                       The Application Exchange Core

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This memo describes Application Exchange (APEX) Core, an extensible,
   asynchronous message relaying service for application layer programs.

Table of Contents

1. Introduction

   Network applications can be broadly distinguished by five operational
   characteristics:

   o   server push or client pull;

   o   synchronous (interactive) or asynchronous (batch);

   o time-assured or time-insensitive;

   o best-effort or reliable; and,

   o stateful or stateless.

   For example:

   o the world-wide web is a pull, synchronous, time-insensitive,
      reliable, stateless service; whilst

   o Internet mail is a push, asynchronous, time-insensitive, best-
      effort (without DSN), stateless service.

   Messaging applications vary considerably in their operational
   requirements. For example, some messaging applications require
   assurance of timeliness and reliability, whilst others do not.

   These features come at a cost, in terms of both infrastructural and
   configuration complexity. Accordingly, the underlying service must
   be extensible to support different requirements in a consistent
   manner.

   This memo defines a core messaging service that supports a range of
   operational characteristics. The core service supports a variety of
   tailored services for both user-based and programmatic exchanges.

1.1 Overview

   APEX provides an extensible, asynchronous message relaying service
   for application layer programs.

   APEX, at its core, provides a best-effort datagram service. Each
   datagram, simply termed "data", is originated and received by APEX
   "endpoints" -- applications that dynamically attach to the APEX
   "relaying mesh".

   The data transmitted specifies:

   o an originating endpoint;

   o an opaque content (via a URI-reference);

   o one or more recipient endpoints; and,

   o zero or more options.

   Options are used to alter the semantics of the service, which may
   occur on a per-recipient or per-data basis, and may be processed by
   either a single or multiple relays.

   Additional APEX services are provided on top of the relaying mesh;
   e.g., access control and presence information.

   APEX is specified, in part, as a BEEP [1] "profile". Accordingly,
   many aspects of APEX (e.g., authentication) are provided within the
   BEEP core. Throughout this memo, the terms "peer", "initiator",
   "listener", "client", and "server" are used in the context of BEEP.
   In particular, Section 2.1 of the BEEP core memo discusses the roles
   that a BEEP peer may perform.

   When reading this memo, note that the terms "endpoint" and "relay"
   are specific to APEX, they do not exist in the context of BEEP.

1.2 Architecture at a Glance

   The APEX stack:
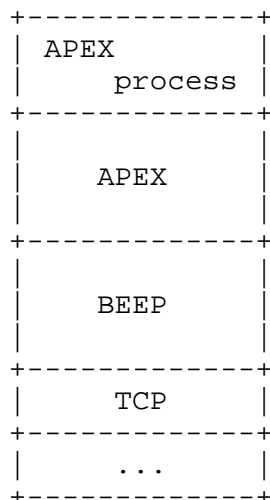
```
       +-------------+
       | APEX        |          an APEX process is either:
       |     process |
       +-------------+
       |             |             - an application attached as an APEX
       |    APEX     |               endpoint; or,
       |             |
       +-------------+             - an APEX relay
       |             |
       |    BEEP     |          APEX services are realized as applications
       |             |          having a special relationship with the APEX
       |             |          relays in their administrative domain
       +-------------+
       |    TCP      |
       +-------------+
       |     ...     |
       +-------------+
```

The APEX entities:

```
           administrative domain #1              administrative domain #2
      +---------------------------+      +----------------------------+
      |    +------+               |      |               +------+     |
      |    |      |               |      |               |      |     |
      |    | appl |               |      |               | appl |     |
      |    |      |               |      |               |      |     |
      |    +......+    +------+    |      |    +------+    +......+     |
      |    |      |    |      |    |      |    |      |    |      |     |
      |    |end-  |    |relay |    |      |    |relay |    |end-  |     |
      |    | point|    |      |    |      |    |      |    | point|     |
      |    +------+    +------+    |      |    +------+    +------+     |
      |    |      |    |      |    |      |    |      |    |      |     |
      |    | APEX |    | APEX |    |      |    | APEX |    | APEX |     |
      |    |      |    |      |    |      |    |      |    |      |     |
      |    +------+    +------+    |      |    +------+    +------+     |
      |      ||          ||  ||    |      |     || ||          ||      |
      |      =============  ================  ============= ==========|
      +---------------------------+      +----------------------------+

              | <---- APEX relaying mesh ----> |
```
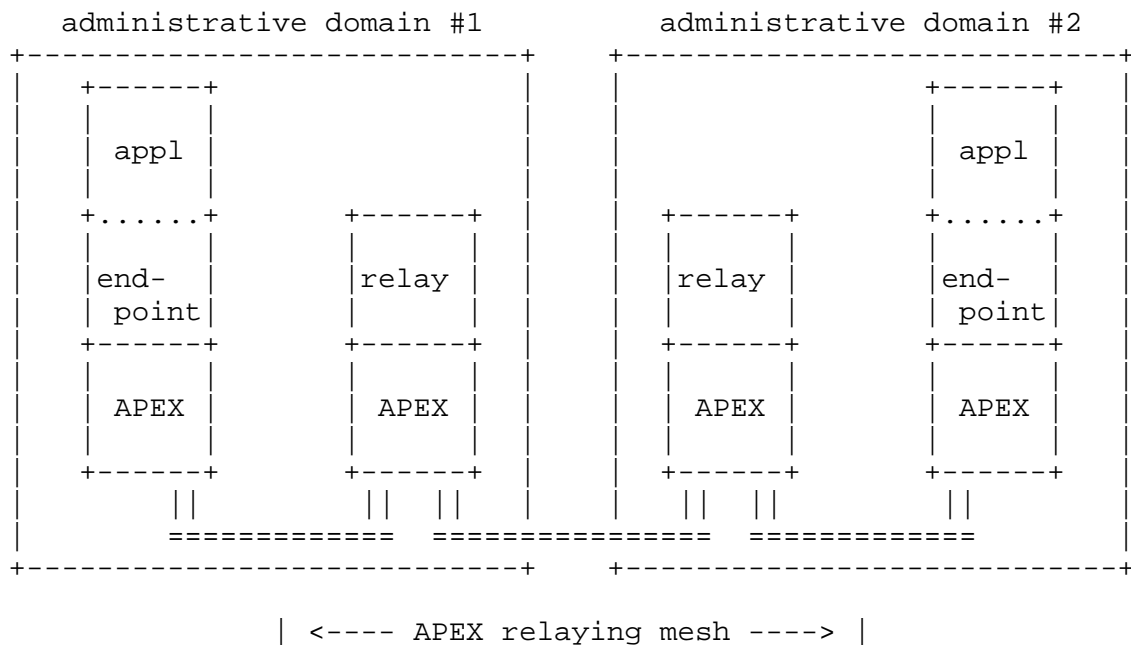
        Note: relaying between administrative domains is configured
              using SRV RRs. Accordingly, the actual number of
              relays between two endpoints is not fixed.

2. Service Principles

2.1 Modes of Operation

   APEX is used in two modes:

   endpoint-relay: in which the endpoint is always the BEEP initiator of
      the service, whilst relays are always the BEEP listeners. In this
      context, applications attach as endpoints, and then the
      transmission of data occurs.

   relay-relay: in which relays typically, though not necessarily,
      reside in different administrative domains. In this context,
      applications bind as relays, and then the transmission of data
      occurs.

   In the endpoint-relay mode, an endpoint (BEEP initiator) may:

   o   attach as one or more endpoints;

   o   send data to other endpoints;

   o receive data from other endpoints; and,

   o terminate any of its attachments.

