Network Working Group   Y. Goland
Internet-Draft   BEA
Expires: May 11, 2006   November 7, 2005

SOA-Reliability (SOA-Rity) for HTTP
draft-goland-http-reliability-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 11, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

SOAR-ity is intended to allow for "reliable" (this term is almost always a misnomer) messaging over HTTP. It achieves this goal by introducing two new request headers, Message-ID which provides a unique ID for a message and MsgCreate which contains the date and time on which the first instance of the message with the associated Message-ID was sent. The purpose of the Message-ID/MsgCreate pair is to allow any HTTP request (e.g. any HTTP method can be used) to be repeated multiple times with a guarantee that the message will be processed no more than one time. In essence it makes any HTTP method

call idempotent.

Table of Contents

1. Introduction

SOAR-ity is intended to allow for "reliable" (this term is almost always a misnomer) messaging over HTTP. It achieves this goal by introducing two new request headers, Message-ID which provides a unique ID for a message and MsgCreate which contains the date and time on which the first instance of the message with the associated Message-ID was sent. The purpose of the Message-ID/MsgCreate pair is to allow any HTTP request (e.g. any HTTP method can be used) to be repeated multiple times with a guarantee that the message will be processed no more than one time. In essence it makes any HTTP method call idempotent.

When a SOAR-ity message is recieved the MsgCreate value is checked to make sure the time/date is within the resource's current "time window". So, for example, if the resource only remembers reliable messages for 10 hours then a MsgCreate value that is more than 10 hours old is outside of the "time window" and has to be rejected. If the MsgCreate value is within the time window then the resource will check to see if it has a record of the Message-ID value. If it does then it will return a cached copy of the response it sent the first time it received this request. If the Message-ID value hasn't been seen before then the resource will process the request and both send and cache its response. It is the cached response that will be used if the request is ever repeated.

The reader may benefit from reviewing both [ExactlyOnce] and [SoaReliableMessaging] to get more background on the motivation for SOA-Rity's design.

2. Terminology

The term resource is used in this specification instead of the more usual HTTP term server in order to indicated that the behavior specified by this specification can change on a resource by resource basis.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The HTTP header definitions are given using ABNF as defined in [RFC2234].

3. Making a Reliable Request over HTTP

A reliable HTTP request MUST contain the following two headers:

Header Name: MsgCreate
Header Value: MsgCreate = "MsgCreate" ":" rfc1123-date ; the rfc1123-date production is defined by 3.3.1 of [RFC2616], note that when the production is used for the MsgCreate header it MUST provide its value in GMT
Header Description: Specifies the date and time when the first instance of a message baring the associated message ID was created. This value MUST be reproduced with the same value on every instance of the message carrying the associated message ID.

and

Header Name: Message-ID
Header Value: Message-ID = "Message-ID" ":" URI ; the URI production is defined by [RFC2396]
Header Description: A message ID specified as a globally unique URI that will make this message unique amongst all messages generated across all senders for all time. It is recommended that implementers look at [RFC4122] as an easy way to meet the previous requirement.

Example:

```
POST /resource/foo HTTP/1.1
Host: www.example.com
Content-Type: text/xml
MsgCreate: 14 Oct 2005 16:20:00 GMT
Message-ID: urn:uuid:72dfcac0-3d09-11da-8cd6-0800200c9a66
...
```

4. Receiving a Response to a Reliable Request over HTTP

The SOARITY response header is used in responses in the manner specified in the rest of this section.

Header Name: SOARITY
Header Value: SOARITY = "SOARITY" ":" ("supported" / "unsupported" / "MsgCreate/Message-ID Rejected")
Header Description: Provides information about the resource's support for SOA-Rity. The "supported" value is returned on all successful responses to confirm to the receiver that the resource has honored the SOA-Rity headers. "Unsupported" appears on rejections to confirm that the receiver does not support SOA-Rity

for that sender at that time. The "MsgCreate/Message-ID Rejected" value specifies that the combination of the submitted MsgCreate and Message-ID values have been rejected, the request SHOULD NOT be repeated with those values.

If a resource receives a request with the MsgCreate header but not with the Message-ID header then the request MUST be rejected with a 400 (Bad Request) response that MAY contain the SOARITY header with an appropriate (e.g. "supported" or "unsupported") value. It is, however, legal to send a request with a Message-ID header but without a MsgCreate header, in that case the request MUST NOT be treated as requesting reliable delivery. The reason for the difference in treatment is that the MsgCreate header is only intended for reliable messaging usage while the Message-ID header has potential utility outside of reliable messaging.

