considerations 34
 - 6.15 Denial of Service 34

Gerhards Expires January 13, 2005 [Page 2]

Internet-Draft The syslog Protocol July 2004

- 6.16 Covert Channels 34
- 7. Notice to RFC Editor 35
- 8. IANA Considerations 36
 - 8.1 Version 36
 - 8.2 SD-IDs 36
 - 8.3 Facility and Severities 36
- 9. Authors and Working Group Chair 37
- 10. Acknowledgements 38
- 11. References 38
 - Author's Address 39
 - Intellectual Property and Copyright Statements 40

Gerhards Expires January 13, 2005 [Page 3]

Internet-Draft The syslog Protocol July 2004

1. Introduction

This document describes the semantics of the syslog protocol, outlines transport mappings and provides a standard format for all syslog messages. It also describes structured data elements, which can be used to precisely define specific message aspects. Many of these structured data elements carry optional information and are optional themselves.


This document describes a layered architecture for syslog. The goal of this architecture is to separate the functionality into separate layers and thus provide easy extensibility.



In order to claim compliance with this document, an implementation **MUST** at least implement all **REQUIRED** parts. Optional parts must not necessarily be implemented.


Gerhards Expires January 13, 2005 [Page 4]
Internet-Draft The syslog Protocol July 2004

2. Definitions and Architecture

The following definitions will be used in this document:




- o An application that can generate a message will be called a "sender".
- o An application that can receive a message will be called a "receiver".
- o An application that can receive the message and forward it to another machine will be called a "relay".
- o An application that receives the message and does not relay it to any other receivers will be called a "collector". s has been commonly known as a "syslog server".

There are applications that both receive messages  forward them to another applicaton D generate syslog messages themselves. An example for this may be an application that operates as a syslog relay as one service while at the same time running other services. These services may be monitored by the same application, generating new syslog messages. Such an application acts both as a relay AND a sender. This case is specifically mentioned as the role an application plays has special significance, for example on formatting. An application as described here may thus have two separate configurations for each of the applications's operations modes.

A given device may run multiple syslog instances. Thus, a device can potentially be a sender, receiver, collector and relay all in once. The key point is that there is not necessarily a direct relationship between the syslog-enabled application and the device it is running on. This is important to note because syslog was traditionally used on many low-powered devices acting only as syslog senders. ple

still tend to think of a syslog sender as a "device" - so it is important to note that this is not necessarily correct.

The architecture of syslog applications may be summarized as follows:

- o Senders send messages to relays or collectors with no knowledge of whether it is a collector or relay.
- o Senders may be configured to send the same message to multiple receivers.
- o Relays may send all or some of the messages that they receive to a subsequent relay or collector.  The case where they do not forward all of their messages, they are acting as both a collector and a relay. In the following diagram, these applications will be designated as relays.
- o Relays may also generate their own messages and send them on to subsequent relays or collectors.  That case it is acting as a sender.  These applications will also be designated as a relay in

Gerhards

Expires January 13, 2005



[Page 5]

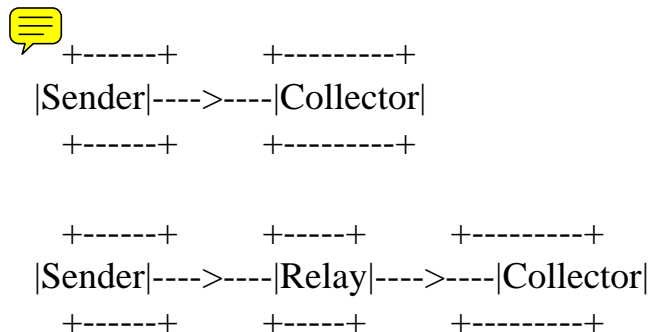
Internet-Draft

The syslog Protocol

July 2004

the following diagram.

The following  architectures shown in Diagram 1 are valid while the first one has been known to be the most prevalent. Other arrangements of these examples are also acceptable. As noted above, in the following diagram relays may pass along all or some of the messages that they receive along with passing along messages that they internally generate. 



```

+-----+ +-----+ +-----+ +-----+
|Sender|-->--|Relay|-->--...-->--|Relay|-->--|Collector|
+-----+ +-----+ +-----+ +-----+

```

```

+-----+ +-----+ +-----+
|Sender|---->----|Relay|---->----|Collector|
| |-\ +-----+ +-----+
+-----+ \
      \ +-----+ +-----+
        \-->--|Relay|---->----|Collector|
          +-----+ +-----+

```

```

+-----+ +-----+
|Sender|---->----|Collector|
| |-\ +-----+
+-----+ \
      \ +-----+ +-----+
        \-->--|Relay|---->----|Collector|
          +-----+ +-----+

```

```

+-----+ +-----+ +-----+
|Sender|---->----|Relay|---->-----|Collector|
| |-\ +-----+ /--| |
+-----+ \ / +-----+
      \ +-----+ /
        \-->--|Relay|-->--/
          +-----+

```

```

+-----+ +-----+ +-----+


```

```

|Sender|---->----|Relay|---->-----|Collector|
| |-\ +-----+ /--| |
+-----+ \ / +-----+
      \ +-----+ /
        \ |+-----+| /

```

```
\-->-||Relay ||->---/  
|+-----|| /  
||Sender||->-/  
|+-----+|  
+-----+
```

Diagram 1. Some possible syslog  hitectures

Gerhards Expires January 13, 2005 [Page 7]

Internet-Draft The syslog Protocol July 2004

3. Transport Layer Protocol

This document DOES NOT specify a specific transport layer protocol. Instead, it describes the format of a syslog message in a transport layer independent way.

Transport mappings being defined MUST ensure that a message formatted according to this document can be transmitted unaltered over the mapping. If the mapping needs to perform temporary transformations, it must be guaranteed that the message received at the final destination is an exact copy of the message sent from the initial originator. This is vital because otherwise cryptographic verifiers (like signatures) would be broken.

3.1 Minimum Required Transport Mapping

To claim compliance with this document, each implementation MUST at least implement the UDP transport mapping described in Anton Okmianski "Transmission of syslog messages over UDP" (draft-ietf-syslog-transport-udp-01). This is to ensure a minimum interoperability between systems implementing this document.

