established. This state contains the tag from the To field in the SUBSCRIBE response, and the Route header set computed from the Record-Routes and Contact headers in the 200 class response. The subscription is indexed by the presentity identification (the To field of the SUBSCRIBE that was generated).

If an unsubscribe request is received from the CPIM side, the gateway checks if the subscription exists. If it does, a SUBSCRIBE is generated as described above. However, the Expires header is set to zero. If the subscription does not exist, the gateway generates a failure response and sends it to the CPIM system. When the response to the SUBSCRIBE request arrives, it is converted to a CPIM response as described above for the initial SUBSCRIBE response. In all cases, any subscription state in the gateway is destroyed.

When a NOTIFY is received from the SIP system, a CPIM notification request is sent. This notification is constructed as follows:

- o The CPIM watcher is set to the URI in the To field of the NOTIFY.
- o The CPIM target is set to the URI in the From field of the NOTIFY.
- o The transID is computed using the same mechanism as for the SUBSCRIBE in Section 7.1
- o The presence component of the notification is extracted from the body of the SIP NOTIFY request.

The gateway generates a 200 response to the SIP NOTIFY and sends it as well.

TODO: some call flow diagrams with the parameters

8 Firewall and NAT Traversal

It is anticipated that presence services will be used by clients and presentities that are connected to proxy servers on the other side of firewalls and NATs. Fortunately, since the SIP presence messages do not establish independent media streams, as INVITE does, firewall and NAT traversal is much simpler than described in [9] and [10].

Generally, data traverses NATs and firewalls when it is sent over TCP or TLS connections established by devices inside the firewall/NAT to devices outside of it. As a result, it is RECOMMENDED that SIP for presence entities maintain persistent TCP or TLS connections to their next hop peers. This includes connections opened to send a SUBSCRIBE,

NOTIFY, and most importantly, REGISTER. By keeping the latter connection open, it can be used by the SIP proxy to send messages from outside the firewall/NAT back to the client. It is also recommended that the client include a Contact cookie as described in [10] in their registration, so that the proxy can map the presentity URI to that connection.

Furthermore, entities on either side of a firewall or NAT should record-route in order to ensure that the initial connection established for the subscription is used for the notifications as well.

9 Security considerations

There are numerous security considerations for presence. Many are outlined above; this section considers them issue by issue.

9.1 Privacy

Privacy encompasses many aspects of a presence system:

- o Subscribers may not want to reveal the fact that they have subscribed to certain users
- o Users may not want to reveal that they have accepted subscriptions from certain users
- o Notifications (and fetch results) may contain sensitive data which should not be revealed to anyone but the subscriber

Privacy is provided through a combination of hop by hop encryption and end to end encryption. The hop by hop mechanisms provide scalable privacy services, disable attacks involving traffic analysis, and hide all aspects of presence messages. However, they operate based on transitivity of trust, and they cause message content to be revealed to proxies. The end-to-end mechanisms do not require transitivity of trust, and reveal information only to the desired recipient. However, end-to-end encryption cannot hide all information, and is susceptible to traffic analysis. Strong end to end authentication and encryption also requires that both participants have public keys, which is not generally the case. Thus, both mechanisms combined are needed for complete privacy services.

SIP allows any hop by hop encryption scheme. It is RECOMMENDED that between network servers (proxies to proxies, proxies to redirect servers), transport mode ESP [11] is used to encrypt the entire message. Between a UAC and its local proxy, TLS [12] is RECOMMENDED. Similarly, TLS SHOULD be used between a presence server and the PUA.

The presence server can determine whether TLS is supported by the receiving client based on the transport parameter in the Contact header of its registration. If that registration contains the token "tls" as transport, it implies that the PUA supports TLS.

Furthermore, we allow for the Contact header in the SUBSCRIBE request to contain TLS as a transport. The Contact header is used to route subsequent messages between a pair of entities. It defines the address and transport used to communicate with the user agent. Even though TLS might be used between the subscriber and its local proxy, placing this parameter in the Contact header means that TLS can also be used end to end for generation of notifications after the initial SUBSCRIBE message has been successfully routed. This would provide end to end privacy and authentication services with low proxy overheads.

SIP encryption MAY be used end to end for the transmission of both SUBSCRIBE and NOTIFY requests. SIP supports PGP based encryption, which does not require the establishment of a session key for encryption of messages within a given subscription (basically, a new session key is established for each message as part of the PGP encryption). Work has recently begun on the application of S/MIME [13] for SIP.