   A relay (BEEP listener), in addition to servicing requests from a
   BEEP initiator, may:

   o terminate any of the endpoint's attachments;

   o deliver data from other endpoints; and,

   o indicate the delivery status of data sent earlier by the endpoint.

   In the relay-relay mode, a relay (BEEP listener or initiator) may:

   o bind as one or more administrative domains;

   o send data;

   o receive data; and,

   o terminate any bindings.

2.2 Naming of Entities

   Endpoints are named using the following ABNF [2] syntax:

       ;; Domain is defined in [3], either a FQDN or a literal
       entity   = local "@" Domain

       local   = address ["/" subaddress]

       address   = token

       subaddress = token

       ;; all non-control characters, excluding "/" and "@" delimiters
       token   = 1*(%x20-2E / %x30-3F / %x41-7E / UTF-8) ;; [4]

   Two further conventions are applied when using this syntax:

   the "apex=" convention: All endpoint identities having a local-part
      starting with "apex=" are reserved for use by APEX services
      registered with the IANA; and,

the "subaddress" convention: If the solidus character ("/", decimal
   code 47) occurs in the local-part, this identifies a subaddress of
   an endpoint identity (e.g., "fred/appl=wb@example.com" is a
   subaddress of the APEX endpoint "fred@example.com").

   All subaddresses starting with "appl=" are reserved for use by
   APEX endpoint applications registered with the IANA.

Relays, although not named, serve of behalf of administrative
domains, as identified by a FQDN or a domain-literal, e.g.,
"example.com" or "[10.0.0.1]".

In APEX, "endpoints" and "relays" are the fundamental entities. APEX
is carried over BEEP, which has the "peer" as its fundamental entity.
The relationship between BEEP peer entities and APEX endpoint and
relay entities are defined by APEX's Access Policies (Section 4.5).

2.2.1 Comparing Endpoints

Note that since the "local" part of an entity is a string of UTF-8
[4] octets, comparison operations on the "local" part use exact
matching (i.e., are case-sensitive).

Accordingly, "fred@example.com" and "Fred@example.com" refer to
different endpoints. Of course, relays serving the "example.com"
administrative domain may choose to treat the two endpoints
identically for the purposes of routing and delivery.

Finally, note that if an APEX endpoint is represented using a
transmission encoding, then, prior to comparison, the encoding is
reversed. For example, if the URL encoding is used, then
"apex:fred@example.com" is identical to "apex:f%72ed@example.com".

3. Service Provisioning

3.1 Connection Establishment

The SRV algorithm [5] is used to determine the IP/TCP addressing
information assigned to the relays for an administrative domain
identified by a FQDN:

service: "apex-edge" (for the endpoint-relay mode), or "apex-mesh"
   (for the relay-relay mode);

protocol: "tcp"; and,

domain: the administrative domain.

If the administrative domain is identified by a domain-literal, then
the IP address information is taken directly from the literal and the
TCP port number used is assigned by the IANA for the registration in
Section 8.2.

3.2 Authentication

Authentication is a matter of provisioning for each BEEP peer (c.f.,
Section 4.5).

An APEX relay might be provisioned to allow a BEEP peer identity to
coincide with a given endpoint identity. For example, a relay in the
"example.com" administrative domain may be configured to allow a BEEP
peer identified as "fred@example.com" to be authorized to attach as
the APEX endpoint "fred@example.com".

3.3 Authorization

Authorization is a matter of provisioning for each BEEP peer (c.f.,
Section 4.5).

Typically, a relay requires that its BEEP peer authenticate as a
prelude to authorization, but an endpoint usually does not require
the same of its BEEP peer.

3.4 Confidentiality

Confidentiality is a matter of provisioning for each BEEP peer.

Typically, any data considered sensitive by an originating endpoint
will have its content encrypted for the intended recipient
endpoint(s), rather than relying on hop-by-hop encryption.
Similarly, an originating endpoint will sign the content if end-to-
end authentication is desired.

3.5 Relaying Integrity

Data are relayed according to SRV entries in the DNS. Accordingly,
relaying integrity is a function of the DNS and the applications
making use of the DNS. Additional assurance is provided if the BEEP
initiator requires that the BEEP listener authenticate itself.

3.6 Traffic Analysis

   Hop-by-hop protection of data transmitted through the relaying mesh
   (endpoint identities and content) is afforded at the BEEP level
   through the use of a transport security profile. Other traffic
   characteristics, e.g., volume and timing of transmissions, are not
   protected from third-party analysis.

4. The APEX

   Section 8.1 contains the BEEP profile registration for APEX.

4.1 Use of XML and MIME

   Each BEEP payload exchanged via APEX consists of an XML document and
   possibly an arbitrary MIME content.

   If only an XML document is sent in the BEEP payload, then the mapping
   to a BEEP payload is straight-forward, e.g.,

```
C: MSG 1 2 . 111 39
C: Content-Type: application/beep+xml
C:
C: <terminate transID='1' />
C: END
```

   Otherwise, if an arbitrary MIME content is present, it is indicated
   by a URI-reference [6] in the XML control document. The URI-
   reference may contain an absolute-URI (and possibly a fragment-
   identifier), or it may be a relative-URI consisting only of a
   fragment-identifier. Arbitrary MIME content is included in the BEEP
   payload by using a "multipart/related" [7], identified using a "cid"
   URL [8], and the XML control document occurs as the start of the
   "multipart/related", e.g.,

```
C: MSG 1 1 . 42 1234
C: Content-Type: multipart/related; boundary="boundary";
C:               start="<1@example.com>";
C:               type="application/beep+xml"
C:
C: --boundary
C: Content-Type: application/beep+xml
C: Content-ID: <1@example.com>
C:
C: <data content='cid:2@example.com'>
C:     <originator identity='fred@example.com' />
C:     <recipient identity='barney@example.com' />
C: </data>
```

```
C: --boundary
C: Content-Type: image/gif
C: Content-Transfer-Encoding: binary
C: Content-ID: <2@example.com>
C:
C: ...
C: --boundary--
C: END
```

   Because BEEP provides an 8bit-wide path, a "transformative" Content-
   Transfer-Encoding (e.g., "base64" or "quoted-printable") should not
   be used. Further, note that MIME [9] requires that the value of the
   "Content-ID" header be globally unique.

   If the arbitrary MIME content is itself an XML document, it may be
   contained within the control document directly as a "data-content"
   element, and identified using a URI-reference consisting of only a
   fragment-identifier, e.g.,

```
C: MSG 1 1 . 42 295
C: Content-Type: application/beep+xml
C:
C: <data content='#Content'>
C:     <originator identity='fred@example.com' />
C:     <recipient identity='barney@example.com' />
C:     <data-content Name='Content'>
C:         <statusResponse transID='86'>
C:             <destination identity='barney@example.com'>
C:                 <reply code='250' />
C:             </destination>
C:         </statusResponse>
C:     </data-content>
C: </data>
C: END
```

4.2 Profile Identification and Initialization

   The APEX is identified as

      http://iana.org/beep/APEX

   in the BEEP "profile" element during channel creation.

   No elements are required to be exchanged during channel creation;
   however, in the endpoint-relay mode, the BEEP initiator will
   typically include an "attach" element during channel creation, e.g.,

```
        <start number='1'>
            <profile uri='http://iana.org/beep/APEX'>
                <![CDATA[<attach endpoint='fred@example.com'
                                  transID='1' />]]>
            </profile>
        </start>
```

   Similarly, in the relay-relay mode, the BEEP initiator will typically
   include an "bind" element during channel creation, e.g.,

```
        <start number='1'>
            <profile uri='http://iana.org/beep/APEX'>
                <![CDATA[<bind relay='example.com'
                               transID='1' />]]>
            </profile>
        </start>
```

4.3 Message Syntax

   Section 9.1 defines the BEEP payloads that are used in the APEX.