Gerhards Expires January 13, 2005 [Page 8]

Internet-Draft The syslog Protocol July 2004

4. Required syslog Format

The syslog message has the following ABNF [8] definition:

; The general syslog message format

SYSLOG-MSG = HEADER STRUCTURED-DATA MSG

HEADER V" VERSION SP SP FACILITY SP
SEVERITY SP TIMESTAMP SP HOSTNAME SP TAG SP

VERSION = 1*3DIGIT

FACILITY = 1*10DIGIT ; range 0..2147483648

SEVERITY = "0" / "1" / "2" / "3" / "4" / "5" /
"6" / "7"

HOSTNAME = 1*255PRINTUSASCII ; a FQDN

TAG = SENDER-NAME SP SENDER-INST; 32 chars max

SENDER-NAME = 1*48VISUAL

SENDER-INST = 1*16VISUAL

VISUAL = (%d33-57/%d59-126) ; all but SP

TIMESTAMP = full-date "T" full-time

date-fullyear = 4DIGIT
 date-month = 2DIGIT ; 01-12
 date-mday = 2DIGIT ; 01-28, 01-29, 01-30, 01-31 based on
 ; month/year
 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

partial-time = time-hour ":" time-minute ":" time-second
 [time-secfrac]

full-date = date-fullyear "-" date-month "-" date-mday

full-time = partial-time time-offset

STRUCTURED-DATA = *STR-DATA-ELT SP

MSG = *OCTET ; VALID UTF-8 String

OCTET = %d00..255

LF = %d10 

CR = %d13 

SP = %d32

PRINTUSASCII = %d33-126

Gerhards Expires January 13, 2005 [Page 9]

Internet-Draft The syslog Protocol July 2004


; Format of structured data element 


STR-DATA-ELT = "[" SD-ID 0*(1*SP SD-PARAM) *SP "]" 

SD-ID = SD-IANA / SD-ID-EXPERI



SD-ID-IANA = 1*1ID-CHAR [1*1ID-CHARNSLASH [1*62ID-CHAR]]

SD-ID-EXPERI = %d120 "-" 1*62ID-CHAR ; "x-" (lower case 'x!')


ID-CHAR = %-33 / %d35-60 / %d62-92 / %d94-126 /
 %d128-254


; all US-ASCII but "" (%d34), '=' (%d61), ']' 



; (%d93)



-CHARNOSLASH = %d32-33 / %d35-44 / %d46-60 / %d62-92 /
 %d94-126 / %d128-254
 ; same as ID-CHAR bwithout '-' (%d45)
 SD-PARAM = PARAM-ID "%d34" PARAM-VALUE "%d34"
 PARAM-ID = 1*64ID-CHAR
 PARAM-VALUE = *(SAFE-CHAR / ESCAPED-CHAR)
 SAFE-CHAR = *((%d32-33) / (%d35-46) / (%d48-92) /
 (%d94-126) / (%d128-254))
 ESCAPED-CHAR = ("\" / %d47.34 / "]") ; 47.34 is \"

4.1 Message Length

The maximum length of any syslog message is 16,000,000 bytes. Any receiver receiving a larger message **MUST** discard the message. A diagnostic message **SHOULD** be logged in this case.

In order to claim compliance with this document, an implementation **MUST** support messages of at least 65,000 bytes. 


A receiver **MAY** support messages longer than the minimum outlined above.  However, it **MUST NOT** do so. The receiver **MAY** support any length between the minimum and the maximum lengths defined. If it receives messages within the maximum length, but with a length larger than it handles, the receiver **MAY** either discard the message as whole or **MAY** truncate it. 


As such, a sender **MUST NOT** assume that a message can be processed by all receivers if it is larger than the minimum length defined here. If the message generated is larger, the most important parts of the message **SHOULD** be placed early in the message (within the minimum length area). This ensures the **n** important parts will be seen by the receiver even if it (or an interim relay) truncates the message.

For performance and usability reasons, a sender **SHOULD** try to limit the message to have a maximum size of 480 bytes (total including the header part).


4.2 HEADER






The header **MUST** contain the token identifying the message as a syslog message complying with this specification, the version of the specification it complies to, the facility, the severity, the timestamp, the hostname and the tag. Each of these fields **MUST** be present and **MUST** be of correct syntax. The code set used in the **HEADER** **MUST** be seven-bit ASCII in an eight-bit field as described in RFC 2234 [8]. These are the ASCII codes as defined in "USA Standard Code for Information Interchange" ANSI.X3-4.1968 [2].

If the header is not syntactically correct, the receiver **MUST NOT** try to sub-parse some of the header fields in order to find a "good" interpretation. However, the receiver **MAY** assume it is a RFC3164 compliant message and **MAY** decide to process it as such. In this case, RFC3164 semantics **MUST** be used. 

As a note to implementors, the "V" token at the very beginning of the message **MAY** be used as a rough indication whether or not the message complies to this document. However, it is not sufficient to assume it complies  this document just because the first character is a "V". As written above, the full header **MUST** be validated to assume this.

4.2.1 VERSION

 The Version field denotes the version of the syslog protocol specification the message is formatted to. It is used to uniquely identify the message format should later revisions of the syslog protocol specification change the format.

 Note well: this document is the first to specify  this format, including the VERSION in the header. Any previous  syslog specification had a different header. As outlined under **HEADER**  above, an invalid **HEADER** will automatically tell the receiver that the message is **NOT** compliant to this specification.  such, all version information is well defined (absence of version information means legacy syslog by the fact that the header is invalid).

The VERSION MUST be a numerical value. It MUST be one of the IANA assigned valid VERSION numbers. It starts at 1, which means the format specified in this document. The VERSION number MUST be incremented for each new syslog protocol specification that changes the format. If MUST NOT be incremented if a new syslog protocol specification does not change the syntax and semantics of the message format.

The sender of the syslog message MUST specify the VERSION of the

Gerhards Expires January 13, 2005 [Page 11]

Internet-Draft The syslog Protocol July 2004

format that the message was formatted to.

