Network Working Group
Request For Comments: 3080
Category: Standards Track

M. Rose
Invisible Worlds, Inc.

March 2001

# The Blocks Extensible Exchange Protocol Core

## Status of this Memo



## Copyright Notice



## Abstract

This memo describes a generic application protocol kernel for connection-oriented, asynchronous interactions called the BEEP (Blocks Extensible Exchange Protocol) core. BEEP permits simultaneous and independent exchanges within the context of a single application user-identity, supporting both textual and binary messages.

Table of Contents

1. Introduction . . . . . 4
2. The BEEP Core . . . . . 5
2.1 Roles . . . . . 6
2.1.1 Exchange Styles . . . . . 6
2.2 Messages and Frames . . . . . 7
2.2.1 Frame Syntax . . . . . 8
2.2.1.1 Frame Header . . . . . 9
2.2.1.2 Frame Payload . . . . . 12
2.2.1.3 Frame Trailer . . . . . 13
2.2.2 Frame Semantics . . . . . 14
2.2.2.1 Poorly-formed Messages . . . . . 14
2.3 Channel Management . . . . . 15
2.3.1 Message Semantics . . . . . 16
2.3.1.1 The Greeting Message . . . . . 16
2.3.1.2 The Start Message . . . . . 17
2.3.1.3 The Close Message . . . . . 20
2.3.1.4 The OK Message . . . . . 23
2.3.1.5 The Error Message . . . . . 23
2.4 Session Establishment and Release . . . . . 25
2.5 Transport Mappings . . . . . 27
2.5.1 Session Management . . . . . 27
2.5.2 Message Exchange . . . . . 27
2.6 Asynchrony . . . . . 28
2.6.1 Within a Single Channel . . . . . 28
2.6.2 Between Different Channels . . . . . 28
2.6.3 Pre-emptive Replies . . . . . 29
2.6.4 Interference . . . . . 29
2.7 Peer-to-Peer Behavior . . . . . 30
3. Transport Security . . . . . 31
3.1 The TLS Transport Security Profile . . . . . 34
3.1.1 Profile Identification and Initialization . . . . . 34
3.1.2 Message Syntax . . . . . 35
3.1.3 Message Semantics . . . . . 36
3.1.3.1 The Ready Message . . . . . 36
3.1.3.2 The Proceed Message . . . . . 36
4. User Authentication . . . . . 37
4.1 The SASL Family of Profiles . . . . . 38
4.1.1 Profile Identification and Initialization . . . . . 39
4.1.2 Message Syntax . . . . . 42
4.1.3 Message Semantics . . . . . 43
5. Registration Templates . . . . . 44
5.1 Profile Registration Template . . . . . 44
5.2 Feature Registration Template . . . . . 44
6. Initial Registrations . . . . . 45
6.1 Registration: BEEP Channel Management . . . . . 45
6.2 Registration: TLS Transport Security Profile . . . . . 45

6.3 Registration: SASL Family of Profiles . . . . . 46
6.4 Registration: application/beep+xml . . . . . 47
7. DTDs . . . . . 48
7.1 BEEP Channel Management DTD . . . . . 48
7.2 TLS Transport Security Profile DTD . . . . . 50
7.3 SASL Family of Profiles DTD . . . . . 51
8. Reply Codes . . . . . 52
9. Security Considerations . . . . . 53
References . . . . . 54
Author's Address . . . . . 55
A. Acknowledgements . . . . . 56
B. IANA Considerations . . . . . 57
Full Copyright Statement . . . . . 58

## 1. Introduction

This memo describes a generic application protocol kernel for connection-oriented, asynchronous interactions called BEEP.

At BEEP's core is a framing mechanism that permits simultaneous and independent exchanges of messages between peers. Messages are arbitrary MIME [1] content, but are usually textual (structured using XML [2]).

All exchanges occur in the context of a channel -- a binding to a well-defined aspect of the application, such as transport security, user authentication, or data exchange.

Each channel has an associated "profile" that defines the syntax and semantics of the messages exchanged. Implicit in the operation of BEEP is the notion of channel management. In addition to defining BEEP's channel management profile, this document defines:

- the TLS [3] transport security profile; and,
- the SASL [4] family of profiles.

Other profiles, such as those used for data exchange, are defined by an application protocol designer.

## 2. The BEEP Core

A BEEP session is mapped onto an underlying transport service. A separate series of documents describe how a particular transport service realizes a BEEP session. For example, [5] describes how a BEEP session is mapped onto a single TCP [6] connection.

When a session is established, each BEEP peer advertises the profiles it supports. Later on, during the creation of a channel, the client supplies one or more proposed profiles for that channel. If the server creates the channel, it selects one of the profiles and sends it in a reply; otherwise, it may indicate that none of the profiles are acceptable, and decline creation of the channel.

Channel usage falls into one of two categories:

initial tuning: these are used by profiles that perform initialization once the BEEP session is established (e.g., negotiating the use of transport security); although several exchanges may be required to perform the initialization, these channels become inactive early in the BEEP session and remain so for the duration.

continuous: these are used by profiles that support data exchange; typically, these channels are created after the initial tuning channels have gone quiet.

Note that because of their special nature, only one tuning channel may be established at any given time; in contrast, BEEP allows multiple data exchange channels to be simultaneously in use.

### 2.1 Roles

Although BEEP is peer-to-peer, it is convenient to label each peer in the context of the role it is performing at a given time:

- When a BEEP session is established, the peer that awaits new connections is acting in the listening role, and the other peer, which establishes a connection to the listener, is acting in the initiating role. In the examples which follow, these are referred to as "L:" and "I:", respectively.

- A BEEP peer starting an exchange is termed the client; similarly, the other BEEP peer is termed the server. In the examples which follow, these are referred to as "C:" and "S:", respectively.

Typically, a BEEP peer acting in the server role is also acting in a listening role. However, because BEEP is peer-to-peer in nature, no such requirement exists.

#### 2.1.1 Exchange Styles

BEEP allows three styles of exchange:

MSG/RPY: the client sends a "MSG" message asking the server to perform some task, the server performs the task and replies with a "RPY" message (termed a positive reply).

MSG/ERR: the client sends a "MSG" message, the server does not perform any task and replies with an "ERR" message (termed a negative reply).

MSG/ANS: the client sends a "MSG" message, the server, during the course of performing some task, replies with zero or more "ANS" messages, and, upon completion of the task, sends a "NUL" message, which signifies the end of the reply.

The first two styles are termed one-to-one exchanges, whilst the third style is termed a one-to-many exchange.

## 2.2 Messages and Frames

A message is structured according to the rules of MIME. Accordingly, each message may begin with "entity-headers" (c.f., MIME's Section 3 [1]). If none, or only some, of the "entity-headers" are present:

- the default "Content-Type" is "application/octet-stream"; and,
- the default "Content-Transfer-Encoding" is "binary".

