

appsawg
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2013

L. Goix
Telecom Italia
K. Li
Huawei Technologies
July 11, 2012

ENUM Service Registration for Social Networking (SN) Services
draft-goix-appsawg-enum-sn-service-02

Abstract

This document registers a Telephone Number Mapping (ENUM) service for Social Networking (SN). Specifically, this document focuses on provisioning 'acct:' URIs (Uniform Resource Identifiers) in ENUM.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 12, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Justification	3
1.2.	Terminology	3
2.	IANA Registration	3
3.	Examples	4
4.	DNS Considerations	5
5.	Security Considerations	5
6.	IANA Considerations	6
7.	Acknowledgements	6
8.	Change log (to be deleted before publication)	6
9.	References	7
9.1.	Normative References	7
9.2.	Informative References	7
	Authors' Addresses	8

1. Introduction

ENUM (E.164 Number Mapping, [RFC6116]) is a system that uses DNS (Domain Name Service, [RFC1034]) to translate telephone numbers, such as '+44 01632 960123', into URIs (Uniform Resource Identifiers, [RFC3986], such as 'acct:user@example.com'. ENUM exists primarily to facilitate the interconnection of systems that rely on telephone numbers with those that use URIs to identify resources.

Social Networking (SN) services allow users to post and retrieve activities (e.g. status updates or media uploads) and related replies. [I-D.saintandre-acct-uri] is proposing a generic URI to identify an SN service account for a particular resource: the 'acct:' URI scheme.

This document registers an enumservice for advertising account information associated with an E.164 number.

1.1. Justification

The requirement to map an SN account with an E.164 number is from the Open Mobile Alliance (OMA) Social Network Web Enabler Release [OMA-SNeW-ER].

In Mobile networks, users are traditionally identified through a Mobile Subscriber ISDN (MSISDN) number. In order to provide SN functionality to mobile subscribers identified by their MSISDN, a mapping is needed to identify a corresponding SN account and provider.

1.2. Terminology

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].

2. IANA Registration

As defined in [RFC6117], the following is a template covering information needed for the registration of the enumservice specified in this document:

```
<record>
  <class>Application-Based, Common</class>
  <type>sn</type>
  <urischeme>acct</urischeme>
  <functionalspec>
    <paragraph>
      This Enumservice indicates that the resource
      can be addressed by the associated URI using
      SN protocols, including
      <xref target='I-D.ietf-appsawg-webfinger' />
      to discover additional information.
    </paragraph>
  </functionalspec>
  <security>
    See <xref target="security" />.
  </security>
  <usage>COMMON</usage>
  <registrationdocs>
    This document.
  </registrationdocs>
  <requesters>
    <xref type="person" data="Laurent_Walter_Goix"/>
  </requesters>
</record>

<people>
  <person id="Laurent_Walter_Goix">
    <name>Laurent-Walter Goix</name>
    <org>Telecom Italia</org>
    <uri>mailto:laurentwalter.goix@telecomitalia.it</uri>
    <updated>2012-07-10</updated>
  </person>
</people>
```

3. Examples

The following is an example of the use of the enumservice registered by this document in a NAPTR resource record for phone number +44 01632 960123.

```
$ORIGIN 3.2.1.0.6.9.2.3.6.1.0.4.4.e164.arpa.
```

```
IN NAPTR 10 100 "u" "E2U+sn" "!.^.*$!acct:john.doe@example.com!" .
```

4. DNS Considerations

If no "E2U+sn" NAPTR record exist for the requested number, the resolution process over issuing ENUM queries may be recursively performed over E.164 numbers identified by other NAPTR records for the requested number pointing to "tel" URIs [RFC3966]. Whilst this process is useful to support, amongst others, number portability as per [RFC4769], Section 4, this may also create potential loops in this resolution process. As such ENUM clients performing such ENUM queries over "tel" URIs to identify "acct" URIs SHOULD understand the "enumdi" indicator defined in [RFC4759]. In particular, if the result of the query does not include "E2U+sn" NAPTR records, and includes a NAPTR resource record containing a "tel" URI that has the same E.164 number, or a "tel" URI with the "enumdi" parameter set, that client SHOULD NOT perform subsequent ENUM queries over such numbers and SHOULD consider that the original requested number cannot be mapped.

Furthermore the client MAY stop performing subsequent ENUM queries after the fifth recursive query as suggested in [RFC6116] section 5.2.1.

5. Security Considerations

DNS, as used by ENUM, is a global, distributed database. Should implementers of this specification use e164.arpa or any other publicly available domain as the tree for maintaining PSTN Enumservice data, this information would be visible to anyone anonymously.

As noted earlier, carriers, service providers, and other users may choose not to publish such information in the public e164.arpa tree. They may instead simply publish this in an internal ENUM infrastructure that is only able to be queried by trusted elements of their network, thus limiting threats.

Per se, this enumservice does not introduce specific security considerations beyond [RFC6116], section 7. However, it has to be acknowledged that the proposed Enumservice could lead to the discovery or disclosure of Personally Identifiable Information (PII) if used in combination with the WebFinger protocol. Please see [I-D.ietf-appsawg-webfinger], section 10 for additional information regarding WebFinger security to avoid unwanted disclosure of information.

Similar security concerns are associated with potential attacks against an underlying Social Networking system. Unlike a traditional

telephone number, and as per the indirect nature of SN communication (typically through an intermediary network element), the resource identified by an 'acct:' URI may not be subject to direct communication with other peers. However (e.g. in case of private message exchange) it may require that peers provide cryptographic credentials for authentication and authorization before messages are exchanged. For this reason, SN protocols should have a number of security requirements that call for authentication, integrity and confidentiality properties, and similar measures to prevent such attacks.

6. IANA Considerations

This document requests the IANA registration of the Enumservice with Type "sn" according to the definitions in this document, [RFC6116] and [RFC6117].

Details of the registration are given in Section 2.

7. Acknowledgements

The authors would like to thank Gonzalo Salgueiro, Paul Jones, Lawrence Conroy, and Enrico Marocco for their valuable feedback to improve this document.

8. Change log (to be deleted before publication)

-02 Draft

- * Updated references to acct: URI I-D and OMA specifications
- * Added changelog section

-01 Draft

- * Made changes to example phone numbers
- * Updated security and DNS considerations
- * Fixed IANA registration template
- * Added acknowledgment section

9. References

9.1. Normative References

- [I-D.ietf-appsawg-webfinger]
Jones, P., Salgueiro, G., and J. Smarr, "WebFinger",
draft-ietf-appsawg-webfinger-00 (work in progress),
July 2012.
- [I-D.saintandre-acct-uri]
Saint-Andre, P., "The 'acct' URI Scheme",
draft-saintandre-acct-uri-01 (work in progress),
July 2012.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
STD 13, RFC 1034, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers",
RFC 3966, December 2004.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
Resource Identifier (URI): Generic Syntax", STD 66,
RFC 3986, January 2005.
- [RFC4759] Stastny, R., Shockey, R., and L. Conroy, "The ENUM Dip
Indicator Parameter for the "tel" URI", RFC 4759,
December 2006.
- [RFC4769] Livingood, J. and R. Shockey, "IANA Registration for an
Enumservice Containing Public Switched Telephone Network
(PSTN) Signaling Information", RFC 4769, November 2006.
- [RFC6116] Bradner, S., Conroy, L., and K. Fujiwara, "The E.164 to
Uniform Resource Identifiers (URI) Dynamic Delegation
Discovery System (DDDS) Application (ENUM)", RFC 6116,
March 2011.
- [RFC6117] Hoeneisen, B., Mayrhofer, A., and J. Livingood, "IANA
Registration of Enumservices: Guide, Template, and IANA
Considerations", RFC 6117, March 2011.

9.2. Informative References

- [OMA-SNeW-ER]
Open Mobile Alliance, "Social Network Web Enabler", OMA-
ER-SNeW-V1_0 20120702-D, July 2012.

Authors' Addresses

Laurent-Walter Goix
Telecom Italia
P.za Einaudi, 8
Milano 20124
Italy

Email: laurentwalter.goix@telecomitalia.it

Kepeng Li
Huawei Technologies
Huawei Base, Bantian, Longgang District
Shenzhen, Guangdong 518129
P. R. China

Phone: +86-755-28974289
Email: likepeng@huawei.com

Applications Area WG (APPSAWG)
Internet-Draft
Intended Status: Informational
Expires: December 9, 2012

S. Moonesamy, Ed.

June 7, 2012

The "about" URI Scheme
draft-ietf-appsawg-about-uri-scheme-07

Abstract

This document describes the "about" URI scheme, which is widely used by web browsers and some other applications to designate access to their internal resources, such as settings, application information, hidden built-in functionality, and so on.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and 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/lid-abstracts.html>

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

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. URI Scheme Specification	2
2.1. URI Scheme Syntax	2
2.2. URI Scheme Semantics	2
2.2.1. Well-known "about" URIs	3
2.3. Encoding Considerations	3
3. "about:blank"	3
4. Security Considerations	3
5. IANA Considerations	4
5.1. URI Scheme Registration	4
5.2. A Registry for Well-known Tokens	4
5.2.1. Registration procedure	5
6. References	5
6.1. Normative References	5
6.2. Informative References	6
Appendix A. Acknowledgments	6
Authors' Addresses	6

1. Introduction

This document describes the "about" Uniform Resource Identifier (URI) scheme. The "about" URI scheme is currently widely used by Web browsers to designate access to their internal resources such as settings, application information, so called "Easter eggs" (i.e. hidden feature or joke in an application).

2. URI Scheme Specification

2.1. URI Scheme Syntax

The "about" URI syntactically conforms to the <about-uri> rule below, expressed using Augmented Backus-Naur Form (ABNF) [RFC5234]:

```
about-uri = "about:" about-token [ about-query ] [ about-fragment ]
about-token = *pchar
about-query = "?" query
about-fragment = "#" fragment
pchar      = <as specified in RFC 3986, Appendix A>
query      = <as specified in RFC 3986, Appendix A>
fragment   = <as specified in RFC 3986, Appendix A>
```

2.2. URI Scheme Semantics

The resource which a particular "about" URI references is denoted by <about-token> part of the URI. It is not a hierarchical element for a naming authority. The <about-query> specifies additional information about its handling and/or the information that should be returned by the resource which the URI references.

It is impossible to specify a binding between all the possible tokens and the semantics of "about" URIs that would contain such tokens. Therefore the resource referenced by the URI is generally considered as specific to a Web browser implementation.

2.2.1. Well-known "about" URIs

Some <about-token>s have been reserved as the behavior when the resource is referenced is well-known (Well-known tokens).

A well-known "about" URI is a URI that has a well-known token as its <about-token> part. It is recommended that such URIs be handled in accordance with the specification referenced in the Well-known Tokens registry (see Section 5.2).

Well-known "about" URIs are intended to be registered when there is a need to codify the behavior of particular <about-token>.

2.3. Encoding Considerations

"about" URIs are subject to encoding rules defined in RFC 3986 [RFC3986].

3. "about:blank"

This document defines one well-known token: "blank". The "about:blank" URI refers to a resource represented in the browser by a blank page.

4. Security Considerations

Security considerations for URIs are discussed in Section 7 of RFC 3986 [RFC3986]. However, most of those provisions do not apply to the "about" URI scheme as they are mainly scoped to schemes used in the Internet.

"about" URIs can sometimes refer to sensitive information, such as user passwords stored in a cache, or parameters that, if changed, could affect user's data. The application therefore needs to ensure that the user's data is secured and no threats are imposed

by "about" URIs.

5. IANA Considerations

5.1. URI Scheme Registration

The registration of the "about" URI scheme in the "URI Schemes" registry is requested. The information below is provided according to the guidelines from RFC 4395 [RFC4395]:

URI scheme name: about

Status: Permanent

URI scheme syntax: see Section 2.1 of RFC xxxx

URI scheme semantics: see Section 2.2 of RFC xxxx

URI scheme encoding considerations: see Section 2.3 of RFC xxxx

Applications that use the scheme: "about" URIs are predominantly used by Web browsers.

Security considerations: see Section 4 of RFC xxxx

Contact: IETF Applications Area Directors <app-ads@tools.ietf.org>

Author/Change controller: IESG <iesg@ietf.org> (on behalf of the IETF)

References: see Section 5 of RFC xxxx

[RFC Editor: Please replace xxxx with assigned RFC number]

5.2. A Registry for Well-known Tokens

This document creates the '"about" URI Well-known Tokens' registry.

The registry entries consist of three fields: Well-known Token, Description and Reference. The Well-known Token field has to conform to <about-token> production defined in Section 2.1. The initial set of assignments is as follows:

Well-known Token	Description	Reference
blank	The about:blank URI references a	RFC xxxx

```
|          | blank page.          |          |
+-----+-----+-----+-----+-----+
```

5.2.1. Registration procedure

The registration policy for this registry is "First Come First Served" as described in RFC 5226 [RFC5226]. The registrant of the token should provide the information mentioned in the following registration template:

- o Registered Token: The desired Well-known token to be used in "about" URIs.
- o Intended usage: A short description of how "about" URIs with the registered token is handled including information about the referenced resource.
- o Contact/Change controller: Person (including contact information) authorized to change this registration.
- o Specification: A stable reference to a document which specifies the registered "about" URI. The question of interoperability does not arise. The key motivation is to have a reference to a specification documenting well-known behavior of the "about" URI in Web browsers. As a rule of thumb if the behavior is common to two or more Web browser implementations it can be considered as well-known. An existing assignment may be duplicated if the registered token is used in more than one Web browser implementation.

The following is a template for the "blank" token:

- o Registered Token: blank
- o Intended usage: The about:blank URI references a blank page.
- o Contact/Change controller: IESG <iesg@ietf.org> (on behalf of IETF).
- o Specification: RFC xxxx. [RFC Editor: Please replace xxxx with assigned RFC number]

6. References

6.1. Normative References

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226,

May 2008.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

6.2. Informative References

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

Appendix A. Acknowledgments

This document has been formed from the draft initially authored by Lachlan Hunt and Joseph Holsten. Additionally, the contributions of Frank Ellermann and Alexey Melnikov are gratefully acknowledged. Barry Leiba and Murray Kucherawy deserve a special credit for providing a great amount of text which has been used in this document.

Lachlan Hunt and Mykyta Yevstifeyev edited previous versions of this document. Tim Bray and John Klensin provided suggestions about how to improve the document.

Authors' Addresses

S. Moonesamy (editor)
76, Ylang Ylang Avenue
Quatre Bornes
Mauritius

EMail: sm+ietf@elandsys.com

Individual submission
Internet-Draft
Intended status: Standards Track
Expires: October 28, 2012

M. Kucherawy
Cloudmark
D. Crocker
Brandenburg InternetWorking
April 26, 2012

Email Greylisting: An Applicability Statement for SMTP
draft-ietf-appsawg-greylisting-09

Abstract

This document describes the art of email greylisting, the practice of providing temporarily degraded service to unknown email clients as an anti-abuse mechanism.

Greylisting is an established mechanism deemed essential to the repertoire of current anti-abuse email filtering systems.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on October 28, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Background	3
1.2. Definitions	4
2. Types of Greylisting	4
2.1. Connection-Level Greylisting	4
2.2. SMTP HELO/EHLO Greylisting	5
2.3. SMTP MAIL Greylisting	5
2.4. SMTP RCPT Greylisting	5
2.5. SMTP DATA Greylisting	6
2.6. Additional Heuristics	7
2.7. Exceptions	7
3. Benefits and Costs	8
4. Unintended Consequences	9
4.1. Unintended Mail Delivery Failures	9
4.2. Unintended SMTP Client Failures	10
4.3. Address Space Saturation	11
5. Recommendations	12
6. Measuring Effectiveness	13
7. IPv6 Applicability	14
8. IANA Considerations	14
9. Security Considerations	14
9.1. Tradeoffs	14
9.2. Database	15
10. References	15
10.1. Normative References	15
10.2. Informative References	15
Appendix A. Acknowledgments	16
Authors' Addresses	16

1. Introduction

Preferred techniques for handling email abuse explicitly identify good actors and bad actors, giving each significantly different qualities service. In some cases an actor does not have a known reputation; this can justify providing degraded service, until there is a basis for providing better service. This latter approach is known as "greylisting". Broadly, the term refers to any degradation of service for an unknown or suspect source, over a period of time (typically measured in minutes or a small number of hours). The narrow use of the term refers to generation of an SMTP temporary failure reply code for traffic from such sources. There are diverse implementations of this basic concept, and, predictably therefore, some blurred terminology.

Absent a perfect abuse detection mechanism that incurs no cost, the current requirement is for an array of techniques to be used by each filtering system. They range in cost and effectiveness and types of abuse techniques they target.

Greylisting happens to be a technique that is cheap and early (in terms of its application in the SMTP sequence) and surprisingly remains useful. Some spamware does indeed route around this technique, but much does not.

The firehose of spam over the Internet represents a wide range of sophistication. Greylisting is useful for removing a large amount of simplistic-but-significant traffic.

This memo documents common greylisting techniques and discusses their benefits and costs. It also defines terminology to enable clear distinction and discussion of these techniques.

There is some confusion in industry that conflates greylisting with an SMTP temporary failure for any reason. The purpose of this memo is also to dispel such confusion.

1.1. Background

For many years, large amounts of spam have been sent through purpose-built software, or "spamware", that supports only a constrained version of SMTP. In particular, such software does not perform retransmission attempts after receiving an SMTP temporary failure. That is, if the spamware cannot deliver a message, it just goes on to the next address in its list since, in spamming, volume counts for far more than reliability. Greylisting exploits this by rejecting mail from unfamiliar sources with a "transient (soft) fail" (4xx) [SMTP] error code. Another application of greylisting is to delay

mail from newly seen IP addresses on the theory that, if it's a spam source, then by the time it retries, it will appear in a list of sources to be filtered, and the mail will not be accepted.

Early references for greylisting descriptions and implementations can be found at [SAUCE] and [PUREMAGIC].

1.2. Definitions

1.2.1. Keywords

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 [KEYWORDS].

1.2.2. E-Mail Architecture Terminology

Readers need to be familiar with the material and terminology discussed in [MAIL] and [EMAIL-ARCH].

2. Types of Greylisting

Greylisting is primarily performed at some phase during an SMTP session. A set of attributes about the client-side SMTP server are used for assessing whether to perform greylisting. At its simplest, the attribute is the IP address of the client and the assessment is whether it has previously connected, recently. More elaborate attribute combinations and more sophisticated assessment, can be performed. The following discussion covers the most common combinations.

2.1. Connection-Level Greylisting

Connection-level greylisting decides whether to accept the (TCP) connection from a "new" [SMTP] client. At this point in the communication between the client and the server, the only information known to the receiving server is the incoming IP address. This, of course, is often (but not always) translatable into a host name.

The typical application of greylisting here is to keep a record of SMTP client IP addresses and/or host names (collectively, "sources") that have been seen. Such a database acts as a cache of known senders and might or might not expire records after some period. If the source is not in the database, or the record of the source has not reached some required minimum age (such as 30 minutes since the initial connection attempt), the server does one of the following, inviting a later retry:

- o returns a 421 SMTP reply, and closes the connection;
- o returns a different 4yz SMTP reply to all further commands in this SMTP session

A useful variant of the basic known/unknown policy is to limit greylisting to those addresses that are on some list of IP addresses known to be affiliated with bad actors. Whereas the simpler policy affects all new connections, including those from good actors, the constrained policy applies greylisting actions only to sites that already have a negative reputation.

2.2. SMTP HELO/EHLO Greylisting

HELO/EHLO greylisting refers to the first command verb in an SMTP session, HELO or EHLO. It includes a single, required parameter that is supposed to contain the client's fully-qualified host name or its literal IP address.

Greylisting implemented at this phase retains a record of sources coupled with HELO/EHLO parameters. It returns 4yz SMTP replies to all commands until the end of the SMTP session if that tuple has not previously been recorded or if the record exists but has not reached some configured minimum age.

2.3. SMTP MAIL Greylisting

MAIL command greylisting refers to the command verb in an SMTP session that initiates a new transaction, MAIL. It includes at least one required parameter that indicates the return email address (RFC5321.MailFrom) of the message being relayed from the client to the server.

Greylisting implemented at this phase retains a record of sources coupled with return email addresses. It returns 4yz SMTP replies to all commands for the remainder of the SMTP session if that tuple has not previously been recorded or if the record exists but has not met some configured minimum age.

2.4. SMTP RCPT Greylisting

RCPT greylisting refers to the command verb in an SMTP session that specifies intended recipients of an email transaction, RCPT. It includes at least one required parameter that indicates the email address of an intended recipient of the message being relayed from the client to the server.

Greylisting implemented at this phase retains a record of tuples that

combines the provided recipient address with any combination of the following:

- o the source, as described above;
- o the return email address;
- o the other recipient addresses of the message (if any)

If the selected tuple is not found in the database, or if the record is present but has not reached some configured minimum age, the greylisting Mail Transfer Agent (MTA) [EMAIL-ARCH] returns 4yz SMTP replies to all commands for the remainder of the SMTP session.

Note that often a match on a tuple involving the first valid RCPT is sufficient to identify a retry correctly, and further checks can be omitted.

2.5. SMTP DATA Greylisting

DATA greylisting refers to the command verb in an SMTP session that transmits the actual message content, DATA, as opposed to its envelope details (see [MAIL]).

This type of greylisting can be performed at two places in the SMTP sequence:

1. on receipt of the DATA command, because at that point the entire envelope has been received (i.e., all MAIL and RCPT commands have been issued);
2. on completion of the DATA command, i.e., after the "." that terminates transmission of the message body, since at that point a digest or other analysis of the message could be performed.

Some implementations do filtering here because there are clients that don't bother checking SMTP reply codes to commands other than DATA. Hence, it can be useful to add greylisting capability at that point in an SMTP session.

Numerous greylisting policies are possible at this point. All of them retain a record of tuples that combine the various parts of the SMTP transaction in some combination, including:

- o the source, as described above;
- o the return email address;

- o the recipients of the message, as a set or individually;
- o identifiers in the message header, such as the contents of the RFC5322.From or RFC5322.To fields;
- o other prominent parts of the content, such as the RFC5322.Subject field;
- o a digest of some or all of the message content, as a test for uniqueness;
- o analysis of arbitrary portions of the message body.

(The last four items in that list are only possible at the end of DATA, not on receipt of the DATA command.)

If the selected tuple is not found in the database, or if the record exists but has not reached some configured minimum age, the greylisting MTA returns 4yz SMTP replies to all commands for the remainder of the SMTP session.

2.6. Additional Heuristics

Since greylisting seeks to target spam senders, it follows that being able to identify spamware within the SMTP context beyond the simple notion of "not seen before" would be desirable. A more targeted approach might also include in its selection such heuristics as:

- o if a [DNSBL] lists an IP address but the implementer wishes to be cautious with mitigation actions rather than blocking traffic from the IP address outright, then subject it to greylisting;
- o if the value found in a PTR record follows common naming patterns for dynamic IP addresses, then subject it to greylisting.

2.7. Exceptions

Most greylisting systems provide for an exception mechanism, allowing one to specify IP addresses, IP address [CIDR] blocks, hostnames or domain names that are exempt from greylisting checks and thus whose SMTP client sessions are not subject to such interference.

Likely candidates to be excepted from greylisting include those known not to retry according to a pattern that will be observed as legitimate, and those that send so rarely that they will age out of the database. In both cases the excepted source is known not to be an abusive one by the site implementing greylisting. Otherwise, typical non-abusive senders will enter the exception list on the

first proper retry, and remain there permanently.