The receiver MUST check the VERSION. If the VERSION is within the set of format versions supported by the receiver, the receiver MUST parse the message according to the correct syslog protocol specification. A receiver MUST NOT parse a previous version with some parsing rules from a later 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. It SHOULD NOT try to process the message, but it MAY try this if the administrator has configured the receiver to do so. In the later case, the results may be undefined. If the administrator has instructed the receiver to parse non-supported version, it SHOULD assume that these messages are legacy syslog messages and parse and process them in respect to RFC 3164. Again, the administrator MAY configure the receiver to use a different algorithm.

To be precise, a receiver receiving an unknown VERSION number, MUST, by default, ignore it. The administrator may configure it to not ignore it. Then, the receiver MUST, by default, parse it according to RFC 3164. The administrator may again override this setting. In this case, the receiver MAY use whatever method the administrator has chosen. In this case, the receiver MUST ensure that no application reliability issue occurs. If there is a chance for this, it MUST NOT

allow the administrator to select an insecure mode.

The spirit of this behaviour is that the administrator may sometime 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 **MUST** use the appropriate defaults specified above. This document is so specific on the defaults and modes because it is common experience that parsing unknown formats often leads to security issues.

4.2.2 FACILITY

The facility is primarily a way to filter messages at the receiver. It is a numerical value. 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 vendors, softwares and senders. Therefore, no specific semantics for facility codes are implied in this document.

FACILITY is just a sender-supplied numerical identifier that can be used for filtering by the receiver. The facility in itself does not have any semantics. Semantics **MUST** be applied by site configuration

Gerhards Expires January 13, 2005 [Page 12]


Internet-Draft The syslog Protocol July 2004

(through the site's administrator).


Any implementation of this document **MUST** support free configuration of the FACILITY on the sender.



4.2.3 SEVERITY



The SEVERITY field is used to indicate the severity that the sender of the message assigned to it. It is a numerical value with just few values. The traditional syslog severity values are reused, because they have proven to be useful and sufficient in reality.

SEVERITY is a numerical field, which **MUST** contain one of the digits from 0 to 7. Any other value is invalid and **MUST NOT** be used. 


Each of the numerical codes has been assigned the following semantics: 

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 today's activity chart 
7	Debug: debug-level messages

All implementations **SHOULD** try to assign the most appropriate severity to their message. Most importantly, test aid like messages or program debugging information **SHOULD** be assigned severity 7. Severity 0 **SHOULD** be reserved for high-privilege core processes of very high importance (like serious hardware failures or a very soon power failure). An implementation **MAY** use severities 0 and 7 for other purposes if this is configured by the administrator.  

In general, a receiver should abide to the fact that severities are often very subjective. As such, a receiver **MUST** not assume that all senders have the same sense of severities.  

4.2.4 TIMESTAMP

The **TIMESTAMP** field is a formalized timestamp as taken from RFC 3339 [12]. 

Note well: RFC 3339 makes allowances for multiple syntaxes for a 

Gerhards Expires January 13, 2005 [Page 13]

Internet-Draft The syslog Protocol July 2004

timestamp to be used in various cases. This document mandates a restricted set of syntaxes. The primary characteristics of **TIMESTAMP** used in this document are as follows.

- o the "T" and "Z" characters in this syntax **MUST** be upper case.
- o usage of the "T" character is mandatory. It **MUST NOT** be replaced by any other character (like a SP character).
- o the sender **SHOULD** include time-secf (fractional seconds) if its clock accuracy and performance permits.
- o the entire length of the **TIMESTAMP** field **MUST NOT** exceed 32 characters.

Please also note that RFC 3339 permits the value "60" in the second part to indicate a leap second. This must not be misinterpreted. As a suggestion for application developer, it is 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 to 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 cryptographical signatures to become invalid. As such, it is suggested that the adjustment is not done when the plain message text is to be stored (e.g. for later verification of signatures).

Two samples of this format are:

```
1985-04-12T23:20:50.52Z
1985-04-12T18:20:50.52-06:00
```

The first represents 20 minutes and 50.52 seconds after the 23rd hour of April 12th, 1985 in UTC. The second represents the same time but expressed in the Eastern US timezone (daylight savings time being observed).

A single space character **MUST** follow the **TIMESTAMP** field.

4.2.4.1 Syslog Senders without knowledge of Time

There is one special case, and this is a syslog sender that is **NOT** aware of time at all. It can be argued if such a syslog sender is something that actually can be found in today's IT infrastructure. However, discussion has indicated that those things may exist in reality and as such there should be a guideline what to do in such a case. The other assumes that those syslog senders will most probably be found in embedded devices.

Note well: an implementation **MUST** emit a valid **TIMESTAMP** if the underlying operating system, programming system and hardware is capable of doing so. A proper **TIMESTAMP** **MUST** be emitted even if it is hard, but doable, to obtain the system time. The rule outlined here **MUST** only be used when there is absolutely no way to obtain time

Gerhards Expires January 13, 2005 [Page 14]
Internet-Draft The syslog Protocol July 2004

information from the system environment. This rule **MUST NOT** be used as an excuse for lazy implementations.

A syslog sender who has absolutely no way of obtaining system time from its environment, **MUST** write the following **TIMESTAMP**:

2000-01-01T00:00:60Z

This **TIMESTAMP** is in the past, but more importantly, it shows a time that never existed, because January, 1st 2000 had no leap second (note the 60 in the second indicator). As such, this **TIMESTAMP** can never exist in a valid syslog message, but it is still syntactically correct in regard to the ABNF above.

If a syslog receiver receives this **TIMESTAMP** it **MUST** treat the **TIMESTAMP** to be well-formed but **MUST** also know that the sender had no idea of what the time actually is. It is left to the application developer what this means for further processing of the message (this is beyond the scope of this document).

4.2.5 HOSTNAME

The **HOSTNAME** field contains the original creator of 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 will be referred to in this document as **HOSTNAME-STD13**.

If the **HOSTNAME-STD13** is not known to the originator, but it knows its

statically assigned IP address, it SHOULD use either its statically assigned IPv4 address or its statically assigned IPv6 address.



If the IP address is not statically assigned, it SHOULD specify its host name (without domain name).

If the host name is not known, it SHOULD specify its dynamically assigned IPv4 or IPv6 address.