9.2 Message integrity and authenticity

It is important for the message recipient to ensure that the message contents are actually what was sent by the originator, and that the recipient of the message be able to determine who the originator really is. This applies to both requests and responses of SUBSCRIBE and NOTIFY. This is supported in SIP through end to end authentication and message integrity. SIP provides PGP based authentication and integrity (both challenge-response and public key signatures), and http basic and digest authentication. HTTP Basic is NOT RECOMMENDED.

9.3 Outbound authentication

When local proxies are used for transmission of outbound messages, proxy authentication is RECOMMENDED. This is useful to verify the identity of the originator, and prevent spoofing and spamming at the originating network.

9.4 Replay prevention

To prevent the replay of old subscriptions and notifications, all signed SUBSCRIBE and NOTIFY requests and responses MUST contain a Date header covered by the message signature. Any message with a date

older than several minutes in the past, or more than several minutes into the future, SHOULD be discarded.

Furthermore, all signed SUBSCRIBE and NOTIFY requests MUST contain a Call-ID and CSeq header covered by the message signature. A user agent or presence server MAY store a list of Call-ID values, and for each, the higest CSeq seen within that Call-ID. Any message that arrives for a Call-ID that exists, whose CSeq is lower than the

highest seen so far, is discarded.

Finally, challenge-response authentication (http digest or PGP) MAY be used to prevent replay attacks.

9.5 Denial of service attacks

Denial of service attacks are a critical problem for an open, inter-domain, presence protocol. Here, we discuss several possible attacks, and the steps we have taken to prevent them.

9.5.1 Smurf attacks through false contacts

Unfortunately, presence is a good candidate for smurfing attacks because of its amplification properties. A single SUBSCRIBE message could generate a nearly unending stream of notifications, so long as a suitably dynamic source of presence data can be found. Thus, a simple way to launch an attack is to send subscriptions to a large number of users, and in the Contact header (which is where notifications are sent), place the address of the target.

The only reliable way to prevent these attacks is through authentication and authorization. End users will hopefully not accept subscriptions from random unrecognized users. Also, the presence client software could be programmed to warn the user when the Contact header in a SUBSCRIBE is from a domain which does not match that of the From field (which identifies the subscriber).

Also, note that as described in [3], if a NOTIFY is not acknowledged or was not wanted, the subscription that generated it is removed. This eliminates the amplification properties of providing false Contact addresses.

10 Example message flows

The following subsections exhibit example message flows, to further clarify behavior of the protocol.

10.1 Client to Client Subscription with Presentity State Changes

This call flow illustrates subscriptions and notifications that do not involve a presence server.

The watcher subscribes to the presentity, and the subscription is accepted, resulting in a 202 Accepted response. The presentity subsequently changes state (is on the phone), resulting in a new notification. The flow finishes with the watcher canceling the subscription.

```
Watcher                          Presentity
-------                          -----------
   |      F1 SUBSCRIBE                |
   | ------------------------------>|
   |      F2 202 Accepted             |
   |<------------------------------|
   |      F3 NOTIFY                   |
   |<------------------------------|
   |      F4 200 OK                   |
   |------------------------------>|
   |      F5 NOTIFY                   |
   |<------------------------------|
   |      F6 200 OK                   |
   |------------------------------>|
   |      F7 SUBSCRIBE (unsub)        |
   |------------------------------>|
   |      F8 202 Accepted             |
   |<------------------------------|
```

Message Details

F1 SUBSCRIBE watcher -> presentity

```
SUBSCRIBE sip:presentity@pres.example.com SIP/2.0
Via: SIP/2.0/UDP  watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:presentity@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq : 1 SUBSCRIBE
Expires: 600
Accept: application/xpidf+xml
Event: presence
Contact: sip:user@watcherhost.example.com
```


F2 202 Accepted presentity->watcher

```
SIP/2.0 202 Accepted
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:presentity@example.com>;tag=88a7s
Call-ID: 3248543@watcherhost.example.com
Cseq: 1 SUBSCRIBE
Event: presence
Expires: 600
Contact: sip:presentity@pres.example.com
```

F3 NOTIFY Presentity->watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/UDP pres.example.com:5060
From: Resource <pres:presentity@example.com>;tag=88a7s
To: User <pres:user@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq: 1 NOTIFY
Event: presence
Content-Type: application/xpidf+xml
Content-Length: 120

