Internet Engineering Task Force IMPP WG
Internet Draft Jonathan Rosenberg
Dean Willis
Robert Sparks
Ben Campbell
dynamicsoft

Henning Schulzrinne
Jonathan Lennox
Columbia U.

Bernard Aboba
Christian Huitema
David Gurle
Microsoft

draft-rosenberg-impp-qauth-00.txt
June 15, 2000
Expires: December, 2000

SIP Extensions for Presence Authorization

STATUS OF THIS MEMO

Abstract

This document proposes a simple SIP extension that allows presence servers to query presence user agents for authorization for a subscription.

1 Introduction

Presence is (indirectly) defined in RFC2778 [1] as subscription to and notification of changes in the communications state of a user. This communications state consists of the set of communications means, communications address, and status of that user. A presence protocol is a protocol for providing such a service over the Internet or any IP network. An example of a presence protocol is described in [2].

A critical component of a presence protocol is authorization. Presence servers will receive subscription requests for presentities served by the presence server. Before accepting or rejecting these subscriptions, the presence server needs to be able to determine if the presentity is willing to authorize the subscription. Such authorization can be determined at the time of subscription (in which case it is an authorization query, or "pull" from presence server to presence user agent), or authorization can be pushed to the server ahead of time.

Authorization policies can be arbitrarily complex. They can depend on any combination of variables available to the presence system, including subscriber profiles, subscription details, and time of day. Supporting these kinds of authorizations through pushes of authorization policies is important, but best left to policy description languages, such as the Call Processing Language (CPL) [3], designed for expressing call processing policies.

However, we recognize that in many cases, a simple policy of "user X may subscribe" and "user Y may not subscribe" will suffice. Furthermore, we realize that there is a critical need to pull subscription authorization, as the PUA cannot possibly know ahead of time all the users that might like to subscribe to it. Thus, to support these requirements, this document defines a simple SIP extension for pulling basic authorization state. This is accomplished by defining a new request method, QAUTH, used to query a PUA as to whether it is willing to authorize a subscriber.

2 Overview of Operation

Operation of this extension is simple. When a PA wishes to obtain authorization for a subscription from a principal or its agent, it formulates a QAUTH request. This request contains the identity of the subscriber in the From field, and the identity of the presentity in the To field. The Call-ID, CSeq, Via and Contact headers are created by the PA for this request; they are not copied from the SUBSCRIBE. The QAUTH request is then sent to a user agent capable of authorizing subscriptions for a presentity. We call such an agent a Subscription Authorizer (SA).

A user agent can indicate that is is an SA through SIP REGISTER requests. REGISTER requests are used to establish address bindings at a registrar, used for routing of messages. An extension to SIP for caller preferences and callee capabilities [4] allows these bindings to contain additional parameters to assist in request routing. One such parameter is the methods parameter, which indicates what methods a user agent can support. A SA includes the value of QAUTH in its registration, indicating that it can support QAUTH requests. This

allows a presentity to have different user agents for authorization of subscriptions, and for processing of subscriptions.

An example REGISTER message containing this parameter is as follows:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP mypc.example.com
To: sip:user@example.com
From: sip:user@example.com
Call-ID: asidhasd@1.2.3.4
CSeq: 39 REGISTER
Contact: sip:user@sa-pc.example.com;methods="QAUTH,SUBSCRIBE"
Content-Length: 0
```

In this case, the same user agent can handle both authorizations and subscriptions.

When the QAUTH request arrives at an SA, the SA authenticates the request (note that the credentials supplied will be those of the PA, not of the original subscriber. This mechanism depends on a trust relationship between the subscription authorizer and its presence agent). If the request is authenticated as coming from the preconfigured PA for the agent, it next determines if the subscriber is authorized. It can do this by prompting the principal, or through pre-configuration of some sort; the mechanism is outside the scope of this specification.

If authorization is granted, a 200 OK response is generated. If denied, a 600 class response is generated. If no authorization can be obtained at all, positive or negative, a 500 class response is generated. The response MAY contain an Expires header indicating the duration for which authorization is granted.

The PA can then use the response to QAUTH to accept or reject the subscription. If authorization is granted for a finite time, the PA MUST destroy the subscription once authorization expires.

3 Detailed Operation

This extension defines a new request method, QAUTH, used to obtain authorization for a subscription.