If a resource receives a request with the SOA-Rity headers but chooses not to honor those headers (for whatever reason) then the resource MUST return a 412 (precondition failed) response with the SOARITY header set to the value "unsupported". Clients who wish to make sure that their requests cannot be mistakenly processed unreliably on a request that included a demand for reliability by resources that do not support this specification should investigate [RFC2774]. Note, however, that this specification does not require that resources support [RFC2774].

If the resource does support the SOA-Rity headers then the resource MUST first validate that the date/time specified in the MsgCreate header is not beyond the resource's time window for remembering IDs. If the value is beyond the resource's time window then the message MUST be rejected with a 403 (Forbidden) response with the SOARITY header set to "MsgCreate/Message-ID Rejected".

If a request contains a MsgCreate and Message-ID header where the Message-ID has been seen previously but with a MsgCreate value that resolved to a different time/date then, assuming the MsgCreate value is within the current time window, the request MUST be rejected with a 403 (Forbidden) response with the SOARITY header set to "MsgCreate/Message-ID Rejected".

The resource MAY assume that if it receives a message with a MsgCreate/Message-ID header combination it has seen before then the new request is identical to the previous request with the same MsgCreate/Message-ID values. In other words if a client makes a mistake and repeats a MsgCreate/Message-ID pair on two different requests then the resource is under no obligation to detect this error. Still, well behaved resources should at least validate that the HTTP method is the same (e.g. a GET and POST with the same

MsgCreate/Message-ID values should be rejected) as well as the request body and key headers. If a resource does choose to remember information about a request other than its MsgCreate/Message-ID pair then it MUST send a 400 (Bad Request) to any clients who use the same MsgCreate/Message-ID pair twice on two materially different messages.

The term 'materially different' in the previous sentence is intended to indicate that not all headers need to be the same for two requests with the same MsgCreate/Message-ID values to be treated as identical. For example, the Date request header, the If-* headers, the Range header, the user-agent header, etc. could all be different on repeated requests but these differences are not 'material' in terms of stating that the repeated request with a specific MsgCreate/Message-ID pair is the same as the original request.

A MsgCreate/Message-ID pair MUST be constrained to only apply to a specific authenticated requester when authentication is in use.

If the MsgCreate is within the local time window then the resource MUST determine if this request has been answered previously. If so then the resource MUST return the same response as the one it originally sent. Note however that the term 'same' only applies to the response code and the body, the value of the headers may need to be altered in some cases although the default behavior should be to return the same header values as on the original response.

If, for whatever reason, the resource cannot honor the requirement to return the same response then the resource MUST either return a 403 (Forbidden) response with the SOARITY header set to "MsgCreate/Message-ID Rejected" (if the condition is permanent) or a 503 (Service Unavailable) with a SOARITY header value of "supported" (if the condition is temporary).

If-* and Range headers on repeated requests should be honored following the resource's normal policies (in the case of range) and HTTP's requirements (in the case of If-*). Although these headers on a repeated request can cause a different response to be received than what was sent in response to the original request the alterations only affect the response body and not the original outcome of the method's processing. In other words, if a client submits a POST using the SOA-Rity headers which generates a HTTP response body and then later repeats the POST but includes a Range request then the returned body (assuming the resource honors Range) would contain only the subset of the original response specified by the range. But, the repeated response would just be a copy of the original HTML response, no new processing would occur and SOA-Rity's idempotent promise would be honored.

If a resource that supports SOA-Rity receives a message whose MsgCreate is within the current time window and with a Message-ID that the resource has not previously seen then the response MUST respond normally to the request but the response MUST contain the SOARITY header set to "supported".

The logic regarding the MsgCreate header requires that the clocks at the client and resource be reasonably synchronized. As such to be compliant with this specification both a client and resource MUST have clocks and MUST take 'reasonable' actions to ensure those clocks are accurate. Deploying and properly configuring a Network Time Protocol (NTP) client [RFC958] is an example of a 'reasonable' action to ensure a clock is accurate. Note however that clocks, even with NTP, can still skew and messages can be delayed for non-trivial periods of time during network transmission, especially if intermediaries are involved. Therefore resources are encouraged to be generous in the size of their time windows and clients are encouraged to be stingy in their expectations of how large the windows will be.