4.4 Message Semantics

4.4.1 The Attach Operation

   When an application wants to attach to the relaying mesh as a given
   endpoint, it sends an "attach" element to a relay, e.g.,

```
            +-------+                    +-------+
            |       | -- attach ----->   |       |
            | appl. |                    | relay |
            |       | <--------- ok --   |       |
            +-------+                    +-------+
```

```
        C: <attach endpoint='fred@example.com' transID='1' />
        S: <ok />
```

   or

```
            +-------+                    +-------+
            |       | -- attach ----->   |       |
            |       |                    |       |
            |       | <--------- ok --   |       |
            | appl. |                    | relay |
            |       | -- attach ----->   |       |
            |       |                    |       |
            |       | <--------- ok --   |       |
            +-------+                    +-------+
```

```
      C: <attach endpoint='fred@example.com' transID='1' />
      S: <ok />
      C: <attach endpoint='wilma@example.com' transID='2' />
      S: <ok />
```

   or

```
         +-------+                      +-------+
         |       | -- attach ----->     |       |
         | appl. |                      | relay |
         |       | <------ error --     |       |
         +-------+                      +-------+
```
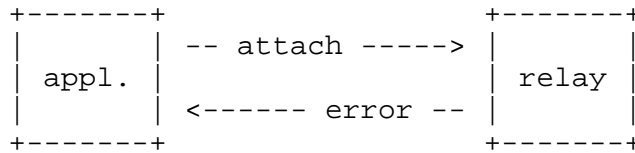
```
      C: <attach endpoint='fred@example.com' transID='1' />
      S: <error code='537'>access denied</error>
```

   The "attach" element has an "endpoint" attribute, a "transID"
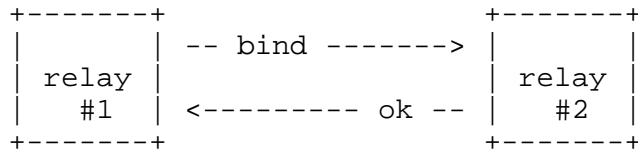   attribute, and contains zero or more "option" elements:

   o the "endpoint" attribute specifies the endpoint that the
      application wants to attach as;

   o the "transID" attribute specifies the transaction-identifier
      associated with this operation; and,

   o the "option" elements, if any, specify additional processing
      options (Section 5).

   When a relay receives an "attach" element, it performs these steps:

   1. If the transaction-identifier refers to a previous, non-terminated
      operation on this BEEP channel, an "error" element having code 555
      is returned.

   2. If the relay is in a different administrative domain than this
      endpoint, an "error" element having code 553 is returned.

   3. If the application is not authorized to attach as this endpoint
      (c.f., Section 4.5.1), an "error" element having code 537 is
      returned.

   4. If any options are present, they are processed.

   5. If another application has already attached as this endpoint, an
      "error" element having code 554 is returned.

   6. Otherwise, the application is bound as this endpoint, and an "ok"
      element is returned.

4.4.2 The Bind Operation

   When an application wants to identify itself as a relay, it sends a
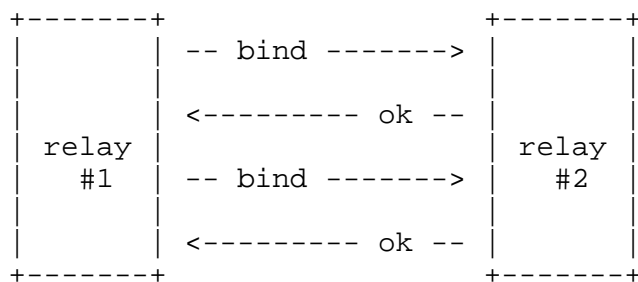   "bind" element to another relay, e.g.,

```
        +-------+              +-------+
        |       | -- bind ------> |       |
        | relay |                 | relay |
        |   #1  | <--------- ok -- |   #2  |
        +-------+              +-------+
```

        C: <bind relay='example.com' transID='1' />
        S: <ok />

   or

```
        +-------+              +-------+
        |       | -- bind ------> |       |
        |       |              |       |
        |       | <--------- ok -- |       |
        | relay |                 | relay |
        |   #1  | -- bind ------> |   #2  |
        |       |              |       |
        |       | <--------- ok -- |       |
        +-------+              +-------+
```

        C: <bind relay='example.com' transID='1' />
        S: <ok />
        C: <bind relay='rubble.com' transID='2' />
        S: <ok />

   or

```
        +-------+              +-------+
        |       | -- bind ------> |       |
        | relay |                 | relay |
        |   #1  | <------ error -- |   #2  |
        +-------+              +-------+
```

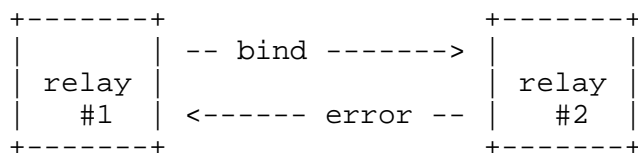   C: <bind relay='example.com' transID='1' />
        S: <error code='537'>access denied</error>

   The "bind" element has a "relay" attribute, a "transID" attribute,
   and contains zero or more "option" elements:

   o the "relay" attribute specifies the administrative domain on whose
      behalf the application wants to serve;

   o the "transID" attribute specifies the transaction-identifier
      associated with this operation; and,

   o the "option" elements, if any, specify additional processing
      options (Section 5).

   When a relay receives an "bind" element, it performs these steps:

   1. If the transaction-identifier refers to a previous, non-terminated
      operation on this BEEP channel, an "error" element having code 555
      is returned.

   2. If the application is not authorized to bind on behalf of this
      administrative domain (c.f., Section 4.5.2), an "error" element
      having code 537 is returned.

   3. If any options are present, they are processed.

   4. Otherwise, the application is accepted as serving this
      administrative domain, and an "ok" element is returned.

4.4.3 The Terminate Operation

   When an application or relay wants to release an attachment or
   binding, it sends a "terminate" element, e.g.,

```
        +-------+                     +-------+
        |       | -- terminate --> |       |
        | appl. |                     | relay |
        |       | <--------- ok -- |       |
        +-------+                     +-------+
```

        C: <terminate transID='1' />
        S: <ok />

   or

```
        +-------+                     +-------+
        |       | -- terminate --> |       |
        | appl. |                     | relay |
        |       | <------ error -- |       |
        +-------+                     +-------+
```
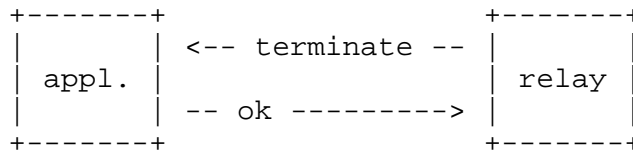
        C: <terminate transID='13' />
        S: <error code='550'>unknown transaction-identifier</error>

   or

```
        +-------+                   +-------+
        |       | <-- terminate -- |       |
        | appl. |                   | relay |
        |       | -- ok --------->  |       |
        +-------+                   +-------+
```

        C: <terminate transID='1' />
        S: <ok />

   The "terminate" element has a "transID" attribute, an optional "code"
   attribute, an optional "xml:lang" attribute, and may contain
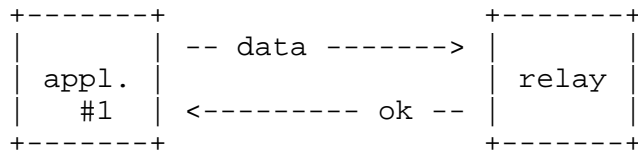   arbitrary textual content:

   o the "transID" attribute specifies the transaction-identifier
      associated with this operation;

   o the "code" attribute, if present, is a three-digit reply code
      meaningful to programs (c.f., Section 10);

   o the "xml:lang" attribute, if present, specifies the language that
      the element's content is written in; and,

   o the textual content is a diagnostic (possibly multiline) which is
      meaningful to implementers, perhaps administrators, and possibly
      even users.