<?xml version="1.0"?>
<presence entityInfo="pres:presentity@example.com">
  <tuple destination="sip:presentity@example.com" status="open"/>
</presence>
```

F4 200 OK watcher->presentity

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pres.example.com:5060
From: Resource <pres:presentity@example.com>
To: User <pres:user@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq: 1 NOTIFY
```


F5 NOTIFY Presentity->watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/UDP pres.example.com:5060
From: Resource <pres:presentity@example.com>
To: User <pres:user@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq: 2 NOTIFY
Event: presence
Content-Type: application/xpidf+xml
Content-Length: 120

<?xml version="1.0"?>
<presence entityInfo="pres:presentity@example.com">
  <tuple destination="sip:presentity@example.com" status="closed"/>
</presence>
```

F6 200 OK watcher->presentity

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pres.example.com:5060
From: Resource <pres:presentity@example.com>
To: User <pres:user@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq: 2 NOTIFY
```

F7 SUBSCRIBE watcher -> presentity

```
SUBSCRIBE sip:presentity@pres.example.com SIP/2.0
Via: SIP/2.0/UDP  watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:presentity@example.com>
Call-ID: 3248543@watcherhost.example.com
Event: presence
CSeq : 2 SUBSCRIBE
Expires: 0
Accept: application/xpidf+xml
Contact: sip:user@watcherhost.example.com
```


F8 202 Accepted presentity->watcher

```
SIP/2.0 202 Accepted
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:presentity@example.com>
Call-ID: 3248543@watcherhost.example.com
Event: presence
Cseq: 2 SUBSCRIBE
Expires:0
```

10.2 Presence Server with Client Notifications

This call flow shows the involvement of a presence server in the handling of subscriptions. In this scenario, the client has indicated that it will handle subscriptions and thus notifications. The message flow shows a change of presence state by the client and a cancellation of the subscription by the watcher.

```
                                Presence
        Watcher                  Server                    PUA
            |                       |  F1 REGISTER          |
            |                       |<----------------------|
            |                       |  F2 200 OK            |
            |                       |---------------------->|
            |  F3 SUBSCRIBE         |                       |
            |--------------------->|                       |
            |                       |  F4 SUBSCRIBE         |
            |                       |---------------------->|
            |                       |  F5 202               |
            |                       |<----------------------|
            |  F6 202               |                       |
            |<---------------------|                       |
            |  F7 NOTIFY            |                       |
            |<--------------------------------------------+
            |  F8  200 OK           |                       |
            |--------------------------------------------->|
            |                       |  F9 REGISTER          |
            |                       |<----------------------|
            |                       |  F10 200 OK           |
            |                       |---------------------->|
            |  F11 NOTIFY           |                       |
```



```
            |<--------------------------------------------+
            |  F12 200 OK           |                       |
            |--------------------------------------------->|
```

Message Details

F1 REGISTER PUA->server

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:resource@example.com>
From: <sip:resource@example.com>
Call-ID: 2818@pua.example.com
CSeq: 1 REGISTER
Contact: <sip:id@pua.example.com>;methods="MESSAGE"
          ;description="open"
Contact: <sip:id@pua.example.com>;methods="SUBSCRIBE"
Expires: 600
```

F2 200 OK .server->PUA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:resource@example.com>
From: <sip:resource@example.com>
Call-ID: 2818@pua.example.com
CSeq: 1 REGISTER
Contact: <sip:id@pua.example.com>;methods="MESSAGE"
          ;description="open"
Contact: <sip:id@pua.example.com>;methods="SUBSCRIBE"
Expires: 600
```

F3 SUBSCRIBE watcher->server


```
SUBSCRIBE sip:resource@example.com SIP/2.0
Via: SIP/2.0/UDP  watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:resource@example.com>
Call-ID: 32485@watcherhost.example.com
CSeq : 1 SUBSCRIBE
Expires: 600
Event: presence
Accept: application/xpidf+xml
Contact: sip:user@watcherhost.example.com
```

F4 SUBSCRIBE server->PUA

```
SUBSCRIBE sip:id@pua.example.com SIP/2.0
Via: SIP/2.0/UDP server.example.com:5060
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:resource@example.com>
Call-ID: 32485@watcherhost.example.com
CSeq : 1 SUBSCRIBE
Event: presence
Expires: 600
Accept: application/xpidf+xml
Contact: sip:user@watcherhost.example.com
```