If the sender does not know its IP address at all, it SHOULD provide the value "0.0.0.0" if it knows it has an IPv4 stack and "0:0:0:0:0:0:0:0" if it knows it has an IPv6 stack.



If the sender does not know anything about its identity or the type of stack used, it MUST provide the value "0:0:0:0:0:0:0:0".



If an IPv4 address is used, it MUST be shown as the dotted decimal notation as used in STD 13 [4], and will be referred to as

Gerhards

Expires January 13, 2005

[Page 15]

Internet-Draft

The syslog Protocol

July 2004

HOSTNAME-IPV4. If an IPv6 address is used, any valid textual representation used in RFC 2373 [9], section 2 MUST be used and will be referred to as HOSTNAME-IPV6.

If a sender has multiple IP addresses, it SHOULD use a consistent value in the HOSTNAME field. This consistent value MUST be one of its actual IP addresses. If a sender is running on a machine which has been assigned addresses both statically and dynamically, that consistent value SHOULD be from the statically assigned addresses. As an alternative, it MAY use the IP address of the interface that is used to send the message.






Please note that the "SHOULD"s above are very strong "SHOULD"s. Please consider them to being "MUST"s, except that an operator-override of the precedence is allowed. The default settings MUST be as specified above.




A single space character **MUST** follow the HOSTNAME field. 



4.2.6 TAG

The TAG is a way to facilitate identification of classes of syslog messages at the receiver. The TAG is composed of a more or less static part, which identifies the sender and a dynamic part, which identifies a specific sender instance (at least this is the intention). 

These two parts are included in their very own fields, which are described below. The author has decided to use the name "TAG" as an umbrella for these two fields, because this is historically well-known syslog terminology. As such, it is easier to understand the meaning of these two fields if the well-known term is being used. 

4.2.6.1 SENDER-NAME

The SENDER-NAME is a string of visible (printing) characters excluding SP, that **MUST NOT** exceed 48 characters in length. The first occurrence of a SP (space) will terminate the SENDER-NAME field, but is not part of it. 

The SENDER-NAME **SHOULD** identify the sender. For example, it may include a configured name (e.g. "MySuperSender") or the (partial) image name of the application sender (e.g. "postfix/smtpd"). The basic idea is that SENDER-NAME should be relatively static during the lifespan of a sender. 

Gerhards

Expires January 13, 2005


[Page 16]

Internet-Draft

The syslog Protocol

July 2004

4.2.6.2 SENDER-INST

The SENDER-INST is a string of visible (printing) characters excluding SP, that **MUST NOT** exceed 16 characters in length. The 

first occurrence of a SP (space) will terminate the SENDER-INST field, but is not part of it.

The SENDER-INST SHOULD identify a specific instance of the sender process. It is recommended to provide an operating system process ID, eventually together with a thread ID, if these things exist. There is no specific format for this information required.

If the sender has no concept (or knowledge) of its instance, a dash "-" SHOULD be placed inside SENDER-INST. A sender aware of its instance SHOULD NOT place a dash inside SENDER-INST. In the remote case the dash may actually be the instance ID, the sender SHOULD escape it in some way, e.g. by specifying "ID:-".

The objective behind SENDER-INST is to provide a quick, but unreliable, way to detect a new instance of the same sender. Properly used, the SENDER-INST can be especially helpful for analysis purposes. However, a receiver MUST NOT assume that the SENDER-INST is actually instance-specific and MUST work flawlessly even if the SENDER-INST never changes.

4.3 STRUCTURED-DATA

While syslog traditionally contains freeform data, the structured data field is used to represent well-structured data. Structured data are special, well defined data elements designed to be easily computable and parsable. They may transport meta data for the syslog protocol as well as application-defined information (e.g. traffic counters, IP addresses and other well-defined elements).

Structured data elements MUST be present in the STRUCTURED-DATA field. This field MUST be followed by a SP character. Structured data elements themselves contain SP characters, but only within a single element. These "in-element" SP characters MUST NOT be mistaken as terminators of the STRUCTURED-DATA field. Structured data elements MUST immediately follow each other, without any SP in between.

The code set used in STRUCTURED-DATA must be UNICODE. It MUST be encoded in UTF-8 as specified in RFC 2279 [7]. A sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence. Most importantly, it must not fail if control characters are present in the STRUCTURED-DATA part.

Gerhards

Expires January 13, 2005

[Page 17]

Internet-Draft

The syslog Protocol

July 2004

There is a certain set of structured data that is under IANA control. These structured data elements are described in this and other RFCs. A second set of structured data elements is not under IANA-control. This set **MUST** be used for experimental or vendor-specific elements.

A syslog message may contain none, one or multiple structured data elements.

4.3.1 Format

Structured data elements **MUST** conform to the ABNF above.

Each structured data element **MUST** begin with the token "[". It **MUST** end with the token "]". These tokens define the beginning and end of the element. SP characters may occur within the element, but not outside of it.

The beginning token **MUST** immediately be followed by the ID of the structured data element. No space is allowed between the beginning token and the SD-ID. The SD-ID uniquely identifies the type and purpose of the element.

IANA controls ALL SD-IDs without a hyphen '-' in the second character position. Experimental or vendor-specific SD-IDs **MUST** start with "x-". Values with a hyphen on the second character position and the first character position not being a lower case "x" are undefined and **SHOULD NOT** be used. Receivers **MAY** accept them.

If a receiver receives a well-formed but unknown SD-ID, the receiver **SHOULD** ignore this element. It **MUST NOT** malfunction because of this unknown SD-ID.

The SD-ID is followed by none, one or many optional parameter/value pairs. Each of them **MUST** start with the parameter name, **MUST** be

followed by an equal sign and a quote sign. There **MUST NOT** be any space between the SD-ID, the equal and the quote sign. This is followed by the parameter value and then another quote sign.

The parameter value may contain any character, but the three special characters '"', '\ ' and ']' **MUST** be escaped. This is necessary to avoid parsing errors. Please note that escaping ']' would actually not be necessary but is required in order 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 message, the receiver **MUST** replace the two-character sequence

Gerhards Expires January 13, 2005 [Page 18]

Internet-Draft The syslog Protocol July 2004

with just the second character received. It is recommended that the receivers logs a diagnostic message in this case. The receiver **MUST** otherwise ignore the invalid escape sequence.

