INTERNET DRAFT
<draft-cohen-gena-p-base-01.txt>
Expires January 1, 1999

J. Cohen, Microsoft
S. Aggarwal, Microsoft

July 9, 1998

General Event Notification Architecture Base

Status of this memo

This document is an Internet-Draft. 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 made obsolete 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 learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited. Please send comments to the HTTP working group at <http-wg@cuckoo.hpl.hp.com>. Discussions of the working group are archived at <URL:http://www.ics.uci.edu/pub/ietf/http/>.

Abstract

This specification defines an HTTP notification architecture that transmits notifications between HTTP resources. An HTTP resource could be any object which might need to send or receive a notification, for example a distribution list, buddy list, print job, etc.

Introduction

The protocol definition provides two main components notification and subscription that can be used independently or in combination.

Any resource can transmit a notification through HTTP to any other resource through the NOTIFY method. Notifications can be triggered by arbitrary events occurring on originating resources for example, a change in the properties of a resource, a notification received by the originating resource, or any other general event.

Additionally, a resource can establish a subscription relationship with any other resource and thereby automatically receive a notification when a specified event occurs on the latter resource.

The notification framework defined herein thus provides powerful capabilities to track arbitrary properties on resources on a real-time basis, such as remote filesystem properties, buddy list online presence information, device availability, etc. The notification framework also allows for more complex routing schemes, such as routing notifications through a succession or hierarchy of resources or sending a notification to more than one resource.

This specification defines the syntax and semantics for asynchronous and polled notification delivery. Asynchronous notifications allow a subscription server to send an event notification at any time, without maintaining a persistent network connection. Polled notifications allow a backward compatible, although less elegant, mechanism for notification delivery across a deployed infrastructure of proxy servers and firewalls. The set of delivery mechanisms is extensible.

1 Definitions

1.1 Resource

A resource is any entity with a URI address that wishes to participate in this notifications protocol.

1.2 Event

Any change in a resource that potentially needs to trigger an event notification.

1.3 Subscription

An established relationship in which a resource has indicated interest in certain events.

1.4 Subscriber

A resource that negotiates a subscription with a subscription server.

1.5 Implied Subscriber

A resource that did not negotiate a subscription with a subscription server but will still be notified of events on that server.

1.6 Subscribed Resource

A subscribed resource is a resource that is the subject of a subscription.

1.7 Event Notification

An event notification is a message sent by a subscribed resource to its subscribers, implied and otherwise, notifying them of an event on that resource.

2 New HTTP Methods

Notification and subscription messages are transmitted via new HTTP methods.

2.1 Subscription Methods

2.1.1 SUBSCRIBE

The SUBSCRIBE method is used to subscribe to a subscription server.

2.1.2 UNSUBSCRIBE

The UNSUBSCRIBE method is used to terminate a subscription.

2.2 Delivery Methods

2.2.1 NOTIFY

NOTIFY is used for asynchronous notification delivery from one resource to another. The Request-URI is the URI of the resource to be notified.

2.2.2 POLL

POLL is used to check a resource for pending events. When a POLL request occurs, the Request-URI is the URI of the subscribed resource.

3 Subscriptions

3.1 SUBSCRIBE request

A subscriber subscribes to subscription servers via the SUBSCRIBE method.

The Request-URI specifies the URI of the resource being subscribed to.

The following headers are provided (detailed header definitions are given in a later section):

3.1.1 Notification-Type

Notification-Type indicates the type of event being subscribed to

3.1.2 Call-Back

Call-Back indicates how event notifications are to be delivered from the subscription server to the subscriber

3.1.3 Subscription-Lifetime

Subscription-Lifetime indicates when the subscription is to expire.

3.2 SUBSCRIBE responses

The response to a SUBSCRIBE method must include a Subscription-ID header that uniquely identifies the subscription.
The response must also include the Delivery-Scheme and Subscription-Lifetime headers, with possibly adjusted values.

4 Delivery Mechanisms

Subscribers use the call-back or delivery-control header to specify the method requested for delivering notifications, as well as any parameters associated with the method.

