Internet Engineering Task Force (IETF)
Request for Comments: 5989
Category: Standards Track
ISSN: 2070-1721

A.B. Roach
Tekelec

October 2010

A SIP Event Package for Subscribing to Changes to an HTTP Resource

Abstract

The Session Initiation Protocol (SIP) is increasingly being used in systems that are tightly coupled with Hypertext Transport Protocol (HTTP) servers for a variety of reasons. In many of these cases, applications can benefit from being able to discover, in near real-time, when a specific HTTP resource is created, changed, or deleted. This document proposes a mechanism, based on the SIP Event Framework, for doing so.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5989.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [3] is increasingly being used in systems that are tightly coupled with Hypertext Transport Protocol (HTTP) [2] servers for a variety of reasons. In many of these cases, applications can benefit from learning of changes to specified HTTP resources in near real-time. For example, user agent terminals may elect to store service-related data in an HTTP tree. When such configuration information is stored and retrieved using HTTP, clients may need to be informed when information changes, so as to make appropriate changes to their local behavior and user interface.

This document defines a mechanism, based on the SIP Event Framework [4], for subscribing to changes in the resource referenced by an HTTP server. Such subscriptions do not necessarily carry the content associated with the resource. In the cases that the content is not conveyed, the HTTP protocol is still used to transfer the contents of HTTP resources. This document further defines a mechanism by which the proper SIP and/or Session Initiation Protocol Secure (SIPS) URI to be used for such subscriptions can be determined from the HTTP server.

2. Terminology

The capitalized terms "MUST", "SHOULD", "MAY", "SHOULD NOT", and "MUST NOT" in this document are to be interpreted as described in RFC 2119 [1].

Note that this document discusses both SIP messages and HTTP messages. Because SIP's syntax was heavily based on HTTP's, the components of these messages have similar or identical names. When referring to message payloads, HTTP documents have historically preferred the hyphenated form "message-body", while SIP documents favor the unhyphenated form "message body". This document conforms to both conventions, using the hyphenated form for HTTP, and the unhyphenated form for SIP.

3. Associating Monitoring SIP URIs with HTTP URLs

One of the key challenges in subscribing to the changes of a resource indicated by an HTTP URL is determining which SIP URI corresponds to a specific HTTP URL. This specification takes the approach of having the HTTP server responsible for the URL in question select an appropriate SIP URI for the corresponding resource and return that URI within an HTTP transaction.

In particular, HTTP servers use link relations -- such as the HTTP Link header field [10], the HTML <link/> element [11], and the Atom <atom:link/> element [5] -- to convey the URI or URIs that can be used to discover changes to the resource. This document defines two new link relation types ("monitor" and "monitor-group") for this purpose, and specifies behavior for SIP and SIPS URIs in link relations of these types. Handling for other URI schemes is out of scope for the current document, although we expect future specifications to define procedures for monitoring via other protocols.

Clients making use of the mechanism described in this document MUST support the HTTP Link header field. Those clients that support processing of HTML documents SHOULD support the HTML <link/> element; those that support processing of Atom documents SHOULD support Atom <atom:link/> elements. These requirements are not intended to preclude the use of any other means of conveying link relations.

The service that provides HTTP access to a resource might provide monitoring of that resource using multiple protocols, so it is perfectly legal for an HTTP response to contain multiple link relationships with relations that allow for monitoring of changes (see [10]). Implementors are cautioned to process all link relations to locate one that corresponds with their preferred change monitoring protocol.

These link relations are scoped to a single HTTP entity. When an HTTP resource is associated with multiple entities (for example, to facilitate content negotiation), the "monitor" and "monitor-group" link relations will generally be different for each entity.

3.1. Monitoring a Single HTTP Resource

If an HTTP server wishes to offer the ability to subscribe to changes in a resource's value using this event package, it returns a link relation containing a SIP or SIPS URI with a relation type of "monitor" in a successful response to a GET or HEAD request on that resource. If the server supports both SIP and SIPS access, it MAY return link relations for both kinds of access.

