Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-iptel-sip-reg-payload-00.ps

IPTEL WG
Lennox/Schulzrinne
Columbia University
February 23, 1999
Expires: September 1999

Transporting User Control Information in SIP REGISTER Payloads

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."

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (c) The Internet Society (1999). All Rights Reserved.

Abstract

Several newly developed languages and interfaces, such as the CPL and SIP CGI, allow users or administrators to specify how Internet telephony servers should process calls. There needs to be a method of transporting scripts for such languages between a client and a server. This document proposes using the payload of SIP REGISTER messages, and their responses, as one method to transport them.

Contents

1 Introduction

Several newly developed languages and interfaces, such as the CPL [1] and SIP CGI [2] allow users or administrators to specify how Internet telephony servers should process calls. Scripts typically can be created on a client, but executed on an Internet telephony server.

There therefore needs to be a method of transporting these scripts from a client to a server, and of retrieving them from the server so the client can know the current status or modify the script. This method should integrate cleanly with the existing infrastructure of Internet telephony, without requiring significant additional protocol traffic or complexity in either a client or a server.

This document proposes using the payload of SIP [3] REGISTER messages, and their responses, as the media to transport these scripts to SIP registration servers alongside the user's registration. Since clients typically will need to register anyway, and servers will need to have registrars to process the clients' registrations, this technique does not impose much additional overhead on servers and clients.

This technique is not appropriate for all environments — most obviously, it is not useful for H.323 [4] servers — and we do not anticipate that it will be the only such transport mechanism developed. Other protocols considered have included transporting scripts over LDAP [5], ACAP [6], or HTTP file upload [7], or transport mechanisms developed from scratch.

2 Transport Details

To upload a script, the registration client places the script in the body of the SIP REGISTER request. Bodies of SIP requests are described in [3]. The Content-Type header field is set to the media type of the submitted script. Currently, we expect to register `application/sip-cgi` as the content type for SIP CGI scripts, and `application/cpl` for CPL scripts.

To inform a client of what types of scripts it supports, a server SHOULD send the media types of its supported scripts in Accept header fields in the response to any successful OPTIONS or REGISTER request.

> Allowing the client to restrict transmitted scripts by media type allows clients connected by a low-bandwidth network avoid downloading lengthy scripts.
>
> Note: this is against the strict wording of the SIP specification, which says that Accept headers are only allowed in requests or 415 (Unsupported Media Type) responses. However, it is always legal to include a header in any request or response, as clients which do not understand it in a given context simply ignore it.

In a successful response to any REGISTER request, whether or not a script payload was included, the server SHOULD return the currently specified script in the response body, with its type specified in Content-Type, unless the REGISTER request included a set of Accept headers which did not include the type of the registration script. The server SHOULD NOT return the currently registered script if response to the registration request was an error condition.

The server MAY perform validation on scripts at the time they are uploaded to the server. If the script is not valid, the server SHOULD return a 400-class error to the registration request indicating the problem. It MAY include in the body of the response an explanation of the problem as, for instance, `text/html` or `text/plain`, if the client specified such a type in its Accept header.

When a script of the same type as an existing script is uploaded, the script is replaced in the server. Which script applies to calls in progress at the time the script was changed is not defined by this document, but MAY be specified by specifications of script languages. If the current or new script affects the handling of REGISTER requests, the registration is handled entirely by the existing script; the new script does not take effect until the registration process is complete.

The result of uploading a script of a different type from the one currently registered is undefined; if a client wants to change its script from one type to another, it SHOULD first delete the old script by the method described below, then upload a new one. A server MAY support simultaneous registration of separate scripts of different types.

> This ambiguity allows a simple server to have a single script per user, while a more complex server might allow users to specify simultaneous CGI and CPL scripts, for example.

To delete a script, a client sends a REGISTER message with its Content-Type set to the type of the current script and a Content-Length of 0. If the request specifies a Content-Type other than that of a currently defined script, the behavior is undefined. When a script is deleted, the server SHOULD return to its default call handling behavior for subsequent calls, just as if no script had ever been uploaded.

Question: should this technique have a Require header? It's possible someone might use REGISTER bodies for some other purpose.

3 Persistence Model

Registrations in SIP are normally transient — the data in the Contact header fields last only for the length of time specified in the registration's Expires header, and clients must refresh their registrations periodically.

In contrast, scripts sent to registration servers using the method described in this document are persistent — they remain in the server until replaced or deleted, and they do not need to be refreshed. Servers SHOULD therefore store uploaded scripts in non-volatile storage so they persist through server restarts or failures. Clients SHOULD only upload scripts when they are explicitly requested to, and SHOULD NOT transmit their scripts in every registration request.

The model of standard SIP registrations is that each client registers itself; if a location changes or hosts die, old registrations naturally time out. Since a user can be simultaneously registered from many locations, several clients re-registering periodically present no conflicts.

The model of scripts is quite different. A user only has one script (or at least only of a given type) at a time, so if clients periodically re-uploaded scripts, two clients with different specified scripts would cause "script flapping," as the behavior specified in the server changed frequently, with unpredictable and probably surprising behavior. Moreover, one of the most important purposes of scripts is to control the processing of a user's requests when he or she is *not* registered from any location; if scripts timed out and had to be refreshed, this goal could not be accomplished.