Normally, a message is sent in a single frame. However, it may be convenient or necessary to segment a message into multiple frames (e.g., if only part of a message is ready to be sent).

Each frame consists of a header, the payload, and a trailer. The header and trailer are each represented using printable ASCII characters and are terminated with a CRLF pair. Between the header and the trailer is the payload, consisting of zero or more octets.

For example, here is a message contained in a single frame that contains a payload of 120 octets spread over 5 lines (each line is terminated with a CRLF pair):

```
C: MSG 0 1 . 52 120
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/OTP' />
C: </start>
C: END
```

In this example, note that the entire message is represented in a single frame.

### 2.2.1 Frame Syntax

The ABNF [7] for a frame is:

```
frame      = data / mapping

data       = header payload trailer

header     = msg / rpy / err / ans / nul

msg        = "MSG" SP common          CR LF
rpy        = "RPY" SP common          CR LF
ans        = "ANS" SP common SP ansno CR LF
err        = "ERR" SP common          CR LF
nul        = "NUL" SP common          CR LF

common     = channel SP msgno SP more SP seqno SP size
channel    = 0..2147483647
msgno      = 0..2147483647
more       = "." / "*"
seqno      = 0..4294967295
size       = 0..2147483647
ansno      = 0..2147483647

payload    = *OCTET

trailer    = "END" CR LF

mapping    = ;; each transport mapping may define additional frames
```

#### 2.2.1.1 Frame Header

The frame header consists of a three-character keyword (one of: "MSG", "RPY", "ERR", "ANS", or "NUL"), followed by zero or more parameters. A single space character (decimal code 32, " ") separates each component. The header is terminated with a CRLF pair.

The channel number ("channel") must be a non-negative integer (in the range 0..2147483647).

The message number ("msgno") must be a non-negative integer (in the range 0..2147483647) and have a different value than all other "MSG" messages on the same channel for which a reply has not been completely received.

The continuation indicator ("more", one of: decimal code 42, "*", or decimal code 46, ".") specifies whether this is the final frame of the message:

intermediate ("*"): at least one other frame follows for the message; or,

complete ("."): this frame completes the message.

The sequence number ("seqno") must be a non-negative integer (in the range 0..4294967295) and specifies the sequence number of the first octet in the payload, for the associated channel (c.f., Section 2.2.1.2).

The payload size ("size") must be a non-negative integer (in the range 0..2147483647) and specifies the exact number of octets in the payload. (This does not include either the header or trailer.)

Note that a frame may have an empty payload, e.g.,

```
S: RPY 0 1 * 287 20
S:     ...
S:     ...
S: END
S: RPY 0 1 . 307 0
S: END
```

The answer number ("ansno") must be a non-negative integer (in the range 0..4294967295) and must have a different value than all other answers in progress for the message being replied to.

There are two kinds of frames: data and mapping. Each transport mapping (c.f., Section 2.5) may define its own frames. For example, [5] defines the SEQ frame. The remainder of this section discusses data frames.

When a message is segmented and sent as several frames, those frames must be sent sequentially, without any intervening frames from other messages on the same channel. However, there are two exceptions: first, no restriction is made with respect to the interleaving of frames for other channels; and, second, in a one-to-many exchange, multiple answers may be simultaneously in progress. Accordingly, frames for "ANS" messages may be interleaved on the same channel -- the answer number is used for collation, e.g.,

```
S: ANS 1 0 * 0 20 0
S:     ...
S:     ...
S: END
S: ANS 1 0 * 20 20 1
S:     ...
S:     ...
S: END
S: ANS 1 0 . 40 10 0
S:     ...
S: END
```

which shows two "ANS" messages interleaved on channel 1 as part of a reply to message number 0. Note that the sequence number is advanced for each frame sent on the channel, and is independent of the messages sent in those frames.

There are several rules for identifying poorly-formed frames:

- if the header doesn't start with "MSG", "RPY", "ERR", "ANS", or "NUL";
- if any of the parameters in the header cannot be determined or are invalid (i.e., syntactically incorrect);
- if the value of the channel number doesn't refer to an existing channel;
- if the header starts with "MSG", and the message number refers to a "MSG" message that has been completely received but for which a reply has not been completely sent;
- if the header doesn't start with "MSG", and refers to a message number for which a reply has already been completely received;
- if the header doesn't start with "MSG", and refers to a message number that has never been sent (except during session establishment, c.f., Section 2.3.1.1);
- if the header starts with "MSG", "RPY", "ERR", or "ANS", and refers to a message number for which at least one other frame has been received, and the three-character keyword starting this frame and the immediately-previous received frame for this message number are not identical;
- if the header starts with "NUL", and refers to a message number for which at least one other frame has been received, and the keyword of of the immediately-previous received frame for this reply isn't "ANS";
- if the continuation indicator of the previous frame received on the same channel was intermediate ("*"), and its message number isn't identical to this frame's message number;
- if the value of the sequence number doesn't correspond to the expected value for the associated channel (c.f., Section 2.2.1.2); or,
- if the header starts with "NUL", and the continuation indicator is intermediate ("*") or the payload size is non-zero.

If a frame is poorly-formed, then the session is terminated without generating a response, and it is recommended that a diagnostic entry be logged.

#### 2.2.1.2 Frame Payload

The frame payload consists of zero or more octets.

Every payload octet sent in each direction on a channel has an associated sequence number. Numbering of payload octets within a frame is such that the first payload octet is the lowest numbered, and the following payload octets are numbered consecutively. (When a channel is created, the sequence number associated with the first payload octet of the first frame is 0.)

The actual sequence number space is finite, though very large, ranging from 0..4294967295 (2**32 - 1). Since the space is finite, all arithmetic dealing with sequence numbers is performed modulo 2**32. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2**32 - 1 to 0 again. Consult Sections 2 through 5 of [8] for a discussion of the arithmetic properties of sequence numbers.

When receiving a frame, the sum of its sequence number and payload size, modulo 4294967296 (2**32), gives the expected sequence number associated with the first payload octet of the next frame received. Accordingly, when receiving a frame if the sequence number isn't the expected value for this channel, then the BEEP peers have lost synchronization, then the session is terminated without generating a response, and it is recommended that a diagnostic entry be logged.

#### 2.2.1.3 Frame Trailer

The frame trailer consists of "END" followed by a CRLF pair.

When receiving a frame, if the characters immediately following the payload don’t correspond to a trailer, then the session is terminated without generating a response, and it is recommended that a diagnostic entry be logged.

### 2.2.2 Frame Semantics

The semantics of each message is channel-specific. Accordingly, the profile associated with a channel must define:

- the initialization messages, if any, exchanged during channel creation;
- the messages that may be exchanged in the payload of the channel; and,
- the semantics of these messages.

A profile registration template (Section 5.1) organizes this information.

