The IMXP Access Service
draft-mrose-imxp-access-01

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 2, 2001.

Copyright Notice

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

Abstract

This memo describes the IMXP access service, addressed as the well-known endpoint "imxp=access". The access service is used to control use of both the IMXP "relaying mesh" and other IMXP services.
Table of Contents

1. Introduction .................................................. 3
2. Management of Access Information ................................. 4
  2.1 Retrieval of Access Information ............................... 5
  2.2 Update of Access Information .................................. 6
3. Format of Access Entries ......................................... 7
4. The Access Service ................................................ 9
  4.1 Use of XML and MIME .......................................... 10
  4.2 The Get Operation ............................................. 11
  4.3 The Set Operation ............................................. 12
  4.4 The Reply Operation ........................................... 14
5. Registration: The Access Service ................................. 15
6. The Access Service DTD ........................................... 16
   References ..................................................... 19
   Authors’ Addresses .............................................. 19
A. Acknowledgements ............................................... 20
B. Changes from draft-mrose-imxp-access-00 .......................... 21
   Full Copyright Statement ....................................... 22
1. Introduction

This memo describes an access service that is built upon the IMXP[1] "relaying mesh", which, in turn, is specified as a BEEP[2] profile.

IMXP at its core, provides a best-effort datagram service. With the exception of a co-resident IMXP report service (used for error reporting), all other IMXP services are provided on top of IMXP’s "relaying mesh", e.g.,

```
+----------+     +----------+    +----------+
|   IMXP   |     |   IMXP   |    |          |
|  access  |     | presence  |    |   ...    |
| service  |     | service   |    |          |
+----------+     +----------+    +----------+
```

Applications communicate with IMXP services by sending data to a "well-known endpoint" (WKE).

The IMXP access service is used to control use of both the relaying mesh and other IMXP services. Although the access service is logically layered above the IMXP core, implementers may choose to physically co-reside the access service with IMXP core software.

IMXP applications communicate with the access service by exchanging data with the well-known endpoint "imxp=access" in the corresponding administrative domain, e.g., "imxp=access@example.com" is the endpoint associated with the access service in the "example.com" administrative domain.

Note that within a single administrative domain, the relaying mesh makes use of the IMXP access service in order to determine if an originator is allowed to transmit data to a recipient (c.f., Step 3.3 of Section 4.4.3.1 of [1]).
2. Management of Access Information

Management of access information falls into two categories:

- applications may retrieve the access entry associated with an endpoint; and,
- applications may modify the access entry associated with an endpoint.

Each is now described in turn.
2.1 Retrieval of Access Information

When an application wants to retrieve the access entry associated with an endpoint, it sends a "get" element to the service, e.g.,

```
+-------+                  +-------+
|       | -- data -------> |       |
| appl. |                  | relay |
+-------+                  +-------+
```

C: <data originator='fred@example.com' content='#Content'>
   <recipient identity='imxp=access@example.com' />
   <data-content Name='Content'>
      <get owner='fred@example.com' transID='1' />
   </data-content>
</data>
S: <ok />

The service immediately responds with a set operation containing the access entry and the same transaction-identifier, e.g.,

```
+-------+                  +-------+
| relay | <------- data ------| access |
|       | -- ok ---------> |       |
+-------+                  +-------+
```

C: <data originator='imxp=access@example.com' content='#Content'>
   <recipient identity='fred@example.com' />
   <data-content Name='Content'>
      <set owner='fred@example.com' transID='1'
      timeStamp='14 May 2000 13:32:00 -0800'>
      <access owner='fred@example.com'
      lastUpdate='14 May 2000 13:02:00 -0800'>
      <entry actor='*@example.com'
      actions='core:data presence:subscribe' />
      <entry actor='*@*' actions='core:data' />
   </access>
   </set>
</data-content>
</data>
S: <ok />
2.2 Update of Access Information