F5 202 Accepted PUA->server

```
SIP/2.0 202 Accepted
Via: SIP/2.0/UDP server.example.com:5060
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:resource@example.com>;tag=ffd2
Call-ID: 32485@watcherhost.example.com
CSeq : 1 SUBSCRIBE
Event: presence
Expires: 600
```


F6 200 OK server->watcher

```
SIP/2.0 202 Accepted
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: User <pres:user@example.com>
To: Resource <pres:resource@example.com>;tag=ffd2
Call-ID: 32485@watcherhost.example.com
CSeq : 1 SUBSCRIBE
Event: presence
Expires: 600
```

F7 NOTIFY PUA->watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/UDP pua.example.com:5060
To: User <pres:user@example.com>
From: Resource <pres:resource@example.com>;tag=ffd2
Call-ID: 32485@watcherhost.example.com
CSeq : 1 NOTIFY
Event: presence
Content-Type: application/xpidf+xml
Content-Length: 120

<?xml version="1.0"?>
<presence entityInfo="pres:resource@example.com">
  <tuple destination="im:resource@example.com" status="open"/>
</presence>
```

F8 200 OK watcher->PUA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pua.example.com:5060
To: User <pres:user@example.com>
From: Resource <pres:resource@example.com>;tag=ffd2
Call-ID: 32485@watcherhost.example.com
CSeq : 1 NOTIFY
```


F9 REGISTER PUA->server

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:resource@example.com>
From: <sip:resource@example.com>
Call-ID: 2818@pua.example.com
CSeq: 2 REGISTER
Contact: <sip:id@pua.example.com>;methods="MESSAGE"
         ;description="busy"
Contact: <sip:id@pua.example.com>;methods="SUBSCRIBE"
Expires: 600
```

F10 200 OK server->PUA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:resource@example.com>
From: <sip:resource@example.com>
Call-ID: 2818@pua.example.com
CSeq: 2 REGISTER
Contact: <sip:id@pua.example.com>;methods="MESSAGE"
         ;description="busy"
Contact: <sip:id@pua.example.com>;methods="SUBSCRIBE"
Expires: 600
```

F11 NOTIFY PUA->watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
```

```
Via: SIP/2.0/UDP pua.example.com:5060
To: User <pres:user@example.com>
From: Resource <pres:resource@example.com>;tag=ffd2
Call-ID: 32485@watcherhost.example.com
CSeq : 2 NOTIFY
Event: presence
Content-Type: application/xpidf+xml
Content-Length: 120
```



```
<?xml version="1.0"?>
<presence entityInfo="pres:resource@example.com">
  <tuple destination="im:resource@example.com" status="busy"/>
</presence>
```

F12 200 OK watcher->PUA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pua.example.com:5060
To: User <pres:user@example.com>
From: Resource <pres:resource@example.com>;tag=ffd2
Call-ID: 32485@watcherhost.example.com
CSeq : 2 NOTIFY
```

10.3 Presence Server Notifications

This message flow illustrates how the presence server can be the responsible for sending notifications for a presentity. The presence server will do this if the presentity has not sent a registration indicating an interest in handling subscriptions. This flow assumes that the watcher has previously been authorized to subscribe to this resource at the server.

```
Watcher              Server                  PUA
    | F1 SUBSCRIBE      |                      |
    |------------------>|                      |
    | F2 202 Accepted   |                      |
    |<------------------|                      |
    | F3 NOTIFY         |                      |
    |<------------------|                      |
    | F4 200 OK         |                      |
    |------------------>|                      |
```

```
|                    |  F5 REGISTER        |
|                    |<-------------------|
|                    |  F6 200 OK          |
|                    |------------------->|
```



```
| F7 NOTIFY          |                    |
|<------------------|                    |
| F8 200 OK          |                    |
|------------------->|                    |
```

Message Details

F1 SUBSCRIBE watcher->server

```
SUBSCRIBE sip:resource@example.com SIP/2.0
Via: SIP/2.0/UDP watcherhost.example.com:5060
To: <pres:resource@example.com>
From: <pres:user@example.com>
Call-ID: 2010@watcherhost.example.com
CSeq: 1 SUBSCRIBE
Event: presence
Accept: application/xpidf+xml
Contact: <sip:user@watcherhost.example.com>
Expires: 600
```