A client wishing to subscribe to the state change of an HTTP resource obtains a SIP or SIPS URI by sending a GET or HEAD request to the HTTP URL it wishes to monitor. This SIP or SIPS URI is then used in a SUBSCRIBE request, according to the event package defined in Section 4.

3.2. Monitoring Multiple HTTP Resources

If a client wishes to subscribe to the state of multiple HTTP resources, it is free to make use of the mechanisms defined in RFC 4662 [6] and/or RFC 5367 [9]. This requires no special support by the server that provides resource state information. These approaches, however, require the addition of a Resource List Server (RLS) as defined in RFC 4662, which will typically subscribe to the state of resources on behalf of the monitoring user. In many cases, this is not a particularly efficient means of monitoring several resources, particularly when such resources reside on the same HTTP server.

As a more efficient alternative, if an HTTP server wishes to offer the ability to subscribe to the state of several HTTP resources in a single SUBSCRIBE request, it returns a link relation containing a SIP or SIPS URI with a relation type of "monitor-group" in a successful response to a GET or HEAD request on any monitorable resource. In general, this monitor-group URI will be the same for all resources on the same HTTP server.

The monitor-group URI corresponds to an RLS service associated with the HTTP server. This RLS service MUST support subscriptions to request-contained resource lists, as defined in RFC 5367 [9]. This RLS service MAY, but is not required to, accept URI lists that include monitoring URIs that are not associated with resources served by its related HTTP server. Not requiring such functionality allows the RLS to be implemented without requiring back-end subscriptions. If a server wishes to reject such requests, the "403" (Forbidden) response code is appropriate. Any "403" responses generated for this reason SHOULD contain a message body of type "application/ resource-lists+xml"; this message body lists the offending URI or URIs. See RFC 4826 [7] for the definition of the "application/ resource-lists+xml" MIME type.

The HTTP server MUST also return a SIP and/or SIPS link relation with a relation type of "monitor" whenever it returns a SIP and/or SIPS link relation with a relation type of "monitor-group". The monitor-group URI corresponds only to an RLS, and never an HTTP resource or fixed set of HTTP resources.

If a client wishes to subscribe to the state of multiple HTTP resources, and has received monitor-group URIs for each of them, it may use the monitor-group URIs to subscribe to multiple resources in the same subscription. To do so, it starts with the set of HTTP resources it wishes to monitor. It then groups these resources by their respective monitor-group URIs. Finally, for each such group,

it initiates a subscription to the group's monitor-group URI; this subscription includes a URI list, as described in RFC 5367. The URI list contains all of the URIs in the group.

For example: consider the case in which a client wishes to monitor the resources http://www.example.com/goat, http://www.example.com/sheep, http://www.example.org/llama, and http://www.example.org/alpaca. It would use HTTP to perform HEAD and/or GET operations on these resources. The responses to these operations will contain link relations for both monitor and monitor-type for each of the four resources. Assume the monitor link for http://www.example.com/goat is sip:a94aa000@example.com; for http://www.example.com/sheep, sip:23ec24c5@example.com; for http://www.example.org/llama, sip:yxbO-UHYxyizU2H3dnEerQ@example.org; and for http://www.example.org/alpaca, sip:-J0piC0ihB9hfNaJc7GCBg@example.org. Further, assume the monitor-group link for http://www.example.com/goat and http://www.example.com/sheep are both sip:httpmon@rls.example.com, while the monitor-group link for http://www.example.org/llama and http://www.example.org/alpaca are both sip:rls@example.org.

Because they share a common monitor-group link, the client would group together http://www.example.com/goat and http://www.example.com/sheep in a single subscription. It sends this subscription to the monitor-group URI (sip:httpmon@rls.example.com), with a resource-list containing the relevant monitor URIs (sip:a94aa000@example.com and sip:23ec24c5@example.com). It then repeats this process for the remaining two HTTP resources, using their monitor-group and monitor URIs in the same way.

4. HTTP Change Event Package

4.1. Event Package Name

The name of this event package is "http-monitor".