   When an application or relay receives a "terminate" element, it
   performs these steps:

   1. If the value of the transaction-identifier is zero, then all
      associations established by this application over this BEEP
      session, either as an endpoint attachment or a relay binding, are
      terminated, and an "ok" element is returned.

   2. Otherwise, if the transaction-identifier does not refer to a
      previous unterminated operation on this BEEP channel, an "error"
      element having code 550 is returned.

   3. Otherwise, the application is no longer bound as an endpoint or a
      relay, and an "ok" element is returned.

4.4.4 The Data Operation

   When an application or relay wants to transmit data over the relaying
   mesh, it sends a "data" element, e.g.,

```
         +-------+                  +-------+
         |       | -- data ------->  |       |
         | appl. |                  | relay |
         |  #1   | <--------- ok -- |       |
         +-------+                  +-------+
```

```
    C: <data content='cid:1@example.com'>
           <originator identity='fred@example.com' />
           <recipient identity='barney@example.com' />
        </data>
    S: <ok />
```
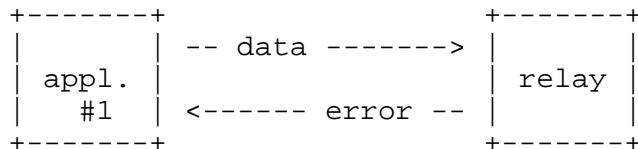
or

```
         +-------+                  +-------+
         |       | -- data ------->  |       |
         | appl. |                  | relay |
         |  #1   | <------ error -- |       |
         +-------+                  +-------+
```

```
    C: <data content='cid:1@example.com'>
           <originator identity='fred@example.com' />
           <recipient identity='barney@example.com' />
        </data>
    S: <error code='537'>access denied</error>
```

or

```
             +-------+                  +-------+
             |       | -- data ------->  |       |
             | relay |                  | appl. |
             |       | <--------- ok -- |   #2  |
             +-------+                  +-------+
```

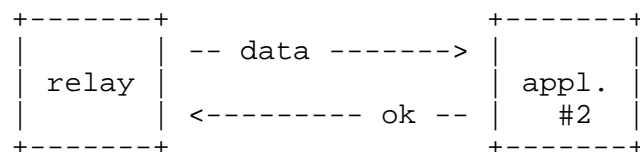```
    C: <data content='cid:1@example.com'>
           <originator identity='fred@example.com' />
           <recipient identity='barney@example.com' />
        </data>
    S: <ok />
```

The "data" element has a "content" attribute, and contains an
"originator" element, one or more "recipient" elements, zero or more
"option" elements, and, optionally, a "data-content" element:

o   the "content" attribute is a URI-reference that specifies the
    contents of the data (c.f., Section 4.1);

o   the "originator" element refers to the endpoint sending the data;

o  each "recipient" element refers to an endpoint destination for the
   data;

o  the "option" elements, if any, specify additional processing
   options (Section 5), termed per-data options; and,

o  the "data-content" element, if present, specifies a nested XML
   entity that is referenced using a URI fragment-identifier as the
   value of the "content" attribute.

The "originator" element has an "identity" attribute, and contains
zero or more option elements:

o  the "identity" attribute specifies the sending endpoint; and

o  the "option" elements, if any, specify additional processing
   options for the originator, termed per-originator options.

Each "recipient" element has an "identity" attribute, and contains
zero or more option elements:

o  the "identity" attribute specifies the destination endpoint; and

o  the "option" elements, if any, specify additional processing
   options for this recipient, termed per-recipient options.

4.4.4.1 Relay Processing of Data

When a relay receives a "data" element, it performs these steps:

1. If the BEEP client is not authorized to originate or relay data on
   behalf of the "originator" endpoint (c.f., Section 4.5), an
   "error" element having code 537 is returned.

2. If any per-data options are present, they are processed.

3. An "ok" element is returned.

4. If any per-originator options are present, they are processed.

5. For each recipient:

   1. If any per-recipient options are present, they are processed.

2. If the recipient endpoint is not in the administrative domain
   associated with the relay, then an APEX session is established
   to a relay that accepts data for the recipient's administrative
   domain, and a new "data" element, containing that "recipient"
   element and all applicable options, is sent to that relay.

   If an APEX session is established, the new "data" is sent, and
   the recipient's relay returns an "ok" element, then the
   recipient is considered to be successfully processed.

3. Otherwise, if the recipient endpoint is in the same
   administrative domain as the relay, the APEX access service
   must check that the originator endpoint is allowed to
   communicate with the recipient endpoint (the access entries
   [10] whose "owner" is the recipient must contain a "core:data"
   token for the originator), and the recipient endpoint must be
   currently attached.

   If so, a new "data" element, containing only that "recipient"
   element, is sent to the corresponding application.  If the
   recipient's endpoint returns an "ok" element, then the
   recipient is considered to be successfully processed.

Providing that these semantics are preserved, a relay may choose to
optimize its behavior by grouping multiple recipients in a single
"data" element that is subsequently transmitted.

Finally, note that a relay receiving a "data" element from an
application may be configured to add administrative-specific options.

Regardless, all relays are expressly forbidden from modifying the
content of the "data" element at any time.

4.4.4.2 Application Processing of Data

When an application receives a "data" element, it performs these
steps:

1. If any per-data or per-originator options are present, they are
   not processed (but may be noted).

2. For each recipient:

   1. If any per-recipient options are present, they are not
      processed (but may be noted).

   2. If the application is not attached as the recipient endpoint,
      then an error in processing has occurred.

    3. Otherwise, the "data" element is further processed in an
       application-specific manner, and the recipient is considered to
       be successfully processed.

  3. If no recipients could be successfully processed, an "error"
     element is returned; otherwise, an "ok" element is returned.

4.5 APEX Access Policies

   Access to APEX is provided by the juxtaposition of:

   o  authenticating as a BEEP peer;

   o  attaching as an APEX endpoint or binding as an APEX relay; and,

   o  being listed as an actor by the APEX access service (c.f., [10]).

   Each of these activities occurs according to the policies of the
   relevant administrative domain:

   o  each administrative domain is responsible for keeping its own
      house in order through "local provisioning"; and,

   o  each administrative domain decides the level of trust to associate
      with other administrative domains.

4.5.1 Access Policies in the Endpoint-Relay Mode

   o  When an application wants to attach to the relaying mesh, local
      provisioning maps BEEP peer identities to allowed APEX endpoints
      (c.f., Step 3 of Section 4.4.1).

      Typically, the identity function is used, e.g., if an application
      authenticates itself as the BEEP peer named as "fred@example.com",
      it is allowed to attach as the APEX endpoint named as
      "fred@example.com".

      However, using the "subaddress" convention of Section 2.2, an
      application authorized to attach as a given APEX endpoint is also
      authorized to attach as any subaddress of that APEX endpoint,
      e.g., an application authorized to attach as the APEX endpoint
      "fred@example.com" is also authorized to attach as the APEX
      endpoint "fred/appl=wb@example.com".

   o  When an application wants to send data, local provisioning maps
      attached endpoints to allowed originators (c.f., Step 1 of Section
      4.4.4.1).

Typically, the identity function is used, e.g., if an application
attaches as the APEX endpoint named as "fred@example.com", it is
allowed to send data originating from the same APEX endpoint.
However, other policies are permissible, for example, the
administrative domain may allow the application attached as the
APEX endpoint named as "wilma@example.com" to send data
originating as either "wilma@example.com" or "fred@example.com".

o  Finally, when a relay is delivering to an endpoint within its own
   administrative domain, it consults the recipient's access entry
   looking for an entry having the originator as an actor (c.f., Step
   5.3 of Section 4.4.4.1).

4.5.2 Access Policies in the Relay-Relay Mode

o  When an application wants to bind as a relay on behalf of an
   administrative domain, local provisioning may map BEEP peer
   identities to allowed APEX relays (c.f., Step 3).

   If so, then typically the identity function is used.  e.g., if an
   application authenticates itself as the BEEP peer named as
   "example.com", it is allowed to bind as a relay on behalf of the
   administrative domain "example.com".

o  When a relay is sending data, no access policies, per se, are
   applied.

o  When a relay is receiving data, local provisioning maps BEEP peer
   identities to allowed originators (c.f., Step 1 of Section
   4.4.4.1).

   Typically, the identity function is used, e.g., if a relay
   authenticates itself as being from the same administrative domain
   as the originator of the data, then the data is accepted.

   In addition, some relays may also be configured as "trusted"
   intermediaries, so that if a BEEP peer authenticates itself as
   being from such a relay, then the data is accepted.

5. APEX Options

   APEX, at its core, provides a best-effort datagram service.  Options
   are used to alter the semantics of the core service.

   The semantics of the APEX "option" element are context-specific.
   Accordingly, the specification of an APEX option must define:

   o  the identity of the option;

o  the context in which the option may appear;

o  what content, if any, is contained within the option; and,

o  the processing rules for the option.

An option registration template (Section 7.1) organizes this
information.

An "option" element is contained within either a "data",
"originator", "recipient", or an "attach" element, all of which are
termed the "containing" element.  The "option" element has several
attributes and contains arbitrary content:

o  the "internal" and the "external" attributes, exactly one of which
   is present, uniquely identify the option;

o  the "targetHop" attribute specifies which relays should process
   the option;

o  the "mustUnderstand" attribute specifies whether the option, if
   unrecognized, must cause an error in processing to occur;

o  the "transID" attribute specifies a transaction-identifier for the
   option; and,

o  the "localize" attribute, if present, specifies one or more
   language tokens, each identifying a desirable language tag to be
   used if textual diagnostics are returned to the originator.

Note that if the containing element is an "attach", then the values
of the "targetHop" and "transID" attributes are ignored.

The value of the "internal" attribute is the IANA-registered name for
the option.  If the "internal" attribute is not present, then the
value of the "external" attribute is a URI or URI with a fragment-
identifier.  Note that a relative-URI value is not allowed.

The "targetHop" attribute specifies which relay(s) should process the
option:

   this: the option applies to this relay, and must be removed prior
   to transmitting the containing element.

   final: the option applies to this relay, only if the relay will
   transmit the containing element directly to the recipient.

all: the option applies to this relay and is retained for the
next.

Note that a final relay does not remove any options as it transmits
the containing element directly to the recipient.

The "mustUnderstand" attribute specifies whether the relay may ignore
the option if it is unrecognized, and is consulted only if the
"targetHop" attribute indicates that the option applies to that
relay.  If the option applies, and if the value of the
"mustUnderstand" attribute is "true", and if the relay does not
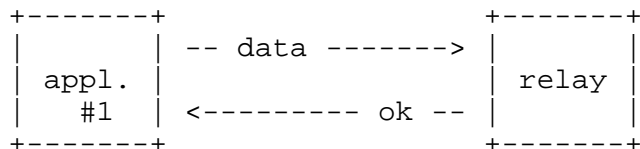"understand" the option, then an error in processing has occurred.

5.1 The statusRequest Option

Section 8.4 contains the APEX option registration for the
"statusRequest" option.

If this option is present, then each applicable relay sends a
"statusResponse" message to the originator.  This is done by issuing
a data operation whose originator is the report service associated
with the issuing relay, whose recipient is the endpoint address of
the "statusRequest" originator, and whose content is a
"statusResponse" element.

A "statusRequest" option MUST NOT be present in any data operation
containing a "statusResponse" element.  In general, applications
should be careful to avoid potential looping behaviors if an option
is received in error.

Consider these examples:

```
    +-------+                    +-------+
    |       | -- data ------->   |       |
    | appl. |                    | relay |
    |  #1   | <--------- ok --   |       |
    +-------+                    +-------+
```

```
  C: <data content='cid:1@example.com'>
         <originator identity='fred@example.com' />
         <recipient identity='barney@example.com' />
         <option internal='statusRequest' targetHop='final'
                 mustUnderstand='true' transID='86' />
     </data>
  S: <ok />
```

```
                             +-------+              +-------+
                             |       | -- data ------> |       |
                             | relay |                 | appl. |
                             |       | <--------- ok -- |   #2  |
                             +-------+              +-------+
```

```
      C: <data content='cid:1@example.com'>
             <originator identity='fred@example.com' />
             <recipient identity='barney@example.com' />
             <option internal='statusRequest' targetHop='final'
                     mustUnderstand='true' transID='86' />
         </data>
      S: <ok />
```
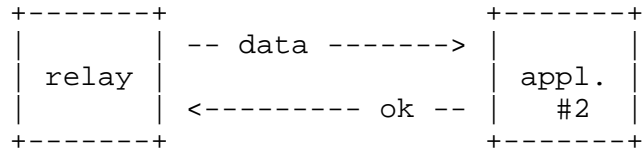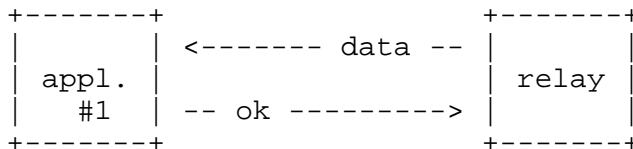
```
        +-------+              +-------+
        |       | <------- data -- |       |
        | appl. |                 | relay |
        |   #1  | -- ok ---------> |       |
        +-------+              +-------+
```

```
      C: <data content='#Content'>
             <originator identity='apex=report@example.com' />
             <recipient identity='fred@example.com' />
             <data-content Name='Content'>
                 <statusResponse transID='86'>
                     <destination identity='barney@example.com'>
                         <reply code='250' />
                     </destination>
                 </statusResponse>
             </data-content>
         </data>
      S: <ok />
```
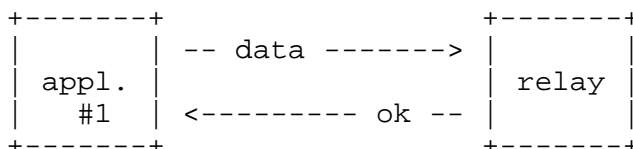
   or

```
        +-------+              +-------+
        |       | -- data ------> |       |
        | appl. |                 | relay |
        |   #1  | <--------- ok -- |       |
        +-------+              +-------+
```