When an application wants to modify the access entry associated with an endpoint, it sends a "set" element to the service, e.g.,

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

C: <data originator='fred@example.com' content='#Content'>
   <recipient identity='imxp=access@example.com' />
   <data-content Name='Content'>
       <set owner='fred@example.com' transID='1'
            timeStamp='14 May 2000 13:32:00 -0800'> ... </set>
   </data-content>
</data>
S: <ok />

The service immediately responds with a reply operation containing the same transaction-identifier, e.g.,

```
+-------+                  +-------+
|       | <------- data -- |       |
| relay |                  |access |
|       | -- ok ---------> |  svc. |
+-------+                  +-------+
```

C: <data originator='imxp=access@example.com' content='#Content'>
   <recipient identity='fred@example.com' />
   <data-content Name='Content'>
       <reply code='250' transID='1' />
   </data-content>
</data>
S: <ok />
3. Format of Access Entries

Each administrative domain is responsible for maintaining an "access entry" for each of its endpoints (regardless of whether those endpoints are currently attached to the relaying mesh).

Section 6 defines the syntax for access entries. Each access entry has an "owner" attribute, a "lastUpdate" attribute, and contains one or more "entry" elements:

- the "owner" attribute specifies the endpoint associated with the access entry;
- the "lastUpdate" attribute specifies the date and time that the service last updated the access entry; and,
- each "entry" element specifies, with respect to the owner’s endpoint, an actor and zero or more allowed actions for that actor.

Within an entry, actions are specified as service/operation pairs, (e.g., "presence:publish" refers to the "publish" operation of the "presence" service). To refer to all services and/or all operations, the reserved value "all" is used (e.g., "all:data", "presence:all", and so on). Note that the service specified as "core" is reserved for use by the relaying mesh, e.g., the "core:data" action is consulted by the relaying mesh (c.f., Step 3.3 of Section 4.4.3.1 of [1]).

An actor is an IMXP endpoint and is specified using the "addr-spec" syntax of RFC 822[3], i.e., "local@domain". However, both the "local" and "domain" parts may contain limited wildcarding:

- The "local" part is either:
  * a literal string (e.g., "fred"); or,
  * the value "imxp=*", specifying all IMXP services; or,
  * the value "*", specifying any endpoint other than an IMXP service.

- The "domain" part is either:
  * a FQDN (e.g., "example.com"); or,
  * the value "*", specifying all administrative domains.
Regardless of the "entry" elements present in an access entry, four additional elements are always considered to exist at the end of the access entry:

- `<entry actor='imxp=*@domain' actions='all:all' />
- `<entry actor='imxp=*@*' actions='core:data' />
- `<entry actor='local@domain' actions='all:all' />
- `<entry actor='*@*' actions='' />

where "local@domain" specifies the endpoint associated with the access entry.

Ordering of "entry" elements within an access element is significant: a process examining an access element selects the first "entry" element that matches the actor in question. For example, consider this access entry:

- `<access owner='fred@example.com' lastUpdate='14 May 2000 13:02:00 -0800'>
  - `<entry actor='wilma@example.com' actions='all:all' />
  - `<entry actor='mr.slate@example.com' actions='core:data' />
  - `<entry actor='@example.com' actions='core:data presence:subscribe presence:watch' />
  - `<entry actor='*@*' actions='core:data' />
</access>

Briefly:

- For endpoints within the "example.com" administrative domain:
  - "fred", "wilma", and all IMXP services, are allowed access to all operations for all IMXP services;
  - "mr.slate" is allowed access only to send data through the relaying mesh; and,
  - any other endpoint is allowed access to send data and invoke the "subscribe" and "watch" operations of the IMXP presence service.

- For any endpoint outside the "example.com" administrative domain, the endpoint is allowed access to send data, regardless of whether it is an IMXP service.