#### 2.2.2.1 Poorly-formed Messages

When defining the behavior of the profile, the template must specify how poorly-formed "MSG" messages are replied to. For example, the channel management profile sends a negative reply containing an error message (c.f., Section 2.3.1.5).

If a poorly-formed reply is received on channel zero, the session is terminated without generating a response, and it is recommended that a diagnostic entry be logged.

If a poorly-formed reply is received on another channel, then the channel must be closed using the procedure in Section 2.3.1.3.

## 2.3 Channel Management

When a BEEP session starts, only channel number zero is defined, which is used for channel management. Section 6.1 contains the profile registration for BEEP channel management.

Channel management allows each BEEP peer to advertise the profiles that it supports (c.f., Section 2.3.1.1), bind an instance of one of those profiles to a channel (c.f., Section 2.3.1.2), and then later close any channels or release the BEEP session (c.f., Section 2.3.1.3).

A BEEP peer should support at least 257 concurrent channels.

### 2.3.1 Message Semantics

#### 2.3.1.1 The Greeting Message

When a BEEP session is established, each BEEP peer signifies its availability by immediately sending a positive reply with a message number of zero that contains a "greeting" element, e.g.,

```
L: <wait for incoming connection>
I: <open connection>
L: RPY 0 0 . 0 110
L: Content-Type: application/beep+xml
L:
L: <greeting>
L:    <profile uri='http://iana.org/beep/TLS' />
L: </greeting>
L: END
I: RPY 0 0 . 0 52
I: Content-Type: application/beep+xml
I:
I: <greeting />
I: END
```

Note that this example implies that the BEEP peer in the initiating role waits until the BEEP peer in the listening role sends its greeting -- this is an artifact of the presentation; in fact, both BEEP peers send their replies independently.

The "greeting" element has two optional attributes ("features" and "localize") and zero or more "profile" elements, one for each profile supported by the BEEP peer acting in a server role:

- the "features" attribute, if present, contains one or more feature tokens, each indicating an optional feature of the channel management profile supported by the BEEP peer;

- the "localize" attribute, if present, contains one or more language tokens (defined in [9]), each identifying a desirable language tag to be used by the remote BEEP peer when generating textual diagnostics for the "close" and "error" elements (the tokens are ordered from most to least desirable); and,

- each "profile" element contained within the "greeting" element identifies a profile, and unlike the "profile" elements that occur within the "start" element, the content of each "profile" element may not contain an optional initialization message.

Section 5.2 defines a registration template for optional features.

#### 2.3.1.2 The Start Message

When a BEEP peer wants to create a channel, it sends a "start" element on channel zero, e.g.,

```
C: MSG 0 1 . 52 120
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/OTP' />
C: </start>
C: END
```

The "start" element has a "number" attribute, an optional "serverName" attribute, and one or more "profile" elements:

- the "number" attribute indicates the channel number (in the range 1..2147483647) used to identify the channel in future messages;
- the "serverName" attribute, an arbitrary string, indicates the desired server name for this BEEP session; and,
- each "profile" element contained with the "start" element has a "uri" attribute, an optional "encoding" attribute, and arbitrary character data as content:
  - the "uri" attribute authoritatively identifies the profile;
  - the "encoding" attribute, if present, specifies whether the content of the "profile" element is represented as a base64-encoded string; and,
  - the content of the "profile" element, if present, must be no longer than 4K octets in length and specifies an initialization message given to the channel as soon as it is created.

To avoid conflict in assigning channel numbers when requesting the creation of a channel, BEEP peers acting in the initiating role use only positive integers that are odd-numbered; similarly, BEEP peers acting in the listening role use only positive integers that are even-numbered.

The "serverName" attribute for the first successful "start" element received by a BEEP peer is meaningful for the duration of the BEEP session. If present, the BEEP peer decides whether to operate as the indicated "serverName"; if not, an "error" element is sent in a negative reply.

When a BEEP peer receives a "start" element on channel zero, it examines each of the proposed profiles, and decides whether to use one of them to create the channel. If so, the appropriate "profile" element is sent in a positive reply; otherwise, an "error" element is sent in a negative reply.

When creating the channel, the value of the "serverName" attribute from the first successful "start" element is consulted to provide configuration information, e.g., the desired server-side certificate when starting the TLS transport security profile (Section 3.1).

For example, a successful channel creation might look like this:

```
C: MSG 0 1 . 52 178
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/OTP' />
C:    <profile uri='http://iana.org/beep/SASL/ANONYMOUS' />
C: </start>
C: END
S: RPY 0 1 . 221 87
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/SASL/OTP' />
S: END
```

Similarly, an unsuccessful channel creation might look like this:

```
C: MSG 0 1 . 52 120
C: Content-Type: application/beep+xml
C:
C: <start number='2'>
C:    <profile uri='http://iana.org/beep/SASL/OTP' />
C: </start>
C: END
S: ERR 0 1 . 221 127
S: Content-Type: application/beep+xml
S:
S: <error code='501'>number attribute
S: in &lt;start&gt; element must be odd-valued</error>
S: END
```

Finally, here's an example in which an initialization element is exchanged during channel creation:

```
C: MSG 0 1 . 52 158
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/TLS'>
C:        <![CDATA[<ready />]]>
C:    </profile>
C: </start>
C: END
S: RPY 0 1 . 110 121
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/TLS'>
S:     <![CDATA[<proceed />]]>
S: </profile>
S: END
```

#### 2.3.1.3 The Close Message

When a BEEP peer wants to close a channel, it sends a "close" element on channel zero, e.g.,

```
C: MSG 0 2 . 235 71
C: Content-Type: application/beep+xml
C:
C: <close number='1' code='200' />
C: END
```

The "close" element has a "number" attribute, a "code" attribute, an optional "xml:lang" attribute, and an optional textual diagnostic as its content:

- the "number" attribute indicates the channel number;
- the "code" attribute is a three-digit reply code meaningful to programs (c.f., Section 8);
- the "xml:lang" attribute identifies the language that the element's content is written in (the value is suggested, but not mandated, by the "localize" attribute of the "greeting" element sent by the remote BEEP peer); and,
- the textual diagnostic (which may be multiline) is meaningful to implementers, perhaps administrators, and possibly even users, but never programs.

Note that if the textual diagnostic is present, then the "xml:lang" attribute is absent only if the language indicated as the remote BEEP peer's first choice is used.

If the value of the "number" attribute is zero, then the BEEP peer wants to release the BEEP session (c.f., Section 2.4) -- otherwise the value of the "number" attribute refers to an existing channel, and the remainder of this section applies.

A BEEP peer may send a "close" message for a channel whenever all "MSG" messages it has sent on that channel have been acknowledged. (Acknowledgement consists of the first frame of a reply being received by the BEEP peer that sent the MSG "message".)

After sending the "close" message, that BEEP peer must not send any more "MSG" messages on that channel being closed until the reply to the "close" message has been received (either by an "error" message rejecting the request to close the channel, or by an "ok" message subsequently followed by the channel being successfully started).