Parameter/Value pairs **MUST** be separated by at least one SP character. Multiple SP characters **MAY** be used, but **SHOULD** be avoided.

The structured data element **MUST** be terminated by the character ']', the ending token. This **MUST** follow the last parameter/value pair. There **SHOULD** be no SP in front of the ending token, but there **MAY** be one or multiple SP in front of it.

If multiple structured data elements are written, they **MUST** all sequentially be written and no SP be written between those elements.

4.3.2 Examples

All examples show the structured data part of the message, only. examples should be considered to be on one line. They are wrapped on multiple lines for readability purposes, only.

Example 1

```
[x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"]
```

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

Example 2

```
[x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"][x-adiscon-priority class="high"]
```

This is the same example, but with a second structured data element. Please note that the structured data element does immediately follow the first one.

Example 3 - Invalid

```
[x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"] [x-adiscon-priority class="high"]
```

This is more or less the same example as in example 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

The code set used in MSG must be UNICODE. It MUST be encoded in

UTF-8 as specified in RFC 2279 [7]. A sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence. Most importantly, it must not fail if control characters are present in the MSG part. Cause the STRUCTURED-DATA field to end after the first element. The second element will actually become part of the MSG field.

Example 4 - Invalid

```
[ x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"][x-adiscon-priority class="high"]
```



This example again looks much like example 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 begin bracket, so the SP character makes this an invalid element.

Example 5

```
[sigSig Ver="1" RSID="1234" ... Signature="....."]
```



Example 5 is not a full example. It shows how a hypothetical IANA assigned SD-ID may be used inside an otherwise empty message. Please note that the dots denote missing fields, which have been left out for brevity.



The code set used in MSG must be UNICODE. It MUST be encoded in UTF-8 as specified in RFC 2279 [7]. A sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence. Most importantly, it must not fail if control characters are present in the MSG part.

The code set used in MSG must be UNICODE. It MUST be encoded in UTF-8 as specified in RFC 2279 [7]. A sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence. Most importantly, it must not fail if control characters are present in the MSG part.

The code set used in MSG must be UNICODE. It MUST be encoded in UTF-8 as specified in RFC 2279 [7]. A sender MAY issue any valid UTF-8 sequence. A receiver MUST accept any valid UTF-8 sequence. Most importantly, it must not fail if control characters are present in the MSG part.



Note to implementors: the octet value 0 (0x00), the C string terminator, is a valid character and MAY be present in the MSG part.

The implementor must ensure that reception of 0x00 causes no malfunction, specifically does not cause message truncation. C



programmers please be aware that this requires proper escaping and/or special string handling.

Another note to implementors: please keep the presence of control characters in mind when writing textual log files. For example, LF is a valid character and may be present in the MSG part. Writing this plainly to a log file may cause problems with log parsers and other programs that process the log file. It is good practice to escape non-printable characters in a consistent way when writing to text files.

4.4 Examples

The following examples are given.

Example 1

```
V1 888 4 2003-10-11T22:14:15.003Z mymachine.example.com su: 'su
root' failed for lonvick on /dev/pts/8
```

In this example, the VERSION is 1 (formatted according to this document), the FACILITY has the value of 888 (whatever this means is up to the sender and recipient). The message was created on October, 11th 2003 at 10:14:15pm, 3 milliseconds into the next second. The timestamp is in UTC. Please note that the sender had millisecond resolution. The sender may have actually had a better resolution, but by providing just three digits for the fractional settings, he does not tell us this. The message originated from a host that calls itself "mymachine.example.com". The TAG is "su:". Note that the colon is part of the tag. Note the TWO SP characters following - the second SP character is the STRUCTURED-DATA delimiter. It tells us that no STRUCTURED-DATA is present in this message. The MSG is "'su root' failed for lonvick...". Please note that the SP characters are not part of the MSG, they are the separators.

As a note to implementors: please note that the sender had millisecond time resolution in this example. A common coding bug is that leading zeros are not written for fractional seconds. Very

often, the above timestamp is erroneously being written as: "2003-10-11T22:13:14.3". This would indicate 300 milliseconds instead of the 3 milliseconds that are actually meant. Please make sure that an implementation handles this correctly.

Gerhards Expires January 13, 2005 [Page 21]

Internet-Draft The syslog Protocol July 2004

Example 2

```
V1 20 6 2003-08-24T05:14:15.000003-09:00 10.1.1.1
myproc[10]:%% It's time to
make the do-nuts. %% Ingredients: Mix=OK, Jelly=OK #
Devices: Mixer=OK, Jelly_Injector=OK, Frier=OK # Transport:
Conveyer1=OK, Conveyer2=OK # %%
```

In this example, the VERSION is again 1. The FACILITY is within the legacy syslog range (20), as such we assume the user has specifically configured the sender to use this FACILITY. The severity is 6 ("Notice" semantics). The timestamp now has microsecond resolution, indicated by the additional digits in the fractional seconds. The sender indicates that its local time is -9 hours from UTC. Given the date stamp, we can assume the sender is in the US Pacific time zone during daylight savings time. The HOSTNAME is "10.1.1.1", so the sender did not know its host- and domainname and used the IPv4 address instead. The TAG is "myproc[10]:%%" - we can speculate that the sender actually wanted the tag to be "myproc[10]:", but because there was no SP following it, the TAG continues until the first SP. The message is "It's time to make the do-nuts.....".

Example 3 - An Invalid Message

```
V1 20 6 2003-08-24T05:14:15.000000003-09:00 10.1.1.1
myproc[10]:%% It's time to
make the do-nuts. %% Ingredients: Mix=OK, Jelly=OK #
Devices: Mixer=OK, Jelly_Injector=OK, Frier=OK # Transport:
Conveyer1=OK, Conveyer2=OK # %%
```



This example just just like Example 2, but this time the sender is overdoing with the clock resolution. It is supplying nanosecond resolution. This will result in the TIME-SECFRAC part to be longer than the allowed 6 digits, which invalidates the header and thus the message. A receiver MUST NOT try to "fix" this error. It MUST detect this as an invalid message and SHOULD log a diagnostic entry. If the receiver is capable of processing legacy syslog messages, it MUST assume that this message is legacy syslog and act accordingly.