One could also use a [DNSBL] that lists known good hosts as a greylisting exception set.

3. Benefits and Costs

The most obvious benefit with any of the above techniques is that spamware generally does not retry, and is therefore less likely to succeed, absent a record of a previous delivery attempts.

The most obvious detriment to implementing greylisting is the imposition of delay on legitimate mail. Some popular MTAs do not retry failed delivery attempts for an hour or more, which can cause expensive delays when delivery of mail is time-critical. Worse, some legitimate MTAs do not retry at all. (Note however that non-retrying clients are not fully SMTP-capable, per Section 2.1 of [SMTP]. A client does not know, nor is it entitled to know, the reason for the temporary failure status code being returned; greylisting could be in effect, or it could be caused by a local resource issue at the server. A client therefore needs to be equipped to retry in order to be considered fully capable.)

The counterargument to this "false positive" problem is that email has always been a "best-effort" mechanism, and thus this cost is ultimately low in comparison to the cost of dealing with high volumes of unwanted mail. Still, the actual effect of such delays can be significant, such as altering the tone or flow of a multi-participant discussion to a mailing list.

The cache of information stored about SMTP client history does not benefit legitimate clients that are already listed for acceptance, when the clients are subjected to any kind of reconfiguration, especially such as network renumbering. To the greylisting implementation, such clients are once again unknown, and they will once again be subjected to the delay.

Another obvious cost is for the required database. It has to be large enough to keep the necessary history and fast enough to avoid excessive inefficiencies in the server's operations. The primary consideration is the maximum age of records in the database. If records age out too soon, then hosts that do retry per [SMTP] will be periodically subjected to greylisting even though they are well-behaved; if records age out after too long a period, then eventually spamware that launches a new campaign will not be identified as "unknown" in this manner, and will not be required to retry.

Presuming that known friendly senders will be manually configured as exceptions to the greylisting check, a steady state will eventually be reached wherein the only mail that is delayed is mail from an IP address that has never sent mail before. Experience suggests that the vast majority of mail comes from places on a developed exception list, so after a training period, only a small proportion of mail is actually affected. The training period could be replaced by processing a history of email traffic and adding the IP addresses from which most traffic arrives to the exception list.

Applying greylisting based on actual message content (i.e., post-DATA) is substantially more expensive than any of the other alternatives both in terms of the resources required to accept and temporarily store a complete message body (which can be quite substantial) and any processing that is done on that content. As a consequence, such methods incur more cost during the session and thus is not a typical practice.

4. Unintended Consequences

4.1. Unintended Mail Delivery Failures

There are a few failure modes of greylisting that are worth considering. For example, consider an email message intended for user@example.com. The example.com domain is served by two receiving mail servers, one called mail1.example.com and one called mail2.example.com. On the first delivery attempt, mail1.example.com greylists the client, and thus the client places the message in its outgoing queue for later retry. Later, when a retry is attempted, mail2.example.com is selected for the delivery, either because mail1.example.com is unavailable or because a round-robin [DNS] evaluation produces that result. However, the two example.com hosts do not share greylisting databases, so the second host again denies the attempt. Thus, although example.com has sought to improve its email throughput by having two servers, it has in fact amplified the problem of legitimate mail delay introduced by greylisting.

Similarly, consider a site with multiple outbound MTAs that share a common queue. On a first outbound delivery attempt to example.com, the attempt is grey listed. On a later retry, a different outbound MTA is selected, which means example.com sees a different source, and once again greylisting occurs on the same message. The same effect can result from the use of [DHCP], where the IP address of an outbound MTA changes between attempts.

For systems that do DATA-level greylisting, if any part of the message has changed since the first attempt, the tuple constructed

might be different than the one for the first attempt, and the delivery is again greylisted. Some MTAs do reformulate portions of the message at submission time and this can produce visible differences for each attempt.

A host that sends mail to a particular destination infrequently might not remain "known" in the receiving server's database and will therefore be greylisted for a high percentage of mail despite possibly being a legitimate sender.

All of these and other similar cases can cause greylisting to be applied improperly to legitimate MTAs multiple times, leading to long delays in delivery or ultimately the return of the message to its sender. Other side effects include out-of-order delivery of related sequenced messages.

Address translation technologies such as [NAT] cause distinct MTAs to appear to come from a common IP address. This can cause greylisting to be applied only to the first connection attempt from the shared IP address, meaning future MTAs connecting for the first time will be exempted from the protection greylisting provides.

4.2. Unintended SMTP Client Failures

Atypical SMTP client behaviours also need to be considered when deploying greylisting.

Some clients do not retry messages for very long periods. Popular open source MTAs implement increasing backoff times when messages receive temporary failure messages and/or degrade queue priority for very large messages. This means greylisting introduces even more delay for MTAs implementing such schemes, and the delay can become large enough to become a nuisance to users.

Some clients do not retry messages at all, in violation of [SMTP]. This means greylisting will cause outright delivery failure right away for sources, envelopes, or messages that it has not seen before, regardless of the client attempting the delivery, essentially treating legitimate mail and spam the same.

If a greylisting scheme requires a database record to have reached a certain age rather than merely testing for the presence of the record in the database, and the client has a retry schedule that is too aggressive, the client could be subjected to rate limiting by the MTA independent of the restrictions imposed by greylisting.

Some SMTP implementations make the error of treating all error codes as fatal, contrary to [SMTP]; that is, a 4yz response is treated as

if it were a 5yz response, and the message is returned to the sender as undeliverable. This can result in such things as inadvertent removal from mailing lists in response to the perceived rejections.

Some clients encode message-specific details in the address parameter to the [SMTP] MAIL command. If doing so causes the parameter to change between retry attempts, a greylisting implementation could see it as a new delivery rather than a retry, and disallow the delivery. In such cases, the mail will never be delivered, and will be returned to the sender after the retry timeout expires.

A client subjected to greylisting might move to the next host found in the ordered [DNS] MX record set for the destination domain and re-attempt delivery. This has several considerations of its own:

- o An increase in traffic to those alternate servers merely as a result of greylisting.
- o Alternate (MX) servers SHOULD share the same greylisting database. When they do not -- as is often true when the servers occupy different Administrative Management Domains (ADMDs) -- SMTP clients can see variable treatment if they try to send to different MX hosts.
- o When alternate MX servers relay mail back to the "primary" MX server, the latter SHOULD be configured to permit the other servers to relay mail without being subjected to greylisting.

There are some applications that connect to an SMTP server and simulate a transaction up to the point of sending the RCPT command in an attempt to confirm that an address is valid. Some of these are legitimate applications (e.g., mailing list servers) and others are automated programs that attempt to ascertain valid addresses to which to send spam (a "directory harvesting" attack). Greylisting can interfere with both instances, with harmful effects on the former.

4.3. Address Space Saturation

Greylisting is obviously not a fool-proof solution to avoiding abusive traffic. Bad actors that send mail with just enough frequency to avoid having their records expire will never be caught by this mechanism after the first instance.

Where this is a concern, combining greylisting with some form of reputation service that estimates the likely behaviour for IP addresses that are not intercepted by the greylisting function would be a good choice.

5. Recommendations

The following practices are RECOMMENDED based on collected experience:

1. Implement greylisting based a tuple consisting of (IP address, RFC5321.MailFrom, and the first RFC5321.RcptTo). It has shown sufficient to use only the first RFC5321.RcptTo as legitimate MTAs appear not to reorder recipients between retries. Including RFC5321.MailFrom improves accuracy where the IP address is being matched in clusters (e.g., CIDR blocks) rather than precisely (see below). After a successful retry, allow all further [SMTP] traffic from the IP address in that tuple regardless of envelope information.
2. Include a configurable time window within which a retry from a greylisted host is considered, and ignored otherwise. The time window needs to be configured to contain typical retry times of common MTA configurations, thus anticipating that a fully-capable MTA will retry sometime after the beginning of the window and before the end of it. The default window SHOULD range from one minute to 24 hours. Retries during the period of this window are permitted and satisfy the greylisting test, and thus the client is no longer likely to be a sender of spam; retries after the end of the window SHOULD be considered to be a new message for the purposes of greylisting evaluation (i.e., reset the "first seen" timestamp for that IP address). Some sites use a higher time value for the low end of the window time to match common legitimate MTA retry timeouts, but additional benefit from doing so appears unlikely.
3. Include a timeout for database entries, after which records for IP addresses that have generated no recent traffic are deleted. This step is intended to re-enable greylisting for an IP address in the event that it has changed "owners", and will subject the client to another round of greylisting. The default SHOULD be at least one week.
4. For an Administrative Management Domain (ADMD) all inbound border MTAs listed in the [DNS] SHOULD share a common greylisting database and common greylisting policies. This handles sequences in which a client's retry goes to a different server after the first 4yz reply, and it lets all servers share the list of hosts that did retry successfully.
5. To accommodate those senders that have clusters of outgoing mail servers, greylisting servers MAY track CIDR blocks of a size of its own choosing, such as /24, rather than the full IPv4 address.

(Note, however, that this heuristic will not work for clusters having machines on different networks.) A similar grouping capability MAY be established based on the domain name of the mail server if one can be determined.

6. Include a manual override capability for adding specific IP addresses or network blocks that always bypass checks. There are legitimate senders that simply don't respond well to greylisting for a variety of reasons, most of which do not conflict with [SMTP]. There are also some highly visible online entities such as email service providers that will be certain to retry, and thus those that are known SHOULD be allowed to bypass the filter.
7. Greylisting SHOULD NOT be applied by an ADMD's submission service (see [SUBMISSION]) for authenticated client hosts. It also SHOULD not be applied against any authenticated ADMD session. Authentication can include whatever mechanisms are deemed appropriate for the ADMD, such as known internal IP addresses, protocol-level client authentication, or the like.

There is no specific recommendation as to the specific choice of 4yz code to be returned as a result of a greylisting delay. Per [SMTP], however, the only two reasonable choices are 421 if the implementation wishes to terminate the connection immediately, and 450 otherwise. It is possible that some clients treat different 4yz codes differently, but no data are available on whether using 421 versus some other 4yz code is particularly advantageous.

There is also no specific recommendation as to the choice of text to include in the SMTP reply, if any. Some implementers argue that indicating that greylisting is in effect can give spamware a hint as to when to try again for successful delivery, while others suspect that it won't matter to spamware and thus the more likely audience is legitimate senders seeking to understand why their mail is being delayed.

6. Measuring Effectiveness

A few techniques are common when measuring the effectiveness of greylisting in a particular installation:

- o Arrange to log the spam vs. legitimate determinations of messages and what the greylisting decision would have been if enabled; then determine whether there is a correlation (and, of course, whether too much legitimate email would also be affected);

- o Continuing from the previous point, query the set of IP addresses subjected to greylisting in any popular [DNSBL] to see if there is a strong correlation.

7. IPv6 Applicability

The descriptions and recommendations presented in this memo are based on many years of experience with greylisting in the IPv4 Internet environment, and so they clearly pertain to IPv4 deployments only.

The greater size of an IPv6 address seems likely to permit differences in behaviours by bad actors, and this could well mean needing to alter the details for applying greylisting; it might even negate any benefits in using greylisting at all. At a minimum, it is likely to call for different specific choices for any greylisting algorithm variables.

In addition, an obvious consideration is that the size of the database required to store records of all of the IP addresses seen will likely be substantially larger in the IPv6 environment.

8. IANA Considerations

No actions are requested of IANA in this memo.

[RFC Editor: Please remove this section prior to publication.]

9. Security Considerations

This section discusses potential security issues related to greylisting.

9.1. Tradeoffs

The discussion above highlights the fact that, although greylisting provides some obvious and valuable defenses, it can introduce unintentional and detrimental consequences for delivery of legitimate mail. Where timely delivery of email is essential, especially for financial, transactional, or security related applications, the possible consequences of such systems need to be carefully considered.

Specific sources can be exempted from greylisting, but of course that means they have elevated privilege in terms of access to the mailboxes on the greylisting system, and malefactors can seek to

exploit this.

9.2. Database

The database that has to be maintained as part of any greylisting system will grow as the diversity of its SMTP clients' hosts grows, and of course is larger in general depending on the nature of the tuple stored about each delivery attempt. Even with a record aging policy in place, such a database could grow large enough to interfere with the system hosting it, or at least to a point at which greylisting service is degraded. Moreover, an attacker knowing which greylisting scheme is in use could rotate parameters of SMTP clients under its control, in an attempt to inflate the database to the point of denial-of-service.

Implementers could consider configuring an appropriate failure policy so that something locally acceptable happens when the database is attacked or otherwise unavailable.

In practice, this has not appeared as a serious concern, because any reasonable aging policy successfully moderates database growth. It is nevertheless identified here as a consideration as there may be implementations in some environments where this is indeed an issue.

10. References

10.1. Normative References

[EMAIL-ARCH]

Crocker, D., "Internet Mail Architecture", RFC 5598, October 2008.

[KEYWORDS]

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

[SMTP]

Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[SUBMISSION]

Gellens, R. and J. Klensin, "Message Submission for Mail", RFC 6409, November 2011.

10.2. Informative References

[CIDR]

Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation

Plan", RFC 4632, August 2006.

- [DHCP] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.
- [DNS] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [DNSBL] Levine, J., "DNS Blacklists and Whitelists", RFC 5782, February 2010.
- [MAIL] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.
- [NAT] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [PUREMAGIC] Harris, E., "The Next Step in the Spam Control War: Greylisting", August 2003, <<http://projects.puremagic.com/greylisting/whitepaper.html>>.
- [SAUCE] Jackson, I., "GNU SAUCE", 2001, <<http://www.gnu.org/software/sauce>>.

Appendix A. Acknowledgments

The author wishes to acknowledge Mike Adkins, Steve Atkins, Mihai Costea, Dave Crocker, Derek Diget, Peter J. Holzer, John Levine, Chris Lewis, Jose-Marcio Martins da Cruz, John Klensin, S. Moonesamy, Suresh Ramasubramanian, Mark Risher, Jordan Rosenwald, Gregory Shapiro, Joe Sniderman, Roland Turner, and Michael Wise for their contributions to this memo. The various participants of the MAAWG Open Sessions about greylisting were also valued contributors.

Authors' Addresses

Murray S. Kucherawy
Cloudmark
128 King St., 2nd Floor
San Francisco, CA 94107
US

Phone: +1 415 946 3800
Email: msk@cloudmark.com

D. Crocker
Brandenburg InternetWorking
675 Spruce Dr.
Sunnyvale 94086
USA

Phone: +1.408.246.8253
Email: dcrocker@bbiw.net
URI: <http://bbiw.net>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 12, 2013

A. Petersson
M. Nilsson
Opera Software
October 9, 2012

Forwarded HTTP Extension
draft-ietf-appsawg-http-forwarded-10

Abstract

This document defines an HTTP extension header field that allows proxy components to disclose information lost in the proxying process, for example, the originating IP address of a request or IP address of the proxy on the user-agent-facing interface. In a path of proxying components, this makes it possible to arrange it so that each subsequent component will have access to, for example, all IP addresses used in the chain of proxied HTTP requests.

This document also specifies guidelines for a proxy administrator to anonymize the origin of a request.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on April 12, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Notational Conventions	5
3. Syntax Notations	5
4. Forwarded HTTP Header Field	5
5. Parameters	6
5.1. Forwarded By	7
5.2. Forwarded For	7
5.3. Forwarded Host	8
5.4. Forwarded Proto	8
5.5. Extensions	8
6. Node Identifiers	8
6.1. IPv4 and IPv6 Identifiers	10
6.2. The "unknown" Identifier	10
6.3. Obfuscated Identifier	10
7. Implementation Considerations	10
7.1. HTTP Lists	10
7.2. Header Field Preservation	11
7.3. Relation to Via	11
7.4. Transition	11
7.5. Example Usage	12
8. Security Considerations	12
8.1. Header Validity and Integrity	13
8.2. Information Leak	13
8.3. Privacy Considerations	13
9. IANA Considerations	14
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Appendix A. Change Log (to be removed by RFC Editor before publication)	16
A.1. Since draft-petersson-forwarded-for-00	16
A.2. Since draft-petersson-forwarded-for-01	16
A.3. Since draft-petersson-forwarded-for-02	17
A.4. Since draft-ietf-appsawg-http-forwarded-00	17
A.5. Since draft-ietf-appsawg-http-forwarded-01	17
A.6. Since draft-ietf-appsawg-http-forwarded-02	17
A.7. Since draft-ietf-appsawg-http-forwarded-03	18
A.8. Since draft-ietf-appsawg-http-forwarded-04	18
A.9. Since draft-ietf-appsawg-http-forwarded-05	18
A.10. Since draft-ietf-appsawg-http-forwarded-06	18
A.11. Since draft-ietf-appsawg-http-forwarded-07	19
A.12. Since draft-ietf-appsawg-http-forwarded-08	19
A.13. Since draft-ietf-appsawg-http-forwarded-09	19
Appendix B. Acknowledgments	19
Authors' Addresses	19

1. Introduction

In today's HTTP landscape, there are a multitude of different applications that act as proxies for the user agents. In many cases, these proxies exist without the action or knowledge of the end-user. These cases occur, for example, when the proxy exists as a part of the infrastructure within the organization running the web server. Such proxies may be used for features such as loadbalancing or crypto offload. Another example is when the proxy is used within the same organization as the user, and the proxy is used to cache resources. However, these proxies make the requests appear as if they originated from the proxy's IP address, and may change other information in the original request. This represents a loss of information from the original request.

This loss of information can cause problems for a web server that has a specific use for the clients' IP addresses which will not be met by using the address of the proxy or other information changed by the proxy. The main uses of this information are for diagnostics, access control, and abuse management. Diagnostic functions can include event logging, trouble-shooting, and statistics gathering, and the information collected is usually only stored for short periods of time and only gathered in response to a particular problem or a complaint from the client. Access control can be operated by configuring a list of client IP addresses from which access is permitted, but this approach will not work if a proxy is used, unless the proxy is trusted and is, itself, configured with a list of allowed client addresses for the server. Cases of abuse require identification of the abuser and this uses many of the same features identified for diagnostics.

Most of the time that a proxy is used, this loss of information is not the primary purpose, or even a desired effect, of using the proxy. Thus, to restore the desired functionality when a proxy is in use, a way of disclosing the original information at the HTTP level is needed. Clearly, however, when the purpose of using a proxy is to provide client anonymity, the proxy will not use the feature defined in this document.

It should be noted that the use of a reverse proxy also hides information. Again, where the loss of information is not a deliberate function of the use of the reverse proxy, it can be desirable to find a way to encode the information within the HTTP messages so that the consumer can see it.

A common way to disclose this information is by using the non-standard header fields such as X-Forwarded-For, X-Forwarded-By, and X-Forwarded-Proto. There are many benefits to using a standardized

approach to commonly desired protocol function: not least is interoperability between implementations. This document standardizes a header field called "Forwarded" and provides the syntax and semantics for disclosing such information. "Forwarded" also combines all the information within one single header field, making it possible to correlate that information. With the header field format described in this document, it is possible to know what information belongs together, as long as the proxies are trusted. Such conclusions are not possible to make with the X-Forwarded class of header fields. The header field defined in this document is optional such that implementations of proxies that are intended to provide privacy are not required to operate or implement the header field.

Note that similar issues to those described for proxies also arise with use of NATs. This is discussed further in [RFC6269].

2. Notational Conventions

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].

3. Syntax Notations

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with the list rule extension defined in Section 3.2.5 of [I-D.ietf-httpbis-pl-messaging].

4. Forwarded HTTP Header Field

The "Forwarded" HTTP header field is an OPTIONAL header field that, when used, contains a list of parameter-identifier pairs that disclose information that is altered or lost when a proxy is involved in the path of the request. Due to the sensitive nature of the data passed in this header field (see Section 8.2 and Section 8.3) this header field should be turned off by default. Further, each parameter should be configured individually. "Forwarded" is only for use in HTTP requests and is not to be used in HTTP responses. This applies to forwarding proxies, as well as reverse proxies. Information passed in this header field can be, for example, the source IP address of the request, the IP address of the incoming interface on the proxy, or whether HTTP or HTTPS was used. If the request is passing through several proxies, each proxy can add a set of parameters; it can also remove earlier added Forwarded-header fields.

The top-level list is represented as a list of HTTP header field-values as defined in Section 3.2 of [I-D.ietf-httpbis-pl-messaging]. The first element in this list holds information added by the first proxy that implements and uses this header field, and each subsequent element holds information added by each subsequent proxy. Because this header field is optional, any proxy in the chain may choose not to update this header field. Each field-value is a semicolon-separated list; this sub-list consists of parameter-identifier pairs. Parameter-identifier pairs are grouped together by an equals sign. Each parameter **MUST NOT** occur more than once per field-value. The parameter names are case-insensitive. The header field value can be defined in augmented BNF syntax as:

```
Forwarded    = 1#forwarded-element

forwarded-element =
    [ forwarded-pair ] *( ";" [ forwarded-pair ] )

forwarded-pair = token "=" value
value          = token / quoted-string

token = <Defined in
    [I-D.ietf-httpbis-pl-messaging], Section 3.2.4>
quoted-string = <Defined in
    [I-D.ietf-httpbis-pl-messaging], Section 3.2.4>
```

Examples:

```
Forwarded: for="_gazonk"
Forwarded: For="[2001:db8:cafe::17]:4711"
Forwarded: for=192.0.2.60;proto=http;by=203.0.113.43
Forwarded: for=192.0.2.43, for=198.51.100.17
```

Note that as ":" and "[" are not valid characters in "token", IPv6 addresses are written as "quoted-string".

A proxy server that wants to add a new "Forwarded" header field value can either append it to the last existing "Forwarded" header field after a comma separator or add a new field at the end of the header block. A proxy **MAY** remove all "Forwarded" header fields from a request. It **MUST**, however, ensure that the correct header field is updated in case of multiple "Forwarded" header fields.

5. Parameters

This document specifies a number of parameters and valid values for

each of them:

- o "by" identifies the user-agent facing interface of the proxy.
- o "for" identifies the node making the request to the proxy.
- o "host" is the host request header field as received by the proxy.
- o "proto" indicates what protocol was used to make the request.

5.1. Forwarded By

The "by" parameter is used to disclose the interface where the request came in to the proxy server. When proxies choose to use the "by" parameter, its default configuration SHOULD contain an obfuscated identifier as described in Section 6.3. If the server receiving proxied requests requires some address-based functionality, this parameter MAY instead contain an IP-address (and, potentially, a port number). A third option is the "unknown" identifier described in Section 6.2.

The syntax of a "by" value, after potential quoted-string unescaping, conforms to the "node" ABNF described in Section 6.

This is primarily added by reverse proxies that wish to forward this information to the backend server. It can also be interesting in a multi-homed environment to signal to backend servers where the request came from.

5.2. Forwarded For

The "for" parameter is used to disclose information about the client that initiated the request and following proxies in a chain of proxies. When proxies choose to use the "for" parameter, its default configuration SHOULD contain an obfuscated identifier as described in Section 6.3. If the server receiving proxied requests requires some address-based functionality, this parameter MAY instead contain an IP-address (and, potentially, a port number). A third option is the "unknown" identifier described in Section 6.2.

The syntax of a "for" value, after potential quoted-string unescaping, conforms to the "node" ABNF described in Section 6.

In a chain of proxy servers where this is fully utilized, the first for-parameter will disclose the client where the request was first made, followed by any subsequent proxy identifiers. The last proxy in the chain is not part of the list of for-parameters. The last proxy's IP address, and optionally a port number, are, however,

readily available as the remote IP address at the transport layer. It can, however, be more relevant to read information about the last proxy from preceding "Forwarded" header field's by-parameter, if present.

