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

Individual Submission
Lennox/Schulzrinne
Columbia University
March 6, 2000
Expires: September 2000

Transporting User Control Information in SIP REGISTER Payloads

Status of this Memo

Copyright Notice

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 Conventions Of This Document

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [8] and indicate requirement levels for compliant SIP CGI implementations.

> Some paragraphs are indented, like this; they give motivations of design choices, or questions for future discussion in the development of the specification. They are not normative to the specification of the protocol.

3 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/cpl` for CPL scripts; however, this document is not a MIME registration for this type. Clients SHOULD upload SIP CGI scripts as an appropriate media type for the language the script is written in (for example, `application/x-perl`), or `application/octet-stream` if no such media type exists or is known. Registrars MAY perform validation on the media types if they know certain types of scripts cannot be executed on their servers, but SHOULD be permissive about unknown or ambiguous media types for SIP CGI scripts.

> Which types of SIP CGI scripts can be sucessfully run on a server depends on the server's environment, including which scripting languages are installed on it. It is possible that the user has more knowledge of this environment than the server.

Script uploads MUST also be accompanied by two new SIP headers: Content-Purpose and Content-Action. The grammar of these headers is as follows:

```
Content-Purpose  =  "Content-Purpose" ":" purpose
                    *( ";" extension-attribute )
purpose          =  "script" | "sip-cgi" | token

Content-Action   =  "Content-Action" ":" action
                    *( ";" extension-attribute )
action           =  "add" | "delete" | token
```

The grammar symbols "token" and "extension-attribute" are defined in RFC 2543 [3].

The Content-Purpose header serves to define the purpose of the specified content. Currently, two purposes are defined: "script" refers to CPL scripts and other, future scripting environments whose calling conventions can be uniquely determined by their media type; "sip-cgi" refers to SIP CGI scripts, which can be any media type executable on the server platform. Additional purposes may be registered with IANA.

We anticipate that this mechanism in this specification can also be used for purposes such as users' speed-dial lists or device configuration files, and that new purposes would be registered for these.

A similar syntax has been proposed for including multiple bodies in SIP INVITE requests, containing information such as billing information, ISUP interworking, and so forth, in addition to the session description. Christian Huitema proposed a specific solution [9] some time ago, but working group consensus was that a more general "Content-" family header would be more appropriate. However, no further Internet-Drafts seem to have come of this. If or when such progress is made, it is our intention that the Content-Purpose header of this draft should be unified with that header.

The Content-Action header is used when uploading scripts, to specify what the server should do with the script uploaded. If a non-zero-length script is specified, the action "add" MUST be given. The action "delete" MUST only be used with a zero-length script.

A script registrar's normal behavior is to enter the script in its database, as specified in section 4. However, if a zero-length script is submitted with the action "delete," any existing scripts are instead deleted from the database.

Note that, therefore, a zero-length script and the absence of any script are quite distinct phenomena, and both are legal.

If multipart MIME types [10] are used, these headers MUST be included in the MIME part headers, not in the general SIP headers.

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 request directed at the server's registrar address. It MAY also include this in the response to a successful REGISTER request, and of course SHOULD include it in any response which rejected a registration on the grounds of an unsupported media type (as specified in section 7.4.15 of RFC 2543 [3].

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. The revised SIP specification (not yet published as an Internet-Draft, but available from `http://www.cs.columbia.edu/~hgs/sip/`) is expected to fix this problem.

Question: a Accept-Purpose header has also been proposed to list the content purposes the server supports. Is this a good idea?

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, and its purpose in the Content-Purpose, unless the REGISTER request included a set of Accept headers which did not include the type of the registration script. The reply sent by the server SHOULD NOT include Content-Action headers, as the server is not requesting that the client perform any actions. The server SHOULD NOT return the currently registered script if response to the registration request was an error condition.

Allowing the client to restrict transmitted scripts by media type allows clients connected by a low-bandwidth network avoid downloading lengthy scripts.

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 why the script was considered invalid, if the registration included an Accept header with an appropriate media type for such an explanation (such as `text/html` or `text/plain`)

When a script with the same purpose 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. Scripts with different purposes are stored and deleted independently. However, a server MAY choose not to execute some scripts if scripts with another purpose are present, for instance only executing one of a CPL and SIP CGI script.

To delete a script, a client sends a REGISTER message with its Content-Purpose header set to the appropriate value, a Content-Length of 0, and a Content-Action of "delete". If there is no script defined with the specified purpose, this message does nothing. 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. As for changing scripts, the effect of deleting scripts on calls currently in progress is not defined by this specification.

Question: should the technique described in this specification have Require or Supported headers defined?

4 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.

5 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 7.

```
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/x-perl, application/sdp, text/html
Authorization: Basic am9lOnBhc3N3b3JkAFBX
Content-Type: application/x-perl
Content-Length: 137
Content-Purpose: sip-cgi
Content-Action: add

#!/usr/bin/perl
if ($ENV{HTTP_FROM} =~ /telemarketers.com/) {
    print "SIP/2.0 603 Go away\n"
} else {
    exit(0); # Default action
}
```

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/x-perl, 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/cpl, */*
Content-Type: application/x-perl
Content-Length: 137
Content-Purpose: sip-cgi

#!/usr/bin/perl
if ($ENV{HTTP_FROM} =˜ /telemarketers.com/) {
    print "SIP/2.0 603 Go away\n"
} else {
    exit(0); # Default action
}
```

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/x-perl, application/sdp, text/html
Content-Length: 0
Content-Purpose: script
Content-Action: delete


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/x-perl, application/cpl, */*
Content-Length: 0
```

6 Usage notes

Because scripts can be long, clients which upload scripts, or which present an Allow header which could cause scripts to be returned, SHOULD send their REGISTER messages over TCP rather than UDP.

7 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.

8 IANA Considerations

The Content-Purpose header in section 3 can have additional purposes defined.

TODO: need specific registration procedure. TBD if consensus is that this approach is the right way to do things.

9 Changes from earier versions

This document was originally published as `draft-iptel-sip-reg-payload-00`, but the consensus of the IPTel working group was that this should not be a work item of that group.

9.1 Changes from IPTel draft -00

The changebars in the Postscript and PDF versions of this document indicate significant changes from this version.

- Added Content-Purpose and Content-Action headers.
- Changed the procedure by which scripts are deleted.
- Eliminated the pseudo-media-type `application/sip-cgi`, as it is counter to the spirit of MIME. Instead, established that SIP CGI scripts can be any media type.
- Added "Conventions," "Usage Notes," and "IANA Considerations" sections.
- Updated examples to use the syntax of the current version of SIP CGI.
- Updated references to refer to the latest versions of all documents.

10 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, Mar. 1999. Work in progress.

[2] J. Lennox, J. Rosenberg, and H. Schulzrinne, "Common gateway interface for SIP," Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.

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

[4] International Telecommunication Union, "Packet based multimedia communication systems," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

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

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

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

[8] S. Bradner, "Key words for use in RFCs to indicate requirement levels," Request for Comments (Best Current Practice) 2119, Internet Engineering Task Force, Mar. 1997.

[9] C. Huitema, "The multipart/sip-id media type," Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.

[10] N. Freed and N. Borenstein, “Multipurpose internet mail extensions (MIME) part two: Media types,” Request for Comments (Draft Standard) 2046, Internet Engineering Task Force, Nov. 1996.

Full Copyright Statement