4.2. Event Package Parameters

This event package defines a single parameter to be used with the Event header field. The syntax for this parameter is shown below, using the ABNF format defined in RFC 5234 [8]. The use of the construction "EQUAL" is as defined by RFC 3261 [3].

```
body-event-param = "body" EQUAL ( "true" / "false" )
```

If present and set to "true" in a SUBSCRIBE request, this parameter indicates to the server that the client wishes to receive a message-body component in the message/http message bodies sent in NOTIFY messages.

If a server receives a SUBSCRIBE message with an Event header field "body" parameter set to "true", it MAY choose to include a message-body component in the message/http message bodies that it sends in NOTIFY messages. Alternatively, it MAY decline to send such message-bodies, even when this parameter is present, based on local policy. In particular, it would be quite reasonable for servers to have a policy of not including HTTP message-bodies larger than a relatively small number of bytes.

When absent, the value of this parameter is assumed to be "false".

> Note that this parameter refers to the message-body component of the HTTP message, not the message body component of the SIP message.

4.3. SUBSCRIBE Bodies

This event package defines no message bodies to be used in the SUBSCRIBE message.

4.4. Subscription Duration

Reasonable values for the duration of subscriptions to the http-monitor event package vary widely with the nature of the HTTP resource being monitored. Some HTTP resources change infrequently (if ever), while others can change comparatively rapidly. For rapidly changing documents, the ability to recover more rapidly from a subscription failure is relatively important, so implementations will be well served by selecting smaller durations for their subscriptions, on the order of 1800 to 3600 seconds (30 minutes to an hour).

Subscriptions to slower-changing resources lack this property, and the need to periodically refresh subscriptions render short subscriptions wasteful. For these types of subscriptions, expirations as long as 604800 seconds (one week) or even longer may well make sense.

The subscriber is responsible for selecting an expiration time that is appropriate for its purposes, taking the foregoing considerations into account. Keep in mind that the goal behind selecting subscription durations is to balance server load against time to

recover in the case of a failure. In particular, short subscription expiration times guard against the loss of subscription server state, albeit at the expense of additional load on the server.

In the absence of an expires value in a subscription, the notifier can assume a default expiration period according to local policy. This local policy might choose to take various aspects of the monitored resource into account, such as its age and presumed period of validity. Absent any other information, it would not be unreasonable for a server to assume a default expiration value of 86400 seconds (one day) when the client fails to provide one.

4.5. NOTIFY Bodies

By default, the message bodies of NOTIFY messages for the http-monitor event package will be of content-type "message/http," as defined in RFC 2616 [2].

4.5.1. Use of message/http in HTTP Monitor Event Package

The message/http NOTIFY message bodies used in the HTTP monitor event package reflect a subset of the response that would be returned if the client performed an HTTP HEAD operation on the HTTP resource.

An example of a message/http message body as used in this event package is shown below.

```
HTTP/1.1 200 OK
Date: Sat, 13 Nov 2010 17:18:52 GMT
ETag: 38fe6-58b-1840e7d0
Content-MD5: 4e3b50421829c7c379a5c6154e560449
Last-Modified: Sat, 13 Nov 2010 03:29:00 GMT
Accept-Ranges: bytes
Content-Location: http://www.example.com/pet-profiles/alpacas/
Content-Length: 12511
Content-Type: text/html
```

When used in the HTTP monitor event package defined in this document, the message/http SHOULD contain at least one of an ETag or Content-MD5 header field, unless returning a null state as described in Section 4.7. Inclusion of a Last-Modified header field is also RECOMMENDED. Additionally, the message/http message body MUST contain a Content-Location field that identifies the resource being monitored. Note that this is not necessarily the same URL from which the link association was originally obtained; see RFC 2616 [2] for details.

Except for the foregoing normative requirements, the decision regarding which HTTP header fields to include is at the discretion of the notifier.

When used in the HTTP monitor event package, the message/http MUST NOT contain a message-body component, unless the corresponding subscription has explicitly indicated the desire to receive such bodies as described in Section 4.2.

