Network Working Group
Internet-Draft
Expires: March 26, 2001

M.T. Rose
Invisible Worlds, Inc.

September 25, 2000

The Blocks Extensible Exchange Protocol Framework
draft-ietf-beep-framework-02

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

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

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

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

This Internet-Draft will expire on March 26, 2001.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

This memo describes a generic application protocol framework for connection-oriented, asynchronous interactions. The framework 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 Framework 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
2.2.2.2 XML-based Profiles 15
2.3 Channel Management 16
2.3.1 Message Semantics 17
2.3.1.1 The Greeting Message 17
2.3.1.2 The Start Message 19
2.3.1.3 The Close Message 22
2.3.1.4 The OK Message 24
2.3.1.5 The Error Message 24
2.4 Session Establishment and Release 26
2.5 Transport Mappings 28
2.5.1 Session Management 28
2.5.2 Message Exchange 28
2.6 Parallelism 29
2.6.1 Within a Single Channel 29
2.6.2 Between Different Channels 29
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
7. DTDs 47
7.1 BEEP Channel Management DTD 47
7.2 TLS Transport Security Profile DTD 49
7.3 SASL Family of Profiles DTD 50
8. Reply Codes 51
9. Security Considerations 52
10. IANA Considerations 53
References 54
Author's Address 55
A. Changes from draft-ietf-beep-framework-01 56
B. Changes from draft-ietf-beep-framework-00 57
C. Changes from draft-mrose-bxxp-framework-01 58
D. Changes from draft-mrose-bxxp-framework-00 59
E. Acknowledgements 60
Full Copyright Statement 61

1. Introduction

This memo describes a generic application protocol framework for connection-oriented, asynchronous interactions. Consult [1] for a description of the framework's design principles.

At the core of the BEEP framework is a framing mechanism that permits simultaneous and independent exchanges of messages between peers. Messages are arbitrary MIME[2] content, but are usually textual (structured using XML[3]).

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[4] transport security profile; and,

- the SASL[5] family of profiles.

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

2. The BEEP Framework

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, [6] describes how a BEEP session is mapped onto a single TCP[7] connection.

When a session is established, each BEEP peer advertises the profile 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.

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[2]'s Section 3). If none, or only some, of the "entity-headers" are present:

o the default "Content-Type" is "application/octet-stream"; and,

o the default "Content-Transfer-Encoding" is "binary".

Normally, a message is sent in a single frame. However, it may be convenient or necesary 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 119 octets spread over 5 lines (each line is terminated with a CRLF pair):

```
C: MSG 0 1 . 40 120
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
C: </start>
C: END
C:
```

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

2.2.1 Frame Syntax

The ABNF[8] 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 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.

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:
S: RPY 0 1 . 307 0
S: END
S:
```

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, [6] 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 sequentally, 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:
S: ANS 1 0 * 20 20 1
S:     ...
S:     ...
S: END
S:
S: ANS 1 0 . 40 10 0
S:     ...
S: END
S:
```

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 not 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", "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 reply 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.

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.

2.2.2.2 XML-based Profiles

If a profile uses XML[3] to structure its messages, then only XML's baseline facilities (as described in the XML 1.0 specification[3]) are allowed. Additional XML facilities (e.g., namespaces) are made available only by being explicitly permitted in a given profile's specification.

In particular this limitation allows use of only the five predefined general entities references ("&", "<", ">", "'", and """) and numeric entity references in the messages exchanged.

Further, because the profile registration template defines the messages exchanged over a channel, the XML documents exchanged in each message needn't have either a "XML" declaration (e.g., <?xml version="1.0" ?>) or a "DOCTYPE" declaration (e.g., <!DOCTYPE ...>). All other XML 1.0 instructions (e.g., CDATA blocks, processing instructions, and so on) are allowed.

Finally, because the "XML" declaration isn't present, the default character set for XML-based profiles is UTF-8. If another character set is desired, the "Content-Type" entity-header must indicate this.