F2 202 OK server->watcher

```
SIP/2.0 202 Accepted
Via: SIP/2.0/UDP watcherhost.example.com:5060
To: <pres:resource@example.com>;tag=ffd2
From: <pres:user@example.com>
Call-ID: 2010@watcherhost.example.com
CSeq: 1 SUBSCRIBE
Event: presence
Expires: 600
Contact: sip:example.com
```

F3 NOTIFY server-> watcher

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/UDP server.example.com:5060
```

```
From: <pres:resource@example.com>;tag=ffd2
To: <pres:user@example.com>
Call-ID: 2010@watcherhost.example.com
Event: presence
CSeq: 1 NOTIFY
Content-Type: application/xpidf+xml
Content-Length: 120

<?xml version="1.0"?>
<presence entityInfo="pres:resource@example.com">
  <tuple destination="im:resource@example.com" status="open"/>
</presence>
```

F4 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server.example.com:5060
From: <pres:resource@example.com>;tag=ffd2
To: <pres:user@example.com>
Call-ID: 2010@watcherhost.example.com
CSeq: 1 NOTIFY
```

F5 REGISTER

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:resource@example.com>
From: <sip:resource@example.com>
Call-ID: 110@pua.example.com
CSeq: 2 REGISTER
Contact: <sip:id@pua.example.com>;methods="MESSAGE"
         ;description="Away from keyboard"
Expires: 600
```

F6 200 OK

```
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:resource@example.com>
From: <sip:resource@example.com>
Call-ID: 110@pua.example.com
CSeq: 2 REGISTER
Contact: <sip:id@pua.example.com>;methods="MESSAGE"
         ; description="Away from keyboard"
         ; expires=600
```

F7 NOTIFY

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/UDP server.example.com:5060
From: <pres:resource@example.com>;tag=ffd2
To: <pres:user@example.com>
Call-ID: 2010@watcherhost.example.com
CSeq: 2 NOTIFY
Event: presence
Content-Type: application/xpidf+xml
Content-Length: 120

<?xml version="1.0"?>
<presence entityInfo="pres:resource@example.com">
  <tuple destination="im:resource@example.com" status="Away from keyboard"/>
</presence>
```