This document defines two delivery schemes Asynchronous and Polled. The set of delivery schemes is extensible.

4.1 Asynchronous

Asynchronous notification delivery occurs through a NOTIFY HTTP message from the subscription server to the subscriber. This behavior may cause problems with some existing systems which expect HTTP requests to initiate from inside of the firewall.

4.2 Polled

Polled delivery indicates that the subscription client will poll the resource at a specified interval agreed upon by the server and the client.

A Poll-Interval parameter specifies intent to issue the next POLL after the specified Poll-Interval; the server may buffer notifications until the next POLL. Subsequent POLL requests may also have a Poll-Interval header specifying the requested duration until the next POLL.

A wait-timeout parameter specifies a request for the server to keep the SUBSCRIBE connection open until the timeout value expires or a notification is received. [[Issue: does Connection-timeout conflict with HTTP?]] The wait-timeout header can also appear on a POLL request, with identical semantics.

5 New HTTP Headers

5.1 Notification-Type General Header

The Notification-Type request header is used with a SUBSCRIBE request message to indicate the requested notification type.

The Notification-Type response header is used with a SUBSCRIBE response to indicate the accepted notification type.

Notification-type := notification-Type: ntype

Ntype := gena:update | ntype-extension
Ntype-extension := < URL >

This draft declares one base notification type, gena:update.

5.2 Call-Back General Header

The call-back request header is used with a SUBSCRIBE request to indicate the subscriber's preferred call-back addresses.

The call-back response header is used with a SUBSCRIBE response to indicate the order in which the subscription server intends to call the listed addresses. If an address submitted with a call-back request header is not listed in the call-back response header then the server has rejected its use. If the server can not accept any of the addresses specified in the call-back request header then the method MUST fail with

The format of the call-back header is an ordered list of space separated URIs.

The call-back parameter is an ordered series of URIs in a space separated string. Each URI represents an event notification receiver interested in receiving notifications (via NOTIFY or other methods) when a triggering event occurs on the subscribed resource.

The call-back with the highest preference is listed first while the call-back with the least preference is listed last in the string.

Call-back := Call-Back: URI [URI]

An example call-back header:

Call-Back: http://switchblade.microsoft.com:8888/mailclient
mailto:joshco@microsoft.com

This example indicates a preferred call-back via HTTP to the switchblade machine on port 8888 and a secondary call-back via an email to joshco@microsoft.com

URIs can be any valid URI type. For example, an http URI like http://foo:port/bar indicates that an http transaction via NOTIFY is to be sent upon event notification. An httpu URI like Httpu://address:port/ indicates that a UDP datagram containing an HTTP request with NOTIFY is to be sent. A mail URI like mailto:user@domain indicates that an email message containing the same HTTP message with NOTIFY is to be sent as the body of an email message.

5.3 Subscription-Lifetime:

This is used on a SUBSCRIBE request as well as response. The requester indicates the timeout it is requesting. The responder may respond with the same or different timeout, but the responder's value is binding.

Subscription-Lifetime: 1*DIGIT

In seconds.

5.4 Delivery-control:

This can be used on a SUBSCRIBE response, or a POLL request or response. (Subscribe requests have an analogous version within the Delivery-Scheme header.) In a request, the value indicates the Delivery-control the client would like; the server responds with the Delivery-control acceptable to it, which may be different from the requested Delivery-control.

The Delivery-control header is a generic control mechanism for poll related parameters.

Delivery-control := Delivery-control: 1#poll-parameter
Poll-parameter := poll-interval | wait-time | poll-provoke | batch-mode

Wait-time := wait-time = seconds
Poll-Interval := poll-interval = seconds
Poll-provoke := poll-provoke
Batch-Mode := multipart/related

Wait-time: Specifies in seconds how long the client requests the server to keep the connection open (the server may close the connection earlier, at its discretion).

Poll-Interval: On request, specifies the requested polling interval initiated by the subscriber. On response, it indicates the servers adjusted recommended poll-interval.