Example 4 - with STRUCTURED-DATA

```
V1 8 4 2003-10-11T22:14:15.003Z mymachine.example.com
EvtSLog: [x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"] An application event log entry.
```



This example is modelled after example one. However, this time it

Gerhards Expires January 13, 2005 [Page 22]

Internet-Draft The syslog Protocol July 2004

contains a STRUCTURED-DATA, a single element with the value "[x-adiscon-iut iut="3" EventSource="Application" EventID="1011"]". The MSG itself is "An application event log entry."

Example 5 - subtle error

```
V1 888 4 2003-10-11T22:14:15.003Z mymachine.example.com
EvtSLog: [x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"] [x-adiscon-priority class="high"] An
application event log entry.
```



Here, the sender seems to have intended to send two structured data

elements. However, he included a SP character AFTER the first element. That space terminates the STRUCTURED-DATA field. So it will be interpreted as follows. STRUCTURED-DATA is: "[x-adiscon-iut iut="3" EventSource="Application" EventID="1011"]". The MSG is "[x-Adiscon-priority class="high"] An application event log entry."

Example 6 - Correct form of Example 5

```
V1 888 4 2003-10-11T22:14:15.003Z mymachine.example.com
EvtSLog: [x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"][x-adiscon-priority class="high"] An
application event log entry.
```

This is the corrected form of what has been shown in example 5. Now the STRUCTURED-DATA is "[x-adiscon-iut iut="3" EventSource="Application" EventID="1011"] [x-Adiscon-priority class="high"]" and the MSG is "An application event log entry."

Example 7 - Just STRUCTURED-DATA

```
V1 888 4 2003-10-11T22:14:15.003Z mymachine.example.com
EvtSLog: [x-adiscon-iut iut="3" EventSource="Application"
EventID="1011"][x-adiscon-priority class="high"]
```

This example shows that a message with just STRUCTURED-DATA (and no MSG part). This is a valid case.

5. Structured Data IDs

This section defines the currently IANA-registered structured data IDs (SD-IDs). See Section 4.3 for a definition of structured data elements.

5.1 time

The SD-ID "time" is used by the original sender to describe its notation 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 if its time zone information is correct. It MAY be written in any other case. The main use of this structured data element is to provide some information on how much the TIMESTAMP described in Section 4.2.4 can be trusted.

5.1.1 tzknown

The "tzknown" parameter indicates if the original sender knows its timezone, as specified in the TIMESTAMP. If so, the value "1" MUST be used. If the time zone information is in doubt, the value "0" MUST be used. Please note that if the sender KNOWS its timezone but decides to emit UTC, the value "1" should still be used (because the time zone is known).

It is suggested that an implementation uses "0" as default and changes to "1" only 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 acknowledges the correctness of the time zone information.

If a system is properly synchronized to an external time zone, the value "1" should be used in most cases. However, we know of external time zone synchronizations that do NOT provide the exact time zone information, just a precise local time. In such (rare) cases, the "time" structured data element should indicate a properly synced time but the absence of time zone information by setting the "tzknown" value to "0".

5.1.2 issynced

The "issynced" parameter indicates if the original sender is synchronized to a reliable external time source, e.g. via NTP. If so, the value "1" MUST be provided. If not, the value "0" MUST be provided.

Gerhards Expires January 13, 2005 [Page 24]

Internet-Draft The syslog Protocol July 2004

5.1.3 syncaccuracy

The "syncaccuracy" parameter indicates how accurate the original sender thinks the time synchronization it participates in is.


If the value "0" is used for "issynced", this parameter MUST NOT be written. If the value "1" is used for "issynced" but the "syncaccuracy" parameter is absent, a receiver should assume that the time information provided is accurate enough to be considered correct. The "syncaccuracy" parameter should ONLY be written if the original sender actually has knowledge of the reliability of the external time source. In reality, in most cases, it does not have this - then the "syncaccuracy" parameter MUST not be written to prevent false impressions.

The "syncaccuracy" parameter is an interger describing the maximum number of milliseconds that the clock may be off between synchoronization intervals.


5.1.4 Examples

The following is an example of a system that knows that it does neither know its time zone nor if it is being synchronized:


```
[time tzknown="0" issynced="0"]
```


With this information, the sender indicates that its time information can not be trusted. 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:

```
[time tzknown="1" issynced="1"]
```

The author considers this to be the typical case. While we do not know the accuracy of the external time synchronization, the time stamp should be good enough for all message correlations with other senders' messages. 

 Note well: this case **SHOULD** be assumed by a receiver if no "time" structured data element is provided by the sender.

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:

Gerhards Expires January 13, 2005 [Page 25]

Internet-Draft The syslog Protocol July 2004

```
[time tzknown="1" issynced="1" syncaccuracy="60000"]
```

The difference between this and the previous example is that the sender knows that its clock will be kept within 60 seconds (more or less) 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.

Knowing the accuracy of the time synchronization can be helpful when correlating syslog messages.

It is important to not create a false impression of accuracy. A sender **MUST** 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.

5.2 origin

The SD-ID "origin" is optional. It MAY be used by a sender to indicate the origin of a syslog message. It has the following parameters:

5.2.1 format

The "format" parameter is optional. If it is present, it denotes the format that this message was originally been created in. Its value MUST be the number of the RFC it complies to. If the message complies to an Internet-Draft format, it must specify the full internet draft name. For example, as of this writing, format may either hold the string "3164" (RFC 3164) or "draft-ietf-syslog-protocol-05.txt".

5.2.2 ip

The "ip" parameter is optional. If it is present, it denotes the IP address that the sender knows it had at the time of sending this message. It must be either the textual representation of an IPv4 address or the textual representation of an IPv6 address as outlined in Section 4.2.5. A host name or FQDN MUST NOT be used inside the "ip" parameter.

If a sender has multiple IP addresses, it MAY either use a single of its IP addresses in the "ip" parameter or it MAY include MULTIPLE "ip" parameters in a single "origin" structured data element.