5.3. Forwarded Host

The "host" parameter is used to forward the original value of the "Host" header field. This can be used, for example, by the origin server if a reverse proxy is rewriting the "Host" header field to some internal host name.

The syntax for a "host" value, after potential quoted-string unescaping, MUST conform to the Host ABNF described in Section 5.4 of [I-D.ietf-httpbis-pl-messaging].

5.4. Forwarded Proto

The "proto" parameter has the value of the used protocol type. The syntax of a "proto" value, after potential quoted-string unescaping, MUST conform to the URI scheme name as defined in Section 3.1 in [RFC3986] and registered to IANA according to [RFC4395]. Typical values are "http" or "https".

For example, in an environment where a reverse proxy is also used as a crypto offloader, this allows the origin server to rewrite URLs in a document to match the type of connection as the user agent requested, even though all connections to the origin server are unencrypted HTTP.

5.5. Extensions

Extensions allow for additional parameters and values. Extensions can be particularly useful in reverse proxy environments. All extension parameters SHOULD be registered in the "HTTP Forwarded Parameter" registry. If certain extensions are expected to have widespread deployment, they SHOULD also be standardized. This is further discussed in Section 9.

6. Node Identifiers

The node identifier is one of the following:

- o The client's IP address, with an optional port number
- o A token indicating that the IP address of the client is not known to the proxy server

- o A generated token, allowing for tracing and debugging, while allowing the internal structure or sensitive information to be hidden

The node identifier is defined by the augmented BNF syntax as:

```
node      = nodename [ ":" node-port ]
nodename  = IPv4address / "[" IPv6address "]" /
           "unknown" / obfnodename

IPv4address = <Defined in [RFC3986], Section 3.2.2>
IPv6address = <Defined in [RFC3986], Section 3.2.2>
obfnodename = "_" 1*( ALPHA / DIGIT / "." / "_" / "-" )

node-port   = port / obfport
port        = 1*5DIGIT
obfport     = "_" 1*(ALPHA / DIGIT / "." / "_" / "-")

DIGIT = <Defined in [RFC5234], Section 3.4>
ALPHA = <Defined in [RFC5234], Section B.1>
```

Each of the identifiers may optionally have the port identifier, for example, allowing the identification of the end point in a NATted environment. The "node-port" can be identified either by its port number or by a generated token obfuscating the real port number. An obfuscated port may be used in situations where the possessor of the proxy wants the ability to trace requests -- for example, in debug purposes -- but does not want to reveal internal information.

Note that the ABNF above also allows port numbers to be appended to the the "unknown" identifier. Interpretation of such notation is, however, left to the possessor of a proxy adding such a value to the header field. To distinguish an "obfport" from a port, the "obfport" MUST have a leading underscore. Further, it MUST also consist of only "ALPHA", "DIGIT", and the characters ".", "_" and "-".

It is important to note that an IPv6 address and any nodename with node-port specified MUST be quoted, since ":" is not an allowed character in "token".

Examples:

```
"192.0.2.43:47011"
"[2001:db8:cafe::17]:47011"
```

6.1. IPv4 and IPv6 Identifiers

The ABNF rules for "IPv6address" and "IPv4address" are defined in [RFC3986]. The "IPv6address" SHOULD comply with textual representation recommendations [RFC5952] (for example, lowercase, compression of zeros).

Note that the IP address may be one from the internal nets, as defined in [RFC1918] and [RFC4193]. Also, note that an IPv6 address is always enclosed in square brackets.

6.2. The "unknown" Identifier

The "unknown" identifier is used when the identity of the preceding entity is not known, but the proxy server still wants to signal that a forwarding of the request was made. One example would be a proxy server process generating an outgoing request without direct access to the incoming request TCP socket.

6.3. Obfuscated Identifier

A generated identifier may be used where there is a wish to keep the internal IP addresses secret, while still allowing the "Forwarded" header field to be used for tracing and debugging. This can also be useful if the proxy uses some sort of interface labels and it is desired to pass them rather than an IP address. Unless static assignment of identifiers is necessary for the server's use of the identifiers, obfuscated identifiers SHOULD be randomly generated for each request. If the server requires that identifiers persist across requests, they SHOULD NOT persist longer than client IP addresses. To distinguish the obfuscated identifier from other identifiers, it MUST have a leading underscore "_". Furthermore, it MUST also consist of only "ALPHA", "DIGIT" and the characters ".", "_", and "-". Example:

```
Forwarded: for=_hidden, for=_SEVKISEK
```

7. Implementation Considerations

7.1. HTTP Lists

Note that an HTTP list allows white spaces to occur between the identifiers, and the list may be split over multiple header fields. As an example, the header field

```
Forwarded: for=192.0.2.43,for="[2001:db8:cafe::17]",for=unknown
```

is equivalent to the header field

```
Forwarded: for=192.0.2.43, for="[2001:db8:cafe::17]", for=unknown
```

which is equivalent to the header fields

```
Forwarded: for=192.0.2.43
```

```
Forwarded: for="[2001:db8:cafe::17]", for=unknown
```

7.2. Header Field Preservation

There are some cases when this header field should be kept and some cases where it should not be kept. A directly forwarded request should preserve and possibly extend it. If a single incoming request causes the proxy to make multiple outbound requests, special care must be taken to decide whether the header field should be preserved or not. In many cases the header field should be preserved, but if the outbound request is not a direct consequence of the incoming request, the header field should not be preserved. Consider also the case when a proxy has detected a content mismatch in a 304 response and is following the instructions in [I-D.ietf-httpbis-p4-conditional] Section 4.1 to repeat the request unconditionally, in which case the new request is still basically a direct consequence of the origin request, and the header field should probably be kept.

7.3. Relation to Via

The "Via" header field [I-D.ietf-httpbis-p4-conditional] Section 6.2 is a header field with similar use case as this header field. The "Via" header field, however, only provides information about the proxy itself, and is thereby leaving out the information about the client connecting to the proxy server. The "Forwarded" header field, on the other hand, has relaying information from the client facing side of the proxy server as its main purpose. As "Via" is already widely deployed, its format can not be changed to address the problems that "Forwarded" addresses.

Note that it is not possible to combine information from this header field with the information from the Via header field. Some proxies will not update the "Forwarded" header field, some proxies will not update the Via header field, and some proxies will update both.

7.4. Transition

If a proxy gets incoming requests with X-Forwarded-* header fields present, it is encouraged to convert these into the header field described in this document, if it can be done in a sensible way. If

the request only contains one type -- for example, X-Forwarded-For -- this can be translated to "Forwarded", by prepending each element with "for=". Note that IPv6 addresses may not be quoted in X-Forwarded-For, and may not be enclosed by square brackets, but they are quoted and enclosed in square brackets in "Forwarded".

```
X-Forwarded-For: 192.0.2.43, 2001:db8:cafe::17
```

becomes:

```
Forwarded: for=192.0.2.43, for="[2001:db8:cafe::17]"
```

Special care must, however, be taken if, for example, both X-Forwarded-For and X-Forwarded-By exist. In such cases, it may not be possible to do a conversion, since it is not possible to know in which order the already existing fields were added. Also, note that removing the X-Forwarded-For header field may cause issues for parties that have not yet implemented support for this new header field.

7.5. Example Usage

A request from a client with IP address 192.0.2.43 passes through a proxy with IP address 198.51.100.17, then through another proxy with IP address 203.0.113.60 before reaching a origin server. This could, for example, be an office client behind a corporate malware filter talking to a origin server through a reverse proxy.

- o The HTTP request between the client and the first proxy has no "Forwarded" header field.
- o The HTTP request between the first and second proxy has a "Forwarded: for=192.0.2.43" header field.
- o The HTTP request between the second proxy and the origin server has a "Forwarded: for=192.0.2.43, for=198.51.100.17;by=203.0.113.60;proto=http;host=example.com" header field.

Note that, at some points in a connection chain, the information might not be updated in the "Forwarded" header field, either because of lack of support of this HTTP extension or because of a policy decision not to disclose information about this network component.

8. Security Considerations

8.1. Header Validity and Integrity

The "Forwarded" HTTP header field cannot be relied upon to be correct, as it may be modified, whether mistakenly or for malicious reasons, by every node on the way to the server, including the client making the request.

One approach is to verify the correctness of proxies and to whitelist them as trusted. This approach has at least two weaknesses. First, the chain of IP addresses listed before the request came to the proxy cannot be trusted. Second, unless the communication between proxies and the end point is secured, the data can be modified by an attacker with access to the network.

8.2. Information Leak

The "Forwarded" HTTP header field can reveal internal structures of the network setup behind the NAT or proxy setup, which may be undesired. This can be addressed either by using obfuscated elements, preventing the internal nodes from updating the HTTP header field, or by having an egress proxy removing entries that reveals internal network information.

This header field should never be copied into response messages by origin servers or intermediaries, as it can reveal the whole proxy chain to the client. As a side effect, special care must be taken in hosting environments not to allow the TRACE request where the "Forwarded" field is used, as it would appear in the body of the response message.

8.3. Privacy Considerations

In recent years, there have been growing concerns about privacy. There is a trade-off between ensuring privacy for users versus disclosing information that is useful, for example for debugging, statistics and generating location-dependent content. The "Forwarded" HTTP header field, by design, exposes information that some users consider privacy sensitive, in order to allow for such uses. A proxy that intends or is widely used to anonymize the user MUST NOT use the header field described in this document.

The clients IP address, that may be forwarded in the "for" parameter of this header field, is considered to be privacy sensitive by many people, as the IP address may be able to uniquely identify a client, what operator the user is using, and possibly a rough estimation of where the user is geographically located.

Proxies using this extension will preserve the information of a

direct connection. This has an end-user privacy impact regardless of whether the end-user or deployer knows or expects that this is the case.

Implementers and deployers of such proxies need to consider whether, and how, deploying this extension affects user privacy.

The default configuration for both the "by" and "for" parameters SHOULD contain obfuscated identifiers. These identifiers SHOULD be randomly generated per request. If identifiers are required that persist across requests, their lifetimes SHOULD be limited and they SHOULD NOT persist longer than client IP addresses. When generating obfuscated identifiers, care must be taken not to include potentially sensitive information in them.

Note that users' IP addresses may already be forwarded by proxies using the header field X-Forwarded-For, which is widely used. It should also be noted that if the user were doing the connection directly without passing the proxy, the client's IP address would be sent to the web server. Users that do not actively choose a anonymizing proxy can not rely on having their IP address shielded. These users who want to minimize the risk of being tracked must also note that there are other ways information may leak, for example by browser header field fingerprinting. The Forwarded header field itself, even when used without a uniquely identifying client identifier, may make fingerprinting more feasible by revealing the chain of proxies traversed by the client's request.

9. IANA Considerations

This document specifies the HTTP header field listed below, which should be added to the permanent HTTP header field registry defined in [RFC3864].

Header field: Forwarded
Applicable protocol: http
Status: standard
Author/Change controller:
 IETF (iesg@ietf.org)
 Internet Engineering Task Force
Specification document(s): this specification (Section 4)
Related information: None

The "Forwarded" header field contains parameters for which IANA is to create and maintain a new registry entitled "HTTP Forwarded parameters". Initial registrations are given below; for future assignments, the registration procedure to be used is IETF review

[RFC5226]. The security and privacy implications of all new parameters should be thoroughly documented. New parameters and their values MUST conform the forwarded-pair as defined in ABNF in Section 4. Further, a short description should be provided in the registration.

Parameter name	Description	Definition
by	IP-address of incoming interface of a proxy	Section 5.1
for	IP-address of client making a request through a proxy	Section 5.2
host	Host header field of the incoming request	Section 5.3
proto	Application protocol used for incoming request	Section 5.4

Table 1: Initial assignments

10. References

10.1. Normative References

- [I-D.ietf-httpbis-pl-messaging]
Fielding, R., Lafon, Y., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing", draft-ietf-httpbis-pl-messaging-19 (work in progress), March 2012.
- [I-D.ietf-httpbis-p4-conditional]
Fielding, R., Lafon, Y., and J. Reschke, "HTTP/1.1, part 4: Conditional Requests", draft-ietf-httpbis-p4-conditional-19 (work in progress), March 2012.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864,

September 2004.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

10.2. Informative References

- [RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

Appendix A. Change Log (to be removed by RFC Editor before publication)

A.1. Since draft-petersson-forwarded-for-00

Added IANA considerations.

Expanded scope and add parameterized list.

A.2. Since draft-petersson-forwarded-for-01

Removed "x-" from private extensions.

Allow for any protocol name.

Rename kv-v to forwarded-element and kv to forwarded-value.

Add informative reference RFC6269.

A.3. Since draft-petersson-forwarded-for-02

Name change to draft-ietf-appsawg-http-forwarded-00.

Updated proto in list under section 5 Parameters.

Remove "hidden" but mention _hidden as an example in 6.3 Obfuscated identifier.

Clarify that IPv6-addresses must be enclosed by square brackets.

Restrict ext-value: do not allow "," or ";".

A.4. Since draft-ietf-appsawg-http-forwarded-00

Write IP address instead of IP number.

Remove BNF for IP addresses.

A.5. Since draft-ietf-appsawg-http-forwarded-01

Refer to httpbis instead of RFC2616. Thereby also change to RFC5234 ABNF.

Split up ABNF to be more general on top level.

Add some comments on draft-ietf-httpbis-p2-semantics-19#section-3.1 to "Implementation Considerations"

Removal of ABNF appendix.

Merging of the sections "Private extensions" and "Future extensions".

A.6. Since draft-ietf-appsawg-http-forwarded-02

Require obfport to start with an underscore.

Include "._-" as valid characters in obfnod.

Remove MAY-references from section 5.

Add a section about the relation to the via-header field.

Add some privacy considerations.

Encourage proxies to convert X-Forwarded-* to this format, when possible.

Mention and demonstrate that IPv6-addresses must be quoted.

Add motivation for the obfnode.

Add some notes on when this header field should be preserved or not.

Fix some typos and make some clarifications.

A.7. Since draft-ietf-appsawg-http-forwarded-03

Require that each parameter only occur once per instance.

Request for a new registry at IANA.

A.8. Since draft-ietf-appsawg-http-forwarded-04

Add ABNF references for token, quoted-string, IPv4address, IPv6address, DIGIT and ALPHA.

Only define the content of the Forwarded header field.

Remove https from "applicable protocol" in Section 9, as this is implied.

A.9. Since draft-ietf-appsawg-http-forwarded-05

Grouped all ABNF.

Change registration from "RFC required" to "Specification required".

Extended the section describing the relation to Via.

Extended Privacy Considerations.

Made some clarifications and language fixes.

A.10. Since draft-ietf-appsawg-http-forwarded-06

Break up the ABNF again.

Update the Privacy Considerations section.

Update the IANA registration policy.

Change back to *(";" [forwarded-pair]) .

Some minor clarifications.

Consistently quote "Forwarded".

A.11. Since draft-ietf-appsawg-http-forwarded-07

Expanded the Introduction.

Expanded the Privacy Considerations discussion.

A.12. Since draft-ietf-appsawg-http-forwarded-08

Change registration procedure to IETF review.

Expand Introduction.

Encourage this head field to be off by default.

A.13. Since draft-ietf-appsawg-http-forwarded-09

Change from "header" to "header field" in some places.

Fix copy-paste error in the "for"-section.

Appendix B. Acknowledgments

Thanks to Per Cederqvist, Alissa Cooper, Adrian Farrel, Stephen Farrell, Ned Freed, Per Hedbor, Amos Jeffries, Poul-Henning Kamp, Murray S. Kucherawy, Barry Leiba, Salvatore Loreto, Alexey Melnikov, S. Moonesamy, Susan Nichols, Mark Nottingham, Julian Reschke, John Sullivan, Willy Tarreau and Dan Wing for their feedback.

Authors' Addresses

Andreas Petersson
Opera Software
S:t Larsgatan 12
Linköping SE-582 24

Email: pettson@opera.com

Martin Nilsson
Opera Software
S:t Larsgatan 12
Linköping SE-582 24

Email: nilsson@opera.com

Applications Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2013

P. Bryan, Ed.
Salesforce.com
M. Nottingham, Ed.
Akamai
January 20, 2013

JSON Patch
draft-ietf-appsawg-json-patch-10

Abstract

JSON Patch defines a JSON document structure for expressing a sequence of operations to apply to a JavaScript Object Notation (JSON) document, suitable for use with the HTTP PATCH method. The "application/json-patch" media type is used to identify such patch documents.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on July 24, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	3
3. Document Structure	3
4. Operations	4
4.1. add	4
4.2. remove	6
4.3. replace	6
4.4. move	6
4.5. copy	7
4.6. test	7
5. Error Handling	8
6. IANA Considerations	8
7. Security Considerations	9
8. Acknowledgements	10
9. References	10
9.1. Normative References	10
9.2. Informative References	10
Appendix A. Examples	11
A.1. Adding an Object Member	11
A.2. Adding an Array Element	11
A.3. Removing an Object Member	12
A.4. Removing an Array Element	12
A.5. Replacing a Value	12
A.6. Moving a Value	13
A.7. Moving an Array Element	13
A.8. Testing a Value: Success	14
A.9. Testing a Value: Error	14
A.10. Adding a nested Member Object	14
A.11. Ignoring Unrecognized Elements	15
A.12. Adding to a Non-existent Target	15
A.13. Invalid JSON Patch Document	16
A.14. ~ Escape Ordering	16
A.15. Comparing Strings and Numbers	16
A.16. Adding an Array Value	17
Authors' Addresses	17

1. Introduction

JavaScript Object Notation (JSON) [RFC4627] is a common format for the exchange and storage of structured data. HTTP PATCH [RFC5789] extends the Hypertext Transfer Protocol (HTTP) [RFC2616] with a method to perform partial modifications to resources.

JSON Patch is a format (identified by the media type "application/json-patch") for expressing a sequence of operations to apply to a target JSON document, suitable for use with the HTTP PATCH method.

This format is also potentially useful in other cases where necessary to make partial updates to a JSON document, or to a data structure that has similar constraints (i.e., they can be serialised as an object or an array using the JSON grammar).

2. Conventions

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 RFC 2119 [RFC2119].

See Section 5 for information about handling errors.

3. Document Structure

A JSON Patch document is a JSON [RFC4627] document that represents an array of objects. Each object represents a single operation to be applied to the target JSON document.

An example JSON Patch document, transferred in a HTTP PATCH request:

```
PATCH /my/data HTTP/1.1
Host: example.org
Content-Length: 326
Content-Type: application/json-patch
If-Match: "abc123"

[
  { "op": "test", "path": "/a/b/c", "value": "foo" },
  { "op": "remove", "path": "/a/b/c" },
  { "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] },
  { "op": "replace", "path": "/a/b/c", "value": 42 },
  { "op": "move", "from": "/a/b/c", "path": "/a/b/d" },
  { "op": "copy", "from": "/a/b/d", "path": "/a/b/e" }
]
```

Evaluation of a JSON Patch document begins against a target JSON document. Operations are applied sequentially in the order they appear in the array. Each operation in the sequence is applied to the target document; the resulting document becomes the target of the next operation. Evaluation continues until all operations are successfully applied, or an error condition is encountered.

4. Operations

Operation objects MUST have exactly one "op" member, whose value indicates the operation to perform. Its value MUST be one of "add", "remove", "replace", "move", "copy" or "test"; other values are errors. The semantics of each is defined below.

Additionally, operation objects MUST have exactly one "path" member. That member's value is a string containing a [JSON-Pointer] value that references a location within the target document (the "target location") where the operation is performed.

The meanings of other members of operation objects are defined by operation (see the subsections below). Members that are not explicitly defined for the operation in question MUST be ignored (i.e., the operation will complete as if the undefined member did not appear in the object).

Note that the ordering of members in JSON objects is not significant; therefore, the following operation objects are equivalent:

```
{ "op": "add", "path": "/a/b/c", "value": "foo" }
{ "path": "/a/b/c", "op": "add", "value": "foo" }
{ "value": "foo", "path": "/a/b/c", "op": "add" }
```

Operations are applied to the data structures represented by a JSON document; i.e., after any unescaping (see [RFC4627], Section 2.5) takes place.

4.1. add

The "add" operation performs the following function, depending upon what the target location references:

- o If the target location specifies an array index, a new value is inserted into the array at the specified index.
- o If the target location specifies an object member that does not already exist, a new member is added to the object.

- o If the target location specifies an object member that does exist, that member's value is replaced.

The operation object MUST contain a "value" member whose content specifies the value to be added.

For example:

```
{ "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] }
```

When the operation is applied, the target location MUST reference one of:

- o The root of the target document - whereupon the specified value becomes the entire content of the target document.
- o A member to add to an existing object - whereupon the supplied value is added to that object at the indicated location. If the member already exists, it is replaced by the specified value.
- o An element to add to an existing array - whereupon the supplied value is added to the array at the indicated location. Any elements at or above the specified index are shifted one position to the right. The specified index MUST NOT be greater than the number of elements in the array. If the "-" character is used to index the end of the array (see [JSON-Pointer]), this has the effect of appending the value to the array.

Because this operation is designed to add to existing objects and arrays, its target location will often not exist. Although the pointer's error handling algorithm will thus be invoked, this specification defines the error handling behaviour for "add" pointers to ignore that error and add the value as specified.

However, the object itself or an array containing it does need to exist, and it remains an error for that not to be the case. For example, an "add" with a target location of "/a/b" starting with this document:

```
{ "a": { "foo": 1 } }
```

is not an error, because "a" exists, and "b" will be added to its value. It is an error in this document:

```
{ "q": { "bar": 2 } }
```

because "a" does not exist.

4.2. remove

The "remove" operation removes the value at the target location.

The target location MUST exist for the operation to be successful.

For example:

```
{ "op": "remove", "path": "/a/b/c" }
```

If removing an element from an array, any elements above the specified index are shifted one position to the left.

4.3. replace

The "replace" operation replaces the value at the target location with a new value. The operation object MUST contain a "value" member whose content specifies the replacement value.

The target location MUST exist for the operation to be successful.

For example:

```
{ "op": "replace", "path": "/a/b/c", "value": 42 }
```

This operation is functionally identical to a "remove" operation for a value, followed immediately by an "add" operation at the same location with the replacement value.

4.4. move

The "move" operation removes the value at a specified location and adds it to the target location.

The operation object MUST contain a "from" member, a string containing a JSON Pointer value that references the location in the target document to move the value from.

The "from" location MUST exist for the operation to be successful.

For example:

```
{ "op": "move", "from": "/a/b/c", "path": "/a/b/d" }
```

This operation is functionally identical to a "remove" operation on the "from" location, followed immediately by an "add" operation at the target location with the value that was just removed.

The "from" location MUST NOT be a proper prefix of the "path" location; i.e., a location cannot be moved into one of its children.

4.5. copy

The "copy" operation copies the value at a specified location to the target location.

The operation object MUST contain a "from" member, a string containing a JSON Pointer value that references the location in the target document to copy the value from.

The "from" location MUST exist for the operation to be successful.

For example:

```
{ "op": "copy", "from": "/a/b/c", "path": "/a/b/e" }
```

This operation is functionally identical to an "add" operation at the target location using the value specified in the "from" member.

4.6. test

The "test" operation tests that a value at the target location is equal to a specified value.

The operation object MUST contain a "value" member that conveys the value to be compared to that at the target location.

The target location MUST be equal to the "value" value for the operation to be considered successful.