Poll-Provoke: The poll-provoke keyword is typically used in a NOTIFY message, possibly over UDP, to indicate to the subscriber that a notification has occurred and to poll the server for the details. This is included as a security measure for unencrypted and or datagram notifications where eavesdropping is common. This allows the server to enforce ACLs on the subsequent provoked poll.

Batch-Mode: The batch mode parameter specifies if and how multiple event notifications may be batched together into a single notification message. In a request, the batch-control parameter indicates the preferred method of batching from the subscriber. In a response, the batch-control parameter indicates the allowed method on the server. This draft specifies a single batch mechanism for multipart/related multipart MIME. It is specified later in this document. Future specifications may define additional batch mechanisms such as XML.
Support for batch mode is OPTIONAL.

5.5 Mandatory header for version control:

The GENA architecture makes use of the Mandatory specification as part of the HTTP extension framework. [13.3]

5.6 Mandatory feature use

This is included on every request and response, to indicate the notifications protocol version for cross-compatibility.

Mandatory := Man: tokenURI

This draft specifies the version token URI:
http://extensions.iana.org/http/GENA/

This token is expressed as a URI for use with a future http extension framework such as Mandatory [13.3]

5.7 Subscription-ID

This is contained in initial SUBSCRIBE responses, on all subsequent SUBSCRIBE requests (for renewing existing subscriptions), on all POLL and UNSUBSCRIBE requests, and on NOTIFY requests generated as a result of a prior subscription. [[Issue: exact subscription ID semantics depends on whether we have a From header.]]

Subscription-ID:= subscription-id = string

5.8 Route-ID:

Route-ID: indicates a route identifier for the message. Each proxy or gateway in the path should append a route-id header to the subscription message. When adding a new route-ID header, the proxy should use an integer greater than the current highest in the rid field.

```
Syntax:
Route-ID: rid ; route-options
rid :=  integer
```

route-options := delivery-uri = URI

5.9 Depth Header

Depth = "Depth" ":" ("0" | "1" | "infinity")

The Depth header is used with methods executed on resources which could potentially have internal members to indicate whether the method is to be applied only to the resource (Depth = 0), to the resource and its immediate children, (Depth = 1), or the resource and all its progeny (Depth = infinity).

5.10 Extended-Response Header

Extended-Response := Extended-Response: response-code ; #(ext-param)

Ext-param := namespace | comment-field

Response-code := numeric-digits
Namespace := ns = URI
Comment-field := comment = string

Example:
Extended-Response: 10219 ; ns = http://microsoft.com/gena comment = subscription failed

5.11 The URI scheme httpu:

The URI scheme httpu: is defined to include a host and a port to which a UDP datagram can be sent. It is specified as a URI with scheme `httpu'.

When used with the notification system, the payload of the UDP datagram is to be the same as the HTTP messages defined here as well.

6 Response Codes

In addition to various HTTP response code messages, GENA response code messages are also defined. These messages are returned through the Extended-response header.

GENA responses are comprised of a major response code number and an optional minor response code number. GENA responses always use an appropriate x00 code as the major response code, and a minor response code indicated with the Extended-Response code header.

HTTP response codes used for general responses:

200 Success

In other cases, rather than allocate arbitrary response codes from the already limited HTTP response code namespace, extended responses are used.

When used to indicate a successful operation, the 200 Ok HTTP response code indicates the major response code, and the minor response code may be:

20241 Subscription Succeeded
20242 Notification Acknowledged
20243 Subscription Terminated

When used to indicate an operation failure, the 400 HTTP response code indicates the major response code, and the minor response code may be:

20441 Subscription Failed
20442 No valid call-backs
20443 Unsupported Notification-Type

7 Proxy Routing

Often, a client may reach a resource via a proxy server. In this case, with a standard proxy server, asynchronous call-back addresses may not be visible to an external server. Because of this, the Route-ID: header is introduced.

Subscription clients should poll their proxy chain to detect which version of HTTP that proxy supports. In addition, they should poll for support of the `route-id' extension. This polling should be done via OPTIONS.

This draft does not address fan-out or aggregation of subscriptions or notification delivery.