NOTE WELL: until a positive reply to the request to close the channel is received, the BEEP peer must be prepared to process any "MSG" messages that it receives on that channel.

When a BEEP peer receives a "close" message for a channel, it may, at any time, reject the request to close the channel by sending an "error" message in a negative reply.

Otherwise, before accepting the request to close the channel, and sending an "ok" message in a positive reply, it must:

- finish sending any queued "MSG" messages on that channel of its own;
- await complete replies to any outstanding "MSG" messages it has sent on that channel; and,
- finish sending complete replies to any outstanding "MSG" messages it has received on that channel, and ensure that the final frames of those replies have been successfully delivered, i.e.,
  - for transport mappings that guarantee inter-channel ordering of messages, the replies must be sent prior to sending the "ok" message in a positive reply; otherwise,
  - the replies must be sent and subsequently acknowledged by the underlying transport service prior to sending the "ok" message in a positive reply.

Briefly, a successful channel close might look like this:

```
C: MSG 0 2 . 235 71
C: Content-Type: application/beep+xml
C:
C: <close number='1' code='200' />
C: END
S: RPY 0 2 . 392 46
S: Content-Type: application/beep+xml
S:
S: <ok />
S: END
```

Similarly, an unsuccessful channel close might look like this:

```
C: MSG 0 2 . 235 71
C: Content-Type: application/beep+xml
C:
C: <close number='1' code='200' />
C: END
S: ERR 0 2 . 392 79
S: Content-Type: application/beep+xml
S:
S: <error code='550'>still working</error>
S: END
```

#### 2.3.1.4 The OK Message

When a BEEP peer agrees to close a channel (or release the BEEP session), it sends an "ok" element in a positive reply.

The "ok" element has no attributes and no content.

#### 2.3.1.5 The Error Message

When a BEEP peer declines the creation of a channel, it sends an "error" element in a negative reply, e.g.,

```
I: MSG 0 1 . 52 115
I: Content-Type: application/beep+xml
I:
I: <start number='2'>
I:    <profile uri='http://iana.org/beep/FOO' />
I: </start>
I: END
L: ERR 0 1 . 221 105
L: Content-Type: application/beep+xml
L:
L: <error code='550'>all requested profiles are
L: unsupported</error>
L: END
```

The "error" element has a "code" attribute, an optional "xml:lang" attribute, and an optional textual diagnostic as its content:

- the "code" attribute is a three-digit reply code meaningful to programs (c.f., Section 8);
- the "xml:lang" attribute identifies the language that the element's content is written in (the value is suggested, but not mandated, by the "localize" attribute of the "greeting" element sent by the remote BEEP peer); and,
- the textual diagnostic (which may be multiline) is meaningful to implementers, perhaps administrators, and possibly even users, but never programs.

Note that if the textual diagnostic is present, then the "xml:lang" attribute is absent only if the language indicated as the remote BEEP peer's first choice is used.

In addition, a BEEP peer sends an "error" element whenever:

- it receives a "MSG" message containing a poorly-formed or unexpected element;
- it receives a "MSG" message asking to close a channel (or release the BEEP session) and it declines to do so; or
- a BEEP session is established, the BEEP peer is acting in the listening role, and that BEEP peer is unavailable (in this case, the BEEP acting in the listening role does not send a "greeting" element).

In the final case, both BEEP peers terminate the session, and it is recommended that a diagnostic entry be logged by both BEEP peers.

## 2.4 Session Establishment and Release

When a BEEP session is established, each BEEP peer signifies its availability by immediately sending a positive reply with a message number of zero on channel zero that contains a "greeting" element, e.g.,

```
L: <wait for incoming connection>
I: <open connection>
L: RPY 0 0 . 0 110
L: Content-Type: application/beep+xml
L:
L: <greeting>
L:    <profile uri='http://iana.org/beep/TLS' />
L: </greeting>
L: END
I: RPY 0 0 . 0 52
I: Content-Type: application/beep+xml
I:
I: <greeting />
I: END
```

Alternatively, if the BEEP peer acting in the listening role is unavailable, it sends a negative reply, e.g.,

```
L: <wait for incoming connection>
I: <open connection>
L: ERR 0 0 . 0 60
L: Content-Type: application/beep+xml
L:
L: <error code='421' />
L: END
I: RPY 0 0 . 0 52
I: Content-Type: application/beep+xml
I:
I: <greeting />
I: END
I: <close connection>
L: <close connection>
L: <wait for next connection>
```

and the "greeting" element sent by the BEEP peer acting in the initiating role is ignored. It is recommended that a diagnostic entry be logged by both BEEP peers.

Note that both of these examples imply that the BEEP peer in the initiating role waits until the BEEP peer in the listening role sends its greeting -- this is an artifact of the presentation; in fact, both BEEP peers send their replies independently.

When a BEEP peer wants to release the BEEP session, it sends a "close" element with a zero-valued "number" attribute on channel zero. The other BEEP peer indicates its willingness by sending an "ok" element in a positive reply, e.g.,

```
C: MSG 0 1 . 52 60
C: Content-Type: application/beep+xml
C:
C: <close code='200' />
C: END
S: RPY 0 1 . 264 46
S: Content-Type: application/beep+xml
S:
S: <ok />
S: END
I: <close connection>
L: <close connection>
L: <wait for next connection>
```

Alternatively, if the other BEEP doesn't want to release the BEEP session, the exchange might look like this:

```
C: MSG 0 1 . 52 60
C: Content-Type: application/beep+xml
C:
C: <close code='200' />
C: END
S: ERR 0 1 . 264 79
S: Content-Type: application/beep+xml
S:
S: <error code='550'>still working</error>
S: END
```

If session release is declined, the BEEP session should not be terminated, if possible.

### 2.5 Transport Mappings

All transport interactions occur in the context of a session -- a mapping onto a particular transport service. Accordingly, this memo defines the requirements that must be satisfied by any document describing how a particular transport service realizes a BEEP session.

#### 2.5.1 Session Management

A BEEP session is connection-oriented. A mapping document must define:

- how a BEEP session is established;
- how a BEEP peer is identified as acting in the listening role;
- how a BEEP peer is identified as acting in the initiating role;
- how a BEEP session is released; and,
- how a BEEP session is terminated.

#### 2.5.2 Message Exchange

A BEEP session is message-oriented. A mapping document must define:

- how messages are reliably sent and received;
- how messages on the same channel are received in the same order as they were sent; and,
- how messages on different channels are sent without ordering constraint.

## 2.6 Asynchrony

BEEP accommodates asynchronous interactions, both within a single channel and between separate channels. This feature allows pipelining (intra-channel) and parallelism (inter-channel).

### 2.6.1 Within a Single Channel

A BEEP peer acting in the client role may send multiple "MSG" messages on the same channel without waiting to receive the corresponding replies. This provides pipelining within a single channel.