F8 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server.example.com:5060
From: <sip:resource@example.com>;tag=ffd2
To: <pres:user@example.com>
Call-ID: 2010@watcherhost.example.com
CSeq: 2 NOTIFY
```

11 Open Issues

The following is the list of known open issues:

- o This draft recommends that the To and From field are populated with presence URLs rather than sip URLs. Is that reasonable? Will this lead to incompatibilities in proxies? Is there any issues with CPIM if the SIP URL format is used? This depends on what components of a message are signed in CPIM.

- o Rate limitations on NOTIFY: do we want that? How do we pick a value? 5 seconds is arbitrary.

- o Merging of presence data from multiple PA has been removed. Is that OK?

- o Placing IM URLs in the Contact header of a REGISTER: is that OK? What does it mean?

- o The process of migrating subscriptions from a presence server to PUA is not likely to work in the case where subscription refreshes use tags and Route headers. So, we have a choice. Either migration is disallowed, and we keep with leg oriented subscriptions, or migration is allowed, and there is no tags or Route's associated with subscriptions.

- o Converting SIP URLs back to pres URLs.

- o The SIP to CPIM and CPIM to SIP gateways are not stateless, because of the need to maintain Route, Call-ID, CSeq, and other parameters. Perhaps we can ask CPIM to define a token value which is sent in a CPIM request and returned in a CPIM response. Would that help?

- o Need to specify how to take Contacts from REGISTER and build a presence document. One obvious thing is that the contact addresses don't go in there directly; you probably want to put the address of record, otherwise calls might not go through the proxy.

12 Changes from -00

The document has been completely rewritten, to reflect the change from a sales pitch and educational document, to a more formal protocol specification. It has also been changed to align with the SIP event architecture and with CPIM. The specific protocol changes resulting from this rewrite are:

- o The Event header must now be used in the SUBSCRIBE and NOTIFY requests.

- o The SUBSCRIBE message can only have a single Contact header.

-00 allowed for more than one.

- o The From and To headers can contain presence URIs.
- o The Request-URI can contain a presence URI.
- o Subscriptions are responded to with a 202 if they are pending or accepted.
- o Presence documents are not returned in the body of the SUBSCRIBE response. Rather, they are sent in a separate NOTIFY. This more cleanly separates subscription and notification, and is mandated by alignment with CPIM.
- o Authentication is now mandatory at the PA. Authorization is now mandatory at the PA.
- o Fake NOTIFY is sent for pending or rejected subscriptions.
- o A rate limit on notifications was introduced.
- o Merging of presence data has been removed.
- o The subscriber rejects notifications received with tags that don't match those in the 202 response to the SUBSCRIBE. This means that only one PA will hold subscription state for a particular subscriber for a particular presentity.
- o IM URLs allowed in Contacts in register
- o CPIM mappings defined.
- o Persistent connections recommended for firewall traversal.

Obtaining Authorization

When a subscription arrives at a PA, the subscription needs to be authorized by the presentity. In some cases, the presentity may have provided authorization ahead of time. However, in many cases, the subscriber is not pre-authorized. In that case, the PA needs to query the presentity for authorization.

In order to do this, we define an implicit subscription at the PA. This subscription is for a virtual presentity, which is the "set of

subscriptions for presentity X", and the subscriber to that virtual presentity is X itself. Whenever a subscription is received for X, the virtual presentity changes state to reflect the new subscription for X. This state changes for subscriptions that are approved and for ones that are pending. As a result of this, when a subscription arrives for which authorization is needed, the state of the virtual presentity changes to indicate a pending subscription. The entire state of the virtual presentity is then sent to the subscriber (the

presentity itself). This way, the user behind that presentity can see that there are pending subscriptions. It can then use some non-SIP means to install policy in the server regarding this new user. This policy is then used to either accept or reject the subscription.

A call flow for this is shown in Figure 3.

In the case where the presentity is not online, the problem is also straightforward. When the user logs into their presence client, it can fetch the state of the virtual presentity for X, check for pending subscriptions, and for each of them, upload a new policy which indicates the appropriate action to take.

A data format to represent the state of these virtual presentities can be found in [14].

A Acknowledgements

We would like to thank the following people for their support and comments on this draft:

Rick Workman	Nortel
Adam Roach	Ericsson
Sean Olson	Ericsson
Billy Biggs	University of Waterloo
Stuart Barkley	UUNet
Mauricio Arango	SUN
Richard Shockey	Shockey Consulting LLC
Jorgen Bjorkner	Hotsip
Henry Sinnreich	MCI Worldcom
Ronald Akers	Motorola

B Authors Addresses

Jonathan Rosenberg


```
|  SUBSCRIBE X          |                        |
| ------------------->  |                        |
|                       |                        |
|  202 Accepted         |                        |
| <-------------------  | NOTIFY X-subscriptions|
|                       |----------------------->|
|                       |                        |
|                       | 200 OK                 |
```