Note that although the four additional elements are always present, the ordering semantics cause the final element to be unused.
4. The Access Service

Section 5 contains the IMXP service registration for the access service:

- Within an administrative domain, the service is addressed using the well-known endpoint of "imxp=access".

- Section 6 defines the syntax of the operations exchanged with the service.

- A consumer of the service initiates communications by sending data containing either the get or set operation.

- The service replies to these operations, and does not initiate communications.

An implementation of the service must maintain information about access entries in persistent storage.

Consult Section 6.1.1 of [1] for a discussion on the properties of long-lived transaction-identifiers.
4.1 Use of XML and MIME

Section 4.1 of [1] describes how arbitrary MIME content is exchanged as a BEEP payload. For example, to transmit:

```
<data originator='fred@example.com' content='...'>
  <recipient identity='imxp=access@example.com' />
</data>
```

where "..." refers to:

```
<get owner='fred@example.com' transID='1' />
```

then the corresponding BEEP operation might look like this:

```
C: MSG 1 2 . 42 1234
C: Content-Type: multipart/related; boundary="boundary";
  start="<1@example.com>";
  type="text/xml"
C: --boundary
C: Content-Type: text/xml
C: Content-ID: <1@example.com>
C: <data originator='fred@example.com'
  content='cid:2@example.com'>
C:   <recipient identity='imxp=access@example.com' />
C: </data>
C: --boundary
C: Content-Type: text/xml
C: Content-Transfer-Encoding: binary
C: Content-ID: <2@example.com>
C: <get owner='fred@example.com' transID='1' />
C: --boundary--
C: END
```

or this:

```
C: MSG 1 1 . 42 255
C: Content-Type: text/xml
C: <data originator='fred@example.com' content='#Content'>
  <recipient identity='imxp=access@example.com' />
  <data-content Name='Content'>
    <get owner='fred@example.com' transID='1' />
  </data-content>
</data>
C: END
```
4.2 The Get Operation

When an application wants to retrieve the access entry associated with an endpoint, it sends a "get" element to the service.

The "get" element has an "owner" attribute, a "transID" attribute, and no content:

- the "owner" attribute specifies the endpoint associated with the access entry; and,
- the "transID" attribute specifies the transaction-identifier associated with this operation.

When the service receives a "get" element, we refer to the "owner" attribute of that element as the "subject", and the service performs these steps:

1. If the subject is outside of this administrative domain, a "reply" element having code 553 is sent as data to the originator.

2. If the subject does not refer to a valid endpoint, a "reply" element having code 550 is sent as data to the originator.

3. If the subject’s access entry does not contain a "access:get" token for the originator, a "reply" element having code 537 is sent as data to the originator.

4. Otherwise, a "set" element, corresponding to the subject’s access entry, is sent as data to the originator.

Regardless of whether a "set" or "reply" element is sent to the originator, the "transID" attribute is identical to the value found in the "get" element sent by the originator.
4.3 The Set Operation

When an application wants to modify the access entry associated with an endpoint, it sends a "set" element to the service.

The "set" element has an "owner" attribute, a "transID" attribute, and contains an "access" element:

- the "owner" attribute specifies the endpoint to be associated with the access entry;

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

- the "timeStamp" attribute specifies the current date and time; and,

- the "access" element contains the desired access entry for the endpoint.
When the service receives a "set" element, we refer to the "owner" attribute of that element as the "subject", and the service performs these steps:

1. If the "owner" attribute of the "set" element doesn’t match the "owner" attribute of the "access" element contained in the "set" element, a "reply" element having code 503 is sent as data to the originator.

2. If the subject is outside of this administrative domain, a "reply" element having code 553 is sent as data to the originator.

3. If the subject does not refer to a valid endpoint, a "reply" element having code 550 is sent as data to the originator.

4. If the subject’s access entry does not contain a "access:set" token for the originator, a "reply" element having code 537 is sent as data to the originator.

