Network Working Group
Request for Comments: 3195
Category: Standards Track

D. New
M. Rose
Dover Beach Consulting, Inc.

November 2001

Reliable Delivery for syslog

Status of this Memo

Copyright Notice

Abstract

The BSD Syslog Protocol describes a number of service options related to propagating event messages. This memo describes two mappings of the syslog protocol to TCP connections, both useful for reliable delivery of event messages. The first provides a trivial mapping maximizing backward compatibility. The second provides a more complete mapping. Both provide a degree of robustness and security in message delivery that is unavailable to the usual UDP-based syslog protocol, by providing encryption and authentication over a connection-oriented protocol.

Table of Contents

1. Introduction

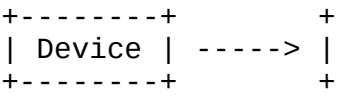
The syslog protocol [1] presents a spectrum of service options for provisioning an event-based logging service over a network. Each option has associated benefits and costs. Accordingly, the choice as to what combination of options is provisioned is both an engineering and administrative decision. This memo describes how to realize the syslog protocol when reliable delivery is selected as a required service. It is beyond the scope of this memo to argue for, or against, the use of reliable delivery for the syslog protocol.

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 [2].

2. The Model

The syslog service supports three roles of operation: device, relay, and collector.

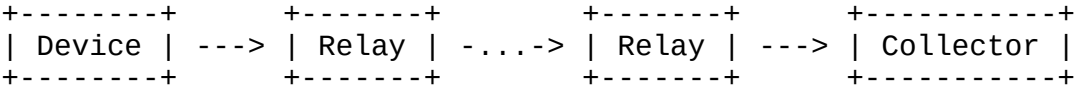
Devices and collectors act as sources and sinks, respectively, of syslog entries. In the simplest case, only a device and collector are present. E.g.,



The relationship between devices and collectors is potentially many-to-many. I.e., a device might communicate with many collectors; similarly, a collector might communicate with many devices.

A relay operates in both modes, accepting syslog entries from devices and other relays and forwarding those entries to collectors and other relays.

For example,



As shown, more than one relay may be present between any particular device and collector.

A relay may be necessary for administrative reasons. For example, a relay might run as an application proxy on a firewall. Also, there might be one relay per company department, which authenticates all the devices in the department, and which in turn authenticates itself to a company-wide collector.

A relay can also serve to filter messages. For example, one relay may collect the syslog information from an entire web server farm, summarizing hit counts for report generation, forwarding "page not found" messages (indicating a possible broken link) to a collector that presents it to the webmaster, and sending more urgent messages (such as hardware failure reports) to a collector that gateways them to a pager. A relay may also be used to convert formats from a device’s output to a collector’s input.

It should be noted that a role of device, relay, or collector is relevant only to a particular BEEP channel (q.v., below). A single server can serve as a device, a relay, and a collector, all at once,

if so configured. It can even serve as a relay and a collector to the same device at the same time using different BEEP channels over the same connection-oriented session; this might be useful to collect status yet relay urgent error messages.

To provide reliable delivery when realizing the syslog protocol, this memo defines two BEEP profiles. BEEP [3] is a generic application protocol framework for connection-oriented, asynchronous interactions. Within BEEP, features such as authentication, privacy, and reliability through retransmission are provided. There are two profiles defined in this memo:

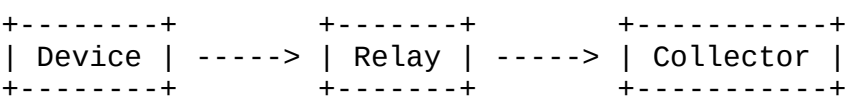
- The RAW profile is designed to provide a high-performance, low-impact footprint, using essentially the same format as the existing UDP-based syslog service.

- The COOKED profile is designed to provide a structured entry format, in which individual entries are acknowledged (either positively or negatively).

Note that both profiles run over BEEP. BEEP defines "transport mappings," specifying how BEEP messages are carried over the underlying transport technologies. At the time of this writing, only one such transport is defined, in [4], which specifies BEEP over TCP. All transport mappings are required to support enough reliability and sequencing to allow all BEEP messages on a given channel to be delivered reliably and in order. Hence, both the RAW and COOKED profile provide reliable delivery of their messages.

The choice of profile is independent of the operational roles discussed above.

For example, in



the device-to-relay link could be configured to use the RAW profile, while the relay-to-collector link could be configured to use the COOKED profile. (For example, the relay may be parsing the RAW syslog messages from the device, knowing the details of their formats, before passing them to a more generic collector.) Indeed, the same device may use different profiles, depending on the collector to which it is sending entries.

Devices and relays MAY discover relays and collectors via the DNS SRV algorithm [5]. If so configured, the service used is "syslog" and the protocol used is "tcp". This allows for central administration of addressing, fallback for failed relays and collectors, and static load balancing. Security policies and hardware configurations may be such that device configuration is more secure than the DNS server. Hardware devices may be of such limited resources that DNS SRV access is inappropriate. Firewalls and other restrictive routing mechanisms may need to be dealt with before a reliable syslog connection can be established. In these cases, DNS might not be the most appropriate configuration mechanism.