A BEEP peer acting in the server role must process all "MSG" messages for a given channel in the same order as they are received. As a consequence, the BEEP peer must generate replies in the same order as the corresponding "MSG" messages are received on a given channel.

Note that in one-to-many exchanges (c.f., Section 2.1.1), the reply to the "MSG" message consists of zero or more "ANS" messages followed by a "NUL" message. In this style of exchange, the "ANS" messages comprising the reply may be interleaved. When the BEEP peer acting in the server role signifies the end of the reply by generating the "NUL" message, it may then process the next "MSG" message received for that channel.

### 2.6.2 Between Different Channels

A BEEP peer acting in the client role may send multiple "MSG" messages on different channels without waiting to receive the corresponding replies. The channels operate independently, in parallel.

A BEEP peer acting in the server role may process "MSG" messages received on different channels in any order it chooses. As a consequence, although the replies for a given channel appear to be generated in the same order in which the corresponding "MSG" messages are received, there is no ordering constraint for replies on different channels.

### 2.6.3 Pre-emptive Replies

A BEEP peer acting in the server role may send a negative reply before it receives the final "MSG" frame of a message. If it does so, that BEEP peer is obliged to ignore any subsequent "MSG" frames for that message, up to and including the final "MSG" frame.

If a BEEP peer acting in the client role receives a negative reply before it sends the final "MSG" frame for a message, then it is required to send a "MSG" frame with a continuation status of complete (".") and having a zero-length payload.

### 2.6.4 Interference

If the processing of a particular message has sequencing impacts on other messages (either intra-channel or inter-channel), then the corresponding profile should define this behavior, e.g., a profile whose messages alter the underlying transport mapping.

## 2.7 Peer-to-Peer Behavior

BEEP is peer-to-peer -- as such both peers must be prepared to receive all messages defined in this memo. Accordingly, an initiating BEEP peer capable of acting only in the client role must behave gracefully if it receives a "MSG" message. Accordingly, all profiles must provide an appropriate error message for replying to unexpected "MSG" messages.

As a consequence of the peer-to-peer nature of BEEP, message numbers are unidirectionally-significant. That is, the message numbers in "MSG" messages sent by a BEEP peer acting in the initiating role are unrelated to the message numbers in "MSG" messages sent by a BEEP peer acting in the listening role.

For example, these two messages

```
I: MSG 0 1 . 52 120
I: Content-Type: application/beep+xml
I:
I: <start number='1'>
I:    <profile uri='http://iana.org/beep/SASL/OTP' />
I: </start>
I: END
L: MSG 0 1 . 221 116
L: Content-Type: application/beep+xml
L:
L: <start number='2'>
L:    <profile uri='http://iana.org/beep/APEX' />
L: </start>
L: END
```

refer to different messages sent on channel zero.

## 3. Transport Security

When a BEEP session is established, plaintext transfer, without privacy, is provided. Accordingly, transport security in BEEP is achieved using an initial tuning profile.

This document defines one profile:

o the TLS transport security profile, based on TLS version one [3].

Other profiles may be defined and deployed on a bilateral basis. Note that because of their intimate relationship with the transport service, a given transport security profile tends to be relevant to a single transport mapping (c.f., Section 2.5).

When a channel associated with transport security begins the underlying negotiation process, all channels (including channel zero) are closed on the BEEP session. Accordingly, upon completion of the negotiation process, regardless of its outcome, a new greeting is issued by both BEEP peers. (If the negotiation process fails, then either BEEP peer may instead terminate the session, and it is recommended that a diagnostic entry be logged.)

A BEEP peer may choose to issue different greetings based on whether privacy is in use, e.g.,

```
L: <wait for incoming connection>
I: <open connection>
L: RPY 0 0 . 0 110
L: Content-Type: application/beep+xml
L:
L: <greeting>
L:    <profile uri='http://iana.org/beep/TLS' />
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 . 52 158
I: Content-Type: application/beep+xml
I:
```

```
      I: <start number='1'>
      I:    <profile uri='http://iana.org/beep/TLS'>
      I:        <![CDATA[<ready />]]>
      I:    </profile>
      I: </start>
      I: END
      L: RPY 0 1 . 110 121
      L: Content-Type: application/beep+xml
      L:
      L: <profile uri='http://iana.org/beep/TLS'>
      L:     <![CDATA[<proceed />]]>
      L: </profile>
      L: END

          ... successful transport security negotiation ...

      L: RPY 0 0 . 0 221
      L: Content-Type: application/beep+xml
      L:
      L: <greeting>
      L:    <profile uri='http://iana.org/beep/SASL/ANONYMOUS' />
      L:    <profile uri='http://iana.org/beep/SASL/OTP' />
      L:    <profile uri='http://iana.org/beep/APEX' />
      L: </greeting>
      L: END
      I: RPY 0 0 . 0 52
      I: Content-Type: application/beep+xml
      I:
      I: <greeting />
      I: END
```

Of course, not all BEEP peers need be as single-minded:

```
      L: <wait for incoming connection>
      I: <open connection>
      L: RPY 0 0 . 0 268
      L: Content-Type: application/beep+xml
      L:
      L: <greeting>
      L:    <profile uri='http://iana.org/beep/SASL/ANONYMOUS' />
      L:    <profile uri='http://iana.org/beep/SASL/OTP' />
      L:    <profile uri='http://iana.org/beep/APEX' />
      L:    <profile uri='http://iana.org/beep/TLS' />
      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 . 52 158
I: Content-Type: application/beep+xml
I:
I: <start number='1'>
I:    <profile uri='http://iana.org/beep/TLS'>
I:        <![CDATA[<ready />]]>
I:    </profile>
I: </start>
I: END
L: RPY 0 1 . 268 121
L: Content-Type: application/beep+xml
L:
L: <profile uri='http://iana.org/beep/TLS'>
L:     <![CDATA[<proceed />]]>
L: </profile>
L: END

    ... failed transport security negotiation ...

L: RPY 0 0 . 0 268
L: Content-Type: application/beep+xml
L:
L: <greeting>
L:    <profile uri='http://iana.org/beep/SASL/ANONYMOUS' />
L:    <profile uri='http://iana.org/beep/SASL/OTP' />
L:    <profile uri='http://iana.org/beep/APEX' />
L:    <profile uri='http://iana.org/beep/TLS' />
L: </greeting>
L: END
I: RPY 0 0 . 0 52
I: Content-Type: application/beep+xml
I:
I: <greeting />
I: END
```

### 3.1 The TLS Transport Security Profile

Section 6.2 contains the registration for this profile.

#### 3.1.1 Profile Identification and Initialization

The TLS transport security profile is identified as:

```
http://iana.org/beep/TLS
```

in the BEEP "profile" element during channel creation.