5. If the "lastUpdate" attribute of the "set" element is not semantically identical to the last update time of the subject’s access entry, a "reply" element having code 555 is sent as data to the originator. (This allows a basic mechanism for atomic updates.)

6. Otherwise:
   1. The subject’s access entry is updated from the "set" element.
   2. The last update time of the access entry is set to the current time.
   3. A "reply" element having code 250 is sent as data to the originator.

When sending the "reply" element, the "transID" attribute is identical to the value found in the "set" element sent by the originator.
4.4 The Reply Operation

While processing operations, the service may respond with a "reply" element. Consult Sections 10.2 and 6.1.2 of [1], respectively, for the syntax and semantics of the reply operation.
5. Registration: The Access Service

Well-Known Endpoint: imxp=access

Syntax of Messages Exchanged: c.f., Section 6

Sequence of Messages Exchanged: c.f., Section 4

Access Control Tokens: access:get, access:set

Contact Information: c.f., the "Authors’ Addresses" section of this memo
6. The Access Service DTD

<!--
DTD for the IMXP access service, as of 2000-09-19

Refer to this DTD as:

<!ENTITY % IMXPACCESS PUBLIC "-//Blocks//DTD IMXP ACCESS//EN"
  "http://xml.resource.org/profiles/IMXP/imxp-access.dtd">
%IMXPACCESS;
-->

<!ENTITY % IMXPCORE PUBLIC "-//Blocks//DTD IMXP CORE//EN"
  "http://xml.resource.org/profiles/IMXP/imxp-core.dtd">
%IMXPCORE;

<!--
DTD data types:

<table>
<thead>
<tr>
<th>entity</th>
<th>syntax/reference</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>access actor</td>
<td>an ENDPOINT or a wildcard</td>
<td>*@example.com</td>
</tr>
</tbody>
</table>
-->

<!ENTITY % ACTOR "CDATA">
Synopsis of the IMXP access service

service WKE: imxp=access

message exchanges:

<table>
<thead>
<tr>
<th>consumer initiates</th>
<th>service replies</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>set or reply</td>
</tr>
<tr>
<td>set</td>
<td>reply</td>
</tr>
</tbody>
</table>

access control:

<table>
<thead>
<tr>
<th>token</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>access:get for &quot;owner&quot; of &quot;access&quot; element</td>
<td></td>
</tr>
<tr>
<td>access:set for &quot;owner&quot; of &quot;access&quot; element</td>
<td></td>
</tr>
</tbody>
</table>

<!ELEMENT get         EMPTY>
<!ATTLIST get
  owner       %ENDPOINT;        #REQUIRED
  transID     %UNIQID;          #REQUIRED>

<!ELEMENT set         (access)>  
<!ATTLIST set
  owner       %ENDPOINT;        #REQUIRED
  transID     %UNIQID;          #REQUIRED
  timeStamp   %TIMESTAMP;       #REQUIRED>

<!ELEMENT access      (entry+)>
<!ATTLIST access
  owner       %ENDPOINT;        #REQUIRED
  lastUpdate  %TIMESTAMP;       #REQUIRED>

<!ELEMENT entry       EMPTY>
<!ATTLIST entry
  actor       %ACTOR;           #REQUIRED
  actions     NMTOKENS          "">
References


Authors’ Addresses

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/

Graham Klyne
Content Technologies Limited
1220 Parkview
Arlington Business Park
Theale, Reading RG7 4SA
UK
Phone: +44 118 930 1300
EMail: gk@acm.org

David H. Crocker
Brandenburg Consulting
675 Spruce Drive
Sunnyvale, CA 94086
US
Phone: +1 408 246 8253
EMail: dcrocker@brandenburg.com
URI: http://www.brandenburg.com/
Appendix A. Acknowledgements

The authors gratefully acknowledge the contributions of: Darren New and Scott Pead.
Appendix B. Changes from draft-mrose-imxp-access-00

- Updated to reflect the current BEEP framework[2].
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.