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: text/xml
L:
L: <greeting>
L:    <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
L:
I: RPY 0 0 . 0 40
I: Content-Type: text/xml
I:
I: <greeting />
I: END
I:
```

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.

At present, there are no optional features defined for the channel management profile.

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 . 40 120
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
C: </start>
C: END
C:
```

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 . 40 197
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
C:    <profile
C:       uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
C: </start>
C: END
C:
S: RPY 0 1 . 252 87
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
S: END
S:
```

Similarly, an unsuccessful channel creation might look like this:

```
C: MSG 0 1 . 40 120
C: Content-Type: text/xml
C:
C: <start number='2'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
C: </start>
C: END
C:
S: ERR 0 1 . 252 115
S: Content-Type: text/xml
S:
S: <error code='501'>number attribute
S: in &lt;start&gt; element must be odd-valued</error>
S: END
S:
```

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

```
C: MSG 0 1 . 40 158
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/TLS'>
C:        <![CDATA[<ready />]]>
C:    </profile>
C: </start>
C: END
C:
S: RPY 0 1 . 110 121
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/TLS'>
S:     <![CDATA[<proceed />]]>
S: </profile>
S: END
S:
```

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 . 223 59
C: Content-Type: text/xml
C:
C: <close number='1' code='200' />
C: END
C:
```

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.

When a BEEP peer receives a "close" element on channel zero, it decides whether it is willing to close the channel. If so, an "ok" element is sent in a positive reply; otherwise, an "error" element is sent in a negative reply.

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

```
C: MSG 0 2 . 223 59
C: Content-Type: text/xml
C:
C: <close number='1' code='200' />
C: END
C:
S: RPY 0 2 . 423 34
S: Content-Type: text/xml
S:
S: <ok />
S: END
S:
```

Similarly, an unsuccessful channel close might look like this:

```
C: MSG 0 2 . 223 59
C: Content-Type: text/xml
C:
C: <close number='1' code='200' />
C: END
C:
S: ERR 0 2 . 423 67
S: Content-Type: text/xml
S:
S: <error code='550'>still working</error>
S: END
S:
```

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 . 40 115
I: Content-Type: text/xml
I:
I: <start number='2'>
I:    <profile uri='http://xml.resource.org/profiles/FOO' />
I: </start>
I: END
I:
L: ERR 0 1 . 252 93
L: Content-Type: text/xml
L:
L: <error code='550'>all requested profiles are
L: unsupported</error>
L: END
L:
```

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: text/xml
L:
L: <greeting>
L:    <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
L:
I: RPY 0 0 . 0 40
I: Content-Type: text/xml
I:
I: <greeting />
I: END
I:
```

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 48
L: Content-Type: text/xml
L:
L: <error code='421' />
L: END
L:
I: RPY 0 0 . 0 40
I: Content-Type: text/xml
I:
I: <greeting />
I: END
I:
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 . 40 48
C: Content-Type: text/xml
C:
C: <close code='200' />
C: END
C:
S: RPY 0 1 . 252 34
S: Content-Type: text/xml
S:
S: <ok />
S: END
S:
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 . 40 48
C: Content-Type: text/xml
C:
C: <close code='200' />
C: END
C:
S: ERR 0 1 . 252 67
S: Content-Type: text/xml
S:
S: <error code='550'>still working</error>
S: END
S:
```

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 satisified 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 Parallelism

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.

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.

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.

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 . 40 120
I: Content-Type: text/xml
I:
I: <start number='1'>
I:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
I: </start>
I: END
I:
L: MSG 0 1 . 252 116
L: Content-Type: text/xml
L:
L: <start number='2'>
L:    <profile uri='http://xml.resource.org/profiles/IMXP' />
L: </start>
L: END
L:
```

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[4].

Other profiles may be defined and deployed on a bilateral basis. Note that because of their intimate relationship with the tranpsort service, a given transport security profile tends to be relevant to a single transort 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: text/xml
L:
L: <greeting>
L:    <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
L:
I: RPY 0 0 . 0 40
I: Content-Type: text/xml
I:
I: <greeting />
I: END
I:
I: MSG 0 1 . 40 158
I: Content-Type: text/xml
I:
I: <start number='1'>
I:    <profile uri='http://xml.resource.org/profiles/TLS'>
I:        <![CDATA[<ready />]]>
```

```
I:    </profile>
```

```
      I: </start>
      I: END
      I:
      L: RPY 0 1 . 110 121
      L: Content-Type: text/xml
      L:
      L: <profile uri='http://xml.resource.org/profiles/TLS'>
      L:     <![CDATA[<proceed />]]>
      L: </profile>
      L: END
      L:

          ... successful transport security negotiation ...