If the change to the resource being communicated represents a renaming of the HTTP resource, the message/http start line will contain the same 3xx-class HTTP response that would be returned if a user agent attempted to access the relocated HTTP resource with a HEAD request (e.g., "301 Moved Permanently"). The message/http also SHOULD contain a Location header field that communicates the new name of the resource.

If the change to the resource being communicated represents a deletion of the HTTP resource, the start line will contain the same 4xx-class HTTP response that would be returned if a user agent attempted to access the missing HTTP resource with a HEAD request (e.g., "404 Not Found" or "410 Gone").

4.6. Notifier Processing of SUBSCRIBE Requests

Upon receipt of a SUBSCRIBE request, the notifier applies authorization according to local policy. Typically, this policy will be aligned with the HTTP server authorization policies regarding access to the resource whose change state is being requested.

4.7. Notifier Generation of NOTIFY Requests

NOTIFY messages are generated whenever the underlying resource indicated by the corresponding HTTP URL has been modified.

In the case that the notifier has insufficient information to return any useful information about the underlying HTTP resource, it MUST return a message body that is zero bytes long (subject to any mechanisms that would suppress sending of a NOTIFY message).

4.8. Subscriber Processing of NOTIFY Requests

Upon receipt of a NOTIFY message, the subscriber applies any information in the message/http to update its view of the underlying HTTP resource. In most cases, this results in an invalidation of its view of the HTTP resource. It is up to the subscriber implementation to decide whether it is appropriate to fetch a new copy of the HTTP resource as a reaction to a NOTIFY message.

4.9. Handling of Forked Requests

Multiple notifiers for a single HTTP resource is semantically nonsensical. In the aberrant circumstance that a SUBSCRIBE request is forked, the subscriber SHOULD terminate all but one subscription, as described in Section 4.4.9 of RFC 3265 [4].

4.10. Rate of Notifications

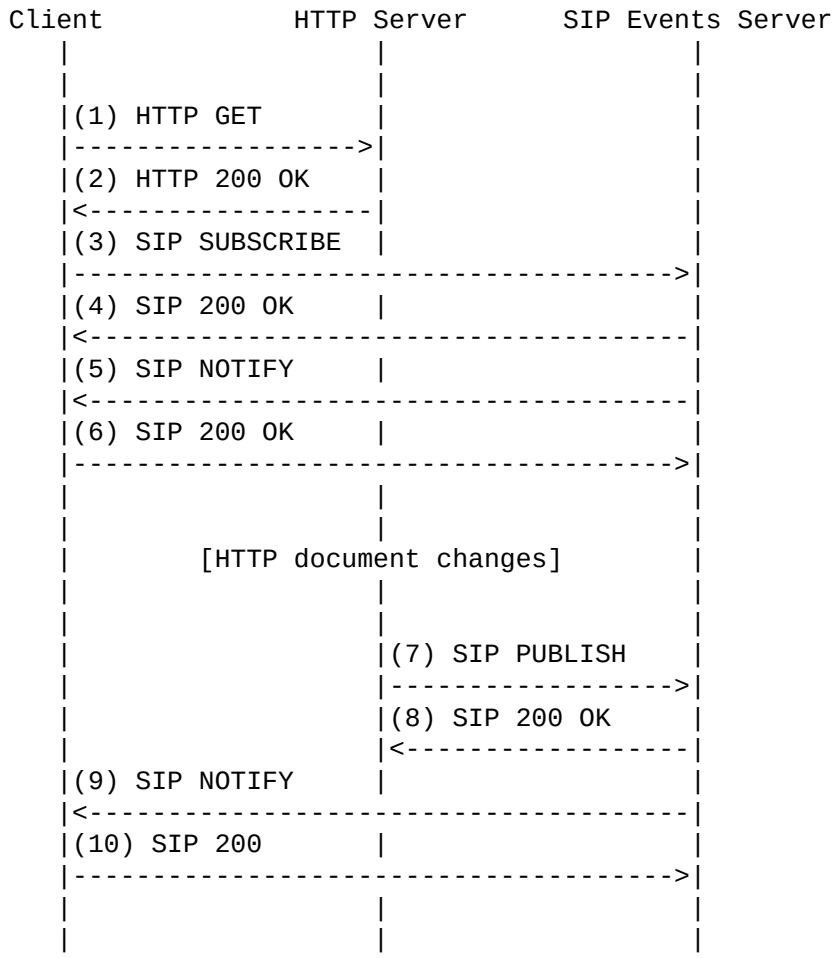
Because the data stored in HTTP for the purpose of SIP services may change rapidly due to user input, and because it may potentially be rendered to users and/or used to impact call routing, a high degree of responsiveness is appropriate. However, for the protection of the network, notifiers for the http-monitor event package SHOULD NOT send notifications more frequently than once every second.