Here, "equal" means that the value at the target location and that conveyed by "value" are of the same JSON type, and considered equal by the following rules for that type:

- o strings: are considered equal if they contain the same number of Unicode characters and their code points are position-wise equal.
- o numbers: are considered equal if their values are numerically equal.
- o arrays: are considered equal if they contain the same number of values, and each value can be considered equal to the value at the corresponding position in the other array, using this list of type-specific rules.

- o objects: are considered equal if they contain the same number of members, and each member can be considered equal to a member in the other object, by comparing their keys as strings, and values using this list of type-specific rules.
- o literals (false, true and null): are considered equal if they are the same.

Note that this is a logical comparison; e.g., whitespace between the member values of an array is not significant.

Also, note that ordering of the serialisation of object members is not significant.

For example:

```
{ "op": "test", "path": "/a/b/c", "value": "foo" }
```

5. Error Handling

If a normative requirement is violated by a JSON Patch document, or if an operation is not successful, evaluation of the JSON Patch document SHOULD terminate and application of the entire patch document SHALL NOT be deemed successful.

See [RFC5789], Section 2.2 for considerations regarding handling errors when JSON Patch is used with the HTTP PATCH method, including suggested status codes to use to indicate various conditions.

Note that the HTTP PATCH method is atomic, as per [RFC5789]. Therefore, the following patch would result in no changes being made to the document at all (because the "test" operation results in an error).

```
[  
  { "op": "replace", "path": "/a/b/c", "value": 42 },  
  { "op": "test", "path": "/a/b/c", "value": "C" }  
]
```

6. IANA Considerations

The Internet media type for a JSON Patch document is application/json-patch.

Type name: application

Subtype name: json-patch

Required parameters: none

Optional parameters: none

Encoding considerations: binary

Security considerations:
See Security Considerations in section 7.

Interoperability considerations: N/A

Published specification:
[this memo]

Applications that use this media type:
Applications that manipulate JSON documents.

Additional information:

Magic number(s): N/A

File extension(s): .json-patch

Macintosh file type code(s): TEXT

Person & email address to contact for further information:
Paul C. Bryan <pbryan@anode.ca>

Intended usage: COMMON

Restrictions on usage: none

Author: Paul C. Bryan <pbryan@anode.ca>

Change controller: IETF

7. Security Considerations

This specification has the same security considerations as JSON [RFC4627] and [JSON-Pointer].

A few older Web browsers can be coerced into loading an arbitrary JSON document whose root is an array, leading to a situation where a

JSON Patch document containing sensitive information could be exposed to attackers, even if access is authenticated. This is known as a Cross-Site Request Forgery (CSRF) attack [CSRF].

However, such browsers are not widely used (estimated to comprise less than 1% of the market, at the time of writing). Publishers who are nevertheless concerned about this attack are advised to avoid making such documents available with HTTP GET.

8. Acknowledgements

The following individuals contributed ideas, feedback and wording to this specification:

Mike Acar, Mike Amundsen, Cyrus Daboo, Paul Davis, Stefan Koegl, Murray S. Kucherawy, Dean Landolt, Randall Leeds, James Manger, Julian Reschke, James Snell, Eli Stevens and Henry S. Thompson.

The structure of a JSON Patch document was influenced by the XML Patch document [RFC5261] specification.

9. References

9.1. Normative References

- [JSON-Pointer] Bryan, P., Zyp, K., and M. Nottingham, "JSON Pointer", draft-ietf-appsawg-json-pointer-07 (work in progress), November 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

9.2. Informative References

- [CSRF] Barth, A., Jackson, C., and J. Mitchell, "Robust Defenses for Cross-Site Request Forgery".
- [RFC2616] 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.
- [RFC5261] Urpalainen, J., "An Extensible Markup Language (XML) Patch

Operations Framework Utilizing XML Path Language (XPath) Selectors", RFC 5261, September 2008.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

Appendix A. Examples

A.1. Adding an Object Member

An example target JSON document:

```
{ "foo": "bar" }
```

A JSON Patch document:

```
[  
  { "op": "add", "path": "/baz", "value": "qux" }  
]
```

The resulting JSON document:

```
{  
  "baz": "qux",  
  "foo": "bar"  
}
```

A.2. Adding an Array Element

An example target JSON document:

```
{ "foo": [ "bar", "baz" ] }
```

A JSON Patch document:

```
[  
  { "op": "add", "path": "/foo/1", "value": "qux" }  
]
```

The resulting JSON document:

```
{ "foo": [ "bar", "qux", "baz" ] }
```

A.3. Removing an Object Member

An example target JSON document:

```
{
  "baz": "qux",
  "foo": "bar"
}
```

A JSON Patch document:

```
[
  { "op": "remove", "path": "/baz" }
]
```

The resulting JSON document:

```
{ "foo": "bar" }
```

A.4. Removing an Array Element

An example target JSON document:

```
{ "foo": [ "bar", "qux", "baz" ] }
```

A JSON Patch document:

```
[
  { "op": "remove", "path": "/foo/1" }
]
```

The resulting JSON document:

```
{ "foo": [ "bar", "baz" ] }
```

A.5. Replacing a Value

An example target JSON document:

```
{
  "baz": "qux",
  "foo": "bar"
}
```

A JSON Patch document:

```
[
  { "op": "replace", "path": "/baz", "value": "boo" }
]
```

```
]
```

The resulting JSON document:

```
{
  "baz": "boo",
  "foo": "bar"
}
```

A.6. Moving a Value

An example target JSON document:

```
{
  "foo": {
    "bar": "baz",
    "waldo": "fred"
  },
  "qux": {
    "corge": "grault"
  }
}
```

A JSON Patch document:

```
[
  { "op": "move", "from": "/foo/waldo", "path": "/qux/thud" }
]
```

The resulting JSON document:

```
{
  "foo": {
    "bar": "baz"
  },
  "qux": {
    "corge": "grault",
    "thud": "fred"
  }
}
```

A.7. Moving an Array Element

An example target JSON document:

```
{ "foo": [ "all", "grass", "cows", "eat" ] }
```

A JSON Patch document:

```
[
  { "op": "move", "from": "/foo/1", "path": "/foo/3" }
]
```

The resulting JSON document:

```
{ "foo": [ "all", "cows", "eat", "grass" ] }
```

A.8. Testing a Value: Success

An example target JSON document:

```
{
  "baz": "qux",
  "foo": [ "a", 2, "c" ]
}
```

A JSON Patch document that will result in successful evaluation:

```
[
  { "op": "test", "path": "/baz", "value": "qux" },
  { "op": "test", "path": "/foo/1", "value": 2 }
]
```

A.9. Testing a Value: Error

An example target JSON document:

```
{ "baz": "qux" }
```

A JSON Patch document that will result in an error condition:

```
[
  { "op": "test", "path": "/baz", "value": "bar" }
]
```

A.10. Adding a nested Member Object

An example target JSON document:

```
{ "foo": "bar" }
```

A JSON Patch document:

```
[
  { "op": "add", "path": "/child", "value": { "grandchild": { } } }
]
```

The resulting JSON document:

```
{
  "foo": "bar",
  "child": {
    "grandchild": {
    }
  }
}
```

A.11. Ignoring Unrecognized Elements

An example target JSON document:

```
{ "foo": "bar" }
```

A JSON Patch document:

```
[
  { "op": "add", "path": "/baz", "value": "qux", "xyz": 123 }
]
```

The resulting JSON document:

```
{
  "foo": "bar",
  "baz": "qux"
}
```

A.12. Adding to a Non-existent Target

An example target JSON document:

```
{ "foo": "bar" }
```

A JSON Patch document:

```
[
  { "op": "add", "path": "/baz/bat", "value": "qux" }
]
```

This JSON Patch document, applied to the target JSON document above, would result in an error (therefore not being applied) because the "add" operation's target location that references neither the root of the document, nor a member of an existing object, nor a member of an existing array.

A.13. Invalid JSON Patch Document

A JSON Patch document:

```
[
  { "op": "add", "path": "/baz", "value": "qux", "op": "remove" }
]
```

This JSON Patch document cannot be treated as an "add" operation since there is a later "op": "remove" element. JSON requires that object member names be unique with a "SHOULD" requirement, and there is no standard error handling for duplicates.

A.14. ~ Escape Ordering

An example target JSON document:

```
{
  "/": 9,
  "~1": 10
}
```

A JSON Patch document:

```
[
  { "op": "test", "path": "/~01", "value": 10 }
]
```

The resulting JSON document:

```
{
  "/": 9,
  "~1": 10
}
```

A.15. Comparing Strings and Numbers

An example target JSON document:

```
{
  "/": 9,
  "~1": 10
}
```

A JSON Patch document:

```
[
  { "op": "test", "path": "/~01", "value": "10" }
]
```

```
]
```

This results in an error, because the test fails; the document value is numeric, whereas the value tested for is a string.

A.16. Adding an Array Value

An example target JSON document:

```
{ "foo": ["bar"] }
```

A JSON Patch document:

```
[  
  { "op": "add", "path": "/foo/-", "value": ["abc", "def"] }  
]
```

The resulting JSON document:

```
{ "foo": ["bar", ["abc", "def"]] }
```

Authors' Addresses

Paul C. Bryan (editor)
Salesforce.com

Phone: +1 604 783 1481
Email: pbryan@anode.ca

Mark Nottingham (editor)
Akamai

Email: mnot@mnot.net

Applications Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 26, 2013

P. Bryan, Ed.
Salesforce.com
K. Zyp
SitePen (USA)
M. Nottingham, Ed.
Akamai
January 22, 2013

JavaScript Object Notation (JSON) Pointer
draft-ietf-appsawg-json-pointer-09

Abstract

JSON Pointer defines a string syntax for identifying a specific value within a JavaScript Object Notation (JSON) document.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on July 26, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	3
3. Syntax	3
4. Evaluation	3
5. JSON String Representation	5
6. URI Fragment Identifier Representation	6
7. Error Handling	7
8. IANA Considerations	7
9. Security Considerations	7
10. Acknowledgements	7
11. References	8
11.1. Normative References	8
11.2. Informative References	8
Authors' Addresses	8

1. Introduction

This specification defines JSON Pointer, a string syntax for identifying a specific value within a JavaScript Object Notation (JSON) [RFC4627] document. It is intended to be easily expressed in JSON string values as well as Uniform Resource Identifier (URI) [RFC3986] fragment identifiers.

2. Conventions

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].

This specification expresses normative syntax rules using Augmented Backus-Naur Form (ABNF) [RFC5234] notation.

3. Syntax

A JSON Pointer is a Unicode string (see [RFC4627], Section 3) containing a sequence of zero or more reference tokens, each prefixed by a '/' (%x2F) character.

Because the characters '~' (%x7E) and '/' (%x2F) have special meanings in JSON Pointer, '~' needs to be encoded as '~0' and '/' needs to be encoded as '~1' when these characters appear in a reference token.

The ABNF syntax of a JSON Pointer is:

```
json-pointer = *( "/" reference-token )
reference-token = *( unescaped / escaped )
unescaped = %x00-2E / %x30-7D / %x7F-10FFFF
             ; %x2F ( '/' ) and %x7E ( '~' ) are excluded from 'unescaped'
escaped = "~" ( "0" / "1" )
            ; representing '~' and '/', respectively
```

It is an error condition if a JSON Pointer value does not conform to this syntax (see Section 7).

Note that JSON Pointers are specified in characters, not as bytes.

4. Evaluation

Evaluation of a JSON Pointer begins with a reference to the root

value of a JSON document and completes with a reference to some value within the document. Each reference token in the JSON Pointer is sequentially evaluated.

Evaluation of each reference token begins by decoding any escaped character sequence; this is performed by first transforming any occurrence of the sequence '~1' to '/', then transforming any occurrence of the sequence '~0' to '~'. By performing the substitutions in this order, an implementation avoids the error of turning '~01' first into '~1' and then into '/', which would be incorrect (the string '~01' correctly becomes '~1' after transformation).

The reference token then modifies which value is referenced according to the following scheme:

- o If the currently referenced value is a JSON object, the new referenced value is the object member with the name identified by the reference token. The member name is equal to the token if it has the same number of Unicode characters as the token and their code points are position-wise equal. No Unicode character normalization is performed. If a referenced member name is not unique in an object, the member that is referenced is undefined, and evaluation fails (see below).
- o If the currently referenced value is a JSON array, the reference token MUST contain either:
 - * characters comprised of digits (see ABNF below; note that leading zeros are not allowed) that represent an unsigned base-10 integer value, making the new referenced value the array element with the zero-based index identified by the token, or
 - * exactly the single character "-", making the new referenced value the (non-existent) member after the last array element.

The ABNF syntax for array indices is:

```
array-index = %x30 / ( %x31-39 * (%x30-39) )  
              ; "0", or digits without a leading "0"
```

Implementations will evaluate each reference token against the document's contents, and will raise an error condition if it fails to resolve a concrete value for any of the JSON pointer's reference tokens. For example, if an array is referenced with a non-numeric token, an error condition will be raised. See Section 7 for details.

Note that the use of the "-" character to index an array will always result in such an error condition because by definition it refers to a non-existent array element. Applications of JSON Pointer thus need to specify how it is to be handled, if it is to be useful.

Any error condition that does not have a specific action defined for it by the JSON Pointer application results in termination of evaluation.

5. JSON String Representation

A JSON Pointer can be represented in a JSON string value. Per [RFC4627], Section 2.5, all instances of quotation mark '"' (%x22), reverse solidus '\' (%x5C) and control (%x00-1F) characters MUST be escaped.

Note that before processing a JSON string as a JSON Pointer, backslash escape sequences must be unescaped.

For example, given the JSON document

```
{
  "foo": ["bar", "baz"],
  "": 0,
  "a/b": 1,
  "c%d": 2,
  "e^f": 3,
  "g|h": 4,
  "i\\j": 5,
  "k\\l": 6,
  " ": 7,
  "m~n": 8
}
```

Then the following JSON strings evaluate to the accompanying values:

```
" "           // the whole document
"/foo"        ["bar", "baz"]
"/foo/0"      "bar"
"/"           0
"/a~1b"       1
"/c%d"        2
"/e^f"        3
"/g|h"        4
"/i\\j"       5
"/k\\l"       6
"/ "          7
"/m~0n"       8
```

6. URI Fragment Identifier Representation

A JSON Pointer can be represented in a URI fragment identifier by encoding it into octets using UTF-8 [RFC3629], percent-encoding those characters not allowed by the fragment rule in [RFC3986].

Note that a given media type needs to specify JSON Pointer as its fragment identifier syntax explicitly (usually, in its registration [RFC4288]); i.e., just because a document is JSON does not imply that JSON Pointer can be used as its fragment identifier syntax. In particular, the fragment identifier syntax for application/json is not JSON Pointer.

Given the same example document as above, the following URI fragment identifiers evaluate to the accompanying values:

```
#           // the whole document
#/foo       ["bar", "baz"]
#/foo/0     "bar"
#/"         0
#/a~1b      1
#/c%25d     2
#/e%5Ef     3
#/g%7Ch     4
#/i%5Cj     5
#/k%22l     6
#/%20       7
#/m~0n      8
```

7. Error Handling

In the event of an error condition, evaluation of the JSON Pointer fails to complete.

This includes, but is not limited to:

- o Invalid pointer syntax
- o A pointer that references a non-existent value

This specification does not define how errors are handled; an application of JSON Pointer SHOULD specify the impact and handling of each type of error.

For example, some applications might stop pointer processing upon an error; others may attempt to recover from missing values by inserting default ones.

8. IANA Considerations

This document has no impact upon IANA.

9. Security Considerations

A given JSON Pointer is not guaranteed to reference an actual JSON value. Therefore, applications using JSON Pointer should anticipate this by defining how a pointer that does not resolve ought to be handled.

Note that JSON pointers can contain the NUL (Unicode U+0000) character. Care is needed not to misinterpret this character in programming languages that use NUL to mark the end of a string.

10. Acknowledgements

The following individuals contributed ideas, feedback and wording to this specification:

Mike Acar, Carsten Bormann, Tim Bray, Jacob Davies, Martin J. Duerst, Bjoern Hoehrmann, James H. Manger, Drew Perttula, Julian Reschke.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

11.2. Informative References

- [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

Authors' Addresses

Paul C. Bryan (editor)
Salesforce.com

Phone: +1 604 783 1481
Email: pbryan@anode.ca

Kris Zyp
SitePen (USA)

Phone: +1 650 968 8787
Email: kris@sitepen.com

Mark Nottingham (editor)
Akamai

Email: mnot@mnot.net

APPSAWG
Internet-Draft
Intended status: Informational
Expires: May 26, 2014

M. Kucherawy
G. Shapiro
N. Freed
November 22, 2013

Advice for Safe Handling of Malformed Messages
draft-ietf-appsawg-malformed-mail-11

Abstract

Although Internet mail formats have been precisely defined since the 1970s, authoring and handling software often show only mild conformance to the specifications. The malformed messages that result are non-standard. Nonetheless, decades of experience has shown that handling with some tolerance the malformations that result is often an acceptable approach, and is better than rejecting the messages outright as nonconformant. This document includes a collection of the best advice available regarding a variety of common malformed mail situations, to be used as implementation guidance.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on May 26, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. The Purpose Of This Work	3
1.2. Not The Purpose Of This Work	4
1.3. General Considerations	4
2. Document Conventions	5
2.1. Examples	5
3. Background	5
4. Invariant Content	5
5. Mail Submission Agents	6
6. Line Termination	7
7. Header Anomalies	7
7.1. Converting Obsolete and Invalid Syntaxes	7
7.1.1. Host-Address Syntax	8
7.1.2. Excessive Angle Brackets	8
7.1.3. Unbalanced Angle Brackets	8
7.1.4. Unbalanced Parentheses	8
7.1.5. Commas in Address Lists	9
7.1.6. Unbalanced Quotes	9
7.1.7. Naked Local-Parts	10
7.2. Non-Header Lines	10
7.3. Unusual Spacing	11
7.4. Header Malformations	12
7.5. Header Field Counts	12
7.5.1. Repeated Header Fields	14
7.5.2. Missing Header Fields	15
7.5.3. Return-Path	16
7.6. Missing or Incorrect Charset Information	16
7.7. Eight-Bit Data	17
8. MIME Anomalies	18
8.1. Missing MIME-Version Field	18
8.2. Faulty Encodings	18
9. Body Anomalies	19
9.1. Oversized Lines	19
10. Security Considerations	19
11. IANA Considerations	19
12. References	20
12.1. Normative References	20
12.2. Informative References	20
Appendix A. RFC Editor Notes	21
Appendix B. Acknowledgements	21

1. Introduction

1.1. The Purpose Of This Work

The history of email standards, going back to [RFC733] and beyond, contains a fairly rigid evolution of specifications. However, implementations within that culture have also long had an undercurrent known formally as the robustness principle, also known informally as Postel's Law: "Be liberal in what you accept, and conservative in what you send." [RFC1122]

Jon Postel's directive is often misinterpreted to mean that any deviance from a specification is acceptable. Rather, it was intended only to account for legitimate variations in interpretation within specifications, as well as basic transit errors, like bit errors. Taken to its unintended extreme, excessive tolerance would imply that there are no limits to the liberties that a sender might take, while presuming a burden on a receiver to guess "correctly" at the meaning of any such variation. These matters are further compounded by receiver software -- the end users' mail readers -- which are also sometimes flawed, leaving senders to craft messages (sometimes bending the rules) to overcome those flaws.

In general, this served the email ecosystem well by allowing a few errors in implementations without obstructing participation in the game. The proverbial bar was set low. However, as we have evolved into the current era, some of these lenient stances have begun to expose opportunities that can be exploited by malefactors. Various email-based applications rely on strong application of these standards for simple security checks, while the very basic building blocks of that infrastructure, intending to be robust, fail utterly to assert those standards.

The distributed and non-interactive nature of email has often prompted adjustments to receiving software, to handle these variations, rather than trying to gain better conformance by senders, since the receiving operator is primarily driven by complaints from recipient users and has no authority over the sending side of the system. Processing with such flexibility comes at some cost, since mail software is faced with decisions about whether to permit non-conforming messages to continue toward their destinations unaltered, adjust them to conform (possibly at the cost of losing some of the original message), or outright rejecting them.

This document includes a collection of the best advice available regarding a variety of common malformed mail situations, to be used as implementation guidance. These malformations are typically based around loose interpretations or implementations of specifications

such as Internet Message Format [MAIL] and Multipurpose Internet Mail Extensions [MIME].

1.2. Not The Purpose Of This Work

It is important to understand that this work is not an effort to endorse or standardize certain common malformations. The code and culture that introduces such messages into the mail stream needs to be repaired, as the security penalty now being paid for this lax processing arguably outweighs the reduction in support costs to end users who are not expected to understand the standards. However, the reality is that this will not be fixed quickly.

Given this, it is beneficial to provide implementers with guidance about the safest or most effective way to handle malformed messages when they arrive, taking into consideration the tradeoffs of the choices available especially with respect to how various actors in the email ecosystem respond to such messages in terms of handling, parsing, or rendering to end users.

1.3. General Considerations

Many deviations from message format standards are considered by some receivers to be strong indications that the message is undesirable, such as spam or something containing malware. These receivers quickly decide that the best handling choice is simply to reject or discard the message. This means malformations caused by innocent misunderstandings or ignorance of proper syntax can cause messages with no ill intent also to fail to be delivered.

Senders that want to ensure message delivery are best advised to adhere strictly to the relevant standards (including, but not limited to, [MAIL], [MIME], and [DKIM]), as well as observe other industry best practices such as may be published from time to time either by the IETF or independently.

Receivers that haven't the luxury of strict enforcement of the standards on inbound messages are usually best served by observing the following guidelines for handling of malformed messages:

1. Whenever possible, mitigation of syntactic malformations should be guided by an assessment of the most likely semantic intent. For example, it is reasonable to conclude that multiple sets of angle brackets around an address are simply superfluous and can be dropped.
2. When the intent is unclear, or when it is clear but also impractical to change the content to reflect that intent,

mitigation should be limited to cases where not taking any corrective action would clearly lead to a worse outcome.

3. Security issues, when present, need to be addressed and may force mitigation strategies that are otherwise suboptimal.

2. Document Conventions

2.1. Examples

Examples of message content include a number within braces at the end of each line. These are line numbers for use in subsequent discussion, and are not actually part of the message content presented in the example.

Blank lines are not numbered in the examples.

3. Background

The reader would benefit from reading [EMAIL-ARCH] for some general background about the overall email architecture. Of particular interest is the Internet Message Format, detailed in [MAIL]. Throughout this document, the use of the term "message" should be assumed to mean a block of text conforming to the Internet Message Format.

4. Invariant Content

An agent handling a message could use several distinct representations of the message. One is an internal representation, such as separate blocks of storage for the header and body, some header or body alterations, or tables indexed by header name, set up to make particular kinds of processing easier. The other is the representation passed along to the next agent in the handling chain. This might be identical to the message input to the module, or it might have some changes such as added or reordered header fields or body elisions to remove malicious content.

Message handling is usually most effective when each in a sequence of handling modules receives the same content for analysis. A module that "fixes" or otherwise alters the content passed to later modules can prevent the later modules from identifying malicious or other content that exposes the end user to harm. It is important that all processing modules can make consistent assertions about the content. Modules that operate sequentially sometimes add private header fields to relay information downstream for later filters to use (and possibly remove), or they may have out-of-band ways of doing so. However, even the presence of private header fields can impact a

downstream handling agent unaware of its local semantics, so an out-of-band method is always preferable.

The above is less of a concern when multiple analysis modules are operated in parallel, independent of one another.

Often, abuse reporting systems can act effectively only when a complaint or report contains the original message exactly as it was generated. Messages that have been altered by handling modules might render a complaint inactionable as the system receiving the report may be unable to identify the original message as one of its own.