5.2.3 enterpriseID

The "enterpriseID" is optional. If it is present, it uniquely

Gerhards Expires January 13, 2005 [Page 26]


Internet-Draft The syslog Protocol July 2004



identifies the vendor whose software or device created the message.

By specifying an enterpriseID, the vendor allows more specific

parsing of the message. This may be of aid to log analysers and similar processes.

The enterprise ID is an integer. It **MUST** be the enterprise ID assigned by IANA to the vendor whoms software or device created the syslog message. 

5.2.4 software

The "software" parameter is optional. If it is present, it unquily identifies the software that created this message. Please note that when "software" is specified "enterpriseID" **SHOULD** also be specified, so that a specific vendor's software can be identified.

Specifying the "software" parameter is an aid to log analysers and similar processes.

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

5.2.5 version

The "version" parameter is optional. If it is present, it unquily identifies the version of the originator that wrote this log message. This is most useful in combination with the "software" and "enterpriseID" parameters. If it is used, the "software" and "enterpriseID" parameters **SHOULD** be provided, too.

Specifying the "version" parameter is an aid to log analysers and similar processes.

The "version" is a string. It **MUST** not be longer than 32 characters.

5.2.6 Example

The following is an example with multiple IP addresses:


```
[origin format="draft-ietf-syslog-protocol-05.txt" ip="192.0.2.1"  
ip="192.0.2.129"]
```

This example is wrapped for readability. With it, the sender indicates that it has formatted the message according to this draft and that it has two ip address, one being 192.0.2.1 and the other one

being 192.0.2.129.

Gerhards Expires January 13, 2005 [Page 27]
Internet-Draft The syslog Protocol July 2004

6. Security Considerations

This section is to be updated once the rest of the document has been confirmed. The current content is incomplete and potentially not in sync with the rest of the draft. 

6.1 Diagnostic Logging

This document, in multiple sections, recommends that an implementation writes a diagnostic message to indicate unusual situations or other things noteworthy. Diagnostic messages are a very useful tool in finding configuration issues as well as a 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, e.g. in a circular buffer, the flood of diagnostic log entries would eventually overwrite useful previous diagnostics.

Besides this risk, diagnostic message, if they occur too frequently, can become meaningless to many administrators. Common practice is to turn off diagnostic logging if it turns out to be too verbose. This potentially removes the administrator of important diagnostic information.

While this document recommends to write meaningful diagnostic logs,

the author also recommends to allow an administrator to limit the amount of diagnostic logging. At least, an implementation SHOULD differentiate between critical, informational and debugging diagnostic message. Critical messages should only be issued in real critical states, e.g. expected or happening malfunction of the application or parts of it. A strong indication of an ongoing attack can eventually also be considered critical. As a guideline, there should be very few critical message. Informational message should indicate all conditions 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 administrator configuring multiple syslog applications in a complex environment). An application may

Gerhards Expires January 13, 2005 [Page 28]

Internet-Draft The syslog Protocol July 2004

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 diagnostics should not be written but discarded. An implementor may create as many different levels of diagnostic messages as he see useful - the above recommendation is just based on real-world experience of what is considered useful. Please not that experience also 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 this section. To solve this, it is recommended that an implementation allows to set a limit of how many "same" diagnostic messages will be generated within a limited amount of time. E.g. an administrator should be able to say that only the first 50 identical messages are logged within a 30 minute interval.

All subsequent identical messages will be discarded until the next time interval. While this causes some information loss, it is considered a good compromise between avoiding overruns and providing most in-depth diagnostic information. An implementation offering this feature should allow the administrator to configure the number of identical messages as well as the time interval to whatever the administrator thinks to be reasonable for his needs. It is up to the implementor of what the term "identical" means. Some may decide that only totally identical (in byte-to-byte comparison) messages are actually identical, some other may say that a message which is of identical type but with just some changed parameter (e.g. changed remote host address) is also considered identical. Both approaches have their advantages and disadvantages. Probably, it is best to leave this, too, configurable and allow the administrator to set the mode.

This document does NOT require nor enforce the outlined diagnostic message categorization or the duplicate suppression feature. It just would like to show some real-world solutions, which may be helpful for implementors. A system MAY claim to be compliant to this document even if it does not implement anything of the above.

6.2 Control Characters

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

Gerhards Expires January 13, 2005 [Page 29]

Internet-Draft The syslog Protocol July 2004

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 also often carried over to programs and script languages written in those languages.

As such, NUL characters must be taken with great care and be properly handled. An attacker may deliberately include NUL characters to hide information after the NUL characters. Wrong 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. This means that NUL characters should never be written to a file in an unencoded way - otherwise it would potentially render the file unreadable.

The same is true for other control characters. For example, deliberately included backspace characters may be used by an attacker to render parts of the log message unreadable. Similiar approaches exist for almost all control characters.

Finally, invalid UTF-8 sequences may be used to inject ASCII control characters. This is why invalid UTF-8 sequences are not allowed and **MUST** be rejected.

6.3 Packet Parameters

The message length **MUST NOT** exceed the maximum value outlined in Section 4. Various problems may result if a sender sends out messages with a greater length. While this document forbids oversize messages, an attacker may deliberately introduce them. As such, it is vital that each receiver performs the necessary sanity checks.

6.4 Single Source to a Destination

The syslog messages are usually presented (placed in a file, displayed on the console, etc.) in the order in which they are received. This is not always in accordance with the sequence in which they were generated. As they are transmitted across an IP network, some out of order receipt should be expected. This may lead to some confusion a messages may be received that would indicate that a process has stopped before it was started. This is somewhat rectified as the sender had timestamped each of the messages before transmission. Please note, however, that the **TIMESTAMP** is not always accurate.

It is desirable to use a transport with guaranteed delivery, if such one is available.

6.5 Multiple Sources to a Destination

In syslog, there is no concept of unified event numbering. Single senders are free to include a sequence number within the MSG but that can hardly be coordinated between multiple senders. In such cases, multiple senders may report that each one is sending message number one. Again, this may be rectified somewhat by the `TIMESTAMP`. As has been noted, however, even messages from a single sender to a single collector may be received out of order. This situation is compounded when there are several senders configured to send their syslog messages to a single collector. Messages from one sender may be delayed so the collector receives messages from another sender first even though the messages from the first sender were generated before the messages from the second. If there is no sufficiently-precise timestamp or coordinated sequence number, then the messages may be presented in the order in which they were received which may give an inaccurate view of the sequence of actual events.