```
      C: <data content='cid:1@example.com'>
             <originator identity='fred@example.com' />
             <recipient identity='barney@example.com' />
             <option internal='statusRequest' targetHop='final'
                     mustUnderstand='true' transID='86' />
         </data>
      S: <ok />
```
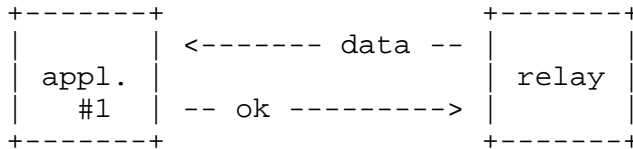
```
        +-------+                      +-------+
        |       | <------- data -- |       |
        | appl. |                      | relay |
        |   #1  | -- ok --------> |       |
        +-------+                      +-------+
```

```
C: <data content='#Content'>
       <originator identity='apex=report@example.com' />
       <recipient identity='fred@example.com' />
       <data-content Name='Content'>
           <statusResponse transID='86'>
               <destination identity='barney@example.com'>
                   <reply code='550'>unknown endpoint
                                     identity</reply>
               </destination>
           </statusResponse>
       </data-content>
   </data>
S: <ok />
```
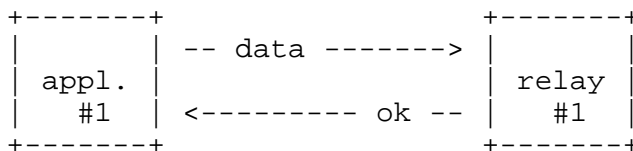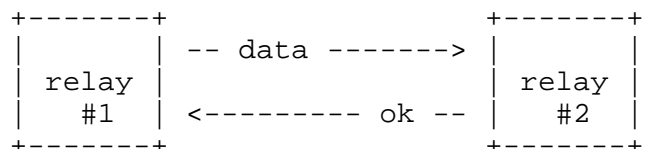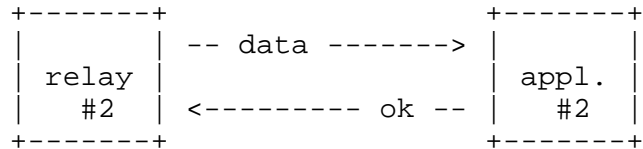
   or

```
        +-------+                      +-------+
        |       | -- data -------> |       |
        | appl. |                      | relay |
        |   #1  | <--------- ok -- |   #1  |
        +-------+                      +-------+
```

```
C: <data content='cid:1@example.com'>
       <originator identity='fred@example.com' />
       <recipient identity='barney@rubble.com' />
       <option internal='statusRequest' targetHop='final'
               mustUnderstand='true' transID='86' />
   </data>
S: <ok />
```

```
                                 +-------+            +-------+
                                 |       | -- data -------> |       |
                                 | relay |                  | relay |
                                 |   #1  | <--------- ok -- |   #2  |
                                 +-------+            +-------+
```

```
C: <data content='cid:1@example.com'>
       <originator identity='fred@example.com' />
       <recipient identity='barney@rubble.com' />
       <option internal='statusRequest' targetHop='final'
               mustUnderstand='true' transID='86' />
   </data>
S: <ok />
```

```
                                 +-------+              +-------+
                                 |       | -- data ------->  |       |
                                 | relay |              | appl. |
                                 |  #2   | <--------- ok -- |  #2   |
                                 +-------+              +-------+
```

```
      C: <data content='cid:1@example.com'>
              <originator identity='fred@example.com' />
              <recipient identity='barney@example.com' />
              <option internal='statusRequest' targetHop='final'
                      mustUnderstand='true' transID='86' />
         </data>
      S: <ok />
```
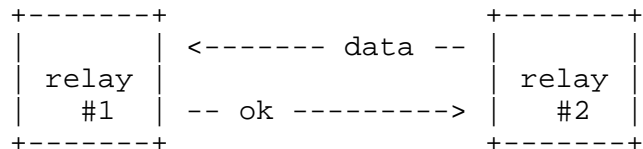
```
                                 +-------+              +-------+
                                 |       | <------- data -- |       |
                                 | relay |              | relay |
                                 |  #1   | -- ok ---------> |  #2   |
                                 +-------+              +-------+
```

```
      C: <data content='#Content'>
              <originator identity='apex=report@rubble.com' />
               <recipient identity='fred@example.com' />
               <data-content Name='Content'>
                  <statusResponse transID='86'>
                      <destination identity='barney@rubble.com'>
                          <reply code='250' />
                      </destination>
                  </statusResponse>
               </data-content>
          </data>
      S: <ok />
```
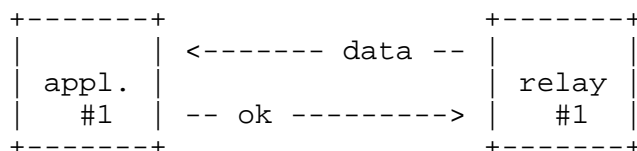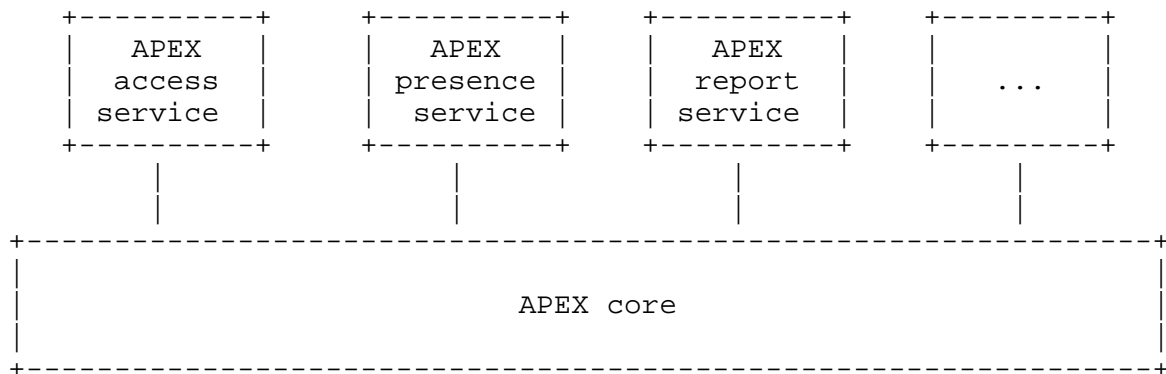
```
        +-------+              +-------+
        |       | <------- data -- |       |
        | appl. |              | relay |
        |  #1   | -- ok ---------> |  #1   |
        +-------+              +-------+
```

```
      C: <data content='#Content'>
              <originator identity='apex=report@rubble.com' />
              <recipient identity='fred@example.com' />
              <data-content Name='Content'>
                  <statusResponse transID='86'>
                      <destination identity='barney@rubble.com'>
                          <reply code='250' />
                      </destination>
                  </statusResponse>
```

```
            </data-content>
          </data>
      S: <ok />
```

   Note that a trace of a data's passage through the relaying mesh can
   be achieved by setting the "targetHop" attribute to "all".

6. APEX Services

   APEX, at its core, provides a best-effort datagram service. Within
   an administrative domain, all relays must be able to handle messages
   for any endpoint within that administrative domain. APEX services
   are logically defined as endpoints but, given their ubiquitous
   semantics, they do not necessarily need to be associated with a
   single physical endpoint. As such, they may be provisioned co-
   resident with each relay within an administrative domain, even though
   they are logically provided on top of the relaying mesh, i.e.,