During channel creation, the corresponding "profile" element in the BEEP "start" element may contain a "ready" element. If channel creation is successful, then before sending the corresponding reply, the BEEP peer processes the "ready" element and includes the resulting response in the reply, e.g.,

```
C: MSG 0 1 . 52 158
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/TLS'>
C:        <![CDATA[<ready />]]>
C:    </profile>
C: </start>
C: END
S: RPY 0 1 . 110 121
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/TLS'>
S:     <![CDATA[<proceed />]]>
S: </profile>
S: END
```

Note that it is possible for the channel to be created, but for the encapsulated operation to fail, e.g.,

```
C: MSG 0 1 . 52 173
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/TLS'>
C:        <![CDATA[<ready version="oops" />]]>
C:    </profile>
C: </start>
C: END
S: RPY 0 1 . 110 193
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/TLS'>
S:     <![CDATA[<error code='501'>version attribute
S: poorly formed in &lt;ready&gt; element</error>]]>
S: </profile>
S: END
```

In this case, a positive reply is sent (as channel creation succeeded), but the encapsulated response contains an indication as to why the operation failed.

### 3.1.2 Message Syntax

Section 7.2 defines the messages that are used in the TLS transport security profile.

### 3.1.3 Message Semantics

#### 3.1.3.1 The Ready Message

The "ready" element has an optional "version" attribute and no content:

- the "version" element defines the earliest version of TLS acceptable for use.

When a BEEP peer sends the "ready" element, it must not send any further traffic on the underlying transport service until a corresponding reply ("proceed" or "error") is received; similarly, the receiving BEEP peer must wait until any pending replies have been generated and sent before it processes a "ready" element.

#### 3.1.3.2 The Proceed Message

The "proceed" element has no attributes and no content. It is sent as a reply to the "ready" element.

When a BEEP peer receives the "ready" element, it must not send any further traffic on the underlying transport service until it generates a corresponding reply. If the BEEP peer decides to allow transport security negotiation, it implicitly closes all channels (including channel zero), and sends the "proceed" element, and awaits the underlying negotiation process for transport security.

When a BEEP peer receives a "proceed" element in reply to its "ready" message, it implicitly closes all channels (including channel zero), and immediately begins the underlying negotiation process for transport security.

## 4. User Authentication

When a BEEP session is established, anonymous access, without trace information, is provided. Accordingly, user authentication in BEEP is achieved using an initial tuning profile.

This document defines a family of profiles based on SASL mechanisms:

- each mechanism in the IANA SASL registry [15] has an associated profile.

Other profiles may be defined and deployed on a bilateral basis.

Whenever a successful authentication occurs, on any channel, the authenticated identity is updated for all existing and future channels on the BEEP session; further, no additional attempts at authentication are allowed.

Note that regardless of transport security and user authentication, authorization is an internal matter for each BEEP peer. As such, each peer may choose to restrict the operations it allows based on the authentication credentials provided (i.e., unauthorized operations might be rejected with error code 530).

### 4.1 The SASL Family of Profiles

Section 6.3 contains the registration for this profile.

Note that SASL may provide both user authentication and transport security. Once transport security is successfully negotiated for a BEEP session, then a SASL security layer must not be negotiated; similarly, once any SASL negotiation is successful, a transport security profile must not begin its underlying negotiation process.

Section 4 of the SASL specification [4] requires the following information be supplied by a protocol definition:

service name: "beep"

initiation sequence: Creating a channel using a BEEP profile corresponding to a SASL mechanism starts the exchange. An optional parameter corresponding to the "initial response" sent by the client is carried within a "blob" element during channel creation.

exchange sequence: "Challenges" and "responses" are carried in exchanges of the "blob" element. The "status" attribute of the "blob" element is used both by a server indicating a successful completion of the exchange, and a client aborting the exchange, The server indicates failure of the exchange by sending an "error" element.

security layer negotiation: When a security layer starts negotiation, all channels (including channel zero) are closed on the BEEP session. Accordingly, upon completion of the negotiation process, regardless of its outcome, a new greeting is issued by both BEEP peers.

If a security layer is successfully negotiated, it takes effect immediately following the message that concludes the server’s successful completion reply.

use of the authorization identity: This is made available to all channels for the duration of the BEEP session.

#### 4.1.1 Profile Identification and Initialization

Each SASL mechanism registered with the IANA is identified as:

```
http://iana.org/beep/SASL/mechanism
```

where "MECHANISM" is the token assigned to that mechanism by the IANA.

Note that during channel creation, a BEEP peer may provide multiple profiles to the remote peer, e.g.,

```
C: MSG 0 1 . 52 178
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/ANONYMOUS' />
C:    <profile uri='http://iana.org/beep/SASL/OTP' />
C: </start>
C: END
S: RPY 0 1 . 221 87
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/SASL/OTP' />
S: END
```

During channel creation, the corresponding "profile" element in the BEEP "start" element may contain a "blob" element. Note that it is possible for the channel to be created, but for the encapsulated operation to fail, e.g.,

```
C: MSG 0 1 . 52 183
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/OTP'>
C:        <![CDATA[<blob>AGJsb2NrbWFzdGVy</blob>]]>
C:    </profile>
C: </start>
C: END
S: RPY 0 1 . 221 178
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/SASL/OTP'>
S:     <![CDATA[<error code='534'>authentication mechanism is
S: too weak</error>]]>
S: </profile>
S: END
```

In this case, a positive reply is sent (as channel creation succeeded), but the encapsulated response contains an indication as to why the operation failed.

Otherwise, the server sends a challenge (or signifies success), e.g.,

```
C: MSG 0 1 . 52 183
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/OTP'>
C:        <![CDATA[<blob>AGJsb2NrbWFzdGVy</blob>]]>
C:    </profile>
C: </start>
C: END
S: RPY 0 1 . 221 171
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/SASL/OTP'>
S:    <![CDATA[<blob>b3RwLXNoYTEgOTk5NyBwaXh5bWlzYXM4NTgwNSBleHQ=
                                                    </blob>]]>
S: </profile>
S: END
```

Note that this example implies that the "blob" element in the server's reply appears on two lines -- this is an artifact of the presentation; in fact, only one line is used.

If a challenge is received, then the client responds and awaits another reply, e.g.,

```
C: MSG 1 0 . 0 97
C: Content-Type: application/beep+xml
C:
C: <blob>d29yZDpmZXJuIGhhbmcgYnJvdyBib25nIGhlcmQgdG9n</blob>
C: END
S: RPY 1 0 . 0 66
S: Content-Type: application/beep+xml
S:
S: <blob status='complete' />
S: END
```

Of course, the client could abort the authentication process by sending "<blob status='abort' />" instead.