3. The RAW Profile

3.1 RAW Profile Overview

The RAW profile is designed for minimal implementation effort, high efficiency, and backwards compatibility. It is appropriate especially in cases where legacy syslog processing will be applied.

It should be noted that even though the RAW profile uses the same format for message payloads as the UDP version of syslog uses, delivery is reliable. The RAW syslog profile is a profile of BEEP [3], and BEEP guarantees ordered reliable delivery of messages within each individual channel.

When the profile is started, no piggyback data is supplied. All BEEP messages in the RAW profile are specified as having a MIME Content-Type [6] of application/octet-stream. Once the channel is open, the listener (not the initiator) sends a MSG message indicating it is ready to act as a syslog sink. (Refer to [3]'s Section 2.1 for a discussion of roles that a BEEP peer may perform, including definitions of the terms "listener", "initiator", "client", and "server".)

The initiator uses ANS replies to supply one or more syslog entries in the current UDP format, as specified in [1]'s Section 3. When the initiator has no more entries to send, it finishes with a NUL reply and closes the channel.

An example might appear as follows:

```
L: <wait for incoming connection>
I: <establish connection>
L: RPY 0 0 . 0 201
L: Content-type: application/beep+xml
L:
L: <greeting>
L:   <profile
L:     uri='http://xml.resource.org/profiles/syslog/COOKED' />
L:   <profile uri='http://xml.resource.org/profiles/syslog/RAW' />
L: </greeting>
L: END
I: RPY 0 0 . 0 52
I: Content-type: application/beep+xml
I:
I: <greeting />
I: END
I: MSG 0 1 . 52 133
I: Content-type: application/beep+xml
```

```
I:
I: <start number='1'>
I:   <profile uri='http://xml.resource.org/profiles/syslog/RAW' />
I: </start>
I: END
L: RPY 0 1 . 201 100
L: Content-type: application/beep+xml
L:
L: <profile uri='http://xml.resource.org/profiles/syslog/RAW' />
L: END
L: MSG 1 0 . 0 50
L:
L: Central Services. This has not been a recording.
L: END
I: ANS 1 0 . 0 61 0
I:
I: <29>Oct 27 13:21:08 ductwork imxpd[141]: Heating emergency.END
I: ANS 1 0 . 61 58 1
I:
I: <29>Oct 27 13:22:15 ductwork imxpd[141]: Contact Tuttle.END
I: NUL 1 0 . 119 0
I: END
L: MSG 0 3 . 301 70
L: Content-Type: application/beep+xml
L:
L: <close number='1' code='200' />
L: END
I: RPY 0 3 . 185 46
I: Content-Type: application/beep+xml
I:
I: <ok />
I: END
I: MSG 0 4 . 231 72
I: Content-Type: application/beep+xml
I:
I: <close number='0' code='200' />
I: END
L: RPY 0 4 . 371 46
L: Content-type: application/beep+xml
L:
L: <ok />
L: END
L: <closes connection>
I: <closes connection>
L: <awaits next connection>
```

Here we see a BEEP session established, followed by the use of the RAW profile. The initiator is a device, while the listener is a collector. The initiator opens the channel, but the listener sends the first MSG. This allows the initiator to send any number of ANS replies carrying syslog event messages. The initiator sends a NUL reply to indicate it is finished. Upon receiving the NUL, the listener closes the RAW channel. The initiator has the choice of closing the entire BEEP session or opening a new syslog channel (RAW or COOKED) for more transfers. In this example, the initiator chooses to close the entire BEEP session.

The overhead for one ANS frame is about thirty octets, once the initial handshakes have been exchanged. If this overhead is too high, then messages are likely being generated at a high rate. In this case, multiple syslog messages can be aggregated into a single ANS frame, each separated by a CRLF sequence from the preceding. The final message still MUST NOT end with a CRLF.

For example,

```
L: MSG 1 0 . 0 50
L:
L: Central Services. This has not been a recording.
L: END
I: ANS 1 0 . 0 119 0
I:
I: <29>Oct 27 13:21:08 ductwork imxpd[141]: Heating emergency.
I: <29>Oct 27 13:21:09 ductwork imxpd[141]: Contact Tuttle.END
I: NUL 1 0 . 119 0
I: END
```

3.2 RAW Profile Identification and Initialization

The RAW syslog profile is identified as

http://xml.resource.org/profiles/syslog/RAW

in the BEEP "profile" element during channel creation.

No data is piggybacked during channel creation.

3.3 RAW Profile Message Syntax

All BEEP messages in this profile have a MIME content-type of application/octet-stream. The listener's first BEEP message is ignored and indeed may be empty except for headers; hence, any syntax is acceptable.

The ANS replies the initiator sends in response MUST be formatted according to Section 4 of [1]. In particular, If the receiver is acting as a relay, then it MUST follow the rules as laid out in Section 4.2.2 of [1].

If multiple syslog messages are included in a single ANS reply, each is separated from the preceding with a CRLF. There is no ending delimiter, but each syslog event message body length MUST be 1024 bytes or less, excluding BEEP framing overhead. Note that there MUST NOT be a CRLF between the text of the final syslog event message and the "END" marking the trailer of the BEEP frame.