```
       +---------+    +---------+    +---------+    +---------+
       |  APEX   |    |  APEX   |    |  APEX   |    |         |
       | access  |    | presence|    | report  |    |  ...    |
       | service |    | service |    | service |    |         |
       +---------+    +---------+    +---------+    +---------+
            |              |              |              |
            |              |              |              |
            |              |              |              |
     +------------------------------------------------------------+
     |                                                            |
     |                         APEX core                          |
     |                                                            |
     +------------------------------------------------------------+
```

   That is, applications communicate with an APEX service by exchanging
   data with a "well-known endpoint" (WKE).

   For example, APEX applications communicate with the report service by
   exchanging data with the well-known endpoint "apex=report" in the
   corresponding administrative domain, e.g., "apex=report@example.com"
   is the endpoint associated with the report service in the
   "example.com" administrative domain.

   The specification of an APEX service must define:

   o the WKE of the service;

   o the syntax and sequence of messages exchanged with the service;

   o what access control tokens are consulted by the service.

   A service registration template (Section 7.2) organizes this
   information.

   Finally, note that within a single administrative domain, the
   relaying mesh makes use of the APEX access service in order to
   determine if an originator is allowed to transmit data to a recipient
   (c.f., Step 5.3 of Section 4.4.4.1).

6.1 Use of the APEX Core DTD

   The specification of an APEX service may use definitions found in the
   APEX core DTD (Section 9.1). For example, the reply operation
   (Section 6.1.2) is defined to provide a common format for responses.

6.1.1 Transaction-Identifiers

   In using APEX's transaction-identifiers, note the following:

   o In the endpoint-relay and relay-relay modes, transaction-
      identifiers are meaningful only during the lifetime of a BEEP
      channel.

      For example, when an application issues the attach operation, the
      associated transaction-identifier has meaning only within the
      context of the BEEP channel used for the attach operation. When
      the BEEP connection is released, the channel no longer exists and
      the application is no longer attached to the relaying mesh.

   o In contrast, when an application communicates with an APEX
      service, transaction-identifiers are often embedded in the data
      that is sent. This means that transaction-identifiers are
      potentially long-lived.

      For example, an application may attach as an endpoint, send data
      (containing an embedded transaction-identifier) to a service, and,
      some time later, detach from the relaying mesh. Later on, a
      second application may attach as the same endpoint, and send data
      of its own (also containing embedded transaction-identifiers).
      Subsequently, the second application may receive data from the
      service responding to the first application's request and
      containing the transaction-identifier used by the first
      application.

   To minimize the likelihood of ambiguities with long-lived
   transaction-identifiers, the values of transaction-identifiers
   generated by applications should appear to be unpredictable.

6.1.2 The Reply Element

   Many APEX services make use of a reply operation. Although each
   service defines the circumstances in which a "reply" element is sent,
   the syntax of the "reply" element is defined in Section 9.1.

   The "reply" element has a "code" attribute, a "transID" attribute, an
   optional "xml:lang" attribute, and may contain arbitrary textual
   content:

   o   the "code" element specifies a three-digit reply code (c.f.,
       Section 10);

   o   the "transID" attribute specifies the transaction-identifier
       corresponding to this reply;

   o   the "xml:lang" attribute, if present, specifies the language that
       the element's content is written in; and,

   o   the textual content is a diagnostic (possibly multiline) which is
       meaningful to implementers, perhaps administrators, and possibly
       even users.

6.2 The Report Service

   Section 8.5 contains the APEX service registration for the report
   service:

   o   Within an administrative domain, the service is addressed using
       the well-known endpoint of "apex=report".

   o   Section 9.2 defines the syntax of the operations exchanged with
       the service.

   o   A consumer of the service does not initiate communications with
       the service.

   o   The service initiates communications by sending data containing
       the "statusResponse" operation.

   If a relay processes a "statusRequest" option (Section 5.1), then it
   sends data to the originator containing a "statusResponse" element
   (Section 9.2).

   The "statusResponse" element has a "transID" attribute and contains
   one or more "destination" elements:

   o the "transID" attribute specifies the value contained in the
      "statusRequest" option; and,

   o each "destination" element has an "identity" attribute and
      contains a "reply" element:

      * the "identity" attribute specifies the recipient endpoint that
         is being reported on; and,

      * the "reply" element (Section 6.1.2) specifies the delivery
         status of that recipient.

7. Registration Templates

7.1 APEX Option Registration Template

   When an APEX option is registered, the following information is
   supplied:

   Option Identification: specify the NMTOKEN or the URI that
      authoritatively identifies this option.

   Present in: specify the APEX elements in which the option may appear.

   Contains: specify the XML content that is contained within the
      "option" element.

   Processing Rules: specify the processing rules associated with the
      option.

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

7.2 APEX Service Registration Template

   When an APEX service is registered, the following information is
   supplied:

   Well-Known Endpoint: specify the local-part of an endpoint identity,
      starting with "apex=".

   Syntax of Messages Exchanged: specify the elements exchanged with the
      service.

   Sequence of Messages Exchanged: specify the order in which data is
      exchanged with the service.

   Access Control Tokens: specify the token(s) used to control access to
      the service (c.f., [10]).

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

   Note that the endpoints "apex=all" and "apex=core" may not be
   assigned.

7.3 APEX Endpoint Application Registration Template

   When an APEX endpoint application is registered, the following
   information is supplied:

   Endpoint Application: specify the subaddress used for an endpoint
      application, starting with "appl=".

   Application Definition: specify the syntax and semantics of the
      endpoint application identified by this registration.

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

8. Initial Registrations

8.1 Registration: The APEX Profile

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

   Messages exchanged during Channel Creation: "attach", "bind"

   Messages starting one-to-one exchanges: "attach", "bind",
      "terminate", or "data"

   Messages in positive replies: "ok"

   Messages in negative replies: "error"

   Messages in one-to-many exchanges: none

   Message Syntax: c.f., Section 9.1

   Message Semantics: c.f., Section 4.4

   Contact Information: c.f., the "Authors' Addresses" section of this
      memo

8.2 Registration: The System (Well-Known) TCP port number for apex-mesh

   Protocol Number: TCP

   Message Formats, Types, Opcodes, and Sequences: c.f., Section 9.1

   Functions: c.f., Section 4.4

   Use of Broadcast/Multicast: none

   Proposed Name: APEX relay-relay service

   Short name: apex-mesh

   Contact Information: c.f., the "Authors' Addresses" section of this
      memo

8.3 Registration: The System (Well-Known) TCP port number for apex-edge

   Protocol Number: TCP

   Message Formats, Types, Opcodes, and Sequences: c.f., Section 9.1

   Functions: c.f., Section 4.4

   Use of Broadcast/Multicast: none

   Proposed Name: APEX endpoint-relay service

   Short name: apex-edge

   Contact Information: c.f., the "Authors' Addresses" section of this
      memo

8.4 Registration: The statusRequest Option

   Option Identification: statusRequest

   Present in: APEX's "data" and "recipient" elements

   Contains: nothing

   Processing Rules: c.f., Section 5.1

   Contact Information: c.f., the "Authors' Addresses" section of this
      memo

8.5 Registration: The Report Service

   Well-Known Endpoint: apex=report

   Syntax of Messages Exchanged: c.f., Section 9.2

   Sequence of Messages Exchanged: c.f., Section 6.2

   Access Control Tokens: none

   Contact Information: c.f., the "Authors' Addresses" section of this
      memo

9. DTDs

9.1 The APEX Core DTD

   <!--
      DTD for the APEX core, as of 2001-07-09

      Refer to this DTD as:

        <!ENTITY % APEXCORE PUBLIC "-//IETF//DTD APEX CORE//EN" "">
        %APEXCORE;
      -->

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

   <!--
      DTD data types:

           entity   syntax/reference   example
           ======   ================   =======
      APEX endpoint
         ENDPOINT   entity,   fred@example.com
                           c.f., Section 2.2

      domain, either a FQDN or a literal
         DOMAIN   c.f., [RFC-2821]   example.com or [10.0.0.1]

      seconds
         SECONDS   0..2147483647   600

      timestamp
         TIMESTAMP   c.f., [12]   2000-05-15T13:02:00-08:00

      unique-identifier