6.6 Multiple Sources to Multiple Destinations

The plethora of configuration options available to the network administrators may further skew the perception of the order of events. It is possible to configure a group of senders to send the status messages -or other informative messages- to one collector, while sending messages of relatively higher importance to another collector. Additionally, the messages may be sent to different files on the same collector. If the messages do not contain sufficiently-precise timestamps from the source, it may be difficult to order the messages if they are kept in different places. An administrator may not be able to determine if a record in one file occurred before or after a record in a different file. This may be somewhat alleviated by placing marking messages with a timestamp into all destination files. If these have coordinated timestamps, then there will be some indication of the time of receipt of the

individual messages. As such, it is highly recommended to use the best available precision in the `TIMESTAMP` and use automatic time synchronization on each systems (as, for example, can be done via `NTP`).

6.7 Replaying

This needs also to be addressed in each transport mapping. Here is the general information on the issue, the transport mapping should address the specifics for the transport in question.

Gerhards Expires January 13, 2005 [Page 31]

Internet-Draft The syslog Protocol July 2004

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.

6.8 Reliable Delivery

As there is no mechanism within either the syslog process or the protocol to ensure delivery, and since the underlying transport may be lossy (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 the drop of 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.

RFC 3195 may be used for the reliable delivery of all syslog messages.

6.9 Message Integrity

Besides being discarded, syslog messages may be damaged in transit, or an attacker may maliciously modify them. In the case of a packet containing a syslog message being damaged, there are various mechanisms built into the link layer as well as into the IP [9] and UDP protocols which may detect the damage. An intermediary router may discard a damaged IP packet [10]. Damage to a UDP packet may be detected by the receiving UDP module, which may silently discard it. Depending on the transport used, additional or other mechanisms may be available. In any case, 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.

6.10 Large Size Messages

Messages larger than the minimum size required to be supported by the transport may be discarded or truncated by either a relay or the

Gerhards Expires January 13, 2005 [Page 32]

Internet-Draft The syslog Protocol July 2004

receiver. This can also lead to message loss, as already indicated above.

In order to avoid this, it is recommended that no sender does generate messages larger than the minimum size, at least not by default.

6.11 Message Observation

While there are no strict guidelines pertaining to the event message 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 of 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.

6.12 Misconfiguration

Since 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 recipient. Cases have been noted where senders were inadvertently configured to send syslog messages to the wrong receiver. In many cases, the inadvertent receiver 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.

6.13 Forwarding Loop

As it is shown in Figure 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 Priority 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 to not cause such a death spiral.

6.14 Load Considerations

Network administrators must take the time to estimate the appropriate size 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, it is recommended to use transports with guaranteed delivery.

6.15 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 receiving syslog messages from known IP addresses.

6.16 Covert Channels

Nothing in this protocol attempts to eliminate covert channels. Indeed, the unformatted message syntax in the packets could be very amenable to sending embedded secret messages. In fact, just about every aspect of syslog messages lends itself to the conveyance of covert signals. For example, a collusionist could send odd and even PRI values to indicate Morse Code dashes and dots.

Gerhards Expires January 13, 2005 [Page 34]

Internet-Draft The syslog Protocol July 2004

7. Notice to RFC Editor

This is a notice 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, these references will be updated to use the RFC numbers. This section will be removed at that time.

Gerhards Expires January 13, 2005 [Page 35]

Internet-Draft The syslog Protocol July 2004

8. IANA Considerations

8.1 Version

IANA MUST maintain a registry of VERSION values.

To register a VERSION number, a specification is required. The values and their meaning must be documented in an RFC or other permanent and readily available reference, in sufficient detail so that interoperability between independent implementations is possible. IANA MUST NOT register version "0" or any negative version number.

For this document, IANA must register the version "1".

8.2 SD-IDs

IANA MUST maintain a registry of SD-ID-IANA values. These are the SD-IDs which do NOT have a hyphen ("-") in the second character position.

To register one of these SD-IDs, a specification is required. The values and their meaning must be documented in an RFC or other permanent and readily available reference, in sufficient detail so that interoperability between independent implementations is possible.

For this document, IANA must register the SD-IDs "time" and "origin".

8.3 Facility and Severities

This document also upholds the Facilities and Severities listed in RFC 3164 [11]. Those values range from 0 to 191. This document also instructs the IANA to reserve all other possible values of the Severities and Facilities above the value of 191 and to distribute them via the consensus process as defined in RFC 2434 [10].

Gerhards Expires January 13, 2005 [Page 36]

Internet-Draft The syslog Protocol July 2004

9. 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@hq.adiscon.com

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

Adiscon GmbH
Mozartstrasse 21
97950 Grossrinderfeld
Germany

Gerhards Expires January 13, 2005 [Page 37]

Internet-Draft The syslog Protocol July 2004

10. Acknowledgements

The authors wish to thank Chris Lonvick, Jon Callas, Andrew Ross, Albert Mietus, Anton Okmianski, Tina Bird, David Harrington and all other people who commented on various versions of this proposal.

11 References

- [1] Okmianski, A., "Transmission of syslog messages over UDP", ANSI X3.4, May 2004.
- [2] American National Standards Institute, "USA Code for Information Interchange", ANSI X3.4, 1968.
- [3] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [4] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [5] Malkin, G., "Internet Users' Glossary", RFC 1983, August 1996.
- [6] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [7] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.
- [8] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.
- [9] Hinden, R. and S. Deering, "IP Version 6 Addressing

Architecture", RFC 2373, July 1998.

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

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

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

Gerhards Expires January 13, 2005 [Page 38]

Internet-Draft The syslog Protocol July 2004

Author's Address

Rainer Gerhards
Adiscon GmbH
Mozartstrasse 21
Grossrinderfeld, BW 97950
Germany

EMail: rgerhards@hq.adiscon.com

Gerhards Expires January 13, 2005 [Page 39]

Internet-Draft The syslog Protocol July 2004

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 (2004). 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.