3.4 RAW Profile Message Semantics

The listener's opening BEEP MSG message has no semantics. (It is a good place to put in an identifying greeting.) The initiator's ANS replies MUST specify a facility, severity, and textual message, as described in [1].

4. The COOKED Profile

4.1 COOKED Profile Overview

The COOKED profile is designed for new implementations of syslog protocol handlers. It provides a much finer grain of information tagging, allowing a better degree of automation in processing. Naturally, it includes more overhead as well in support of this.

The COOKED profile supports three elements of interest:

- The "iam" element identifies the sender to the receiver, allowing each peer to name itself for the other, and specifying the roles (device, relay, or collector) each is taking on.

- The "entry" element provides a parsed version of the syslog entry, with the various fields of interest broken out.

- The "path" element identifies a list of relays through which a tagged collection of "entry" elements has passed, along with a set of flags indicating what assurances of security have been in effect throughout its delivery.

4.2 COOKED Profile Identification and Initialization

The COOKED syslog profile is identified as

    http://xml.resource.org/profiles/syslog/COOKED

in the BEEP "profile" element during channel creation.

During channel creation, the corresponding "profile" element in the BEEP "start" element may contain an "iam" element. If channel creation is successful, then before sending the corresponding reply, the BEEP peer processes the "iam" element and includes the resulting response in the reply. This response will be an "ok" element or an "error" element. The choice of which element is returned is dependent on local provisioning of the recipient. Including an "iam" in the initial "start" element has exactly the same semantics as passing it as the first MSG message on the channel.

4.3 COOKED Profile Message Syntax

All BEEP messages in this profile have a MIME Content-Type [6] of application/beep+xml. The syntax of the individual elements is specified in Section 7.

4.4 COOKED Profile Message Semantics

Initiators issue two elements: "iam" and "entry", each using a "MSG" message. The listener issues "ok" in "RPY" messages and "error" in "ERR" messages. (See [3]'s Section 2.3.1 for the definitions of the "error" and "ok" elements.)

4.4.1 The IAM Element

The "iam" element serves to identify a device, relay, or collector at one end of the BEEP channel to the device, relay, or collector at the other end of the channel. The "iam" element includes the type of peer (device, relay, or collector), the fully qualified domain name of the peer, and an IP address of the peer. (The IP address chosen SHOULD be the IP address associated with the underlying transport protocol carrying the channel.) The character data of the element is free-form human-readable text. It may be used to further identify the peer, such as by describing the physical location of the machine.

An "iam" element may be sent by the initiator of the channel at any time. The listener responds to an "iam" element with an "ok" (indicating acceptance), or an "error" (indicating rejection). The identity and role in effect is specified by the most recent "iam" answered with an "ok".

An "iam" could be rejected (with an "error" element) by the listener if the privacy or authentication that has been negotiated is inadequate or if the authenticated user does not have authorization to serve in the specified role. It is expected that most installations will require an "iam" from the peer before accepting any "entry" messages.

For example, a successful creation might look like this:

```
I: MSG 0 10 . 1832 259
I: Content-type: application/beep+xml
I:
I: <start number='1'>
I:   <profile
I:       uri='http://xml.resource.org/profiles/syslog/COOKED'>
I:     <![CDATA[ <iam fqdn='lowry.example.com' ip='10.0.0.27'
I:       type='device'/> ]]>
I:   </profile>
I: </start>
L: END
L: RPY 0 10 . 704 138
L: Content-type: application/beep+xml
L:
L: <profile uri='http://xml.resource.org/profiles/syslog/COOKED'>
L:   <![CDATA[ <ok /> ]]>
L: </profile>
L: END
```

A creation with an embedded "iam" that fails might look like this:

```
C: MSG 0 12 . 1832 259
C: Content-type: application/beep+xml
C:
C: <start number='1'>
C:   <profile
C:       uri='http://xml.resource.org/profiles/syslog/COOKED'>
C:     <![CDATA[ <iam fqdn='tuttle.example.com' ip='10.0.0.29'
C:       type='relay'/> ]]>
C:   </profile>
C: </start>
C: END
S: RPY 0 12 . 704 241
S: Content-type: application/beep+xml
S:
S: <profile uri='http://xml.resource.org/profiles/syslog/COOKED'>
S:   <![CDATA[
S:     <error code='535'>User 'buttle.example.com' not allowed
S:       to "iam" for 'tuttle.example.com'</error> ]]>
S: </profile>
S: END
```

In this case, the error code indicates that the user "buttle.example.com" has logged in via some SASL profile, but the syslog COOKED profile implementation is claiming to be "tuttle.example.com", a mismatch that the server is disallowing.

4.4.2 The ENTRY Element

The "entry" element carries the details of a single syslog entry. The attributes of an "entry" element include "facility", "severity", "timestamp", "hostname", and "tag". "Facility" and "severity" have the semantics defined in [1]'s 4.1. The other attributes have the semantics as in Sections 4.2.1 and 4.2.3 of [1]. An "entry" element can also contain a "pathID" attribute, described below.