4.11. State Agents

Decomposition of the authority for the HTTP resource into an HTTP server and a SIP Events server is likely to be useful, due to the potentially different scaling properties associated with serving HTTP resources and managing subscriptions. In the case of such decomposition, implementors are encouraged to familiarize themselves with the PUBLISH mechanism described in RFC 3903 [14].

5. Example Message Flow

The following is a simple example message flow, to aid in understanding how this event package can be used. It is included for illustrative purposes only, and does not form any portion of the specification of the mechanisms defined in this document.



The following messages illustrate only the portions of the messages that are relevant to the example. They intentionally elide fields that, while typical or mandatory, are not key to understanding the foregoing message flow.

1. The client issues a GET request to retrieve the document identified by the URL "http://www.example.com/pet-profiles/alpacas/".

```
GET /pet-profiles/alpacas/ HTTP/1.1
Host: www.example.com
```

2. The HTTP server responds with the document, and several relevant pieces of meta-data. Of key interest for this example is the Link header field with a "rel" parameter of "monitor". This is the SIP URL that the client will use to monitor changes to the state of the HTTP resource. Note that, since the message-body

is an HTML document, the "monitor" link relation could alternately be indicated in the HTML document itself, through the use of a <link/> element.

Note also the presence of the ETag, Content-MD5, and Last-Modified header fields. These can be used by the client to identify the version of the entity returned by the HTTP server.

```
HTTP/1.1 200 OK
ETag: 38fe6-58b-1840e7d0
Content-MD5: 4e3b50421829c7c379a5c6154e560449
Last-Modified: Sat, 13 Nov 2010 03:29:00 GMT
Content-Location: http://www.example.com/pet-profiles/alpacas/
Link: <sip:23ec24c5@example.com>; rel="monitor"
Link: <sip:httpmon@rls.example.com>; rel="monitor-group"
Content-Length: 12511
Content-Type: text/html

[HTML message-body]
```

3. The client sends a SUBSCRIBE request to the SIP URI indicated in the "monitor" link relation, indicating an event type of "http-monitor".

```
SUBSCRIBE sip:23ec24c5@example.com SIP/2.0
To: <sip:23ec24c5@example.com>
From: <sip:adam@example.org>;tag=57dac993-0b5b-4f04
Event: http-monitor
Contact: <sip:adam@198.51.100.17:2487>
```

4. The SIP Events server acknowledges receipt of the subscription request, and establishes a dialog for the resulting subscription.

```
SIP/2.0 200 OK
To: <sip:23ec24c5@example.com>;tag=907A953576E6
From: <sip:adam@example.org>;tag=57dac993-0b5b-4f04
Contact: <sip:23ec24c5@203.0.113.72>
```

5. The SIP Events server sends a NOTIFY message containing the current state of the HTTP resource. The client can compare the contents of the ETag, Content-MD5, or Last-Modified header fields against those received in the HTTP "200" response to verify that it has the most recent version of the entity.

```
NOTIFY sip:adam@198.51.100.17:2487 SIP/2.0
To: <sip:adam@example.org>;tag=57dac993-0b5b-4f04
From: <sip:23ec24c5@example.com>;tag=907A953576E6
Contact: <sip:23ec24c5@203.0.113.72>
Event: http-monitor
Subscription-State: active
Content-Type: message/http

HTTP/1.1 200 OK
ETag: 38fe6-58b-1840e7d0
Content-MD5: 4e3b50421829c7c379a5c6154e560449
Last-Modified: Sat, 13 Nov 2010 03:29:00 GMT
Content-Location: http://www.example.com/pet-profiles/alpacas/
Content-Length: 12511
Content-Type: text/html
```