Some message changes alter syntax without changing semantics. For example, Section 7.4 describes a situation where an agent removes additional header whitespace. This is a syntax change without a change in semantics, though some systems (such as DKIM) are sensitive to such changes. Message system developers need to be aware of the downstream impact of making either kind of change.

Where a change to content between modules is unavoidable, adding trace data (such as prepending a standard Received field) will at least allow tracing of the handling by modules that actually see different input.

There will always be local handling exceptions, but these guidelines should be useful for developing integrated message processing environments.

In most cases, this document only discusses techniques used on internal representations. It is occasionally necessary to make changes between the input and output versions; such cases will be called out explicitly.

5. Mail Submission Agents

Within the email context, the single most influential component that can reduce the presence of malformed items in the email system is the Mail Handling Service (MHS; see [EMAIL-ARCH]), which includes the Mail Submission Agent (MSA). This is the component that is essentially the interface between end users that create content and the mail stream.

MHSes need to become more strict about enforcement of all relevant email standards, especially [MAIL] and the [MIME] family of documents.

More strict conformance by relaying Mail Transfer Agents (MTAs) will also be helpful. although preventing the dissemination of malformed

messages is desirable, the rejection of such mail already in transit also has a support cost, namely the creation of a [DSN] that many end users might not understand.

6. Line Termination

For interoperable Internet Mail messages, the only valid line separation sequence during a typical SMTP session is ASCII 0x0D ("carriage return", or CR) followed by ASCII 0x0A ("line feed", or LF), commonly referred to as CRLF. This is not the case for binary mode SMTP (see [BINARYSMTP]).

Common UNIX user tools, however, typically only use LF for internal line termination. This means that a protocol engine that converts between UNIX and Internet Mail formats has to convert between these two end-of-line representations before transmitting a message or after receiving it.

Non-compliant implementations can create messages with a mix of line terminations, such as LF everywhere except CRLF only at the end of the message. According to [SMTP] and [MAIL], this means the entire message actually exists on a single line.

Within modern Internet Mail it is highly unlikely that an isolated CR or LF is valid in common ASCII text. Furthermore, when content actually does need to contain such an unusual character sequence, [MIME] provides mechanisms for encoding that content in an SMTP-safe manner.

Thus, it will typically be safe and helpful to treat an isolated CR or LF as equivalent to a CRLF when parsing a message.

Note that this advice pertains only to the raw SMTP data, and not to decoded MIME entities. As noted above, when MIME encoding mechanisms are used, the unusual character sequences are not visible in the raw SMTP stream.

7. Header Anomalies

This section covers common syntactic and semantic anomalies found in a message header, and presents suggested mitigations.

7.1. Converting Obsolete and Invalid Syntaxes

A message using an obsolete header syntax (see Section 4 of [MAIL]) might confound an agent that is attempting to be robust in its handling of syntax variations. A bad actor could exploit such a weakness in order to get abusive or malicious content through a

filter. This section presents some examples of such variations. Messages including them ought be rejected; where this is not possible, recommended internal interpretations are provided.

7.1.1. Host-Address Syntax

The following obsolete syntax attempts to specify source routing:

To: <@example.net:fran@example.com>

This means "send to fran@example.com via the mail service at example.net". It can safely be interpreted as:

To: <fran@example.com>

7.1.2. Excessive Angle Brackets

The following over-use of angle brackets:

To: <<<user2@example.org>>>

can safely be interpreted as:

To: <user2@example.org>

7.1.3. Unbalanced Angle Brackets

The following use of unbalanced angle brackets:

To: <another@example.net

can usually be treated as:

To: <another@example.net>

The following:

To: second@example.org>

can usually be treated as:

To: second@example.org

7.1.4. Unbalanced Parentheses

The following use of unbalanced parentheses:

To: (Testing <fran@example.com>

can safely be interpreted as:

To: (Testing) <fran@example.com>

Likewise, this case:

To: Testing) <sam@example.com>

can safely be interpreted as:

To: "Testing)" <sam@example.com>

In both cases, it is obvious where the active email address in the string can be found. The former case retains the active email address in the string by completing what appears to be intended as a comment; the intent in the latter case is less obvious, so the leading string is interpreted as a display name.

7.1.5. Commas in Address Lists

This use of an errant comma:

To: <third@example.net, fourth@example.net>

can usually be interpreted as ending an address, so the above is usually best interpreted as:

To: third@example.net, fourth@example.net

7.1.6. Unbalanced Quotes

The following use of unbalanced quotation marks:

To: "Joe <joe@example.com>

leaves software with no obvious "good" interpretation. If it is essential to extract an address from the above, one possible interpretation is:

To: "Joe <joe@example.com>"@example.net

where "example.net" is the domain name or host name of the handling agent making the interpretation. Another possible interpretation, much simpler and likely more correct, is simply:

To: "Joe" <joe@example.com>

7.1.7. Naked Local-Parts

[MAIL] defines a local-part as the user portion of an email address, and the display-name as the "user-friendly" label that accompanies the address specification.

Some broken submission agents might introduce messages with only a local-part or only a display-name and no properly formed address. For example:

```
To: Joe
```

A submission agent ought to reject this or, at a minimum, append "@" followed by its own host name or some other valid name likely to enable a reply to be delivered to the correct mailbox. Where this is not done, an agent receiving such a message will probably be successful by synthesizing a valid header field for evaluation using the techniques described in Section 7.5.2.

7.2. Non-Header Lines

Some messages contain a line of text in the header that is not a valid message header field of any kind. For example:

```
From: user@example.com {1}
To: userpal@example.net {2}
Subject: This is your reminder {3}
about the football game tonight {4}
Date: Wed, 20 Oct 2010 20:53:35 -0400 {5}
```

```
Don't forget to meet us for the tailgate party! {7}
```

The cause of this is typically a bug in a message generator of some kind. Line {4} was intended to be a continuation of line {3}; it should have been indented by whitespace as set out in Section 2.2.3 of [MAIL].

This anomaly has varying impacts on processing software, depending on the implementation:

1. some agents choose to separate the header of the message from the body only at the first empty line (that is, a CRLF immediately followed by another CRLF);
2. some agents assume this anomaly should be interpreted to mean the body starts at line {4}, as the end of the header is assumed by encountering something that is not a valid header field or folded portion thereof;

3. some agents assume this should be interpreted as an intended header folding as described above and thus simply append a single space character (ASCII 0x20) and the content of line {4} to that of line {3};
4. some agents reject this outright as line {4} is neither a valid header field nor a folded continuation of a header field prior to an empty line.

This can be exploited if it is known that one message handling agent will take one action while the next agent in the handling chain will take another. Consider, for example, a message filter that searches message headers for properties indicative of abusive or malicious content that is attached to a Mail Transfer Agent (MTA) implementing option 2 above. An attacker could craft a message that includes this malformation at a position above the property of interest, knowing the MTA will not consider that content part of the header, and thus the MTA will not feed it to the filter, thus avoiding detection. Meanwhile, the Mail User Agent (MUA) which presents the content to an end user, implements option 1 or 3, which has some undesirable effect.

It should be noted that a few implementations choose option 4 above since any reputable message generation program will get header folding right, and thus anything so blatant as this malformation is likely an error caused by a malefactor.

The preferred implementation if option 4 above is not employed is to apply the following heuristic when this malformation is detected:

1. Search forward for an empty line. If one is found, then apply option 3 above to the anomalous line, and continue.
2. Search forward for another line that appears to be a new header field (a name followed by a colon). If one is found, then apply option 3 above to the anomalous line, and continue.

7.3. Unusual Spacing

The following message is valid per [MAIL]:

```
From: user@example.com {1}
To: userpal@example.net {2}
Subject: This is your reminder {3}
{4}
about the football game tonight {5}
Date: Wed, 20 Oct 2010 20:53:35 -0400 {6}
```

Don't forget to meet us for the tailgate party! {8}

Line {4} contains a single whitespace. The intended result is that lines {3}, {4}, and {5} comprise a single continued header field. However, some agents are aggressive at stripping trailing whitespace, which will cause line {4} to be treated as an empty line, and thus the separator line between header and body. This can affect header-specific processing algorithms as described in the previous section.

This example was legal in earlier versions of the Internet Mail format standard, but was rendered obsolete as of [RFC2822] as line {4} could be interpreted as the separator between the header and body.

The best handling of this example is for a message parsing engine to behave as if line {4} was not present in the message and for a message creation engine to emit the message with line {4} removed.

7.4. Header Malformations

Among the many possible malformations, a common one is insertion of whitespace at unusual locations, such as:

```
From: user@example.com {1}
To: userpal@example.net {2}
Subject: This is your reminder {3}
MIME-Version : 1.0 {4}
Content-Type: text/plain {5}
Date: Wed, 20 Oct 2010 20:53:35 -0400 {6}
```

Don't forget to meet us for the tailgate party! {8}

Note the addition of whitespace in line {4} after the header field name but before the colon that separates the name from the value.

The obsolete grammar of Section 4 of [MAIL] permits that extra whitespace, so it cannot be considered invalid. However, a consensus of implementations prefers to remove that whitespace. There is no perceived change to the semantics of the header field being altered as the whitespace is itself semantically meaningless. Therefore, it is best to remove all whitespace after the field name but before the colon and to emit the field in this modified form.

7.5. Header Field Counts

Section 3.6 of [MAIL] prescribes specific header field counts for a valid message. Few agents actually enforce these in the sense that a message whose header contents exceed one or more limits set there are

generally allowed to pass; they typically add any required fields that are missing, however.

Also, few agents that use messages as input, including Mail User Agents (MUAs) that actually display messages to users, verify that the input is valid before proceeding. Some popular open source filtering programs and some popular Mailing List Management (MLM) packages select either the first or last instance of a particular field name, such as From, to decide who sent a message. Absent strict enforcement of [MAIL], an attacker can craft a message with multiple instances of the same field fields if that attacker knows the filter will make a decision based on one but the user will be shown the others.

This situation is exacerbated when message validity is assessed, such as through enhanced authentication methods like DomainKeys Identified Mail [DKIM]. Such methods might cover one instance of a constrained field but not another, taking the wrong one as "good" or "safe". An MUA, for example could show the first of two From fields to an end user as "good" or "safe" while an authentication method actually only verified the second.

In attempting to counter this exposure, one of the following strategies can be used:

1. reject outright or refuse to process further any input message that does not conform to Section 3.6 of [MAIL];
2. remove or, in the case of an MUA, refuse to render any instances of a header field whose presence exceeds a limit prescribed in Section 3.6 of [MAIL] when generating its output;
3. where a field has a limited instance count, combine additional instances into a single instance carrying the same information as the multiple instances;
4. where a field can contain multiple distinct values (such as From) or is free-form text (such as Subject), combine them into a semantically identical single header field of the same name (see Section 7.5.1);
5. alter the name of any header field whose presence exceeds a limit prescribed in Section 3.6 of [MAIL] when generating its output so that later agents can produce a consistent result. Any alteration likely to cause the field to be ignored by downstream agents is acceptable. A common approach is to prefix the field names with a string such as "BAD-".

Selecting a mitigation action from the above list, or some other action, must consider the needs of the operator making the decision, and the nature of its user base.

7.5.1. Repeated Header Fields

There are some occasions where repeated fields are encountered where only one is expected. Two examples are presented. First:

```
From: reminders@example.com {1}
To: jqpublic@example.com {2}
Subject: Automatic Meeting Reminder {3}
Subject: 4pm Today -- Staff Meeting {4}
Date: Wed, 20 Oct 2010 08:00:00 -0700 {5}
```

```
Reminder of the staff meeting today in the small {6}
auditorium. Come early! {7}
```

The message above has two Subject fields, which is in violation of Section 3.6 of [MAIL]. A safe interpretation of this would be to treat it as though the two Subject field values were concatenated, so long as they are not identical, such as:

```
From: reminders@example.com {1}
To: jqpublic@example.com {2}
Subject: Automatic Meeting Reminder {3}
        4pm Today -- Staff Meeting {4}
Date: Wed, 20 Oct 2010 08:00:00 -0700 {5}
```

```
Reminder of the staff meeting today in the small {6}
auditorium. Come early! {7}
```

Second:

```
From: president@example.com {1}
From: vice-president@example.com {2}
To: jqpublic@example.com {3}
Subject: A note from the E-Team {4}
Date: Wed, 20 Oct 2010 08:00:00 -0700 {5}
```

```
This memo is to remind you of the corporate dress {6}
code. Attached you will find an updated copy of {7}
the policy. {8}
...
```

As with the first example, there is a violation in terms of the number of instances of the From field. A likely safe interpretation would be to combine these into a comma-separated address list in a

single From field:

```
From: president@example.com, {1}
      vice-president@example.com {2}
To: jqpublic@example.com {3}
Subject: A note from the E-Team {4}
Date: Wed, 20 Oct 2010 08:00:00 -0700 {5}
```

```
This memo is to remind you of the corporate dress {6}
code. Attached you will find an updated copy of {7}
the policy. {8}
...
```

7.5.2. Missing Header Fields

Similar to the previous section, there are messages seen in the wild that lack certain required header fields. In particular, [MAIL] requires that a From and Date field be present in all messages.

When presented with a message lacking these fields, the MTA might perform one of the following:

1. Make no changes
2. Add an instance of the missing field(s) using synthesized content based on data provided in other parts of the protocol

Option 2 is recommended for handling this case. Handling agents should add these for internal handling if they are missing, but should not add them to the external representation. The reason for this advice is that there are some filter modules that would consider the absence of such fields to be a condition warranting special treatment (for example, rejection), and thus the effectiveness of such modules would be stymied by an upstream filter adding them in a way visible to other components.

The synthesized fields should contain a best guess as to what should have been there; for From, the SMTP MAIL command's address can be used (if not null) or a placeholder address followed by an address literal (for example, unknown@[192.0.2.1]); for Date, a date extracted from a Received field is a reasonable choice.

One other important case to consider is a missing Message-Id field. An MTA that encounters a message missing this field should synthesize a valid one and add it to the external representation, since many deployed tools use the content of that field as a common unique message reference, so its absence inhibits correlation of message processing. Section 3.6.4 of [MAIL] describes advisable practise for

synthesizing the content of this field when it is absent, and establishes a requirement that it be globally unique.

7.5.3. Return-Path

A valid message will have exactly one Return-Path header field, as per Section 4.4 of [SMTP]. Should a message be encountered bearing more than one, all but the topmost one is to be disregarded, as it is most likely to have been added nearest to the mailbox that received that message.

7.6. Missing or Incorrect Charset Information

MIME provides the means to include textual material employing character sets ("charsets") other than US-ASCII. Such material is required to have an identified charset. Charset identification is done using a "charset" parameter in the Content-Type header field, a charset label within the MIME entity itself, or the charset can be implicitly specified by the Content-Type (see [CHARSET]).

It is unfortunately fairly common for required character set information to be missing or incorrect in textual MIME entities. As such, processing agents should perform basic sanity checks, such as:

- o US-ASCII contains bytes between 1 and 127 inclusive only (colloquially, "7-bit" data), so material including bytes outside of that range ("8-bit" data) is necessarily not US-ASCII. (See Section 2.3.1 of [MAIL].)
- o [UTF-8] has a very specific syntactic structure that other 8-bit charsets are unlikely to follow.
- o Null bytes (ASCII 0x00) are not allowed in either 7-bit or 8-bit data.
- o Not all 7-bit material is US-ASCII. The presence of the various escape sequences used for character switching can be used as an indication of the various charsets based on ISO/IEC 2022, such as those defined in [ISO-2022-CN], [ISO-2022-JP], and [ISO-2022-KR].

When a character set error is detected, processing agents should:

- a. apply heuristics to determine the most likely character set and, if successful, proceed using that information; or
- b. refuse to process the malformed MIME entity.

A null byte inside a textual MIME entity can cause typical string

processing functions to mis-identify the end of a string, which can be exploited to hide malicious content from analysis processes. Accordingly, null bytes require additional special handling.

A few null bytes in isolation is likely to be the result of poor message construction practices. Such nulls should be silently dropped.

Large numbers of null bytes are usually the result of binary material that is improperly encoded, improperly labeled, or both. Such material is likely to be damaged beyond the hope of recovery, so the best course of action is to refuse to process it.

Finally, the presence of null bytes may be used as indication of possible malicious intent.

7.7. Eight-Bit Data

Standards-compliant email messages do not contain any non-ASCII data without indicating that such content is present by means of published SMTP extensions. Absent that, MIME encodings are typically used to convert non-ASCII data to ASCII in a way that can be reversed by other handling agents or end users.

The best way to handle non-compliant 8bit material depends on its location.

Non-compliant 8bit material in MIME entity content should simply be processed as if the necessary SMTP extensions had been used to transfer the message. Note that improperly labeled 8bit material in textual MIME entities may require treatment as described in Section 7.6.

Non-compliant 8bit material in message or MIME entity header fields can be handled as follows:

- o Occurrences in unstructured text fields, comments, and phrases, can be converted into encoded-words (see [MIME3] if a likely character set can be determined). Alternatively, 8bit characters can be removed or replaced with some other character.
- o Occurrences in header fields whose syntax is unknown may be handled by dropping the field entirely or by removing/replacing the 8bit character as described above.
- o Occurrences in addresses are especially problematic. Agents supporting [EAI] may, if the 8bit material conforms to 8bit syntax, elect to treat the message as an EAI message and process

it accordingly. Otherwise, it is in most cases best to exclude the address from any sort of processing -- which may mean dropping it entirely -- since any attempt to fix it definitively is unlikely to be successful.

8. MIME Anomalies

The five-part set of MIME specifications includes a mechanism of message extensions for providing text in character sets other than ASCII, non-text attachments to messages, multi-part message bodies, and similar facilities.

Some anomalies with MIME-compliant generation are also common. This section discusses some of those and presents preferred mitigations.

8.1. Missing MIME-Version Field

Any message that uses [MIME] constructs is required to have a MIME-Version header field. Without it, the Content-Type and associated fields have no semantic meaning.

It is often observed that a message has complete MIME structure, yet lacks this header field. It is prudent to disregard this absence and conduct analysis of the message as if it were present, especially by agents attempting to identify malicious material.

Further, the absence of MIME-Version might be an indication of malicious intent, and extra scrutiny of the message may be warranted. Such omissions are not expected from compliant message generators.

8.2. Faulty Encodings

There have been a few different specifications of base64 in the past. The implementation defined in [MIME] instructs decoders to discard characters that are not part of the base64 alphabet. Other implementations consider an encoded body containing such characters to be completely invalid. Very early specifications of base64 (see [PEM], for example) allowed email-style comments within base64-encoded data.

The attack vector here involves constructing a base64 body whose meaning varies given different possible decodings. If a security analysis module wishes to be thorough, it should consider scanning the possible outputs of the known decoding dialects in an attempt to anticipate how the MUA will interpret the data.

9. Body Anomalies

9.1. Oversized Lines

A message containing a line of content that exceeds 998 characters plus the line terminator (1000 total) violates Section 2.1.1 of [MAIL]. Some handling agents may not look at content in a single line past the first 998 bytes, providing bad actors an opportunity to hide malicious content.

There is no specified way to handle such messages, other than to observe that they are non-compliant and reject them, or rewrite the oversized line such that the message is compliant.

To ensure long lines do not prevent analysis of potentially malicious data, handling agents are strongly encouraged to take one of the following actions:

1. Break such lines into multiple lines at a position that does not change the semantics of the text being thus altered. For example, breaking an oversized line such that a [URI] then spans two lines could inhibit the proper identification of that URI.
2. Rewrite the MIME part (or the entire message if not MIME) that contains the excessively long line using a content encoding that breaks the line in the transmission but would still result in the line being intact on decoding for presentation to the user. Both of the encodings declared in [MIME] can accomplish this.

10. Security Considerations

The discussions of the anomalies above and their prescribed solutions are themselves security considerations. The practises enumerated in this document are generally perceived as attempts to resolve security considerations that already exist rather than introducing new ones. However, some of the attacks described here may not have appeared in previous email specifications.

11. IANA Considerations

This document contains no actions for IANA.

[RFC Editor: Please remove this section prior to publication.]

12. References

12.1. Normative References

- [EMAIL-ARCH] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.
- [MAIL] Resnick, P., "Internet Message Format", RFC 5322, October 2008.
- [MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

12.2. Informative References

- [BINARYSMTP] Vaudreuil, G., "SMTP Service Extensions for Transmission of Large and Binary MIME Messages", RFC 3030, December 2000.
- [CHARSET] Melnikov, A. and J. Reschke, "Update to MIME regarding "charset" Parameter Handling in Textual Media Types", RFC 6657, July 2012.
- [DKIM] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011.
- [DSN] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, January 2003.
- [EAI] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, February 2012.
- [ISO-2022-CN] Zhu, HF., Hu, DY., Wang, ZG., Kao, TC., Chang, WCH., and M. Crispin, "Chinese Character Encoding for Internet Messages", RFC 1922, March 1996.
- [ISO-2022-JP] Murai, J., Crispin, M., and E. van der Poel, "Japanese Character Encoding for Internet Messages", RFC 1468, June 1993.
- [ISO-2022-KR] Choi, U., Chon, K., and H. Park, "Korean Character Encoding for Internet Messages", RFC 1557, December 1993.
- [MIME3] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

- [PEM] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I -- Message Encipherment and Authentication Procedures", RFC 1113, August 1989.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.
- [RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001.
- [RFC733] Crocker, D., Vittal, J., Pogran, K., and D. Henderson, Jr., "Standard for the Format of Internet Text Messages", RFC 733, November 1977.
- [SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.
- [URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.
- [UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, 2003.

Appendix A. RFC Editor Notes

[RFC Editor Note: This section can be removed before publication.]

I can't seem to figure out how to do this with xml2rfc, but the ISO-2022 reference above should contain the following URI:
http://www.iso.org/iso/catalogue_detail.htm?csnumber=22747

Appendix B. Acknowledgements

The author wishes to acknowledge the following for their review and constructive criticism of this proposal: Dave Cridland, Dave Crocker, Jim Galvin, Tony Hansen, John Levine, Franck Martin, Alexey Melnikov, and Timo Sirainen

Authors' Addresses

Murray S. Kucherawy

EMail: superuser@gmail.com

Gregory N. Shapiro

EMail: gshapiro@proofpoint.com

N. Freed

EMail: ned.freed@mrochek.com

Network Working Group
Internet-Draft
Obsoletes: 4288 (if approved)
Intended status: BCP
Expires: December 22, 2012

N. Freed
Oracle
J. Klensin
T. Hansen
AT&T Laboratories
June 20, 2012

Media Type Specifications and Registration Procedures
draft-ietf-appsawg-media-type-regs-14

Abstract

This document defines procedures for the specification and registration of media types for use in HTTP, MIME and other Internet protocols.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on December 22, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1.	Introduction	4
1.1.	Historical Note	4
1.2.	Conventions Used in This Document	5
2.	Media Type Registration Preliminaries	5
3.	Registration Trees and Subtype Names	5
3.1.	Standards Tree	5
3.2.	Vendor Tree	6
3.3.	Personal or Vanity Tree	7
3.4.	Unregistered x. Tree	8
3.5.	Additional Registration Trees	8
4.	Registration Requirements	8
4.1.	Functionality Requirement	8
4.2.	Naming Requirements	9
4.2.1.	Text Media Types	10
4.2.2.	Image Media Types	11
4.2.3.	Audio Media Types	11
4.2.4.	Video Media Types	11
4.2.5.	Application Media Types	11
4.2.6.	Multipart and Message Media Types	12
4.2.7.	Additional Top-level Types	12
4.2.8.	Structured Syntax Name Suffixes	12
4.2.9.	Deprecated Aliases	13
4.3.	Parameter Requirements	13
4.4.	Canonicalization and Format Requirements	14
4.5.	Interchange Recommendations	15
4.6.	Security Requirements	16
4.7.	Requirements specific to XML media types	17
4.8.	Encoding Requirements	17
4.9.	Usage and Implementation Non-requirements	18
4.10.	Publication Requirements	18
4.11.	Fragment Identifier Requirements	19
4.12.	Additional Information	19
5.	Media Type Registration Procedures	20
5.1.	Preliminary Community Review	20
5.2.	Submit request to IANA	20
5.2.1.	Provisional Registrations	21
5.3.	Review and Approval	21
5.4.	Comments on Media Type Registrations	22
5.5.	Change Procedures	22
5.6.	Registration Template	23
6.	Structured Syntax Suffix Registration Procedures	24
6.1.	Change Procedures	25
6.2.	Structured Syntax Suffix Registration Template	25

7. Security Considerations	26
8. IANA Considerations	26
9. Acknowledgments	27
10. References	28
10.1. Normative References	28
10.2. Informative References	29
Appendix A. Grandfathered Media Types	30
Appendix B. Changes Since RFC 4288	30
Authors' Addresses	32

1. Introduction

Recent Internet protocols have been carefully designed to be easily extensible in certain areas. In particular, many protocols, including but not limited to HTTP [RFC2616] and MIME [RFC2045], are capable of carrying arbitrary labeled content.