If the client is a relay, the "entry" SHOULD also contain the attributes "deviceFQDN" and "deviceIP", specifying the FQDN and IP address of the device that originally created the entry. These attributes may be added by either the relay or the originating device. If possible, the device SHOULD add these entries, referring to the interface most closely associated with the syslog entry. Before a relay forwards an entry from a device that does not carry these attributes, it SHOULD add them based on the "iam" element it has received from the device, or based on the underlying transport connection address. A relay MUST NOT add these fields if they are missing and an "iam" element on the channel has indicated that messages are coming from another relay.

The "pathID" attribute indicates the path over which this entry has travelled, from device through relays to the final collector. Syntactically, its value is a string of digits that must match the "pathID" attribute of a "path" element sent earlier over the current channel. Semantically, it indicates that the list of relays and flags indicated in that earlier "path" element apply to this "entry" element.

The character data for the element is the unstructured syslog event message being logged. If the original device delivers the message for the first time via the COOKED profile, it may have any structure inside the CDATA. However, for maximum compatibility, the device SHOULD format the CDATA of the message in accordance with Sections 4.2.1 through 4.2.3 of [1].

In the message is being relayed, "tag" SHOULD be those of the original device generating the entry (unless the device cannot supply a tag). The "timestamp" SHOULD be that of the original entry generation time, rather than the time the entry was passed outward from the relay. The "hostname" SHOULD be the host name or IP address by which the device knows itself; this MUST follow the rules established in Sections 4.2.1 through 4.2.3 of [1]. The original contents of the syslog message MUST be preserved in the CDATA of the "entry" element; this includes preservation of exact content during translation from the UDP or RAW formats. In particular, the timestamps MUST NOT be rewritten in the CDATA of the "entry" element, the tag MUST NOT be removed from the CDATA even if presented in the "entry" attributes as well, and so on.

To be consistent with the spirit of [1], a relay receiving a message that does not contain a valid priority, timestamp or hostname will follow the same general rules as described in section 4.2.2 of [1] while including the exact contents of the received syslog packet as the CDATA. The values of the facility and severity will be construed to be 8 and 6 respectively and will be placed into the appropriate attributes of the "entry" element. The hostname will be the name of the device as it is known to the relay and will also be inserted into the "entry" element's attributes. The timestamp would be set to the received time, inserted only into the attributes of the "entry" element. As an example, consider this message received on UDP port 514 and interpreted as a traditional syslog message, assuming the underlying IP source address is that of the "pipeworks" machine:

```
<.....eeeek!
```

To be relayed, it must be modified as follows:

```
C: MSG 1 0 . 2079 156
C: Content-Type: application/beep+xml
C:
C: <entry facility='8' severity='6'
C:   hostname='pipeworks'
C:   timestamp='Oct 31 23:59:59'
C:  >&lt;.....eeeek!</entry>
C: END
S: RPY 1 0 . 933 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

As another example, consider a message being received that does not properly adhere to the conventions described in Section 4.2.2 of [1]. In particular, the timestamp has a year, making it a nonstandard format:

```
<166> 1990 Oct 22 01:00:00 bomb tick[0]: BOOM!
```

This would be relayed as follows:

```
C: MSG 1 0 . 2235 242
C: Content-Type: application/beep+xml
C:
C: <entry facility='160' severity='6'
C:   hostname='bomb'
C:   deviceFQDN='bomb.terrorist.net' deviceIP='10.0.0.83'
C:   timestamp='Oct 22 01:00:04'
C:  >&lt;166> 1990 Oct 22 01:00:00 bomb tick[0]: BOOM!</entry>
C: END
S: RPY 1 0 . 978 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

Note that the tag value was not readily apparent from the received message (due to the failed parsing of the timestamp), so it was not included in the "entry" element.

It is explicitly permitted for a relay to parse raw messages in a more sophisticated way, but all implementations MUST be able to parse messages presented in the format described in [1]. A more sophisticated relay could have recognized the year and completely parsed out the correct time, tag, and hostname, but such additional parsing capability is OPTIONAL.

Consider the following example, in contrast:

```
<166> Oct 22 01:00:00 bomb tick[0]: BOOM!
```

This conformant message would be relayed as follows:

```
C: MSG 1 0 . 2477 248
C: Content-Type: application/beep+xml
C:
C: <entry facility='160' severity='6'
C:   hostname='bomb'
C:   deviceFQDN='bomb.terrorist.net' deviceIP='10.0.0.83'
C:   timestamp='Oct 22 01:00:00' tag='tick'
C:  >&lt;166> Oct 22 01:00:00 bomb tick[0]: BOOM!</entry>
C: END
S: RPY 1 0 . 1023 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

In this case, the tag is detected and the timestamp represents the message generation time rather than the message reception time.

Finally, the "entry" element may also contain an "xml:lang" attribute, indicating the language in which the CDATA content of the tag is presented, as described in [7].

The "entry" element is answered with either an empty "ok" element if everything was successful, or a standard "error" element if there was a problem. An "entry" element can be rejected if no "iam" element has been accepted by the listener. It can also be rejected if the user authenticated on the BEEP session (if any) does not have the authority to generate (as a device) or relay that entry. An error is also possible if the "pathID" attribute refers to an unknown (or rejected) "path" element.

A successful exchange of an "entry" element may look like this:

```
C: MSG 1 0 . 2725 173
C: Content-Type: application/beep+xml
C:
C: <entry facility='24' severity='5'
C:   timestamp='Jan 26 15:16:17'
C:   hostname='pipework' tag='imxp'>
C:     No 27B/6 available</entry>
C: END
S: RPY 1 0 . 1068 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

Here, the device IP address and FQDN are taken from the "iam" element, if any, or from the underlying connection information.