```
Qauth = QAUTH
```

Like all method names, QAUTH is case sensitive. A client sends a QAUTH request if it wishes to obtain authorization for a subscription. The client will often be a presence agent (PA), although that need not be the case. QAUTH requests are sent to user agent servers (UAS) which are believed to be capable of providing authorization. The client sending a QAUTH request MUST only send it to a server whom the client is configured to consider authoritative for providing authorizations.

Tables 1 and 2 extend Tables 4 and 5 of SIP by adding an additional column, defining the headers that can be used in QAUTH requests and responses.

	where	enc.	e-e	QAUTH
Accept	R		e	o
Accept	415		e	o
Accept-Encoding	R		e	o
Accept-Encoding	415		e	o
Accept-Language	R		e	o
Accept-Language	415		e	o
Allow	200		e	o
Allow	405		e	m
Authorization	R		e	o
Authorization	r		e	o
Call-ID	gc	n	e	m
Contact	R		e	o
Contact	2xx		e	o
Contact	3xx		e	o
Contact	485		e	o
Content-Encoding	e		e	o
Content-Length	e		e	m
Content-Type	e		e	*
CSeq	gc	n	e	m
Date	g		e	o
Encryption	g	n	e	o
Expires	g		e	o
From	gc	n	e	m
Hide	R	n	h	o
Max-Forwards	R	n	e	o
Organization	g	c	h	o

Table 1: Summary of header fields, A--O

	where	enc.	e-e	QAUTH
Priority	R	c	e	o
Proxy-Authenticate	407	n	h	o
Proxy-Authorization	R	n	h	o
Proxy-Require	R	n	h	o
Record-Route	R		h	o
Record-Route	2xx,401,484		h	o
Require	R		e	o
Retry-After	R	c	e	-
Retry-After	404,413,480,486	c	e	o
	500,503	c	e	o
	600,603	c	e	o
Response-Key	R	c	e	o
Route	R		h	o
Server	r	c	e	o
Subject	R	c	e	o
Timestamp	g		e	o
To	gc(1)	n	e	m
Unsupported	420		e	o
User-Agent	g	c	e	o
Via	gc(2)	n	e	m
Warning	r		e	o
WWW-Authenticate	R	c	e	o
WWW-Authenticate	401	c	e	o

Table 2: Summary of header fields, P--Z; (1): copied with possible addition of tag; (2): UAS removes first Via header field

A QAUTH request MUST contain a From field. The From field identifies the subscriber requesting authorization. A QUATH request MUST contain a To field, identifying the principal from whom authorization is sought. The request MUST contain a Call-ID and CSeq header, which together with the To and From, uniquely identify the request. Note that subsequent QAUTH requests need not use the same Call-ID. There is no notion of a persisent session for QAUTH requests; it is like the SIP OPTIONS request in this regard. However, QAUTH requests MAY be record-routed, although there is little or no benefit in doing so. Like all other SIP requests, QAUTH MUST also contain a Via header.

QAUTH MAY contain a body, which indicates details about the subscription requeste by the subscriber. Typically, this body will be copied from the SUBSCRIBE which is triggering the QAUTH request. A response to QAUTH MAY contain a body only if an Accept header was present in the request, listing the allowed body types for the response. Absence of the Accept header from the request implies a body MUST NOT be placed in the response.

A QAUTH request SHOULD be authenticated by the UAS receiving it. Note that the credentials supplied will be those of the originator of the QAUTH (typically, the presence server), and *not* the credentials of the originator of the SUBSCRIBE which triggered the QAUTH.

The response to a QAUTH is 200 OK if authorization is approved for the subscriber indicated in the From field, 600 class is the authorization is rejected, and 500 class if no authorization could be obtained at this time. A response to QAUTH copies the To, From, Call-ID, CSeq, Record-Route, and Via headers from the request. The UAS SHOULD additionally sign the response.

A client sending QAUTH SHOULD verify that the response has been signed by the entity listed in the To field.

4 Example Message Flow

The following shows an example message flow using QAUTH. It also includes SUBSCRIBE and NOTIFY requests.

In the message flow, a subscriber asks to subscribe to some presentity. However, neither a PUA or an SA are currently available for that presentity. So, the server authenticates the subscription (not shown in the flow) and tentatively accepts it. When an SA for the presentity becomes available (known to the presence server through a registration), the server creates a QAUTH request to authorize the previous subscription. The response is 200 OK, authorizing the subscription. However, since the original subscription had expired, the server does not generate a NOTIFY. The watcher subscribes again, after the REGISTER has expired, and has its subscription approved. Note that since there are no contacts registered for the resource at that time, the presence document is vacuous.