4 Examples

The first example shows a user uploading a simple call-filtering SIP CGI script written in Perl to his server. Note that he is transmitting both a contact address, which persists only for 30 minutes, the time specified by the Expires header, and a script, which persists indefinitely. This allows him subsequently to register new contact addresses and have his script apply equally to them. (See [2] for an explanation of SIP CGI as used in the script.)

The use of Basic authorization here is for the purposes of the example only; in actual practice much more robust authentication SHOULD be used. See section 5.

```
REGISTER sip:@sip.example.com SIP/2.0
From: Joe User <sip:joe@example.com>
To: "J. User" <sip:joe@example.com>
CSeq: 18 REGISTER
Expires: 1800
Call-ID: 39485832@joespc.example.com
Contact: sip:joe@joespc.example.com
Accept: application/sip-cgi, application/sdp, text/html
Authorization: Basic am9lOnBhc3N3b3JkAFBX
```

```
Content-Type: application/sip-cgi
Content-Length: 168

#!/usr/bin/perl
if ($ENV{HTTP_FROM} =~ /telemarketers.com/) {
    print "SIP/2.0 603 Go away\n"
} else {
    print "CGI-DEFAULT-ACTION dummy SIP/2.0\n"
}
```

In the second example, a few minutes later, the user registers a new contact address, but does not change his script. In the response to the registration, the server reminds him of his contact addresses and his current script.

His client sends this request:

```
REGISTER sip:@sip.example.com SIP/2.0
From: Joe User <sip:joe@example.com>
To: "J. User" <sip:joe@example.com>
CSeq: 19 REGISTER
Expires: 1800
Call-ID: 39485832@joespc.example.com
Contact: sip:joe@joeshome.example.com
Accept: application/sip-cgi, application/sdp, text/html
Authorization: Basic am9lOnBhc3N3b3JkAFBX
Content-Length: 0
```

And the server replies with this response:

```
SIP/2.0 200 OK
From: Joe User <sip:joe@example.com>
To: "J. User" <sip:joe@example.com>
CSeq: 19 REGISTER
Contact: sip:joe@joespc.example.com
Contact: sip:joe@joeshome.example.com
Accept: application/sip-cgi, application/cpl
Content-Type: application/sip-cgi
Content-Length: 168

#!/usr/bin/perl
if ($ENV{HTTP_FROM} =~ /telemarketers.com/) {
    print "SIP/2.0 603 Go away\n"
} else {
    print "CGI-DEFAULT-ACTION dummy SIP/2.0\n"
}
```

Finally, the user decides to eliminate his script, and the server responds in the same manner as it would respond to an ordinary registration, as though no script had ever been uploaded:

```
REGISTER sip:@sip.example.com SIP/2.0
From: Joe User <sip:joe@example.com>
To: "J. User" <sip:joe@example.com>
CSeq: 20 REGISTER
Call-ID: 39485832@joespc.example.com
Contact: sip:joe@joeshome.example.com
Authorization: Basic am9lOnBhc3N3b3JkAFBX
Accept: application/sip-cgi, application/sdp, text/html
Content-Type: application/sip-cgi
Content-Length: 0

SIP/2.0 200 OK
From: Joe User <sip:joe@example.com>
To: "J. User" <sip:joe@example.com>
CSeq: 20 REGISTER
Contact: sip:joe@joespc.example.com
Contact: sip:joe@joeshome.example.com
Accept: application/sip-cgi, application/cpl
Content-Length: 0
```

5 Security Considerations

Since the scripts transported by this mechanism control how a server directs private information intended for a user, the server MUST reject all un-authenticated attempts to submit a script, and SHOULD require that the authentication method used verifies the integrity of the submitted script; for example, by having the entire request, including its body, signed with SIP's PGP authentication method.

6 Authors' Addresses

Jonathan Lennox
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: lennox@cs.columbia.edu

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401

New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu

References

[1] J. Lennox and H. Schulzrinne, “CPL: A language for user control of internet telephony services,” Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.

[2] J. Lennox, J. Rosenberg, and H. Schulzrinne, “Common gateway interface for SIP,” Internet Draft, Internet Engineering Task Force, Nov. 1998. Work in progress.

[3] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, “SIP: session initiation protocol,” Internet Draft, Internet Engineering Task Force, Jan. 1999. Work in progress.

[4] International Telecommunication Union, “Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service,” Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

[5] T. Howes, S. Kille, and M. Wahl, “Lightweight directory access protocol (v3),” Request for Comments (Proposed Standard) 2251, Internet Engineering Task Force, Dec. 1997.

[6] J. Myers and C. Newman, “ACAP – application configuration access protocol,” Request for Comments (Proposed Standard) 2244, Internet Engineering Task Force, Dec. 1997.

[7] E. Nebel and L. Masinter, “Form-based file upload in HTML,” Request for Comments (Experimental) 1867, Internet Engineering Task Force, Nov. 1995.

Full Copyright Statement

Copyright (c) The Internet Society (1999). 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.