An example where an "entry" element is rejected with an "error" element:

```
C: MSG 1 2 . 2898 223
C: Content-Type: application/beep+xml
C:
C: <entry facility='24' severity='5' timestamp='Jan 02 13:22:15'
C:   deviceFQDN='jack.example.net' deviceIP='10.0.0.83'
C:   tag='imxpd'>
C:     Replacement device found in nostril.
C: </entry>
C: END
S: ERR 1 2 . 1113 111
S: Content-Type: application/beep+xml
S:
S: <error code='554'>Not allowed to relay for
S:    jack.example.net</error>
S: END
```

Here, the client attempts to relay an entry on behalf of jack.example.com, but the entry is refused by the collector for administrative reasons. This may occur, for example, if lowry.example.com is in a different department than jack.example.com.

4.4.3 The PATH Element

The "path" element serves to describe a list of the relays through which that element has passed, along with a set of flags that indicate the properties that all links from the device to the relay have shared in common. Each "path" element contains either another "path" element or is empty. An empty "path" element identifies a device, while a "path" element with a nested "path" element identifies a relay. Each "path" element names a FQDN and IP address of the interface that sent the element. Each "path" element also names a FQDN and IP address for the interface that received the element. Each "path" element also carries a "linkprops" attribute, specifying the properties of the link it describes.

Each "path" element has a "pathID" attribute which must be unique for all "path" elements sent on this channel since its inception. Syntactically, the "pathID" attribute is a string of digits. Semantically, it serves to identify one "path" element out of many, and it serves to link a "path" element with one or more "entry" elements. Any "pathID" attribute is unrelated to any "pathID" attribute in nested "path" elements or on other channels.

Each "path" element has a "fromFQDN" attribute and an "fromIP" attribute. The "fromFQDN" attribute SHOULD be the fully qualified domain name of the interface over which the "path" element was sent. (The "fromFQDN" can be omitted if that interface has no DNS entry.) Similarly, the "fromIP" attribute MUST be the IP address of the interface over which the "path" element was sent.

Each "path" element has a "toFQDN" attribute and an "toIP" attribute. The "toFQDN" attribute SHOULD be the fully qualified domain name of the interface over which the "path" element was received. (The "toFQDN" can be omitted if that interface has no DNS entry.) Similarly, the "toIP" attribute MUST be the IP address of the interface over which the "path" element was received.

Finally, each "path" element carries a "linkprops" attribute. This is syntactically a string of individual characters, each indicating one property of the channel over which this "path" element is being carried. Note that outer "path" elements may have stronger guarantees than inner "path" elements; care should be taken in the interpretation of flags. The semantics of each possible character in this string are as follows:

o: When present, "o" (lower-case letter "o") indicates that weak privacy has been negotiated over this link, weakly protecting from observation the content of entries associated with this "path" element. (Weak privacy is encryption with less than 80 bits of key.)

O: When present, "O" (upper-case letter "O") indicates that strong privacy has been negotiated over this link, strongly protecting from observation the content of entries associated with this "path" element. (Strong privacy is encryption with 80 bits or more of key, or a transfer mechanism that is otherwise impossible to eavesdrop upon.)

U: When present, "U" indicates that a valid user has been authenticated (via SASL or TLS) and an "iam" element has been accepted.

A: When present, "A" indicates that this link has been protected by an authentication layer, authenticating the source of every "entry" associated with this path.

R: When present, "R" indicates that this link has been protected against message replay.

I: When present, "I" indicates that this link has been protected against modifications of messages in passing. ("I" stands for message Integrity.)

L: When present, "L" indicates that this link has been protected against loss of messages. That is, this is a reliable delivery link.

D: When present, "D" indicates that the "from" side of this link is a device. If this is not present on the innermost "path" element, "entry" elements associated with this path have not been carried by the COOKED profile for their entire lifetime.

Upon receiving a "path" element, the peer MUST perform the following checks:

- The "fromFQDN" and "fromIP" must match the underlying transport connection.
- The flags in the "linkprops" attribute must match the attributes of the session.
- The "toFQDN" and "toIP" must match the underlying transport connection.

- The "pathID" attribute must be unique with respect to all other "path" elements received on this channel.

If all these checks pass, the "path" element is accepted with an "ok" element. Otherwise, an "error" element is generated with an appropriate code. In addition, if any of the nested "path" elements refer to the machine receiving the element, it may indicate a routing loop in the configuration for the so-identified path, and appropriate measures should be taken.

If the peer receiving an "entry" element is receiving it directly from a device via either syslog-conn profile, and the device has not generated a "path" element, the receiver may itself generate an appropriate "path" element, either to be recorded in the logs (if this peer is a collector) or passed to the next peer (if this peer is a relay). If a peer receives a syslog message via UDP, it may optionally generate an appropriate "peer" element based on any cryptographic information provided in the message itself.