7.1 Example OPTIONS request

The proxy is myproxy.my.com on port 8080

```
OPTIONS * HTTP/1.1
Host: myproxy.my.com:8080
Compliance: uri=http://ietf.org/http-ext/route-id



Compliance: uri=http://ietf.org/http/v11

A successful response:
HTTP/1.1 200 Ok
Compliance: uri=http://ietf.org/http-ext/route-id
```

```
Compliance: uri=http://ietf.org/http/v11
```

Lack of a successful response indicates non-compliance with HTTP/1.1 and Route-ID extensions.

7.2 Routing with a compliant Proxy

Compliant proxy servers are expected to include a Route-ID header as they forward subscription request messages. As they include the route-id header, the rid parameter is to be an integer greater than the highest rid parameter of any other route-id header already in a message. This allows a deterministic route list for the message transit.

Servers who send responses to messages which include route-id headers are expected to consume the highest rid parameter route-id header, strip it from the message and use that address as the connection proxy for notification delivery.

7.3 Routing with a non-compliant Proxy

Since no deterministic way exists to determine an appropriate call-back path for notification delivery, subscribers should NOT select asynchronous call-back as a delivery type.

8 Examples

```
SUBSCRIBE URI HTTP/1.1
Notification-Type: CoreNotificationType
Call-back: URI [ URI  ]
Delivery-control: poll-interval = seconds
Subscription-Lifetime: seconds
Notification-Version: 1.0
```

200 Subscribed

```
Notification-Version: 1.0
Call-back: [ accepted callback URIs]
Delivery-control: poll-interval = acc
Subscription-Lifetime: seconds
Subscription-ID: FFFFF

NOTIFY cbURI HTTP/1.1
Subscription-ID: FFFF (optional)
Notification-Type: UPDATE
Notification-Version: 1.0
Content-Type:


HTTP/1.1 200 Notification Acknowledged

[NOTIFY without subscription-ID is meant for
 Unsolicited notifications like the IM case.]

POLL URI HTTP/1.1
Subscription-ID: FFFF
Delivery-control: Wait-time=20
```

200 None pending

```
Server: NSERV/1.0
Date: Tue Jan 05, 2068

POLL URI HTTP/1.1
Subscription-ID: FFFF
Delivery-control: Wait-time=20

200 Notification
Subscription-ID: FFFF
Notification-Type: UPDATE


UNSUBSCRIBE uri HTTP/1.1
Subscription-Id: NNN
Call-back: uri

HTTP/1.1 200 Ok
Extended-Response: 20243 ;   ns=http://extension.iana.org/http/GENA/1/1/1;
comment = Subscription Terminated
```

9 Security Considerations

Servers responding to subscription requests should be careful to implement a rational security policy for subscriptions which protects the event notification data about resources as well as the resources themselves. Allowing a subscriber to receive notifications on a resource which that subscriber would not normally have access to may unknowingly reveal information about that resource or the contents itself.

10 IANA Considerations

This document introduces a new entry into an HTTP extension registry. This extension is:

http://extension.iana.org/http/GENA/1/1/1

11 Copyright

The following copyright notice is copied from RFC 2026 [Bradner, 1996], Section 10.4, and describes the applicable copyright for this document.

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 assignees.

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.

12 Intellectual Property

The following notice is copied from RFC 2026 [Bradner, 1996], Section 10.4, and describes the position of the IETF concerning intellectual property claims made against this document.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use other technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

13 References

13.1 [Bradner, 1996]

S. Bradner, "The Internet Standards Process - Revision 3." RFC 2026, BCP 9. Harvard University. October, 1996.

13.2 [Fielding et al., 1997]

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1." RFC 2068. U.C. Irvine, DEC, MIT/LCS. January, 1997.

13.3 [Frystyk et al., 1998]

H. Frystyk Nielsen, P. Leach, Microsoft Scott Lawrence, Agranat Systems Mandatory Extensions in HTTP ,<W3C draft-frystyk-http-mandatory>, March 13, 1998

14 Authors' Addresses

Josh Cohen
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Email: <joshco@microsoft.com>

Sonu Aggarwal
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Email: sonuag@microsoft.com