```
     subscriber                 server              PUA
        |                      |                     |
        | F1 SUBSCRIBE         |                     |
        |--------------------->| (PUA not "online")  |
        | F2 202 Accepted      |                     |
        |<---------------------|                     |
        | (subscription expires)                     |
        |                      |  F3 REGISTER        |
        |                      |<--------------------|
        |                      |  F4 200 OK          |
        |                      |-------------------->|
```

```
        |                      |  F5 QAUTH           |
        |                      |-------------------->|
        |                      |  F6 200 OK          |
        |                      |<--------------------|
        |                      (registration expires)|
        | F7 SUBSCRIBE         |                     |
        |--------------------->|                     |
        | F8 200 OK            |                     |
        |<---------------------|                     |
```

Message Details

F1 SUBSCRIBE subscriber->server

```
SUBSCRIBE sip:presentity@example.com SIP/2.0
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: Subscriber <sip:subscriber@example.com>
To: Presentity <sip:presentity@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq: 1 SUBSCRIBE
Date: Mon, 12 Jun 2000 16:00:00 GMT
Expires: 600
Contact: sip:subscriber@watcherhost.example.com
```

F2 202 Accepted   server->subscriber

```
SIP/2.0 202 Accepted
Via: SIP/2.0/UDP watcherhost.example.com:5060
From: Subscriber <sip:subscriber@example.com>
To: Presentity <sip:presentity@example.com>
Call-ID: 3248543@watcherhost.example.com
CSeq: 1 SUBSCRIBE
Expires: 600
```

F3 REGISTER PUA->server

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:presentity@example.com>
From: <sip:presentity@example.com>
Call-ID: 2001@pua.example.com
CSeq: 1 REGISTER
Date: Wed, 14 Jun 2000 09:57:16 GMT
Contact: <sip:messenger@pua.example.com;methods="MESSAGE";
          description="open">
Contact: <sip:authagent@pua.example.com;methods="QAUTH">
Expires: 600
```

F4 200 OK server->PUA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pua.example.com:5060
To: <sip:presentity@example.com>
From: <sip:presentity@example.com>
Call-ID: 2001@pua.example.com
CSeq: 1 REGISTER
Contact: <sip:messenger@pua.example.com;methods="MESSAGE";
          description="open">
Contact: <sip:authagent@pua.example.com;methods="QAUTH">
Expires: 600
```

F5 QAUTH server->PUA

```
QAUTH sip:authagent@pua.example.com SIP/2.0
Via: SIP/2.0/UDP server.example.com:5060
From: Subscriber <sip:subscriber@example.com>
To: Presentity <sip:presentity@example.com>
Call-ID: 47774@server.example.com
CSeq: 1 QAUTH
Expires: Sun, 14 Jun 2036 00:00:00 GMT
Contact: sip:server.example.com
```

```
     F6 200 OK   PUA->server

       SIP/2.0 200 OK
       Via: SIP/2.0/UDP server.example.com:5060
       From: Subscriber <sip:subscriber@example.com>
       To: Presentity <sip:presentity@example.com>
       Call-ID: 47774@server.example.com
       CSeq: 1 QAUTH
       Expires: Sun, 14 Jun 2036 00:00:00 GMT
```

```
     F7 SUBSCRIBE subscriber->server

       SUBSCRIBE sip:presentity@example.com SIP/2.0
       Via: SIP/2.0/UDP watcherhost.example.com:5060
       From: Subscriber <sip:subscriber@example.com>
       To: Presentity <sip:presentity@example.com>
       Call-ID: 3248544@watcherhost.example.com
       CSeq: 2 SUBSCRIBE
       Date: Thu, 15 Jun 2000 07:22:15 GMT
       Expires: 600
       Contact: sip:subscriber@watcherhost.example.com
```

```
     F8 200 OK         server->watcher

       SIP/2.0 200 OK
       Via: SIP/2.0/UDP watcherhost.example.com:5060
       From: Subscriber <sip:subscriber@example.com>
       To: Presentity <sip:presentity@example.com>
       Call-ID: 3248544@watcherhost.example.com
       CSeq: 2 SUBSCRIBE
       Expires: 600
       Content-Type: text/lpidf
       Content-Length: 31

       To: sip:presentity@example.com
```

5 Authors Addresses

Jonathan Rosenberg
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: jdrosen@dynamicsoft.com

Dean Willis
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: dwillis@dynamicsoft.com

Robert Sparks
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
email: rsparks@dynamicsoft.com

Ben Campbell
dynamicsoft
200 Executive Drive
Suite 120
West Orange, NJ 07052
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

6 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] J. Rosenberg, R. Sparks, D. Willis, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, and D. Gurle, "SIP extensions for presence," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

[3] J. Lennox and H. Schulzrinne, "CPL: a language for user control of internet telephony services," Internet Draft, Internet Engineering Task Force, Mar. 1999. Work in progress.

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