Example:

```
HTTP/1.1 200 It's all cool
SOARITY: supported
...
```

5. OPTIONS Support

A resource that supports this specification MUST return the SOARITY header on an OPTIONS response with the value of either "supported" or "unsupported" depending on the resource's support for SOA-Rity for the particular client who made the request (assuming authentication is involved). In other words, it is perfectly legal for a resource to send a SOARITY header with "supported" to one authenticated client and "unsupported" to a different authenticated client.

6. Proxies & Caches

Successful responses to reliable requests SHOULD include the HTTP 1.1 Vary header with values that point to the Message-ID and MsgCreate headers. Please refer to section 14.8 of [RFC2616] for details of how caching and shared caches interact in order to make sure that a private response is not inadvertently cached by a shared cached.

7. Security Considerations

It is, in theory, possible for a client to repeat a request made by some other client but using the same Message-ID and MsgCreate values. If successful such an attempt would allow one client to see the response sent to another client. A simple way to prevent such an attack is to authenticate all requesters and, when recording MsgCreate/Message-ID values, to also record the identity of the requester. In the future if a request is received with the same Message-ID/MsgCreate values it will only be honored if the client is authenticated as the original sender.

The provision of reliable messaging can entail the use of a non-trivial amount of resources. As such large numbers of reliable messaging requests can constitute a denial of service attack. Therefore it is reasonable to only provide SOA-Rity support to authenticated requesters or to use other mechanisms to rate limit who can make requests.

8. IANA Considerations

The following HTTP headers are submitted for provisional message header field registration per [RFC3864].

```
Header Field Name:  MsgCreate
Applicable Protocol:  http
Status:  provisional
Author/Change controller:
   Name:  Yaron Y. Goland
   Email:  soarityietfsubmission@goland.org
   Home Page URI:  http://www.goland.org
Defined In:  Section 3 of this document.

Header Field Name:  Message-ID
Applicable Protocol:  http
Status:  provisional
Author/Change controller:
   Name:  Yaron Y. Goland
   Email:  soarityietfsubmission@goland.org
   Home Page URI:  http://www.goland.org
Defined In:  Section 3 of this document.

Header Field Name:  SOARITY
```

Applicable Protocol: http
Status: provisional
Author/Change controller:
   Name: Yaron Y. Goland
   Email: soarityietfsubmission@goland.org
   Home Page URI: http://www.goland.org
Defined In: Section 4 of this document.

Appendix A. Q&A

Appendix A.1. SOA-Rity?!?!!?!?!

O.k. O.k. I know, it's not the best name in the world. Although I must admit that people I share it with either laugh or wretch when they hear it which is more or less the range of reactions I'm looking for. :) I thought about SOA-R but that sounds like sore. I also thought about SOA-Ring but that seemed forced. I'm open to better ideas.

Appendix A.2. Why not put the MsgCreate/Message-ID values into the URL?

When I was first designing the HTTP version of this protocol I thought of sticking the MsgCreate and Message-ID values directly into the URL. E.g. something along the lines of:

```
POST /resource/foo?MsgCreate=14Oct200516:20:00GMT&Message-ID=urn:
uuid:72dfcac0-3d09-11da-8cd6-0800200c9a66 HTTP/1.1
Host: www.example.com
Content-Type: text/xml
...
```

I changed my mind for a number of reasons:

- o In theory URLs are supposed to be of unlimited length but in practice there are often size limits and the MsgCreate/Message-ID values are not small.
- o There is no standard for how to place arguments on a URL so I would, in effect, be telling people "Your URLS MUST support a query option and MUST use the '&' character as a delimiter." I generally don't like telling people how to form their URLs.
- o I don't think MsgCreate/Message-ID belong in the URL any more than an eTag does.

Appendix A.3. Why is there no time window declaration?

It would be easy to throw in a header that specifies how long a service promises to remember messages but I think the value would be

so misleading as to be, on balance, a bad idea. First, servers crash and forget things (which they shouldn't, but oh well). Second, a service may have different windows for different people at different times. In truth the window value would at best be a 'rough estimate' rather than a real promise. I suspect the best way to present time window information is as part of a human readable description of what the server is and how it works, this is exactly the sort of thing one should get back in an OPTIONS response.

Appendix A.4. **Why introduce the MsgCreate Header? Why not just use the** HTTP Date header?

It would be easy enough to just re-use the HTTP date header rather than introduce the MsgCreate header. The main consequence would probably be to create a SOARITY request header to definitively identify a request as requiring SOA-Rity support.

One could even argue that this is the correct path forward since [RFC2616] explicitly states that the semantics for the HTTP Date header are the same as the [RFC822] orig-date production which is used to specify the email Date header. In email the idea of resending a message is a common one and two different date headers are provided, a Date header reflecting when the message was originally created and a separate Resent-Date header to identify when the message was retried.