Alternatively, the server might reject the response with an error: e.g.,

```
C: MSG 1 0 . 0 97
C: Content-Type: application/beep+xml
C:
C: <blob>d29yZDpmZXJuIGhhbmcgYnJvdyBib25nIGhlcmQgdG9n</blob>
C: END
S: ERR 1 0 . 0 60
S: Content-Type: application/beep+xml
S:
S: <error code='535' />
S: END
```

Finally, depending on the SASL mechanism, an initialization element may be exchanged unidirectionally during channel creation, e.g.,

```
C: MSG 0 1 . 52 125
C: Content-Type: application/beep+xml
C:
C: <start number='1'>
C:    <profile uri='http://iana.org/beep/SASL/CRAM-MD5' />
C: </start>
C: END
S: RPY 0 1 . 221 185
S: Content-Type: application/beep+xml
S:
S: <profile uri='http://iana.org/beep/SASL/CRAM-MD5'>
S: <![CDATA[<blob>PDE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2UucmVzdG9uLm1
                                                  jaS5uZXQ+</blob>]]>
S: </profile>
S: END
```

Note that this example implies that the "blob" element in the server's reply appears on two lines -- this is an artifact of the presentation; in fact, only one line is used.

### 4.1.2 Message Syntax

Section 7.3 defines the messages that are used for each profile in the SASL family.

Note that because many SASL mechanisms exchange binary data, the content of the "blob" element is always a base64-encoded string.

#### 4.1.3 Message Semantics

The "blob" element has an optional "status" attribute, and arbitrary octets as its content:

- the "status" attribute, if present, takes one of three values:

  abort: used by a client to indicate that it is aborting the authentication process;

  complete: used by a server to indicate that the exchange is complete and successful; or,

  continue: used by either a client or server, otherwise.

Finally, note that SASL's EXTERNAL mechanism works with an "external authentication" service, which is provided by one of:

- a transport security profile, capable of providing authentication information (e.g., Section 3.1), being active on the connection;
- a network service, capable of providing strong authentication (e.g., IPSec [12]), underlying the connection; or,
- a locally-defined security service.

For authentication to succeed, two conditions must hold:

- an external authentication service must be active; and,
- if present, the authentication identity must be consistent with the credentials provided by the external authentication service (if the authentication identity is empty, then an authorization identity is automatically derived from the credentials provided by the external authentication service).

# 5. Registration Templates

## 5.1 Profile Registration Template

When a profile is registered, the following information is supplied:

Profile Identification: specify a URI [10] that authoritatively identifies this profile.

Message Exchanged during Channel Creation: specify the datatypes that may be exchanged during channel creation.

Messages starting one-to-one exchanges: specify the datatypes that may be present when an exchange starts.

Messages in positive replies: specify the datatypes that may be present in a positive reply.

Messages in negative replies: specify the datatypes that may be present in a negative reply.

Messages in one-to-many exchanges: specify the datatypes that may be present in a one-to-many exchange.

Message Syntax: specify the syntax of the datatypes exchanged by the profile.

Message Semantics: specify the semantics of the datatypes exchanged by the profile.

Contact Information: specify the postal and electronic contact information for the author of the profile.

## 5.2 Feature Registration Template

When a feature for the channel management profile is registered, the following information is supplied:

Feature Identification: specify a string that identifies this feature. Unless the feature is registered with the IANA, the feature's identification must start with "x-".

Feature Semantics: specify the semantics of the feature.

Contact Information: specify the postal and electronic contact information for the author of the feature.

# 6. Initial Registrations

## 6.1 Registration: BEEP Channel Management

Profile Identification: not applicable

Messages exchanged during Channel Creation: not applicable

Messages starting one-to-one exchanges: "start" or "close"

Messages in positive replies: "greeting", "profile", or "ok"

Messages in negative replies: "error"

Messages in one-to-many exchanges: none

Message Syntax: c.f., Section 7.1

Message Semantics: c.f., Section 2.3.1

Contact Information: c.f., the "Author's Address" section of this memo

## 6.2 Registration: TLS Transport Security Profile

Profile Identification: http://iana.org/beep/TLS

Messages exchanged during Channel Creation: "ready"

Messages starting one-to-one exchanges: "ready"

Messages in positive replies: "proceed"

Messages in negative replies: "error"

Messages in one-to-many exchanges: none

Message Syntax: c.f., Section 7.2

Message Semantics: c.f., Section 3.1.3

Contact Information: c.f., the "Author's Address" section of this memo

### 6.3 Registration: SASL Family of Profiles

Profile Identification: http://iana.org/beep/SASL/mechanism, where "mechanism" is a token registered with the IANA

Messages exchanged during Channel Creation: "blob"

Messages starting one-to-one exchanges: "blob"

Messages in positive replies: "blob"

Messages in negative replies: "error"

Messages in one-to-many exchanges: none

Message Syntax: c.f., Section 7.3

Message Semantics: c.f., Section 4.1.3

Contact Information: c.f., the "Author's Address" section of this memo

### 6.4 Registration: application/beep+xml

MIME media type name: application

MIME subtype name: beep+xml

Required parameters: none

Optional parameters: charset (defaults to "UTF-8" [13])

Encoding considerations: This media type may contain binary content; accordingly, when used over a transport that does not permit binary transfer, an appropriate encoding must be applied

Security considerations: none, per se; however, any BEEP profile which uses this media type must describe its relevant security considerations

Interoperability considerations: n/a

Published specification: This media type is a proper subset of the the XML 1.0 specification [2]. Two restrictions are made.

First, no entity references other than the five predefined general entities references ("&", "<", ">", "'", and """) and numeric entity references may be present.

Second, neither the "XML" declaration (e.g., <?xml version="1.0" ?>) nor the "DOCTYPE" declaration (e.g., <!DOCTYPE ...>) may be present. (Accordingly, if another character set other than UTF-8 is desired, then the "charset" parameter must be present.)

All other XML 1.0 instructions (e.g., CDATA blocks, processing instructions, and so on) are allowed.

Applications which use this media type: any BEEP profile wishing to make use of this XML 1.0 subset

Additional Information: none

Contact for further information: c.f., the "Author's Address" section of this memo