When a peer receives a "path" element, it remembers it for future use. A collector will store it in the log for later reference. A relay will remember it. When an "entry" arrives referencing the received "path" element, and that entry needs to be forwarded to another relay or collector, and no appropriate "path" element has already been generated, an appropriate "path" element is generated and sent over the outbound channel before the entry is forwarded. An appropriate "path" element is created by taking the received "path" element, wrapping it in a new "path" element with the appropriate attributes, and assigning it a new "pathID" attribute. When future "entry" elements arrive with the same incoming "pathID" attribute, and they need to be forwarded to a channel over which an appropriate "pathID" attribute has already been sent, only the "pathID" attribute of the "entry" element needs to be rewritten to refer to the "path" element on the outgoing channel.

It should be noted that the majority of the complexity in managing "path" elements arises only in relays. In particular, devices never need to generate "path" elements and collectors need only verify them, log them, and possibly use them in displays and reports. Collectors do not need to generate "path" elements or rewrite "entry" elements. Hence, only in complex configurations (where they are most useful) do complex "path" configurations occur.

For example, here is a path element sent from lowry.records.example.com to kurtzman.records.example.com. It indicates that entries from lowry to kurtzman tagged with pathID='173' originated from screen.lowry.records.example.com. It indicates that screen.lowry.records.example.com is believed by lowry.records.example.com to be the originating device, and that entries over this path are delivered without loss and without modification, although messages might be replayed or observed. The link between lowry and kurtzman, however, avoids replay attacks, lost messages, and modifications to messages. While screen.lowry.records.example.com has not authenticated itself to lowry.records.example.com, lowry claims to have authenticated itself to kurtzman.

```
C: MSG 2 1 . 3121 426
C: Content-type: application/beep+xml
C:
C: <path fromFQDN='lowry.records.example.com'
C:       fromIP='10.0.0.50'
C:       toFQDN='kurtzman.records.example.com'
C:       toIP='10.0.0.51'
C:       linkprops='ULRI'
C:       pathID='173'>
C: <path fromFQDN='screen.lowry.records.example.com'
C:       fromIP='10.0.0.47'
C:       toFQDN='lowry.records.example.com'
C:       toIP='10.0.0.50'
C:       linkprops='DLI'
C:       pathID='24'>
C: </path>
C: </path>
C: END
S: ERR 2 1 . 1224 114
S: Content-type: application/beep+xml
S:
S: <error code='530'>linkprops includes 'U'
S:   but no 'iam' received</error>
S: END
```

However, kurtzman.records.example.com rejects the "path" element, since the "linkprops" attribute claims that lowry has authenticated itself, but kurtzman disagrees, not having received an "iam" element.

In a second example, this "path" element informs collector.example.com that the records department's firewall will be forwarding "entry" elements with a "pathID" attribute whose value is "17". These "entry" elements will be coming in on the "10.0.0.2" interface of the firewall, to be forwarded out the "134.130.74.56" interface of the firewall. The final hop has all possible guarantees, although the entries transferred within the records department (behind the firewall) may have been observed in passing.

```
C: MSG 2 2 . 3547 813
C: Content-type: application/beep+xml
C:
C: <path fromFQDN='fwall.records.example.com'
C:       fromIP='134.130.74.56'
C:       toFQDN='collector.example.com'
C:       toIP='134.130.74.12'
C:       linkprops='OUARIL'
C:       pathID='17'>
C: <path fromFQDN='kurtzman.records.example.com'
C:       fromIP='10.0.0.50'
C:       toFQDN='fwall.records.example.com'
C:       toIP='10.0.0.2'
C:       linkprops='ULRI'
C:       pathID='120'>
C: <path fromFQDN='lowry.records.example.com'
C:       fromIP='10.0.0.50'
C:       toFQDN='kurtzman.records.example.com'
C:       toIP='10.0.0.51'
C:       linkprops='ULRI'
C:       pathID='173'>
C: <path fromFQDN='screen.lowry.records.example.com'
C:       fromIP='10.0.0.47'
C:       toFQDN='lowry.records.example.com'
C:       toIP='10.0.0.50'
C:       linkprops='DLI'
C:       pathID='24'>
C: </path></path></path></path>
C: END
S: RPY 2 2 . 1338 45
S: Content-type: application/beep+xml
S:
S: <ok/>
S: END
```

As a final example, an "entry" element from Lowry's screen arrives at the firewall. The "path" attribute is rewritten, and it is forwarded on to the collector.

The entry arrives on the 10.0.0.2 interface:

```
C: MSG 2 3 . 4360 250
C: Content-Type: application/beep+xml
C:
C: <entry facility='24' severity='5'
C:   timestamp='Oct 27 13:24:12'
C:   deviceFQDN='screen.lowry.records.example.com'
C:   deviceIP='10.0.0.47'
C:   pathID='173'
C:   tag='dvd'>
C:     Job paused - Boss watching.
C: </entry>
C: END
S: RPY 2 3 . 1383 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

It is forwarded out the 134.130.74.56 interface:

```
C: MSG 7 9 . 9375 276
C: Content-Type: application/beep+xml
C:
C: <entry facility='24' severity='5'
C:   timestamp='Oct 27 13:24:12'
C:   deviceFQDN='screen.lowry.records.example.com'
C:   deviceIP='10.0.0.47'
C:   pathID='17'
C:   tag='dvd'>
C:     Job paused - Boss watching.
C: </entry>
C: END
S: RPY 7 9 . 338 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

A discussion of the wisdom of configuring Lowry's machine to forward such messages via Kurtzman's machine is beyond the scope of this document.