      L: RPY 0 0 . 0 252
      L: Content-Type: text/xml
      L:
      L: <greeting>
      L:    <profile
      L:       uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
      L:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
      L:    <profile uri='http://xml.resource.org/profiles/IMXP' />
      L: </greeting>
      L: END
      L:
      I: RPY 0 0 . 0 40
      I: Content-Type: text/xml
      I:
      I: <greeting />
      I: END
      I:
```

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

```
      L: <wait for incoming connection>
      I: <open connection>
      L: RPY 0 0 . 0 311
      L: Content-Type: text/xml
      L:
      L: <greeting>
      L:    <profile
      L:       uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
      L:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
      L:    <profile uri='http://xml.resource.org/profiles/IMXP' />
      L:    <profile uri='http://xml.resource.org/profiles/TLS' />
      L: </greeting>
      L: END
      L:
```

```
I: RPY 0 0 . 0 40
```

```
I: Content-Type: text/xml
I:
I: <greeting />
I: END
I:
I: MSG 0 1 . 40 158
I: Content-Type: text/xml
I:
I: <start number='1'>
I:    <profile uri='http://xml.resource.org/profiles/TLS'>
I:        <![CDATA[<ready />]]>
I:    </profile>
I: </start>
I: END
I:
L: RPY 0 1 . 311 121
L: Content-Type: text/xml
L:
L: <profile uri='http://xml.resource.org/profiles/TLS'>
L:     <![CDATA[<proceed />]]>
L: </profile>
L: END
L:

    ... failed transport security negotiation ...

L: RPY 0 0 . 0 311
L: Content-Type: text/xml
L:
L: <greeting>
L:    <profile
L:       uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
L:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
L:    <profile uri='http://xml.resource.org/profiles/IMXP' />
L:    <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
L:
I: RPY 0 0 . 0 40
I: Content-Type: text/xml
I:
I: <greeting />
I: END
I:
```

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://xml.resource.org/profiles/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 . 40 158
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/TLS'>
C:        <![CDATA[<ready />]]>
C:    </profile>
C: </start>
C: END
C:
S: RPY 0 1 . 110 121
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/TLS'>
S:     <![CDATA[<proceed />]]>
S: </profile>
S: END
S:
```

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

```
C: MSG 0 1 . 40 173
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/TLS'>
C:        <![CDATA[<ready version="oops" />]]>
C:    </profile>
C: </start>
C: END
C:
S: RPY 0 1 . 110 181
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/TLS'>
S:     <error code='501'>version attribute
S: poorly formed in &lt;ready&gt; element</error>
S: </profile>
S: END
S:
```

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 any channel until a corresponding reply is received; similarly, before processing a "ready" element, the receiving BEEP peer waits until any pending replies have been generated and sent.

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 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[14] 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[5] 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://xml.resource.org/profiles/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 . 40 197
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile
C:       uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
C: </start>
C: END
C:
S: RPY 0 1 . 252 87
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
S: END
S:
```

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 . 40 183
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
C:        <![CDATA[<blob>AGJsb2NrbWFzdGVy</blob>]]>
C:    </profile>
C: </start>
C: END
C:
S: RPY 0 1 . 252 166
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
S:     <error code='534'>authentication mechanism is
S: too weak</error>
S: </profile>
S: END
S:
```

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 . 40 183
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
C:        <![CDATA[<blob>AGJsb2NrbWFzdGVy</blob>]]>
C:    </profile>
C: </start>
C: END
C:
S: RPY 0 1 . 252 171
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
S:    <![CDATA[<blob>b3RwLXNoYTEgOTk5NyBwaXh5bWlzYXM4NTgwNSBleHQ=
                                                        </blob>]]>
S: </profile>
S: END
S:
```

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 85
C: Content-Type: text/xml
C:
C: <blob>d29yZDpmZXJuIGhhbmcgYnJvdyBib25nIGhlcmQgdG9n</blob>
C: END
C:
S: RPY 1 0 . 0 54
S: Content-Type: text/xml
S:
S: <blob status='complete' />
S: END
S:
```

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 85
C: Content-Type: text/xml
C:
C: <blob>d29yZDpmZXJuIGhhbmcgYnJvdyBib25nIGhlcmQgdG9n</blob>
C: END
C:
S: ERR 1 1 . 0 48
S: Content-Type: text/xml
S:
S: <error code='535' />
S: END
S:
```

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

```
C: MSG 0 1 . 40 133
C: Content-Type: text/xml
C:
C: <start number='1'>
C:    <profile
C:        uri='http://xml.resource.org/profiles/sasl/CRAM-MD5' />
C: </start>
C: END
C:
S: RPY 0 1 . 252 173
S: Content-Type: text/xml
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/CRAM-MD5'>
S: <blob>PDE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2UucmVzdG9uLm1jaS5uZXQ+
                                                        </blob>
S: </profile>
S: END
S:
```

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://xml.resource.org/profiles/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://xml.resource.org/profiles/sasl/MECHANISM, where "MECHANISM" is a token registered with the IANA[15]

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

7. DTDs

7.1 BEEP Channel Management DTD