The mechanism used to label such content is a media type, consisting of a top-level type and a subtype, which is further structured into trees. Optionally, media types can define companion data, known as parameters.

A registration process is needed for these labels, so that the set of such values are defined in a reasonably orderly, well-specified, and public manner.

This document specifies the criteria for media type registrations and defines the procedures to be used to register media types (Section 5) as well as media type structured suffixes (Section 6) in the Internet Assigned Numbers Authority (IANA) central registry.

The location of the media type registry managed by these procedures is:

<http://www.iana.org/assignments/media-types/>

1.1. Historical Note

The media type registration process was initially defined for registering media types for use in the context of the asynchronous Internet mail environment. In this mail environment there is a need to limit the number of possible media types, to increase the likelihood of interoperability when the capabilities of the remote mail system are not known. As media types are used in new environments in which the proliferation of media types is not a hindrance to interoperability, the original procedure proved excessively restrictive and had to be generalized. This was initially done in [RFC2048], but the procedure defined there was still part of the MIME document set. The media type specification and registration procedure is now a separate document, to make it clear that it is independent of MIME.

It may be desirable to restrict the use of media types to specific environments or to prohibit their use in other environments. This specification incorporates such restrictions into media type registrations in a systematic way. See Section 4.9 for additional discussion.

1.2. Conventions Used in This Document

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] when they appear in ALL CAPS. They may also appear in lower or mixed case as plain English words, without any normative meaning.

This specification makes use of the Augmented Backus-Naur Form (ABNF) [RFC5234] notation, including the core rules defined in Appendix A of that document.

2. Media Type Registration Preliminaries

Registration of a new media type or types starts with the construction of a registration proposal. Registration may occur within several different registration trees that have different requirements, as discussed below. In general, a new registration proposal is circulated and reviewed in a fashion appropriate to the tree involved. The media type is then registered if the proposal is acceptable. The following sections describe the requirements and procedures used for each of the different registration trees.

3. Registration Trees and Subtype Names

In order to increase the efficiency and flexibility of the registration process, different structures of subtype names can be registered to accommodate the different natural requirements for, e.g., a subtype that will be recommended for wide support and implementation by the Internet community, or a subtype that is used to move files associated with proprietary software. The following subsections define registration "trees" that are distinguished by the use of faceted names, e.g., subtype names that begin with a "tree." prefix. Note that some media types defined prior to this document do not conform to the naming conventions described below. See Appendix A for a discussion of them.

3.1. Standards Tree

The standards tree is intended for types of general interest to the Internet community. Registrations in the standards tree MUST be either:

1. in the case of registrations in IETF specifications, approved directly by the IESG, or

2. registered by a recognized standards body using the "Specification Required" IANA registration policy [RFC5226] (which implies Expert Review).

The first procedure is used for registering registrations from IETF Consensus documents, or in rare cases when registering a grandfathered (see Appendix A) and/or otherwise incomplete registration is in the interest of the Internet community. The registration proposal MUST be published as an RFC. When the RFC is in the IETF stream it is an IETF Consensus RFC, which can be on the Standards Track, a BCP, Informational, or Experimental. Registrations published in non-IETF RFC streams are also allowed, and require IESG approval. A registration can be either in a standalone "registration only" RFC or incorporated into a more general specification of some sort.

In the second case the IESG makes a one time decision on whether the registration submitter represents a recognized standards body; after that, a Media Types Reviewer (Designated Expert or a group of Designated Experts) performs the Expert Review as specified in this document. Subsequent submissions from the same source do not involve the IESG. The format MUST be described by a formal standards specification produced by the submitting standards body.

Media types in the standards tree MUST NOT have faceted names, unless they are grandfathered in using the process described in Appendix A.

The "owner" of a media type registered in the standards tree is assumed to be the standards body itself. Modification or alteration of the specification uses the same level of processing (e.g., a registration submitted on Standards Track can be revised in another Standards Track RFC, but cannot be revised in an Informational RFC) required for the initial registration.

Standards-tree registrations from recognized standards bodies are submitted directly to the IANA, where they will undergo Expert Review [RFC5226] prior to approval. In this case, the Expert Reviewer(s) will, among other things, ensure that the required specification provides adequate documentation.

3.2. Vendor Tree

The vendor tree is used for media types associated with publicly available products. "Vendor" and "producer" are construed very broadly in this context and are considered equivalent. Note that industry consortia as well as non-commercial entities that do not qualify as recognized standards bodies can quite appropriately register media types in the vendor tree.

A registration may be placed in the vendor tree by anyone who needs to interchange files associated with some product or set of products. However, the registration properly belongs to the vendor or organization producing the software that employs the type being registered, and that vendor or organization can at any time elect to assert ownership of a registration done by a third party in order to correct or update it. See Section 5.5 for additional information.

When a third party registers a type on behalf of someone else both entities SHOULD be noted in the Change Controller field in the registration. One possible format for this would be "Foo, on behalf of Bar".

Registrations in the vendor tree will be distinguished by the leading facet "vnd.". That may be followed, at the discretion of the registrant, by either a media subtype name from a well-known producer (e.g., "vnd.mudpie") or by an IANA-approved designation of the producer's name that is followed by a media type or product designation (e.g., vnd.bigcompany.funnypictures).

While public exposure and review of media types to be registered in the vendor tree is not required, using the media-types@iana.org mailing list for review is encouraged to improve the quality of those specifications. Registrations in the vendor tree may be submitted directly to the IANA, where they will undergo Expert Review [RFC5226] prior to approval.

3.3. Personal or Vanity Tree

Registrations for media types created experimentally or as part of products that are not distributed commercially may be registered in the personal or vanity tree. The registrations are distinguished by the leading facet "prs.".

The owner of "personal" registrations and associated specifications is the person or entity making the registration, or one to whom responsibility has been transferred as described below.

While public exposure and review of media types to be registered in the personal tree is not required, using the media-types@iana.org mailing list (see Section 5.1) for review is encouraged to improve the quality of those specifications. Registrations in the personal tree may be submitted directly to the IANA, where they will undergo Expert Review [RFC5226] prior to approval.

3.4. Unregistered x. Tree

Subtype names with "x." as the first facet may be used for types intended exclusively for use in private, local environments. Types in this tree cannot be registered and are intended for use only with the active agreement of the parties exchanging them.

However, with the simplified registration procedures described above for vendor and personal trees, it should rarely, if ever, be necessary to use unregistered types. Therefore, use of types in the "x." tree is strongly discouraged.

Note that types with names beginning with "x-" are no longer considered to be members of this tree (see [I-D.ietf-appsawg-xdash]). Also note that if a generally useful and widely deployed type incorrectly ends up with an "x-" name prefix, it MAY be registered using its current name in an alternate tree by following the procedure defined in Appendix A.

3.5. Additional Registration Trees

From time to time and as required by the community, new top-level registration trees may be created by IETF Standards Action. It is explicitly assumed that these trees may be created for external registration and management by well-known permanent bodies; for example, scientific societies may register media types specific to the sciences they cover. In general, the quality of review of specifications for one of these additional registration trees is expected to be equivalent to registrations in the standards tree by a recognized Standards Development Organization. When the IETF performs such review, it needs to consider the greater expertise of the requesting body with respect to the subject media type.

4. Registration Requirements

Media type registrations are all expected to conform to various requirements laid out in the following sections. Note that requirement specifics sometimes vary depending on the registration tree, again as detailed in the following sections.

4.1. Functionality Requirement

Media types MUST function as actual media formats. Registration of things that are better thought of as a transfer encoding, as a charset, or as a collection of separate entities of another type, is not allowed. For example, although applications exist to decode the base64 transfer encoding [RFC2045], base64 cannot be registered as a

media type.

This requirement applies regardless of the registration tree involved.

4.2. Naming Requirements

All registered media types MUST be assigned top-level type and subtype names. The combination of these names serves to uniquely identify the media type, and the subtype name facet (or the absence of one) identifies the registration tree. Both top-level type and subtype names are case-insensitive.

Type and subtype names MUST conform to the following ABNF:

```
type-name = restricted-name
subtype-name = restricted-name

restricted-name = restricted-name-first *126restricted-name-chars
restricted-name-first = ALPHA / DIGIT
restricted-name-chars = ALPHA / DIGIT / "!" / "#" /
                        "$" / "&" / "-" / "^" / "_"
restricted-name-chars =/ "." ; Characters before first dot always
                           ; specify a facet name
restricted-name-chars =/ "+" ; Characters after last plus always
                           ; specify a structured syntax suffix
```

Note that this syntax is somewhat more restrictive than what is allowed by the ABNF in section 5.1 of [RFC2045] or section 4.2 of [RFC4288]. Also note that while this syntax allows names of up to 127 characters, implementation limits may make such long names problematic. For this reason the components of names SHOULD be limited to 64 characters.

Although the name syntax treats "." as equivalent to any other character, characters before any initial "." always specify the registration facet. Note that this means that facet-less standards tree registrations cannot use periods in the subtype name.

Similarly, "+" is used in subtype names to introduce a structured syntax specifier suffix. Structured syntax suffix requirements are specified in Section 4.2.8.

While it is possible for a given media type to be assigned additional names, the use of different names to identify the same media type is discouraged.

These requirements apply regardless of the registration tree

involved.

The choice of top-level type MUST take into account the nature of media type involved. New subtypes of top-level types MUST conform to the restrictions of the top-level type, if any. The following sections describe each of the initial set of top-level types and their associated restrictions. Additionally, various protocols, including but not limited to HTTP and MIME, MAY impose additional restrictions on the media types they can transport. (See [RFC2046] for additional information on the restrictions MIME imposes.)

4.2.1. Text Media Types

The "text" top-level type is intended for sending material that is principally textual in form.

Many subtypes of text, notably including the subtype "text/plain", which is a generic subtype for plain text defined in [RFC2046], define a "charset" parameter. If a "charset" parameter is defined for a particular subtype of text, it MUST be used to specify a charset name defined in accordance to the procedures laid out in [RFC2978].

As specified in [I-D.ietf-appsawg-mime-default-charset], a "charset" parameter SHOULD NOT be specified when charset information is transported inside the payload (e.g., as in "text/xml").

If a "charset" parameter is specified, it SHOULD be a required parameter, eliminating the options of specifying a default value. If there is a strong reason for the parameter to be optional despite this advice, each subtype MAY specify its own default value, or alternately, it MAY specify that there is no default value. Finally, the "UTF-8" charset [RFC3629] SHOULD be selected as the default. See [I-D.ietf-appsawg-mime-default-charset] for additional information on the use of "charset" parameters in conjunction with subtypes of text.

Regardless of what approach is chosen, all new text/* registrations MUST clearly specify how the charset is determined; relying on the US-ASCII default defined in Section 4.1.2 of [RFC2046] is no longer permitted. If explanatory text is needed this SHOULD be placed in the additional information section of the registration.

Plain text does not provide for or allow formatting commands, font attribute specifications, processing instructions, interpretation directives, or content markup. Plain text is seen simply as a linear sequence of characters, possibly interrupted by line breaks or page breaks. Plain text MAY allow the stacking of several characters in the same position in the text. Plain text in scripts like Arabic and

Hebrew may also include facilities that allow the arbitrary mixing of text segments with different writing directions.

Beyond plain text, there are many formats for representing what might be known as "rich text". An interesting characteristic of many such representations is that they are to some extent readable even without the software that interprets them. It is useful to distinguish them, at the highest level, from such unreadable data as images, audio, or text represented in an unreadable form. In the absence of appropriate interpretation software, it is reasonable to present subtypes of "text" to the user, while it is not reasonable to do so with most non-textual data. Such formatted textual data can be represented using subtypes of "text".

4.2.2. Image Media Types

A top-level type of "image" indicates that the content specifies one or more individual images. The subtype names the specific image format.

4.2.3. Audio Media Types

A top-level type of "audio" indicates that the content contains audio data. The subtype names the specific audio format.

4.2.4. Video Media Types

A top-level type of "video" indicates that the content specifies a time-varying-picture image, possibly with color and coordinated sound. The term 'video' is used in its most generic sense, rather than with reference to any particular technology or format, and is not meant to preclude subtypes such as animated drawings encoded compactly.

Note that although in general the mixing of multiple kinds media in a single body is discouraged [RFC2046], it is recognized that many video formats include a representation for synchronized audio and/or text, and this is explicitly permitted for subtypes of "video".

4.2.5. Application Media Types

The "application" top-level type is to be used for discrete data that do not fit under any of the other type names, and particularly for data to be processed by some type of application program. This is information that must be processed by an application before it is viewable or usable by a user. Expected uses for the "application" type name include but are not limited to file transfer, spreadsheets, presentations, scheduling data, and languages for "active"

(computational) material. (The last, in particular, can pose security problems that must be understood by implementors. The "application/postscript" media type registration in [RFC2046] provides a good example of how to handle these issues.)

For example, a meeting scheduler might define a standard representation for information about proposed meeting dates. An intelligent user agent would use this information to conduct a dialog with the user, and might then send additional material based on that dialog. More generally, there have been several "active" languages developed in which programs in a suitably specialized language are transported to a remote location and automatically run in the recipient's environment. Such applications may be defined as subtypes of the "application" top-level type.

The subtype of "application" will often either be the name or include part of the name of the application for which the data are intended. This does not mean, however, that any application program name may simply be used freely as a subtype of "application"; the subtype needs to be registered.

4.2.6. Multipart and Message Media Types

Multipart and message are composite types, that is, they provide a means of encapsulating zero or more objects, each one a separate media type.

All subtypes of multipart and message MUST conform to the syntax rules and other requirements specified in [RFC2046] and amended by Section 3.5 of [RFC6532].

4.2.7. Additional Top-level Types

In some cases a new media type may not "fit" under any currently defined top-level type names. Such cases are expected to be quite rare. However, if such a case does arise a new type name can be defined to accommodate it. Such a definition MUST be done via standards-track RFC; no other mechanism can be used to define additional type names.

4.2.8. Structured Syntax Name Suffixes

XML in MIME [RFC3023] defined the first such augmentation to the media type definition to additionally specify the underlying structure of that media type. To quote:

This document also standardizes a convention (using the suffix '+xml') for naming media types ... when those media types represent XML MIME (Multipurpose Internet Mail Extensions) entities.

That is, it specified a suffix (in that case, "+xml") to be appended to the base subtype name.

Since this was published, the de facto practice has arisen for using this suffix convention for other well-known structuring syntaxes. In particular, media types have been registered with suffixes such as "+der", "+fastinfoset" and "+json". This specification formalizes this practice and sets up a registry for structured type name suffixes.

The primary guideline for whether a structured type name suffix is registrable is that it be described by a readily-available description, preferably within a document published by an established standards organization, and for which there's a reference that can be used in a Normative References section of an RFC.

Media types that make use of a named structured syntax SHOULD use the appropriate registered "+suffix" for that structured syntax when they are registered. By the same token, media types MUST NOT be given names incorporating suffixes for structured syntaxes they do not actually employ. "+suffix" constructs for as-yet unregistered structured syntaxes SHOULD NOT be used, given the possibility of conflicts with future suffix definitions.

4.2.9. Deprecated Aliases

In some cases a single media type may have been widely deployed prior to registration under multiple names. In such cases a preferred name MUST be chosen for the media type and applications MUST use this to be compliant with the type's registration. However, a list of deprecated aliases the type is known by MAY be supplied as additional information in order to assist applications in processing the media type properly.

4.3. Parameter Requirements

Media types MAY elect to use one or more media type parameters, or some parameters may be automatically made available to the media type by virtue of being a subtype of a content type that defines a set of parameters applicable to any of its subtypes. In either case, the names, values, and meanings of any parameters MUST be fully specified when a media type is registered in the standards tree, and SHOULD be specified as completely as possible when media types are registered

in the vendor or personal trees.

Parameter names have the syntax as media type names and values:

```
parameter-name = restricted-name
```

Note that this syntax is somewhat more restrictive than what is allowed by the ABNF in [RFC2045] and amended by [RFC2231].

Parameter names are case-insensitive and no meaning is attached to the order in which they appear. It is an error for a specific parameter to be specified more than once.

There is no defined syntax for parameter values. Therefore registrations MUST specify parameter value syntax. Additionally, some transports impose restrictions on parameter value syntax, so care needs be taken to limit the use of potentially problematic syntaxes; e.g., pure binary valued parameters, while permitted in some protocols, are best avoided.

Note that a protocol can impose further restrictions on parameter value syntax, depending on how it chooses to represent parameters. Both MIME [RFC2045] [RFC2231] and HTTP [RFC2045] [RFC5987] allow binary parameters as well as parameter values expressed in a specific charset, but other protocols may be less flexible.

New parameters SHOULD NOT be defined as a way to introduce new functionality in types registered in the standards tree, although new parameters MAY be added to convey additional information that does not otherwise change existing functionality. An example of this would be a "revision" parameter to indicate a revision level of an external specification such as JPEG. Similar behavior is encouraged for media types registered in the vendor or personal trees, but is not required.

Changes to parameters (including the introduction of new ones) is managed in the same manner as other changes to the media type; see Section 5.5.

4.4. Canonicalization and Format Requirements

All registered media types MUST employ a single, canonical data format, regardless of registration tree.

A permanent and readily available public specification of the format for the media type MUST exist for all types registered in the standards tree, and this specification MUST provide sufficient detail so that interoperability between independent implementations using

the media type is possible. This specification **MUST** at a minimum be referenced by, if it is not actually included in, the media type registration proposal itself.

The specifications of format and processing particulars may or may not be publicly available for media types registered in the vendor and personal trees, and such registrations are explicitly permitted to limit the information in the registration to which software and version produce or process such media types. As such, references to or inclusion of format specifications in registrations is encouraged but not required. Note, however, that the public availability of a meaningful specification will often make the difference between simply having a name reserved so that there are no conflicts with other uses and having the potential for other implementations of the media type and useful interoperation with them.

Some media types involve the use of patented technology. The registration of media types involving patented technology is specifically permitted. However, the restrictions set forth in BCP 79 [RFC3979] and BCP 78 [RFC5378] on the use of patented technology in IETF standards-track protocols must be respected when the specification of a media type is part of a standards-track protocol. In addition, other standards bodies making use of the standards tree may have their own rules regarding intellectual property that must be observed in their registrations.

IPR disclosures for registrations in the vendor and personal tree are encouraged but not required.

4.5. Interchange Recommendations

Ideally media types will be defined so they interoperate across as many systems and applications as possible. However, some media types will inevitably have problems interoperating across different platforms. Problems with different versions, byte ordering, and specifics of gateway handling can and will arise.

Universal interoperability of media types is not required, but known interoperability issues **SHOULD** be identified whenever possible. Publication of a media type does not require an exhaustive review of interoperability, and the interoperability considerations section is subject to continuing evaluation.

These recommendations in this subsection apply regardless of the registration tree involved.

4.6. Security Requirements

An analysis of security issues **MUST** be done for all types registered in the standards tree. A similar analysis for media types registered in the vendor or personal trees is encouraged but not required. However, regardless of what security analysis has or has not been done, all descriptions of security issues **MUST** be as accurate as possible regardless of registration tree. In particular, the security considerations **MUST NOT** state that there are "no security issues associated with this type". Security considerations for types in the vendor or personal tree **MAY** say that "the security issues associated with this type have not been assessed".

There is absolutely no requirement that media types registered in any tree be secure or completely free from risks. Nevertheless, all known security risks **MUST** be identified in the registration of a media type, again regardless of registration tree.

The security considerations section of all registrations is subject to continuing evaluation and modification, and in particular **MAY** be extended by use of the "comments on media types" mechanism described in Section 5.4 below.

Some of the issues that need to be examined and described in a security analysis of a media type are:

- o Complex media types may include provisions for directives that institute actions on a recipient's files or other resources. In many cases provision is made for originators to specify arbitrary actions in an unrestricted fashion that may then have devastating effects. See the registration of the application/postscript media type in [RFC2046] for an example of such directives and how they can be described in a media type registration.
- o Any security analysis **MUST** state whether or not they employ such "active content", and if they do, they **MUST** state what steps have been taken, or **MUST** be taken by applications of the media type, to protect users of the media type from harm.
- o Complex media types may include provisions for directives that institute actions that, while not directly harmful to the recipient, may result in disclosure of information that either facilitates a subsequent attack or else violates a recipient's privacy in some way. Again, the registration of the application/postscript media type illustrates how such directives can be handled.

- o A media type that employs compression may provide an opportunity for sending a small amount of data that, when received and evaluated, expands enormously to consume all of the recipient's resources. All media types SHOULD state whether or not they employ compression, and if they do they SHOULD discuss what steps need to be taken to avoid such attacks.
- o A media type might be targeted for applications that require some sort of security assurance but not provide the necessary security mechanisms themselves. For example, a media type could be defined for storage of sensitive medical information that in turn requires external confidentiality and integrity protection services, or which is designed for use only within a secure environment. Types SHOULD always document whether or not they need such services in their security considerations.

4.7. Requirements specific to XML media types

There are a number of additional requirements specific to the registration of XML media types. These requirements are specified in [RFC3023].

4.8. Encoding Requirements

Some transports impose restrictions on the type of data they can carry. For example, Internet mail traditionally was limited to 7bit US-ASCII text. Encoding schemes are often used to work around such transport limitations.

It is therefore useful to note what sort of data a media type can consist of as part of its registration. An "encoding considerations" field is provided for this purpose. Possible values of this field are:

7bit: The content of the media type consists solely of CRLF-delimited 7bit US-ASCII text.

8bit: The content of the media type consists solely of CRLF-delimited 8bit text.

binary: The content consists of an unrestricted sequence of octets.

framed: The content consists of a series of frames or packets without internal framing or alignment indicators. Additional out-of-band information is needed to interpret the data properly, including but not necessarily limited to, knowledge of the boundaries between successive frames and knowledge of the transport mechanism. Note that media types of this sort cannot

simply be stored in a file or transported as a simple stream of octets; therefore, such media types are unsuitable for use in many traditional protocols. A commonly used transport with framed encoding is the Real-time Transport Protocol, RTP. Additional rules for framed encodings defined for transport using RTP are given in [RFC4855].