6. The client acknowledges receipt of the NOTIFY message.

```
SIP/2.0 200 OK
To: <sip:adam@example.org>;tag=57dac993-0b5b-4f04
From: <sip:23ec24c5@example.com>;tag=907A953576E6
Contact: <sip:adam@198.51.100.17:2487>
```

7. At some point after the subscription has been established, the entity hosted by the HTTP server changes. It can convey this information to a SIP Events server using a SIP PUBLISH request. The PUBLISH message body contains information regarding the state of the entity.

   Note that SIP PUBLISH is one of many ways such information could be conveyed -- any other means of communicating this information would also be valid.

```
PUBLISH sip:23ec24c5@example.com SIP/2.0
To: <sip:23ec24c5@example.com>
From: <sip:webserver@example.com>;tag=03-5gbK652_jNMr-b8-11Z_G-NsLR
Contact: <sip:webserver@203.0.113.99>
Event: http-monitor
Content-Type: message/http

HTTP/1.1 200 OK
ETag: 3238e-1a3-b83be580
Content-MD5: 10a1ef5b223577059fafba867829abf8
Last-Modified: Sat, 17 Nov 2010 08:17:39 GMT
Content-Location: http://www.example.com/pet-profiles/alpacas/
Content-Length: 17481
Content-Type: text/html
```

8. The SIP Events server acknowledges the changed entity state. Note that the value of the SIP-ETag header field is not related to the ETag header field associated with the HTTP entity.

```
SIP/2.0 200 OK
To: <sip:23ec24c5@example.com>
From: <sip:webserver@example.com>;tag=03-5gbK652_jNMr-b8-11Z_G-NsLR
SIP-ETag: 3psbqi1o5633
```

9. The SIP events server informs the client of the change in state for the subscribed resource using a NOTIFY message.

```
NOTIFY sip:adam@198.51.100.17:2487 SIP/2.0
To: <sip:adam@example.org>;tag=57dac993-0b5b-4f04
From: <sip:23ec24c5@example.com>;tag=907A953576E6
Contact: <sip:23ec24c5@203.0.113.72>
Event: http-monitor
Subscription-State: active
Content-Type: message/http

HTTP/1.1 200 OK
ETag: 3238e-1a3-b83be580
Content-MD5: 10a1ef5b223577059fafba867829abf8
Last-Modified: Sat, 17 Nov 2010 08:17:39 GMT
Content-Location: http://www.example.com/pet-profiles/alpacas/
Content-Length: 17481
Content-Type: text/html
```

10. The client acknowledges receipt of the changed state. At this point, the client may choose to retrieve a fresh copy of the document so that it can act on the new content. Alternately, it may simply mark the previously retrieved document as out of date or discard it, choosing to retrieve a new copy at a later point in time.

```
SIP/2.0 200 OK
To: <sip:adam@example.org>;tag=57dac993-0b5b-4f04
From: <sip:23ec24c5@example.com>;tag=907A953576E6
Contact: <sip:adam@198.51.100.17:2487>
```

6. Security Considerations

Unless secured using Transport Layer Security (TLS), IPsec, or a similar technology, the content of the Link header field is not secure, private, or integrity-protected.

Because an unencrypted Link header field can be intercepted, server implementations are cautioned not to use the value sent in the Link header field as a security token that authenticates a subscriber, or that demonstrates authorization to subscribe to a particular resource.

Because an unsecured Link header field can be tampered with -- or inserted -- in transit, client implementations need to consider the interaction between their application and a forged set of notifications. This issue becomes particularly problematic when the change notifications include entity state (using "body=true").

This mechanism introduces the means to learn information about the state of an HTTP resource using an alternate protocol, and potentially a different server. If the HTTP resource is restricted using some form of access control, special care MUST be taken to ensure that the SIP means of subscribing to the resource state is also restricted in the same way. Otherwise, unauthorized users may learn information that was intended to be confidential (including the actual resource value, in some cases).