5. Additional Provisioning

In more advanced configurations, syslog devices, relays, and collectors can be configured to support various delivery priorities. Multiple channels running the same profile can be opened between two peers, with higher priority syslog messages routed to a channel that is given more bandwidth. Such provisioning is a local matter.

syslog [1] discusses a number of reasons why privacy and authentication of syslog entry messages may be important in a networked computing environment. The nature of BEEP allows for convenient layering of authentication and privacy over any BEEP channel.

5.1 Message Authenticity

Section 6.2 of [1] discusses the dangers of unauthenticated syslog entries. To prevent inauthentic syslog event messages from being accepted, configure syslog peers to require the use of a strong authentication technology for the BEEP session.

If provisioned for message authentication, implementations SHOULD use SASL mechanism DIGEST-MD5 [8] to provision this service.

5.2 Message Replay

Section 6.3.4 of [1] discusses the dangers of syslog message replay. To prevent syslog event messages from being replayed, configure syslog peers to require the use of a strong authentication technology for the BEEP session.

If provisioned to detect message replay, implementations SHOULD use SASL mechanism DIGEST-MD5 [8] to provision this service.

5.3 Message Integrity

Section 6.5 of [1] discusses the dangers of syslog event messages being maliciously altered by an attacker. To prevent messages from being altered, configure syslog peers to require the use of a strong authentication technology for the BEEP session.

If provisioned to protect message integrity, implementations SHOULD use SASL mechanism DIGEST-MD5 [8] to provision this service.

5.4 Message Observation

Section 6.6 of [1] discusses the dangers (and benefits) of syslog messages being visible at intermediate points along the transmission path between device and collector. To prevent messages from being viewed by an attacker, configure syslog peers to require the use of a transport security profile for the BEEP session. (However, other traffic characteristics, e.g., volume and timing of transmissions, remain observable.)

If provisioned to secure messages against unauthorized observation, implementations SHOULD use the TLS profile [3] to provision this service. The cipher algorithm used SHOULD be TLS_RSA_WITH_3DES_EDE_CBC_SHA.

5.5 Summary of Recommended Practices

For the indicated protections, implementations SHOULD be configured to use the indicated mechanisms:

Desired Protection	SHOULD tune using
Authentication	http://iana.org/beep/SASL/DIGEST-MD5
+ Replay	http://iana.org/beep/SASL/DIGEST-MD5
+ Integrity	http://iana.org/beep/SASL/DIGEST-MD5
+ Observation	http://iana.org/beep/TLS

BEEP peer identities used for authentication SHOULD correspond to the FQDN of the initiating peer. That is, a relay running on relay.example.com should use a "user ID" of "relay.example.com" within the SASL authentication profiles, as well as in the FQDN of the "iam" element.