```
           UNIQID        1..2147483647        42

      unique-identifier OR zero
           UNIZID        0..2147483647        0
     -->

   <!ENTITY  % ENDPOINT  "CDATA">
   <!ENTITY  % DOMAIN    "CDATA">
   <!ENTITY  % SECONDS   "CDATA">
   <!ENTITY  % TIMESTAMP "CDATA">
   <!ENTITY  % UNIQID    "CDATA">
   <!ENTITY  % UNIZID    "CDATA">

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

        role         MSG          RPY          ERR
       ======        ===          ===          ===
         I           attach       ok           error

      I or L         bind         ok           error

      I or L         terminate    ok           error

      I or L         data         ok           error
     -->

   <!ELEMENT attach       (option*)>
   <!ATTLIST attach
           endpoint     %ENDPOINT;          #REQUIRED
           transID      %UNIQID;            #REQUIRED>

   <!ELEMENT bind         (option*)>
   <!ATTLIST bind
           relay        %DOMAIN;            #REQUIRED
           transID      %UNIQID;            #REQUIRED>

   <!ELEMENT terminate    (#PCDATA)>
   <!ATTLIST terminate
           code         %XYZ;               "250"
           xml:lang     %LANG;              #IMPLIED
           transID      %UNIZID;            "0">

   <!ELEMENT data         (originator,recipient+,option*,data-content?)>
   <!ATTLIST data
           content      %URI;               #REQUIRED>

   <!ELEMENT originator  (option*)>
```

```
    <!ATTLIST originator
            identity    %ENDPOINT;         #REQUIRED>

    <!ELEMENT recipient    (option*)>
    <!ATTLIST recipient
            identity    %ENDPOINT;         #REQUIRED>

    <!ELEMENT data-content
                         ANY>
    <!ATTLIST Name         ID                #REQUIRED>

    <!ELEMENT ok          EMPTY>

    <!ELEMENT reply       (#PCDATA)>
    <!ATTLIST reply
            code        %XYZ;              #REQUIRED
            transID     %UNIQID;           #REQUIRED
            xml:lang    %LANG;             #IMPLIED>

    <!-- either the "internal" or the "external" attribute is present in
         an option -->

    <!ELEMENT option       ANY>
    <!ATTLIST option
            internal    NMTOKEN            ""
            external    %URI;              ""
            targetHop   (this|final|all)   "final"
            mustUnderstand
                         (true|false)       "false"
            transID     %UNIQID;           #REQUIRED
            localize    %LOCS;             "i-default">
```

9.2 The Report Service DTD

```
    <!--
      DTD for the APEX report service, as of 2000-12-12

      Refer to this DTD as:

        <!ENTITY % APEXREPORT PUBLIC "-//Blocks//DTD APEX REPORT//EN" "">
        %APEXREPORT;
      -->

    <!ENTITY % APEXCORE PUBLIC "-//Blocks//DTD APEX CORE//EN" "">
    %APEXCORE;

    <!--
      Synopsis of the APEX report service
```

```
      service WKE: apex=report

      message exchanges:

          service initiates    consumer replies
          =================    =================
          statusResponse       (nothing)

      access control tokens: none
    -->

  <!ELEMENT statusResponse
                    (destination+)>
  <!ATTLIST statusResponse
          transID    %UNIQID;          #REQUIRED>

  <!ELEMENT destination (reply)>
  <!ATTLIST destination
          identity   %ENDPOINT;        #REQUIRED>
```

10. Reply Codes

```
    code    meaning
    ====    =======
    250     transaction successful

    421     service not available

    450     requested action not taken

    451     requested action aborted

    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

    535     authentication failure

    537     action not authorized for user
```

          538   authentication mechanism requires encryption

          550   requested action not taken

          553   parameter invalid

          554   transaction failed (e.g., policy violation)

          555   transaction already in progress

11. Security Considerations

     Consult Section 3 and Section 4.5 for a discussion of security
     issues, e.g., relaying integrity.

     Although service provisioning is a policy matter, at a minimum, all
     APEX implementations must provide the following tuning profiles:

     for authentication: http://iana.org/beep/SASL/DIGEST-MD5

     for confidentiality: http://iana.org/beep/TLS (using the
        TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher)

     for both: http://iana.org/beep/TLS (using the
        TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher supporting client-side
        certificates)

     Further, APEX endpoint implementations may choose to offer MIME-based
     security services providing message integrity and confidentiality,
     such as OpenPGP [13] or S/MIME [14].

     Regardless, since APEX is a profile of the BEEP, consult [1]'s
     Section 9 for a discussion of BEEP-specific security issues.

     Finally, the statusRequest option (Section 5.1) may be used to expose
     private network topology. Accordingly, an administrator may wish to
     choose to disable this option except at the ingress/egress points for
     its administrative domain.

References

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

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

   [3]   Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April
          2001.

   [4]   Yergeau, F., "UTF-8, a transformation format of Unicode and ISO
          10646", RFC 2044, October 1996.

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

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

   [7]   Levinson, E., "The MIME Multipart/Related Content-type", RFC
          2387, August 1998.

   [8]   Levinson, E., "Content-ID and Message-ID Uniform Resource
          Locators", RFC 2392, August 1998.

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

   [10]   Rose, M., Klyne, G. and D. Crocker, "The Application Exchange
          (APEX) Access Service", RFC 3341, July 2002.

   [11]   Rose, M., Klyne, G. and D. Crocker, "The Application Exchange
          (APEX) Presence Service", Work in Progress.

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

   [13]   Elkins, M., Del Torto, D., Levien, R. and T. Roessler, "MIME
          Security with OpenPGP", RFC 3156, August 2001.

   [14]   Ramsdell, B., "S/MIME Version 3 Message Specification", RFC
          2633, June 1999.

Appendix A. Acknowledgements

   The authors gratefully acknowledge the contributions of: Jeffrey
   Altman, Harald Alvestrand, Eric Dixon, Ronan Klyne, Darren New, Chris
   Newman, Scott Pead, and Bob Wyman.

Appendix B. IANA Considerations

   The IANA has registered "APEX" as a standards-track BEEP profile, as
   specified in Section 8.1.

   The IANA has registered "apex-mesh" as a TCP port number, as
   specified in Section 8.2.

   The IANA has registered "apex-edge" as a TCP port number, as
   specified in Section 8.3.

   The IANA maintains a list of:

   o   APEX options, c.f., Section 7.1;

   o   APEX services, c.f., Section 7.2; and,

   o   APEX endpoint applications, c.f., Section 7.3.

   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 APEX
   options and services, the mailing list apexwg@invisible.net may be
   used to solicit commentary.

   The IANA makes the registrations specified in Section 8.4 and Section
   8.5.

Authors' Addresses

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

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

    Graham Klyne
    Clearswift Corporation
    1310 Waterside
    Arlington Business Park
    Theale, Reading RG7 4SA
    UK

    Phone: +44 11 8903 8903
    EMail: Graham.Klyne@MIMEsweeper.com

    David H. Crocker
    Brandenburg InternetWorking
    675 Spruce Drive
    Sunnyvale, CA 94086
    US

    Phone: +1 408 246 8253
    EMail: dcrocker@brandenburg.com
    URI: http://www.brandenburg.com/

Full Copyright Statement

Acknowledgement