```
<!--
  DTD for BEEP Channel Management, as of 2000-09-17


  Refer to this DTD as:

    <!ENTITY % BEEP PUBLIC "-//Blocks//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, seperated by space
       FTRS          NMTOKENS             "magic"

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

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

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


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

```
   <!--
     BEEP messages

        role       MSG         RSP         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 "-//Blocks//DTD TLS//EN"
               "http://xml.resource.org/profiles/TLS/tls.dtd">
    %TLS;
  -->


<!--
  TLS messages

     role       MSG         RSP         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 "-//Blocks//DTD SASL//EN"
               "http://xml.resource.org/profiles/sasl/sasl.dtd">
    %SASL;
  -->


<!--
  SASL messages

     role       MSG         RSP         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
421	service not available
450	requested action not taken (e.g., lock already in use)
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. 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 [5]'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.

10. IANA Considerations

The IANA registers "beep" as a GSSAPI[13] service name, as specified in Section 4.1.

The IANA maintains a list of:

o BEEP profiles, c.f., Section 5.1; and,

o features for the channel management profile, c.f., Section 5.2.

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.

References

[1] Rose, M.T., "On the Design of Application Protocols", draft-mrose-beep-design-00 (work in progress), July 2000.

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

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

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

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

[6] Rose, M.T., "Mapping the BEEP Framework onto TCP", draft-ietf-beep-tcpmapping-02 (work in progress), September 2000.

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

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

[9] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766, March 1995.

[10] Berners-Lee, T., Fielding, R.T. 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] Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997.

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

[15] <URL:http://www.iana.org/>

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. Changes from draft-ietf-beep-framework-01

- s/BXXP/BEEP/g
- The default "Content-Type:" is now "application/octet-stream".
- Initialization messages are now character data, not XML elements.
- A template is provided for channel management feature registration (c.f., Section 5.2).

Appendix B. Changes from draft-ietf-beep-framework-00

- The names of messages are renamed:
  - "REQ" messages are now "MSG" messages; and,
  - "RSP" messages are now "RPY" (positive), "ANS"/"NULL" (one-to-many), and "ERR" (negative).
- One-to-many exchanges are supported using the "ANS" message.
- Commonly-used paraemters are re-ordered in the header:
  - channel numbers appear in each frame (and are 31-bits wide); and,
  - serial numbers are now message numbers, and are per-channel.
- MIME "entity-headers" are now part of the payload (and there is no longer any header-related processing associated with them).
- An IANA registration for BEEP error codes is no longer required (the error codes are used only within this specification).
- The close message (Section 2.3.1.3) is also used to release the BEEP session.

Appendix C. Changes from draft-mrose-bxxp-framework-01

- Channel numbers are now 31-bits wide (instead of 8-bits).
- Peers should support at least 257 concurrent channels.
- The consistency rules in Section 2.2.1.1 now mandate that any MIME entity-headers occur only in the first frame of a message.
- Discussion of the role of the entity-headers is moved to Section 2.2.1.1.
- Section 2.2.2 requires that a BEEP peer close a channel when a poorly-formed reply is received (unless it's channel zero, in which case the BEEP session is terminated).
- Section 2.2.2 explains that in an XML-based profile, if something other than UTF-8 is sent, then a "Content-Type:" entity-header must be present to specify the character set.
- The close (Section 2.3.1.3) and ok (Section 2.3.1.4) messages were added.
- Both Section 2.3.1.3 and Section 2.3.1.5 clarify that diagnostic text is not to be interpreted by programs.
- Section 5.1 limits the the size of an initialization element to 4K octets.

Appendix D. Changes from draft-mrose-bxxp-framework-00

- The IPR notice is changed to be in full conformance with all provisions of Section 10 of RFC2026.

- At the beginning of Section 2.2 (and in the ABNF in Section 2.2.1) the relationship between messages and frames is clarified.

- A typo involving the final CR LF in the ABNF in Section 2.2.1 is corrected.

- In Section 2.2.1.1, the "contiguous message" rule replaces the "transport-specific" assertion (the sixth rule for identifying poorly-formed frames).

- At the beginning of Section 2.3, an explanation of the relstionship between profiles and channels (and the greeting and start messages) is added.

- In Section 2.3.1, the order of the sections for the greeting and start messages is reversed for readability.

Appendix E. Acknowledgements

The author gratefully acknowledges the contributions of: David Clark, Dave Crocker, Steve Deering, 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.

Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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