6. Initial Registrations

6.1 Registration: The RAW Profile

Profile Identification: http://xml.resource.org/profiles/syslog/RAW

Messages exchanged during Channel Creation: None

Messages starting one-to-one exchanges: Anything

Messages in positive replies: None

Messages in negative replies: None

Messages in one-to-many exchanges: Anything

Message Syntax: See Section 3.3

Message Semantics: See Section 3.4

Contact Information: See the "Authors' Addresses" section of this memo

6.2 Registration: The COOKED Profile

Profile Identification: http://xml.resource.org/profiles/syslog/COOKED

Messages exchanged during Channel Creation: iam

Messages starting one-to-one exchanges: iam, entry, path

Messages in positive replies: ok

Messages in negative replies: error

Messages in one-to-many exchanges: None

Message Syntax: See Section 4.3

Message Semantics: See Section 4.4

Contact Information: See the "Authors' Addresses" section of this memo

7. The syslog DTD

The following is the DTD defining the valid elements for the syslog over BEEP mapping.

```
<!--
  DTD for syslog over BEEP, as of 2000-10-10

  Refer to this DTD as:

    <!ENTITY % SYSLOG PUBLIC "-//Blocks//DTD SYSLOGRELIABLE//EN" "">
    %SYSLOG;
  -->

<!--
  Contents

    Overview

    Includes
    Profile Summaries
    Entity Definitions

    Operations
        iam
        entry
        path
  -->

<!--
  Overview

    Syslog packets delivered via BEEP

  -->

<!-- Includes -->

       <!ENTITY % BEEP PUBLIC "-//Blocks//DTD BEEP//EN"
                  "">
       %BEEP;
```

```
<!--
  Profile summaries

    BEEP profile SYSLOG-RAW

    role        MSG        ANS        ERR
    ====        ===        ===        ===
     L          text       text       text

    BEEP profile SYSLOG-COOKED

    role        MSG        RPY        ERR
    ====        ===        ===        ===
    I or L      iam        ok         error
    I or L      entry      ok         error
    I or L      path       ok         error

-->

<!--
  Entity Definitions

        entity        syntax/reference     example
        ======        ================     =======
    a fully qualified domain name
        FQDN          See [RFC-1034]       www.example.com

    a dotted-quad IP address
        IP            1*3DIGIT "." 1*3DIGIT "."
                       1*3DIGIT "." 1*3DIGIT
                                           10.0.0.27
    a syslog facility
        FACILITY      See [1]
                      1*3DIGIT             80

    a syslog severity
        SEVERITY      See [1]
                      DIGIT                 4

    a timestamp       See [1]               Jan 03 18:43:12
        TIMESTAMP

    an identifying integer
        IDINT         1*DIGIT               1027

-->
```

```
<!ENTITY % FQDN         "CDATA">
<!ENTITY % IP           "CDATA">
<!ENTITY % FACILITY     "CDATA">
<!ENTITY % SEVERITY     "CDATA">
<!ENTITY % TIMESTAMP    "CDATA">
<!ENTITY % IDINT        "CDATA">


<!--
  The iam element declares the role and identity of the peer
  issuing it. The contents of the element may include human-readable
  informative text, such as the physical location of the computer
  issuing the "iam".

  -->

<!ELEMENT iam         (#PCDATA)>
<!ATTLIST iam
          fqdn        %FQDN;                    #REQUIRED
          ip          %IP;                      #REQUIRED
          type        (device|relay|collector) #REQUIRED>

<!--
  The entry element conveys a single syslog message.
  -->

<!ELEMENT entry       (#PCDATA)>
<!ATTLIST entry
          xml:lang    %LANG;                    "i-default"
          facility    %FACILITY;                 #REQUIRED
          severity    %SEVERITY;                 #REQUIRED
          timestamp   %TIMESTAMP;                #IMPLIED
          tag         %ATEXT;                    #IMPLIED
          deviceFQDN  %FQDN;                     #IMPLIED
          deviceIP    %IP;                       #IMPLIED
          pathID      %IDINT;                    #IMPLIED>
```

```
<!--
  The path element conveys a list of relays through which
  entries have passed.
  -->

<!ELEMENT path        (path?)>
<!ATTLIST path
          pathID      %IDINT;                  #REQUIRED
          fromFQDN    %FQDN;                   #IMPLIED
          fromIP      %IP;                     #REQUIRED
          toFQDN      %FQDN;                   #IMPLIED
          toIP        %IP;                     #REQUIRED
          linkprops   %ATEXT;                  #REQUIRED>

<!-- End of DTD -->
```

8. Reply Codes

The following error codes are used in the protocol:

code	meaning
200	success
421	service not available
451	requested action aborted (e.g., local error in processing)
454	temporary authentication failure
500	general syntax error (e.g., poorly-formed XML)
501	syntax error in parameters (e.g., non-valid XML)
504	parameter not implemented
530	authentication required
534	authentication mechanism insufficient (e.g., too weak, sequence exhausted, etc.)
535	authentication failure
537	action not authorized for user
538	authentication mechanism requires encryption
550	requested action not taken (e.g., no requested profiles are acceptable)
553	parameter invalid
554	transaction failed (e.g., policy violation)

9. IANA Considerations

9.1 Registration: BEEP Profiles

The IANA registers the profiles specified in Section 6, and selects IANA-specific URIs "http://iana.org/beep/SYSLOG/RAW" and "http://iana.org/beep/SYSLOG/COOKED".

9.2 Registration: The System (Well-Known) TCP port number for syslog-conn

A single well-known port (601) is allocated to syslog-conn. In-band negotiation determines whether COOKED or RAW syslog-conn is in use.

Protocol Number: TCP

Message Formats, Types, Opcodes, and Sequences: See Section 3.3 and Section 4.4.

Functions: See Section 3.4 and Section 4.4.

Use of Broadcast/Multicast: none

Proposed Name: Reliable syslog service

Short name: syslog-conn

Contact Information: See the "Authors' Addresses" section of this memo

10. Security Considerations

Consult Section 6 of [1] for a discussion of security issues for the syslog service. In addition, since the RAW and COOKED profiles are defined using the BEEP framework, consult [3]'s Section 8 for a discussion of BEEP-specific security issues.

BEEP is used to provide communication security but not object integrity. In other words, the messages "on the wire" can be protected, but a compromised device may undetectably generate incorrect messages, and relays and collectors can modify, insert, or delete messages undetectably. Other techniques must be used to assure that such compromises are detectable.

11. Acknowledgements

The authors gratefully acknowledge the contributions of Christopher Calabrese, Keith McCloghrie, Balazs Scheidler, and David Waitzman.

12. References

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

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

[3] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

[4] Rose, M., "Mapping the BEEP Core onto TCP", RFC 3081, March 2001.

[5] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[6] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[7] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[8] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

Authors' Addresses

Darren New
5390 Caminito Exquisito
San Diego, CA 92130
US

Phone: +1 858 350 9733
EMail: dnew@san.rr.com

Marshall T. Rose
Dover Beach Consulting, Inc.
POB 255268
Sacramento, CA 95865-5268
US

Phone: +1 916 483 8878
EMail: mrose@dbc.mtview.ca.us

Full Copyright Statement

Acknowledgement

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