The problem is that while HTTP may have stated that its Date header was to have [RFC822] semantics in practice nobody I'm aware of has made it a habit of repeating HTTP requests (even idempotent ones) with the same date header as the original request. In fact, I suspect if one were to ask most HTTP implementers what they thought the meaning of the HTTP Date header was they would probably answer "it provides the time the message was sent." The concept that the time when the message was sent and the time when it was actually generated could be radically different (due to retries) hasn't been entertained before in HTTP requests (it is, of course, quite common in cached replies however).

So the question is - if this specification were to replace the MsgCreate header with the Date header and simply mandate that all SOA-Rity requests MUST include a Date header ([RFC2616] makes them optional on requests) as well as a new SOARITY request header would anything break? Would this confuse any proxies? Screw up any servers?

I just don't know so I have erred on the side of caution and introduced the MsgCreate header.

Appendix A.5. Why use RFC 2234 ABNF instead of RFC 2616 ABNF?

My main reason is that I wanted to use Harald Alvestrand's ABNF parser, available at http://www.apps.ietf.org/abnf.html. Near as I can tell the choice shouldn't be a big deal because the only substantive difference my cursory examination of [RFC2616]'s ABNF versus the IETF standard [RFC2234]'s ABNF is that [RFC2616] uses "|" to indicate "or" semantics while [RFC2234] uses "/". If it turns out it matters then I'll just use the [RFC2616] format.

Appendix A.6. Why not require RFC 2774 Support?

Because in this case I think it's more trouble than it's likely worth. If a client is really worried that the resource it's talking to doesn't support SOA-Rity then it can make an OPTIONS request. Yes, this still leaves open some race conditions but I just don't think they are common enough to justify requiring everyone to support RFC 2774.

Appendix A.7. Why Didn't You Use the Expect Header?

Because it's useless. It shows a real failure in the IETF RFC vetting process. HTTP/1.1 was explicitly required to be backwards compatible with HTTP/1.0 so if a 1.0 resource got a 1.1 request then all would be well with the world. But the Expect header's functionality is "If you don't have this feature then fail this request". Well, duh, HTTP/1.0 didn't have an expect header so a 1.0 server would just ignore expect all together. So much for guaranteeing failure. So anyone who actually takes RFC 2616 at its word must conclude that the RFC is, in fact, not backwards compatible with HTTP/1.0 and therefore failed in its mission. The right thing to do would be to remove the Expect header. RFC 2774 showed the right way to support Expect style behavior and retain complete backwards compatibility with HTTP/1.0.

Appendix A.8. What about Clockless Systems?

There are a number of tricks that could be used to make this specification work with clockless systems but they all put an extra burden on those with clocks that I just don't think can be justified. If it should turn out that reliable messaging in clockless systems is a real world use case it will always be possible to come out with an extension to this protocol.

Appendix A.9. What about One Ways?

HTTP only knows about request/responses so the spec only addresses that. If someone sends a one-way request (at the application level)

then at the HTTP level they still have to send some kind of response. The tradition is that the response is just a 200 (O.K.) with no response body. It is that 200 response that would be cached by the algorithm above. So the point is that from SOA-Rity's perspective it doesn't care or need to know if a request is part of a synchronous request/response or a one-way at the application level, it all looks the same at the HTTP level.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2396, BCP 14, March 1997.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2616] Fielding, R., Gettys, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol --- HTTP/1.1", RFC 2616, June 1999.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", RFC 3864, BCP 90, September 2004.

9.2. Informative References

[ExactlyOnce] Goland, Y., "How Should An Exactly Once SOA Reliable Messaging System Be Designed? - http://www.goland.org/exactlyonce", October 2005.

[RFC2774] Frystyk, H., Leach, P., and S. Lawrence, "An HTTP Extension Framework", RFC 2774, February 2000.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", RFC 822, August 1982.

[RFC958]   Mills, D., "Network Time Protocol (NTP)", RFC 958, September 1985.

[SoaReliableMessaging]
Goland, Y., "Does SOA Need A Reliable Messaging Protocol? - http://www.goland.org/soareliablemessaging", October 2005.

Author's Address

Yaron Y. Goland
BEA Systems Inc.
999 North Northlake Way
Seattle, WA 98103
US

Email: soarityietfsubmission@goland.org
URI: http://www.goland.org

Full Copyright Statement

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat 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 on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.