Additional restrictions on 7bit and 8bit text are given in Section 4.1.1 of [RFC2046].

4.9. Usage and Implementation Non-requirements

In the asynchronous mail environment, where information on the capabilities of the remote mail agent is frequently not available to the sender, maximum interoperability is attained by restricting the media types used to those "common" formats expected to be widely implemented. This was asserted in the past as a reason to limit the number of possible media types, and resulted in a registration process with a significant hurdle and delay for those registering media types.

However, the need for "common" media types does not require limiting the registration of new media types. If a limited set of media types is recommended for a particular application, that should be asserted by a separate applicability statement specific for the application and/or environment.

Therefore, universal support and implementation of a media type is NOT a requirement for registration. However, if a media type is explicitly intended for limited use, this MUST be noted in its registration. The "Restrictions on Usage" field is provided for this purpose.

4.10. Publication Requirements

Media types registered in the standards tree by the IETF itself MUST be published as RFCs. RFC publication of vendor and personal media type registrations is allowed but not required. In all cases the IANA will retain copies of all media type registrations and "publish" them as part of the media types registration tree itself.

As stated previously, standards tree registrations for media types defined in documents produced by other standards bodies MUST be described by a formal standards specification produced by that body. Additionally, any copyright on the registration template MUST allow the IANA to copy it into the IANA registry.

Other than IETF registrations in the standards tree, the registration

of a media type does not imply endorsement, approval, or recommendation by the IANA or the IETF or even certification that the specification is adequate. To become an Internet Standard, a protocol or data object must go through the IETF standards process. While it provides additional assurances when it is appropriate, this is too difficult and too lengthy a process for the convenient registration of media types.

The standards tree exists for media types that do require a substantive review and approval process in a recognized standards body. The vendor and personal trees exist for those media types that do not require such a process. It is expected that applicability statements for particular applications will be published from time to time in the IETF, recommending implementation of, and support for, media types that have proven particularly useful in those contexts.

As discussed above, registration of a top-level type requires Standards Action in the IETF and, hence, the publication of a RFC on the Standards Track.

4.11. Fragment Identifier Requirements

Media type registrations can specify how applications should interpret fragment identifiers (specified in section 3.5 of [RFC3986]) associated with the media type.

Media types are encouraged to adopt fragment identifier schemes that are used with semantically similar media types. In particular, media types that use a named structured syntax with a registered "+suffix" MUST follow whatever fragment identifier rules are given in the structured syntax suffix registration.

4.12. Additional Information

Various sorts of optional information SHOULD be included in the specification of a media type if it is available:

- o Magic number(s) (length, octet values). Magic numbers are byte sequences that are always present at a given place in the file and thus can be used to identify entities as being of a given media type.
- o File name extension(s) commonly used on one or more platforms to indicate that some file contains a given media type.
- o Mac OS File Type code(s) (4 octets) used to label files containing a given media type. Some discussion of Macintosh file type codes and their purpose can be found in [MacOSFileTypes].

In the case of a registration in the standards tree, this additional information MAY be provided in the formal specification of the media type format. It is suggested that this be done by incorporating the IANA media type registration form into the format specification itself.

5. Media Type Registration Procedures

The media type registration procedure is not a formal standards process, but rather an administrative procedure intended to allow community comment and sanity checking without excessive time delay.

Normal IETF processes need to be followed for all IETF registrations in the standards tree. The posting of an Internet Draft is a necessary first step, followed by posting to the media-types@iana.org list as discussed below.

5.1. Preliminary Community Review

Notice of a potential media type registration in the standards tree SHOULD be sent to the media-types@iana.org mailing list for review. This mailing list has been established for the purpose of reviewing proposed media and access types. Registrations in other trees MAY be sent to the list for review as well; doing so is entirely OPTIONAL, but is strongly encouraged.

The intent of the public posting to this list is to solicit comments and feedback on the choice of type/subtype name, the unambiguity of the references with respect to versions and external profiling information, and a review of any interoperability or security considerations. The submitter may submit a revised registration proposal or abandon the registration completely and at any time.

5.2. Submit request to IANA

Media types registered in the standards tree by the IETF itself MUST be reviewed and approved by the IESG as part of the normal standards process. Standards tree registrations by recognized standards bodies as well as registrations in the vendor and personal tree are submitted directly to the IANA, unless other arrangements were made as part of a liaison agreement. In either case posting the registration to the media-types@iana.org list for review prior to submission is strongly encouraged.

Registration requests can be sent to iana@iana.org. A web form for registration requests is also available:

<http://www.iana.org/cgi-bin/mediatypes.pl>

5.2.1. Provisional Registrations

Standardization processes often take considerable time to complete. In order to facilitate prototyping and testing it is often helpful to assign identifiers, including but not limited to media types, early in the process. This way identifiers used during standards development can remain unchanged once the process is complete and implementations and documentation do not have to be updated.

Accordingly, a provisional registration process is provided to support early assignment of media type names in the standards tree. A provisional registration MAY be submitted to IANA for standards tree types. The only required fields in such registrations are the media type name and contact information (including the standards body name).

Upon receipt of a provisional registration, IANA will check the name and contact information, then publish the registration in a separate publicly visible provisional registration list.

Provisional registrations MAY be updated or abandoned at any time. When the registration is abandoned the media type is no longer registered in any sense; it can subsequently be registered just like any other unassigned media type name.

5.3. Review and Approval

With the exception of provisional standards tree registrations, registrations submitted to the IANA will be passed on to the media types reviewer. The media types reviewer, who is appointed by the IETF Applications Area Director(s), will review the registration to make sure it meets the requirements set forth in this document. Registrations that do not meet these requirements will be returned to the submitter for revision.

Decisions made by the media types reviewer may be appealed to the IESG using the procedure specified in section 6.5.4 of [RFC2026].

Once a media type registration has passed review, the IANA will register the media type and make the media type registration available to the community.

In the case of standards tree registrations from other standards bodies IANA will also check that the submitter is in fact a recognized standards body. If the submitter is not currently recognized as such the IESG will be asked to confirm their status.

Recognition from the IESG MUST be obtained before a standards tree registration can proceed.

5.4. Comments on Media Type Registrations

Comments on registered media types may be submitted by members of the community to the IANA at iana@iana.org. These comments will be reviewed by the media types reviewer and then passed on to the "owner" of the media type if possible. Submitters of comments may request that their comment be attached to the media type registration itself, and if the IANA, in consultation with the media types reviewer, approves, the comment will be made accessible in conjunction with the type registration.

5.5. Change Procedures

Once a media type has been published by the IANA, the owner may request a change to its definition. The descriptions of the different registration trees above designate the "owners" of each type of registration. The same procedure that would be appropriate for the original registration request is used to process a change request.

Media type registrations may not be deleted; media types that are no longer believed appropriate for use can be declared OBSOLETE by a change to their "intended use" field; such media types will be clearly marked in the lists published by the IANA.

Significant changes to a media type's definition should be requested only when there are serious omissions or errors in the published specification. When review is required, a change request may be denied if it renders entities that were valid under the previous definition invalid under the new definition.

The owner of a media type may pass responsibility to another person or agency by informing the IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a media type. The most common case of this will be to enable changes to be made to types where the author of the registration has died, moved out of contact or is otherwise unable to make changes that are important to the community.

5.6. Registration Template

Type name:

Subtype name:

Required parameters:

Optional parameters:

Encoding considerations:

Security considerations:

Interoperability considerations:

Published specification:

Applications that use this media type:

Fragment identifier considerations:

Additional information:

 Deprecated alias names for this type:

 Magic number(s):

 File extension(s):

 Macintosh file type code(s):

Person & email address to contact for further information:

Intended usage:

(One of COMMON, LIMITED USE or OBSOLETE.)

Restrictions on usage:

(Any restrictions on where the media type can be used go here.)

Author:

Change controller:

Provisional registration? (standards tree only):

(Any other information that the author deems interesting may be added below this line.)

"N/A", written exactly that way, can be used in any field if desired to emphasize the fact that it does not apply or that the question was not omitted by accident. Do not use 'none' or other words that could be mistaken for a response.

Limited use media types should also note in the applications list whether or not that list is exhaustive.

6. Structured Syntax Suffix Registration Procedures

Someone wishing to define a "+suffix" name for a structured syntax for use with a new media type registration SHOULD:

1. Check IANA's registry of media type name suffixes to see whether or not there is already an entry for that well-defined structured syntax.
2. If there is no entry for their suffix scheme, fill out the template (specified in Section 6.2) and include that with the media type registration. The template may be contained in an Internet Draft, alone or as part of some other protocol specification. The template may also be submitted in some other form (as part of another document or as a stand-alone document), but the contents will be treated as an "IETF Contribution" under the guidelines of BCP 78 [RFC5378].
3. Send a copy of the template or a pointer to the containing document (with specific reference to the section with the template) to the mailing list media-types@iana.org, requesting review. This may be combined with a request to review the media type registration. Allow a reasonable time for discussion and comments.
4. Respond to review comments and make revisions to the proposed registration as needed to bring it into line with the guidelines given in this document.
5. Submit the (possibly updated) registration template (or pointer to the document containing it) to IANA at iana@iana.org.

Upon receipt of a structured syntax suffix registration request,

1. IANA checks the submission for completeness; if sections are missing or citations are not correct, IANA rejects the registration request.

2. IANA checks the current registry for an entry with the same name; if such a registry exists, IANA rejects the registration request.
3. IANA requests Expert Review of the registration request against the corresponding guidelines.
4. The Designated Expert may request additional review or discussion, as necessary.
5. If Expert Review recommends registration, IANA adds the registration to the appropriate registry.

The initial registry content specification

[I-D.appsawg-media-type-suffix-regs] provides examples of structured syntax suffix registrations.

6.1. Change Procedures

Registrations may be updated in each registry by the same mechanism as required for an initial registration. In cases where the original definition of the scheme is contained in an IESG-approved document, update of the specification also requires IESG approval.

6.2. Structured Syntax Suffix Registration Template

This template describes the fields that must be supplied in a structured syntax suffix registration request:

Name

Full name of the well-defined structured syntax.

+suffix

Suffix used to indicate conformance to the syntax.

References

Include full citations for all specifications necessary to understand the structured syntax.

Encoding considerations

General guidance regarding encoding considerations for any type employing this syntax should be given here. The same requirements for media type encoding considerations given in Section 4.8 apply here.

Interoperability considerations

Any issues regarding the interoperable use of types employing this structured syntax should be given here. Examples would include the existence of incompatible versions of the syntax, issues

combining certain charsets with the syntax, or incompatibilities with other types or protocols.

Fragment identifier considerations

Generic processing of fragment identifiers for any type employing this syntax should be described here.

Security considerations

Security considerations shared by media types employing this structured syntax must be specified here. The same requirements for media type security considerations given in Section 4.6 apply here, with the exception that the option of not assessing the security considerations is not available for suffix registrations.

Contact

Person (including contact information) to contact for further information.

Author/Change controller.

Person (including contact information) authorized to change this suffix registration.

7. Security Considerations

Security requirements for both media type and media type suffix registrations are discussed in Section 4.6.

8. IANA Considerations

The purpose of this document is to define IANA registries for media types and structured syntax suffixes as well as the procedures for managing these registries. Additionally, this document requires IANA to maintain a list of IESG-recognized standards bodies who are allowed to register types in the standards tree.

The existing media type registry has been extended to include a section for provisional registrations. Only standards tree registrations are allowed in the standards tree and only at the request of a standards body on the IESG-recognized standards body list. See Section 5.2.1 for additional information on provisional registrations.

IANA is also requested to add the following note at the top of the provisional registry:

This registry, unlike some other provisional IANA registries, is only for temporary use. Entries in this registry are either finalized and moved to the main media types registry, or are abandoned and deleted. Entries in this registry are suitable for use for use for development and test purposes only.

The structured syntax name suffix registry is to be created as follows:

- o The name is the "Structured Syntax Suffix" registry.
- o The registration process is specified in Section 6.
- o The information required for a registry entry as well as the entry format are specified in Section 6.2.
- o The initial content of the registry is specified in [I-D.appsaug-media-type-suffix-regs].

Entries in both the media type and structured suffix registries will be annotated by IANA with both the original registration date as well as the date of the most recent update to the entry. Registrations made prior to the implementations of this specification can be marked as "registered under RFC 4288 or earlier".

Since registration entries can be updated multiple times, IANA is also requested to maintain the history of changes to each registration in such a way that the state of the registration at any given time can be determined

Finally, this document calls for the creation of a new email address, `media-types@iana.org`, for the media type review list, which replaces the `ietf-types@iana.org` address specified in RFC 4288. `ietf-types@iana.org` should be retained as an alias.

9. Acknowledgments

The current authors would like to acknowledge their debt to the late Dr. Jon Postel, whose general model of IANA registration procedures and specific contributions shaped the predecessors of this document [RFC2048] [RFC4288]. We hope that the current version is one with which he would have agreed but, as it is impossible to verify that agreement, we have regretfully removed his name as a co-author.

Randy Bush, Francis Dupont, Bjoern Hoehrmann, Barry Leiba, Murray Kucherawy, Alexey Melnikov, S. Moonesamy, Mark Nottingham, Tom Petch, Peter Saint-Andre, and Jeni Tennison provided many helpful review

comments and suggestions.

10. References

10.1. Normative References

- [I-D.appsawg-media-type-suffix-regs]
Hansen, T., "Additional Media Type Structured Syntax Suffixes", draft-appsawg-media-type-suffix-regs-00 (work in progress), April 2012.
- [I-D.ietf-appsawg-mime-default-charset]
Melnikov, A. and J. Reschke, "Update to MIME regarding Charset Parameter Handling in Textual Media Types", draft-ietf-appsawg-mime-default-charset-01 (work in progress), March 2012.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2978] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC3979] Bradner, S., "Intellectual Property Rights in IETF Technology", BCP 79, RFC 3979, March 2005.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5378] Bradner, S. and J. Contreras, "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, November 2008.
- [RFC6532] Yang, A., Steel, S., and N. Freed, "Internationalized Email Headers", RFC 6532, January 2012.

10.2. Informative References

- [I-D.ietf-appsawg-xdash]
Saint-Andre, P. and D. Crocker, "Deprecating the X- Prefix and Similar Constructs in Application Protocols", draft-ietf-appsawg-xdash-05 (work in progress), April 2012.
- [MacOSFileTypes]
Apple Computer, Inc., "Mac OS: File Type and Creator Codes, and File Formats", Apple Knowledge Base Article 55381, June 1993,
<<http://www.info.apple.com/kbnum/n55381>>.
- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.
- [RFC2048] Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 2048, November 1996.
- [RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.
- [RFC2616] 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.
- [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.
- [RFC5987] Reschke, J., "Character Set and Language Encoding for Hypertext Transfer Protocol (HTTP) Header Field

Parameters", RFC 5987, August 2010.

Appendix A. Grandfathered Media Types

A number of media types with unfaceted subtype names, registered prior to 1996, would, if registered under the guidelines in this document, be given a faceted name and placed into either the vendor or personal trees. Reregistration of those types to reflect the appropriate trees is encouraged but not required. Ownership and change control principles outlined in this document apply to those types as if they had been registered in the trees described above.

From time to time there may also be cases where a media type with an unfaceted subtype name has been widely deployed without being registered. (Note that this includes subtype names beginning with the "x-" prefix.) If possible such media type SHOULD be reregistered with a proper faceted subtype name. However, if this is not possible the type can, subject to approval by both the media types reviewer and the IESG, be registered in the proper tree with its unfaceted name.

Appendix B. Changes Since RFC 4288

- o Suffixes to indicate the use of a particular structured syntax are now fully specified and a suffix registration process has been defined.
- o Registration of widely deployed unregistered unfaceted type names in the vendor or personal trees is now allowed, subject to approval by the media types reviewer and the IESG.
- o The standards tree registration process has been revised to include Expert Review and generalized to address cases like media types in non-IETF stream documents.
- o A field for fragment identifiers has been added to the registration template and brief directions for specifying fragment identifiers have been added.
- o The specification requirements for personal tree registrations have been changed to be the same as those for the vendor tree. The text has been changed to encourage (but not require) specification availability.
- o The definition of additional trees has been clarified to state that an IETF Standards Action is required.

- o Widely deployed types with "x-" names can now be registered as an exception in the vendor tree.
- o The requirements on changes to registrations have been loosened so minor changes are easier to make.
- o The registration process has been completely restructured so that with the exception of IETF-generated types in the standards tree, all requests are processed by IANA and not the IESG.
- o A provisional registration process has been added for early assignment of types in the standards tree.
- o Many editorial changes have been made throughout the document to make the requirements and processes it describes clearer and easier to follow.
- o The ability to specify a list of deprecated aliases for a media type has been added.
- o Types with names beginning with "x-" are no longer considered to be members of the unregistered "x." tree. As with any unfaceted type, special procedures have been added to allow registration of such types in an appropriate tree.
- o Changes to a type registered by a third party may now be made by the designated change controller even if that isn't the vendor or organization that created the type. However, the vendor or organization may elect to assert ownership and change controller over the type at any time.
- o Limited use media types are now asked to note whether or not the supplied list of applications employing the media type is exhaustive.
- o The ABNF for media type names has been further restricted to require that names begin with an alphanumeric character.
- o Mailing list review is no longer required prior to registration of media types. Additionally, the address associated with the media type review mailing list has been changed to media-types@iana.org.
- o The rules for text/* media types have been updated to reflect the changes specified in [I-D.ietf-appsawg-mime-default-charset].

Authors' Addresses

Ned Freed
Oracle
800 Royal Oaks
Monrovia, CA 91016-6347
USA

Email: ned+ietf@mrochek.com

John C. Klensin
1770 Massachusetts Ave, #322
Cambridge, MA 02140
USA

Email: john+ietf@jck.com

Tony Hansen
AT&T Laboratories
200 Laurel Ave.
Middletown, NJ 07748
USA

Email: tony+mtsuffix@maillennium.att.com

Network Working Group
Internet-Draft
Updates: 3023 (if approved)
Intended status: Informational
Expires: May 9, 2013

T. Hansen
AT&T Laboratories
A. Melnikov
Isode Ltd
November 5, 2012

Additional Media Type Structured Syntax Suffixes
draft-ietf-appsawg-media-type-suffix-regs-08

Abstract

A content media type name sometimes includes partitioned meta-information distinguish by a Structured Syntax, to permit noting an attribute of the media as a suffix to the name. This document defines several Structured Syntax Suffixes for use with media type registrations. In particular, it defines and registers the "+json", "+ber", "+der", "+fastinfoset", "+wbxml" and "+zip" Structured Syntax Suffixes, and provides a Message Type Structured Syntax Suffix registration form for the "+xml" Structured Syntax Suffix.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on May 9, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. When to Use these Structured Syntax Suffixes	3
3. Initial Structured Syntax Suffix Definitions	4
3.1. The +json Structured Syntax Suffix	4
3.2. The +ber Structured Syntax Suffixes	5
3.3. The +der Structured Syntax Suffixes	6
3.4. The +fastinfoset Structured Syntax Suffix	8
3.5. The +wbxml Structured Syntax Suffix	9
3.6. The +zip Structured Syntax Suffix	10
4. IANA Considerations	11
5. Security Considerations	13
6. References	13
6.1. Normative References	13
6.2. Informative References	14
Appendix A. Change History	14
Authors' Addresses	15

1. Introduction

[RFC3023] created the +xml suffix convention that can be used when defining names for media types whose representation uses XML underneath. That is, they could have been successfully parsed as if the media type had been application/xml in addition to their being parsed as their media type that is using the +xml suffix. [I-D.ietf-appsawg-media-type-regs] defines the Message Type Structured Syntax Suffixes registry to be used for such Structured Syntax Suffixes.

A variety of Structured Syntax Suffixes have already been used in some media type registrations, in particular "+json", "+der", "+fastinfoset" and "+wbxml". This document defines and registers these Structured Syntax Suffixes in the Structured Syntax Suffix registry, along with "+ber" and "+zip". In addition, this document updates [RFC3023] to formally register the "+xml" Structured Syntax Suffix according to procedure defined in [I-D.ietf-appsawg-media-type-regs].

Discussion of this document should occur in the Apps Area Working Group (apps-discuss@ietf.org). [RFC Editor note: remove this paragraph.]

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].

2. When to Use these Structured Syntax Suffixes

Each of the Structured Syntax Suffixes defined in this document is appropriate for use when the media type identifies the semantics of the protocol payload. That is, knowing the semantics of the specific media type provides for more specific processing of the content than that afforded by generic processing of the underlying representation.

At the same time, using the suffix allows receivers of the media types to do generic processing of the underlying representation in cases where

- they do not need to perform special handling of the particular semantics of the exact media type, and,

- there is no special knowledge needed by such a generic processor in order to parse that underlying representation other than what would be needed to parse any example of that underlying representation.

3. Initial Structured Syntax Suffix Definitions

3.1. The +json Structured Syntax Suffix

[RFC4627] defines the "application/json" media type. The suffix "+json" MAY be used with any media type whose representation follows that established for "application/json". The Message Type Structured Syntax Suffix registration form follows. See [I-D.ietf-appsawg-media-type-regs] for definitions of each of the registration form headings.

Name: JavaScript Object Notation (JSON)

+suffix: +json

References: [RFC4627]

Encoding considerations: Per [RFC4627], JSON is allowed to be represented using UTF-8, UTF-16, or UTF-32. When JSON is written in UTF-8, JSON is 8bit compatible ([RFC2045]). When JSON is written in UTF-16 or UTF-32, JSON is binary ([RFC2045]).

Fragment identifier considerations:

The syntax and semantics of fragment identifiers specified for +json SHOULD be as specified for "application/json". (At publication of this document, there is no fragment identification syntax defined for "application/json".)

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+json" SHOULD be processed as follows:

For cases defined in +json, where the fragment identifier resolves per the +json rules, then as specified in +json.

For cases defined in +json, where the fragment identifier does not resolve per the +json rules, then as specified in "xxx/yyy+json".

For cases not defined in +json, then as specified in "xxx/yyy+json".

Interoperability considerations: n/a

Security considerations: See [RFC4627]

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has change control over this registration.

3.2. The +ber Structured Syntax Suffixes

The ITU defined the Basic Encoding Rules (BER) message transfer syntax in [ITU.X690.2008]. The suffix "+ber" MAY be used with any media type whose representation follows the BER message transfer syntax. (The expert reviewer for Message Type Structured Syntax Suffix registrations ought to be aware of the relationship between BER and DER to aid in selecting the proper suffix.) The Message Type Structured Syntax Suffix registration form for +ber follows:

Name: Basic Encoding Rules (BER) message transfer syntax

+suffix: +ber

References: [ITU.X690.2008]

Encoding considerations: BER is a binary encoding.

Fragment identifier considerations:

At publication of this document, there is no fragment identification syntax defined for +ber.

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+ber" SHOULD be processed as follows:

For cases defined in +ber, where the fragment identifier resolves per the +ber rules, then as specified in +ber.

For cases defined in +ber, where the fragment identifier does not resolve per the +ber rules, then as specified in "xxx/yyy+ber".

For cases not defined in +ber, then as specified in "xxx/yyy+ber".

Interoperability considerations: n/a

Security considerations: Each individual media type registered with a +ber suffix can have additional security considerations.

BER has a type-length-value structure, and it is easy to construct malicious content with invalid length fields that can cause buffer overrun conditions.

BER allows for arbitrary levels of nesting, which may make it possible to construct malicious content that will cause a stack overflow.