Intended usage: limited use

Author/Change controller: the IESG

## 7. DTDs

### 7.1 BEEP Channel Management DTD

```
   <!--
     DTD for BEEP Channel Management, as of 2000-10-29


     Refer to this DTD as:

       <!ENTITY % BEEP PUBLIC "-//IETF//DTD BEEP//EN"
                  "http://xml.resource.org/profiles/BEEP/beep.dtd">
       %BEEP;
     -->


   <!--
     DTD data types:

           entity        syntax/reference     example
           ======        ================     =======
       a channel number
           CHAN          1..2147483647        1

       authoritative profile identification
           URI           c.f., [RFC-2396]     http://invisible.net/

       one or more feature tokens, separated by space
           FTRS          NMTOKENS             "magic"

       a language tag
           LANG          c.f., [RFC-1766]     "en", "en-US", etc.

       zero or more language tags
           LOCS          NMTOKENS             "en-US"

       a 3-digit reply code
           XYZ            [1-5][0-9][0-9]     500
   -->


   <!ENTITY % CHAN       "CDATA">
   <!ENTITY % URI        "CDATA">
   <!ENTITY % FTRS       "NMTOKENS">
   <!ENTITY % LANG       "NMTOKEN">
   <!ENTITY % LOCS       "NMTOKEN">
   <!ENTITY % XYZ        "CDATA">
```

```
<!--
  BEEP messages, exchanged as application/beep+xml

     role       MSG         RPY         ERR
    =======     ===         ===         ===
    I and L                 greeting    error

    I or L      start       profile     error

    I or L      close       ok          error
  -->


<!ELEMENT greeting    (profile)*>
<!ATTLIST greeting
          features    %FTRS;            #IMPLIED
          localize    %LOCS;            "i-default">

<!ELEMENT start       (profile)+>
<!ATTLIST start
          number      %CHAN;             #REQUIRED
          serverName  CDATA              #IMPLIED>

<!-- profile element is empty if contained in a greeting -->
<!ELEMENT profile     (#PCDATA)>
<!ATTLIST profile
          uri         %URI;              #REQUIRED
          encoding    (none|base64)      "none">

<!ELEMENT close       (#PCDATA)>
<!ATTLIST close
          number      %CHAN;             "0"
          code        %XYZ;              #REQUIRED
          xml:lang    %LANG;             #IMPLIED>

<!ELEMENT ok          EMPTY>

<!ELEMENT error       (#PCDATA)>
<!ATTLIST error
          code        %XYZ;              #REQUIRED
          xml:lang    %LANG;             #IMPLIED>
```

## 7.2 TLS Transport Security Profile DTD

```
<!--
  DTD for the TLS Transport Security Profile, as of 2000-09-04


  Refer to this DTD as:

    <!ENTITY % TLS PUBLIC "-//IETF//DTD TLS//EN"
              "http://xml.resource.org/profiles/TLS/tls.dtd">
    %TLS;
  -->


<!--
  TLS messages, exchanged as application/beep+xml

     role       MSG         RPY         ERR
    ======      ===         ===         ===
    I or L      ready       proceed     error
  -->


<!ELEMENT ready       EMPTY>
<!ATTLIST ready
          version     CDATA              "1">

<!ELEMENT proceed     EMPTY>
```

## 7.3 SASL Family of Profiles DTD

```
<!--
  DTD for the SASL Family of Profiles, as of 2000-09-04


  Refer to this DTD as:

    <!ENTITY % SASL PUBLIC "-//IETF//DTD SASL//EN"
               "http://xml.resource.org/profiles/sasl/sasl.dtd">
    %SASL;
  -->


<!--
  SASL messages, exchanged as application/beep+xml

     role       MSG         RPY         ERR
    ======      ===         ===         ===
    I or L      blob        blob        error
  -->


<!ELEMENT blob        (#PCDATA)>
<!ATTLIST blob
          xml:space   (default|preserve)
                                        "preserve"
          status      (abort|complete|continue)
                                         "continue">
```

## 8. Reply Codes

| code | meaning |
|---|---|
| 200 | success |
| 421 | service not available |
| 450 | requested action not taken<br>(e.g., lock already in use) |
| 451 | requested action aborted<br>(e.g., local error in processing) |
| 454 | temporary authentication failure |
| 500 | general syntax error<br>(e.g., poorly-formed XML) |
| 501 | syntax error in parameters<br>(e.g., non-valid XML) |
| 504 | parameter not implemented |
| 530 | authentication required |
| 534 | authentication mechanism insufficient<br>(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<br>(e.g., no requested profiles are acceptable) |
| 553 | parameter invalid |
| 554 | transaction failed<br>(e.g., policy violation) |

## 9. Security Considerations

The BEEP framing mechanism, per se, provides no protection against attack; however, judicious use of initial tuning profiles provides varying degrees of assurance:

1. If one of the profiles from the SASL family is used, refer to [4]'s Section 9 for a discussion of security considerations.

2. If the TLS transport security profile is used (or if a SASL security layer is negotiated), then:

    1. A man-in-the-middle may remove the security-related profiles from the BEEP greeting or generate a negative reply to the "ready" element of the TLS transport security profile. A BEEP peer may be configurable to refuse to proceed without an acceptable level of privacy.

    2. A man-in-the-middle may cause a down-negotiation to the weakest cipher suite available. A BEEP peer should be configurable to refuse weak cipher suites.

    3. A man-in-the-middle may modify any protocol exchanges prior to a successful negotiation. Upon completing the negotiation, a BEEP peer must discard previously cached information about the BEEP session.

    As different TLS ciphersuites provide varying levels of security, administrators should carefully choose which ciphersuites are provisioned.

As BEEP is peer-to-peer in nature, before performing any task associated with a message, each channel should apply the appropriate access control based on the authenticated identity and privacy level associated with the BEEP session.

References

[1] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[2] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0", W3C XML, February 1998, <http://www.w3.org/TR/1998/REC-xml-19980210>.

[3] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[4] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

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

[6] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[7] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[8] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

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

[10] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[11] Newman, C., "The One-Time-Password SASL Mechanism", RFC 2444, October 1998.

[12] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[13] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[14] Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997.

[15] <http://www.isi.edu/in-notes/iana/assignments/sasl-mechanisms>

Author's Address

Marshall T. Rose
Invisible Worlds, Inc.
1179 North McDowell Boulevard
Petaluma, CA 94954-6559
US

Phone: +1 707 789 3700
EMail: mrose@invisible.net
URI: http://invisible.net/

## Appendix A. Acknowledgements

The author gratefully acknowledges the contributions of: David Clark, Dave Crocker, Steve Deering, Wesley Michael Eddy, Huston Franklin, Marco Gazzetta, Danny Goodman, Steve Harris, Robert Herriot, Ken Hirsch, Greg Hudson, Ben Laurie, Carl Malamud, Michael Mealling, Keith McCloghrie, Paul Mockapetris, RL 'Bob' Morgan, Frank Morton, Darren New, Chris Newman, Joe Touch, Paul Vixie, Gabe Wachob, Daniel Woods, and, James Woodyatt. In particular, Dave Crocker provided helpful suggestions on the nature of segmentation in the framing mechanism.

## Appendix B. IANA Considerations

The IANA registers "beep" as a GSSAPI [14] service name, as specified in Section 4.1.

The IANA maintains a list of:

- standards-track BEEP profiles, c.f., Section 5.1; and,
- standards-track features for the channel management profile, c.f., Section 5.2.

For each list, the IESG is responsible for assigning a designated expert to review the specification prior to the IANA making the assignment. As a courtesy to developers of non-standards track BEEP profiles and channel management features, the mailing list bxxpwg@invisible.net may be used to solicit commentary.

The IANA makes the registrations specified in Section 6.2 and Section 6.3. It is recommended that the IANA register these profiles using the IANA as a URI-prefix, and populate those URIs with the respective profile registrations.

## Full Copyright Statement



## Acknowledgement

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