```
|                        |<----------------------|
|                        |                       |
|                        |                       |
|                        | HTTP POST w/ policy   |
|                        |<----------------------|
|                        |                       |
|                        | 200 OK                |
|                        |---------------------->|
|                        |                       |
|                        |                       |
|                        |                       |
```

Figure 3: Sequence diagram for online authorization

dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Dean Willis
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
email: dwillis@dynamicsoft.com

Robert Sparks
dynamicsoft
5100 Tennyson Parkway

Suite 1200
Plano, Texas 75024
email: rsparks@dynamicsoft.com

Ben Campbell
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
email: bcampbell@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

Jonathan Lennox
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: lennox@cs.columbia.edu

Christian Huitema
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
email: huitema@microsoft.com

Bernard Aboba
Microsoft Corporation

One Microsoft Way
Redmond, WA 98052-6399
email: bernarda@microsoft.com

David Gurle
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
email: dgurle@microsoft.com

David Oran
Cisco Systems
170 West Tasman Dr.
San Jose, CA 95134
email: oran@cisco.com

C Bibliography

[1] M. Day, J. Rosenberg, and H. Sugano, "A model for presence and instant messaging," Request for Comments 2778, Internet Engineering Task Force, Feb. 2000.

[2] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[3] A. Roach, "Event notification in SIP," Internet Draft, Internet Engineering Task Force, Oct. 2000. Work in progress.

[4] D. Crocker et al. , "A common profile for instant messaging (CPIM)," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[5] P. Calhoun, A. Rubens, H. Akhtar, and E. Guttman, "DIAMETER base protocol," Internet Draft, Internet Engineering Task Force, Sept. 2000. Work in progress.

[6] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote authentication dial in user service (RADIUS)," Request for Comments 2865, Internet Engineering Task Force, June 2000.

[7] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry, "The COPS (common open policy service) protocol," Request for Comments 2748, Internet Engineering Task Force, Jan. 2000.

[8] H. Schulzrinne and J. Rosenberg, "SIP caller preferences and callee capabilities," Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[9] J. Rosenberg, D. Drew, and H. Schulzrinne, "Getting SIP through firewalls and NATs," Internet Draft, Internet Engineering Task Force, Feb. 2000. Work in progress.

[10] J. Rosenberg and H. Schulzrinne, "SIP traversal through enterprise and residential NATs and firewalls," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[11] S. Kent and R. Atkinson, "IP encapsulating security payload (ESP)," Request for Comments 2406, Internet Engineering Task Force, Nov. 1998.

[12] T. Dierks and C. Allen, "The TLS protocol version 1.0," Request for Comments 2246, Internet Engineering Task Force, Jan. 1999.

[13] B. Ramsdell and Ed, "S/MIME version 3 message specification," Request for Comments 2633, Internet Engineering Task Force, June 1999.

[14] J. Rosenberg et al. , "An XML based format for watcher

information," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

Table of Contents

1	Introduction	1
2	Definitions	2
3	Overview of Operation	3
4	Naming	4
5	Presence Event Package	5
5.1	Package Name	5
5.2	SUBSCRIBE bodies	5
5.3	Expiration	5
5.4	NOTIFY Bodies	6
5.5	Processing Requirements at the PA	6
5.6	Generation of Notifications	7
5.7	Rate Limitations on NOTIFY	8
5.8	Refresh Behavior	9

6	Publication	9
6.1	Migrating the PA Function	10
7	Mapping to CPIM	11
7.1	SIP to CPIM	11
7.2	CPIM to SIP	14
8	Firewall and NAT Traversal	16
9	Security considerations	17
9.1	Privacy	17
9.2	Message integrity and authenticity	18
9.3	Outbound authentication	18
9.4	Replay prevention	18
9.5	Denial of service attacks	19
9.5.1	Smurf attacks through false contacts	19
10	Example message flows	19
10.1	Client to Client Subscription with Presentity State Changes	19
10.2	Presence Server with Client Notifications	23
10.3	Presence Server Notifications	28
11	Open Issues	32
12	Changes from -00	32
A	Acknowledgements	34
B	Authors Addresses	34
C	Bibliography	37

Network Working Group
Internet Draft

D. Atkins, Telcordia Technologies
G. Klyne, Baltimore Technologies

13 June 2001
Expires: December 2001

Common Presence and Instant Messaging: Message Format
<draft-ietf-impp-cpim-msgfmt-03.txt>

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

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/1id-abstracts.html

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

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Copyright Notice

Copyright (C) The Internet Society 2001. All Rights Reserved.

Abstract

This memo defines the mime type 'message/cpim', a message format for protocols that conform to the Common Profile for Instant Messaging (CPIM) specification.

Discussion of this document

Please send comments to: <impp@iastate.edu>. To subscribe: send a message with the body 'subscribe' to <impp-request@iastate.edu>. The mailing list archive is at <http://www.imppwg.org>.

Table of Contents

1. INTRODUCTION
1.1 Motivation
1.2 Background
1.3 Goals
1.4 Terminology and conventions
2. OVERALL MESSAGE STRUCTURE
2.1 Message/cpim MIME headers
2.2 Message headers
2.3 Character escape mechanism
2.4 Message content
3. MESSAGE HEADER SYNTAX
3.1 Header names
3.2 Header Value
3.3 Language Tagging
3.4 Namespaces for header name extensibility
3.5 Mandatory-to-recognize features
3.6 Collected message header syntax
4. HEADER DEFINITIONS
4.1 The 'From' header
4.2 The 'To' header
4.3 The 'cc' header
4.4 The 'DateTime' header
4.5 The 'Subject' header
4.6 The 'NS' header
4.7 The 'Require' header
5. EXAMPLES
5.1 An example message/cpim message
5.2 An example using MIME multipart/signed
6. APPLICATION DESIGN CONSIDERATIONS
7. IANA CONSIDERATIONS
8. INTERNATIONALIZATION CONSIDERATIONS
9. SECURITY CONSIDERATIONS
10. ACKNOWLEDGEMENTS
11. REFERENCES
12. AUTHORS' ADDRESSES
Appendix A: Amendment history
Full copyright statement