Similarly, if the HTTP resource is encrypted or integrity protected in transit -- for example, by using HTTP over TLS [12] -- then the SIP means of subscribing to the HTTP resource MUST also have appropriate encryption or integrity protection applied. Examples of mechanisms for providing such protection include the use of the SIPS URI scheme [17], and the use of S/MIME bodies [13].

7. IANA Considerations

7.1. New Link Relations

The following entries have been added to the "Link Relation Types" registry, as created by the "Web Linking" specification [10].

7.1.1. New Link Relation: monitor

- Relation Name: monitor
- Description: Refers to a resource that can be used to monitor changes in an HTTP resource.
- Reference: RFC 5989

7.1.2. New Link Relation: monitor-group

- Relation Name: monitor-group
- Description: Refers to a resource that can be used to monitor changes in a specified group of HTTP resources.
- Reference: RFC 5989

7.2. New SIP Event Package: http-monitor

The following entry is to be added to the "SIP Events" registry, as created by the SIP Event Framework [4].

Package Name: http-monitor

Type: package

Contact: Adam Roach, adam@nostrum.com

Reference: RFC 5989

7.3. New Event Header Field Parameter: body

The following entry is to be added to the SIP "Header Field Parameters and Parameter Values" registry, as created by the SIP Change Framework [15].

Header Field: Event

Parameter Name: body

Predefined Values: yes

Reference: RFC 5989

8. Acknowledgements

Thanks to Lisa Dusseault and Mark Nottingham for significant input on the mechanisms to bind an HTTP URL to a SIP URI. Thanks also to Mark Nottingham and Theo Zourzouvillys for thorough feedback on early versions of this document. Thanks to Martin Thompson, Shida Schubert, John Elwell, and Scott Lawrence for their careful reviews and feedback.

9. References

9.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[3] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[4] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[5] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[6] Roach, A., Campbell, B., and J. Rosenberg, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", RFC 4662, August 2006.

[7] Rosenberg, J., "Extensible Markup Language (XML) Formats for Representing Resource Lists", RFC 4826, May 2007.

[8] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[9] Camarillo, G., Roach, A., and O. Levin, "Subscriptions to Request-Contained Resource Lists in the Session Initiation Protocol (SIP)", RFC 5367, October 2008.

[10] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[11] Jacobs, I., Hors, A., and D. Raggett, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

9.2. Informative References

[12] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[13] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[14] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, October 2004.

[15] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", BCP 98, RFC 3968, December 2004.

[16] Dusseault, L., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.

[17] Audet, F., "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)", RFC 5630, October 2009.

[18] Wachob, G., Reed, D., Chasen, L., Tan, W., and S. Churchill, "Extensible Resource Identifier (XRI) Resolution V2.0", February 2008, <http://docs.oasis-open.org/xri/2.0/specs/xri-resolution-V2.0.html>.

Appendix A. Rationale: Other Approaches Considered

Several potential mechanisms for retrieving the SIP URI from the HTTP server were evaluated. Of them, link relations were determined to have the most favorable set of properties. Two key candidates that were considered but rejected in favor of link relations are discussed below.

The HTTP PROPFIND method ([16], Section 9.1) can be used to retrieve the value of a specific property associated with an HTTP URL. However, this cannot be done in conjunction with retrieval of the document itself, which is usually desirable. If a PROPFIND approach is employed, clients will typically perform both a GET and a PROPFIND on resources of interest. Additionally, the use of PROPFIND requires support of the PROPFIND method in HTTP user agents -- which, although fairly well implemented, still lacks the penetration of GET implementations.

Similar to PROPFIND, XRDS (Extensible Resource Descriptor Sequence) [18] can be used to retrieve properties associated with an HTTP URL. It has the advantage of using GET instead of PROPFIND; however, it suffers from both the two-round-trip issue discussed above, as well as an unfortunately large number of options in specifying how to retrieve the properties.

Author's Address

Adam Roach
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

EMail: adam@nostrum.com