Interpreters of the BER structures should be aware of these issues and should take appropriate measures to guard against buffer overflows and stack overruns in particular and malicious content in general.

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has change control over this registration.

3.3. The +der Structured Syntax Suffixes

The ITU defined the Distinguished Encoding Rules (DER) message transfer syntax in [ITU.X690.2008]. The suffix "+der" MAY be used with any media type whose representation follows the DER message transfer syntax. (The expert reviewer for Message Type Structured Syntax Suffix registrations ought to be aware of the relationship between BER and DER to aid in selecting the proper suffix.) The Message Type Structured Syntax Suffix registration form for +der follows:

Name: Distinguished Encoding Rules (DER) message
transfer syntax

+suffix: +der

References: [ITU.X690.2008]

Encoding considerations: DER is a binary encoding.

Fragment identifier considerations:

At publication of this document, there is no
fragment identification syntax defined for
+der.

The syntax and semantics for fragment
identifiers for a specific "xxx/yyy+der"
SHOULD be processed as follows:

For cases defined in +der, where the
fragment identifier resolves per the +der
rules, then as specified in +der.

For cases defined in +der, where the
fragment identifier does not resolve per
the +der rules, then as specified in "xxx/
yyy+der".

For cases not defined in +der, then as
specified in "xxx/yyy+der".

Interoperability considerations: n/a

Security considerations: Each individual media type registered with
a +der suffix can have additional security
considerations.

DER has a type-length-value structure, and it is
easy to construct malicious content with invalid
length fields that can cause buffer overrun
conditions.

DER allows for arbitrary levels of nesting, which
may make it possible to construct malicious
content that will cause a stack overflow.

Interpreters of the DER structures should be aware of these issues and should take appropriate measures to guard against buffer overflows and stack overruns in particular and malicious content in general.

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has change control over this registration.

3.4. The +fastinfofet Structured Syntax Suffix

The ITU defined the Fast Infofet document format as a binary representation of the XML Information Set in [ITU.X891.2005]. These documents further define the "application/fastinfofet" media type. The suffix "+fastinfofet" MAY be used with any media type whose representation follows that established for "application/fastinfofet". The Message Type Structured Syntax Suffix registration form follows:

Name: Fast Infofet document format

+suffix: +fastinfofet

References: [ITU.X891.2005]

Encoding considerations: Fast Infofet is a binary encoding. The binary, quoted-printable and base64 content-transfer-encodings are suitable for use with Fast Infofet.

Fragment identifier considerations:

The syntax and semantics of fragment identifiers specified for +fastinfofet SHOULD be as specified for "application/fastinfofet". (At publication of this document, there is no fragment identification syntax defined for "application/fastinfofet".)

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+fastinfofet" SHOULD be processed as follows:

For cases defined in +fastinfofet, where the fragment identifier resolves per the +fastinfofet rules, then as specified in +fastinfofet.

For cases defined in +fastinfofet, where the fragment identifier does not resolve per the +fastinfofet rules, then as specified in "xxx/yyy+fastinfofet".

For cases not defined in +fastinfofet, then as specified in "xxx/yyy+fastinfofet".

Interoperability considerations: n/a

Security considerations: There are no security considerations inherent in Fast Infofet. Each individual media type registered with a +fastinfofet suffix can have additional security considerations.

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has change control over this registration.

3.5. The +wbxml Structured Syntax Suffix

The WAP Forum has defined the WAP Binary XML (WBXML) document format as a binary representation of XML in [WBXML]. This document further defines the "application/vnd.wap.wbxml" media type. The suffix "+wbxml" MAY be used with any media type whose representation follows that established for "application/vnd.wap.wbxml". The Message Type Structured Syntax Suffix registration form follows:

Name: WAP Binary XML (WBXML) document format

+suffix: +wbxml

References: [WBXML]

Encoding considerations: WBXML is a binary encoding.

Fragment identifier considerations:

The syntax and semantics of fragment identifiers specified for +wbxml SHOULD be as specified for "application/vnd.wap.wbxml". (At publication of this document, there is no

fragment identification syntax defined for
"application/vnd.wap.wbxml".)

The syntax and semantics for fragment
identifiers for a specific "xxx/yyy+wbxml"
SHOULD be processed as follows:

For cases defined in +wbxml, where the
fragment identifier resolves per the +wbxml
rules, then as specified in +wbxml.

For cases defined in +wbxml, where the
fragment identifier does not resolve per
the +wbxml rules, then as specified in
"xxx/yyy+wbxml".

For cases not defined in +wbxml, then as
specified in "xxx/yyy+wbxml".

Interoperability considerations: n/a

Security considerations: There are no security considerations
inherent in WBXML. Each individual media type
registered with a +wbxml suffix can have
additional security considerations.

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has
change control over this registration.

3.6. The +zip Structured Syntax Suffix

The ZIP format is a public domain, cross-platform, interoperable file
storage and transfer format, originally defined by PKWARE, Inc.; it
supports compression and encryption and is used as the underlying
representation by a variety of file formats. The media type
"application/zip" has been registered for such files. The suffix
"+zip" MAY be used with any media type whose representation follows
that established for "application/zip". The Message Type Structured
Syntax Suffix registration form follows:

Name: ZIP file storage and transfer format

+suffix: +zip

References: [ZIP]

Encoding considerations: ZIP is a binary encoding.

Fragment identifier considerations:

The syntax and semantics of fragment identifiers specified for +zip SHOULD be as specified for "application/zip". (At publication of this document, there is no fragment identification syntax defined for "application/zip".)

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+zip" SHOULD be processed as follows:

For cases defined in +zip, where the fragment identifier resolves per the +zip rules, then as specified in +zip.

For cases defined in +zip, where the fragment identifier does not resolve per the +zip rules, then as specified in "xxx/yyy+zip".

For cases not defined in +zip, then as specified in "xxx/yyy+zip".

Interoperability considerations: n/a

Security considerations: ZIP files support two forms of encryption: Strong Encryption and AES 128-bit, 192-bit and 256-bit encryption; see the specification for further details. Each individual media type registered with a +zip suffix can have additional security considerations.

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has change control over this registration.

4. IANA Considerations

See the Message Type Structured Syntax Suffix registration forms in Section 3.1 - Section 3.6.

The following Structured Syntax Suffix registration for "+xml" shall be used to reflect the information found in [RFC3023], with the addition of fragment identifier considerations:

Name: Extensible Markup Language (XML)

+suffix: +xml

References: [RFC3023]

Encoding considerations: Per [RFC3023], XML is allowed to be represented using both 7-bit and 8-bit encodings. When XML is written in UTF-8, XML is 8bit compatible ([RFC2045]). When XML is written in UTF-16 or UTF-32, XML is binary ([RFC2045]).

Fragment identifier considerations:

The syntax and semantics of fragment identifiers specified for +xml SHOULD be as specified for "application/xml". (At publication of this document, the fragment identification syntax considerations for "application/xml" are defined in [RFC3023], sections 5 and 7.)

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+xml" SHOULD be processed as follows:

For cases defined in +xml, where the fragment identifier resolves per the +xml rules, then as specified in +xml.

For cases defined in +xml, where the fragment identifier does not resolve per the +xml rules, then as specified in "xxx/yyy+xml".

For cases not defined in +xml, then as specified in "xxx/yyy+xml".

Interoperability considerations: See [RFC3023].

Security considerations: See [RFC3023]

Contact: Apps Area Working Group (apps-discuss@ietf.org)

Author/Change controller: The Apps Area Working Group. IESG has change control over this registration.

5. Security Considerations

See the Security considerations sections found in the Message Type Structured Syntax Suffix registration forms from Section 3.1 - Section 3.5.

When updating a +<suffix> registration, care should be taken to review all previously-registered xxx/yyy+<suffix> media types as to whether they might be affected by the updated +<suffix> registration. Because the generic fragment identifier processing rules take precedence over media-type-specific rules, introducing new or changing existing definitions may break the existing registrations of specific media types, as well as particular implementations of applications that process affected media types. Such changes can introduce interoperability and security issues.

When updating the fragment identifier processing rules for a specific xxx/yyy+<suffix> media type, care should be taken to review the generic fragment identifier processing rules for the +<suffix> registration and not introduce any conflicts. Because the generic fragment identifier processing rules take precedence over media-type-specific rules, such conflicting processing requirements should be ignored by an implementation, but such conflicts can introduce interoperability and security issues.

Note that [FRAGID-BP] provides additional advice to designers of fragment identifier rules for media type suffixes and specific media types.

6. References

6.1. Normative References

- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [ITU.X690.2008] International Telecommunications Union, "Recommendation ITU-T X.690 | ISO/IEC 8825-1 (2008), ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules

(DER)", ITU-T Recommendation X.690, November 2008.

[ITU.X891.2005]

International Telecommunications Union, "Recommendation ITU-T X.891 | ISO/IEC 24824-1 (2007), Generic applications of ASN.1: Fast infoset", ITU-T Recommendation X.891, May 2005.

[WBXML]

Open Mobile Alliance, "Binary XML Content Format Specification", OMA Wireless Access Protocol WAP-192-WBXML-20010725-a, July 2001.

[ZIP]

PKWARE, Inc., "APPNOTE.TXT - .ZIP File Format Specification", PKWARE .ZIP File Format Specification - Version 6.3.2, September 2007.

[RFC2045]

Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2119]

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

[RFC3023]

Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

6.2. Informative References

[I-D.ietf-appsawg-media-type-regs]

Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", draft-ietf-appsawg-media-type-regs-14 (work in progress), June 2012.

[FRAGID-BP]

Tennison, J., "Best Practices for Fragment Identifiers and Media Type Definitions", July 2012,
<<http://www.w3.org/TR/fragid-best-practices/>>.

Appendix A. Change History

This section is to be removed before publication.

draft-ietf-appsawg-media-type-suffix-regs-07 Added information based on IANA and GEN-ART reviews.

- draft-ietf-appsawg-media-type-suffix-regs-06 Clarified why this document updates RFC 3023.
- draft-ietf-appsawg-media-type-suffix-regs-05 Added an Informative reference to <http://www.w3.org/TR/fragid-best-practices/>. Minor editorial changes.
- draft-ietf-appsawg-media-type-suffix-regs-03 Added generic fragment identifier rules to +ber/+der to make them consistent with other registrations. Added some warning about how adding/changing fragment identifier rules for a +suffix can affect fragment identifier processing rules for previously registered xxx/yyy+suffix media types.
- draft-ietf-appsawg-media-type-suffix-regs-02 Added BER/DER security considerations. Reworked fragment identifier wording some more.
- draft-ietf-appsawg-media-type-suffix-regs-01 Reordered the sections. Cleaned up some MUSTard. Fixed some references. Added encoding considerations. Reworked fragment identifier wording.
- draft-ietf-appsawg-media-type-suffix-regs-00 Added the fragment identifier consideration sections. Added a note about +xml fragment identifier considerations.
- draft-hansen-media-type-suffix-regs-02 Added +zip. Fixed up the ISO document references. Minor changes.
- draft-hansen-media-type-suffix-regs-01 Added +ber. Minor changes.

Authors' Addresses

Tony Hansen
AT&T Laboratories
200 Laurel Ave. South
Middletown, NJ 07748
USA

Email: tony+sss@maillennium.att.com

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

Applications Area Working Group
Internet-Draft
Updates: 2046 (if approved)
Intended status: Standards Track
Expires: November 10, 2012

A. Melnikov
Isode Limited
J. Reschke
greenbytes
May 9, 2012

Update to MIME regarding Charset Parameter Handling in
Textual Media Types
draft-ietf-appsawg-mime-default-charset-04

Abstract

This document changes RFC 2046 rules regarding default charset parameter values for text/* media types to better align with common usage by existing clients and servers.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft should take place on the Apps Area Working Group mailing list (apps-discuss@ietf.org), which is archived at <http://www.ietf.org/mail-archive/web/apps-discuss>.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on November 10, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction and Overview	3
2.	Conventions Used in This Document	3
3.	New rules for default charset parameter values for text/* media types	3
4.	Default charset parameter value for text/plain media type	4
5.	Security Considerations	4
6.	IANA Considerations	5
7.	References	5
7.1.	Normative References	5
7.2.	Informative References	5
Appendix A.	Acknowledgements	5
	Authors' Addresses	6

1. Introduction and Overview

RFC 2046 specified that the default charset parameter (i.e. the value used when the parameter is not specified) is "US-ASCII" (Section 4.1.2 of [RFC2046]). RFC 2616 changed the default for use by HTTP (Hypertext Transfer Protocol) to be "ISO-8859-1" (Section 3.7.1 of [RFC2616]). This encoding is not very common for new text/* media types and a special rule in the HTTP specification adds confusion about which specification ([RFC2046] or [RFC2616]) is authoritative in regards to the default charset for text/* media types.

Many complex text subtypes such as text/html [RFC2854] and text/xml [RFC3023] have internal (to their format) means of describing the charset. Many existing User Agents ignore the default of "US-ASCII" rule for at least text/html and text/xml.

This document changes RFC 2046 rules regarding default charset parameter values for text/* media types to better align with common usage by existing clients and servers. It does not change the defaults for any currently registered media type.

2. Conventions Used in This Document

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].

3. New rules for default charset parameter values for text/* media types

Section 4.1.2 of [RFC2046] says:

The default character set, which must be assumed in the absence of a charset parameter, is US-ASCII.

As explained in the Introduction section this rule is considered to be outdated, so this document replaces it with the following set of rules:

Each subtype of the "text" media type which uses the "charset" parameter can define its own default value for the "charset" parameter, including the absence of any default.

In order to improve interoperability with deployed agents, "text/*" media type registrations SHOULD either

- a. specify that the "charset" parameter is not used for the defined subtype, because the charset information is transported inside the payload (such as in "text/xml"), or
- b. require explicit unconditional inclusion of the "charset" parameter eliminating the need for a default value.

In accordance with option (a), above, registrations for "text/*" media types that can transport charset information inside the corresponding payloads (such as "text/html" and "text/xml") SHOULD NOT specify the use of a "charset" parameter, nor any default value, in order to avoid conflicting interpretations should the charset parameter value and the value specified in the payload disagree.

New subtypes of the "text" media type, thus, SHOULD NOT define a default "charset" value. If there is a strong reason to do so despite this advice, they SHOULD use the "UTF-8" [RFC3629] charset as the default.

Regardless of what approach is chosen, all new text/* registrations MUST clearly specify how the charset is determined; relying on the default defined in Section 4.1.2 of [RFC2046] is no longer permitted. However, existing text/* registrations that fail to specify how the charset is determined still default to US-ASCII.

Specifications covering the "charset" parameter, and what default value, if any, is used, are subtype-specific, NOT protocol-specific. Protocols that use MIME, therefore, MUST NOT override default charset values for "text/*" media types to be different for their specific protocol. The protocol definitions MUST leave that to the subtype definitions.

4. Default charset parameter value for text/plain media type

The default charset parameter value for text/plain is unchanged from [RFC2046] and remains as "US-ASCII".

5. Security Considerations

Guessing of the charset parameter can lead to security issues such as content buffer overflows, denial of services or bypass of filtering mechanisms. However, this document does not promote guessing, but encourages use of charset information that is specified by the sender.

Conflicting information in-band vs out-of-band can also lead to similar security problems, and this document recommends the use of

charset information which is more likely to be correct (for example, in-band over out-of-band).

6. IANA Considerations

This document asks IANA to update the "text" subregistry of the Media Types registry (<<http://www.iana.org/assignments/media-types/text/>>), to add the following preamble: "See [this RFC] for information about 'charset' parameter handling for text media types."

IANA is also asked to add this RFC to the list of references at the beginning of the Application for Media Type (<<http://www.iana.org/cgi-bin/mediatypes.pl>>).

7. References

7.1. Normative References

- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

7.2. Informative References

- [RFC2616] 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.
- [RFC2854] Connolly, D. and L. Masinter, "The 'text/html' Media Type", RFC 2854, June 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

Appendix A. Acknowledgements

Many thanks to Ned Freed and John Klensin for comments and ideas that motivated creation of this document, and to Carsten Bormann, Murray S. Kucherawy, Barry Leiba, and Henri Sivonen for feedback and text

suggestions.

Authors' Addresses

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

Julian F. Reschke
greenbytes GmbH
Hafenweg 16
Muenster, NW 48155
Germany

Email: julian.reschke@greenbytes.de
URI: <http://greenbytes.de/tech/webdav/>

Individual submission
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2012

D. Crocker
Brandenburg InternetWorking
M. Kucherawy
Cloudmark, Inc.
June 27, 2012

Indicating Email Handling States in Trace Fields
draft-ietf-appsawg-received-state-04

Abstract

This document registers a trace field clause for use in indicating transitions between handling queues or processing states, including enacting inter- and intra-host message transitions. This might include message quarantining, mailing list moderation, timed delivery, queueing for further analysis, content conversion, or other similar causes, as well as optionally identifying normal handling queues.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on December 29, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Key Words	3
3. Trace Clause	3
4. Discussion	6
5. Granularity	7
6. IANA Considerations	7
6.1. Mail Parameters Additional-registered-clauses Sub-Registry	8
6.2. Mail Parameters Registered-states Sub-Registry	8
7. Security Considerations	9
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Appendix A. Trace Field Examples	10
A.1. Typical Delivery Without Obvious Extra Handling	11
A.2. Delivery With Moderation	11
Appendix B. Acknowledgements	12

1. Introduction

[SMTP] defines the content of email message trace fields, commonly the "Received" header field. These are typically used to record an audit trail of the path a message follows from origin to destination, with one such field added each time a message moves from one host to the next.

Section 3.7.2 of that document mentions that "the most important use of of Received: lines is for debugging mail faults [...]".

There are some cases where there may be large time gaps between trace fields. Though this might be caused by transient communication issues, they might also be caused by policy decisions or special processing regarding the content of the message, authorization of some identity on the message, or transitions between major software components. Common examples include message quarantines (filters that cause a message to be held pending further evaluation, or delivery of a message pending manual operator action), pending content analysis, or mailing list servers that impose moderation rules (mailing list owner action required regarding mail from authors not subscribed to those lists).

This document registers a new optional clause that can be used in trace fields to indicate that a message entered such a special processing queue or state for some period. This allows inspection of the trace information to reveal that the cause for a time gap in trace fields was imposed by additional processing rather than one caused by transient technical difficulties.

2. Key Words

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 [KEYWORDS].

3. Trace Clause

This specification defines a clause, called "state", which MAY be used when creating a Received header field (see Section 4.4 of [SMTP]) to indicate the nature of additional handling imposed on the relaying of a message toward its recipient(s). It is followed by a single keyword that provides that detail. A Mail Transfer Agent (MTA) or other handling agent that determines a message has entered a state other than normal queueing of messages for relaying or delivery MAY generate a trace field including one of these clauses. That is, the presence of this clause on a trace field is an indication of the entry of the message into that state; a later trace field added would

indicate its departure from that state.

An MTA implementing this specification SHOULD add a Received field as described whenever:

- a. It determines that a special handling condition will occur, and places it into that condition; or
- b. It determines that no special handling is required, and prepares it for relay to the next handling agent.

An MTA need not add a Received field indicating preparation for normal handoff to the next handling agent if it has already added a Received field for some other reason. Trace data added by the next handling agent will imply the message's exit from the special handling condition.

If a single MTA processes a message through multiple special handling conditions, it MAY add a Received for each distinct condition.

For example: Presume a message will be injected into MTA-1, then travel to MTA-3 via MTA-2, and then MTA-3 enacts final delivery. At MTA-2, it is determined that some action will be taken that will cause the message to undergo some handling change that is outside of typical message flow. In this case:

1. MTA-1 adds a typical Received field and relays it to MTA-2
2. MTA-2 determines that the atypical handling will occur and adds a Received field using the extension specified here
3. On completion of the atypical handling, MTA-2 relays the message to MTA-3
4. MTA-3 adds a typical Received field and enacts final delivery of the message

Appropriate use of this mechanism does not include associating meta-data with the message, such as categorizing the message (e.g., the notions of "is spam" or "was 8-bit, converted to 7-bit"). Processing agents also cannot reliably use this mechanism to determine anything about the message content, since there is no guarantee that all agents in the chain of handling made such annotations allowing correct conclusions. The sole purpose here is to allow one to determine the point(s) in the chain of custody of a message at which the message was subjected to handling outside of normal message routing and queueing.

The following state keywords are defined in this document; extensions may define other registered keywords (see Section 6.2):

auth: The message entered a queue pending authentication of some identifier in the message.

content: The message entered a queue pending content analysis, such as scanning for spam or viruses.

convert: The message entered a queue pending content conversion.

moderation: The message entered a hold pending mailing list moderator action.

normal: The message is not in an administrative hold and is queued for or is being handed off to the next handling agent (which may be local delivery). This is the default interpretation when no "state" clause is present.

other: The message entered a hold or queue for reasons not covered by other keywords in this list, and not for transient technology issues.

outbound: The message entered a queue for outbound relaying. This is typically the last case added for a single host, and the next Received header field is expected to be added by some other host.

quarantine: The message entered a hold in an isolation queue pending operator action for local policy reasons.

timed: The message entered a hold in order to meet a requested delivery window, such as is defined in [FUTURERELEASE].

The "state" clause is added in Section 6 to the Additional-Registered-Clauses IANA sub-registry. The ABNF for this clause is:

```
State = CFWS "state" FWS queue-state-keyword [ "/" value ]

queue-state-keyword = ( reg-state-keyword / unreg-state-keyword )

reg-state-keyword = ( "auth" / "content" / "convert" /
                      "moderation" / "normal" / "other" /
                      "outbound" / "quarantine" / "timed" /
                      additional-state-keyword )

additional-state-keyword = token
                          ; MUST be registered; see
                          ; "IANA Considerations" below

value = token

unreg-state-keyword = token
```

"FWS" and "CFWS" are defined in [MAIL]. "token" is defined in [MIME].

A transfer agent making use of this extension MAY also include header field comments to provide additional information.

The "value" is available for providing additional labels as explanation for the state transition. Examples could include:

- o convert/unicode2ascii
- o moderation/not-subscribed
- o quarantine/spam

4. Discussion

Handling agents are not expected to implement or support all of these. Indeed, recording trace information for all of the states described above could make the header of a message inordinately large. Rather, an agent is encouraged to apply state annotations when a message enters a handling queue where a significant condition occurs or where significant additional processing or delay is possible, and especially when a handoff has occurred between two different, independent agents.

For example, an MTA receiving a message, doing message authentication, scanning for viruses and spam, and then putting it in an outbound queue could add four Received header fields denoting each of these states. However, where they are all done as part of a single system process, in a single pass, doing so would be considered unusual (and extremely verbose). This method SHOULD NOT be applied

except when doing detailed analysis of a single component to identify performance issues with those steps.

Rather, an agent that wishes to make a state annotation SHOULD add only a single Received header field including such annotation, thus indicating (a) the time of completion of its handling of the message via the date portion of the field, and (b) the final disposition of that message relative to that agent. For example, an MTA receiving a message that performs various checks on the message before immediately handing it off to a Mailing List Manager (MLM) would only record a "normal" state, assuming it passes those checks. The MLM would then evaluate the message and record its own state once it decides what the next step will be for the handling of that message.

5. Granularity

The degree of granularity -- and therefore the degree of verbosity -- recorded through the use of this additional trace clause is likely to vary depending on circumstances. It will typically be the case that use of this clause will be limited to "unusual" transitions, such as when a message requires additional scrutiny or other processing, or needs to be quarantined.

Somewhat greater granularity might also include transitions of administrative responsibility, such as between an Mail Transfer Agent (MTA) operator and a Mailing List Manager (MLM) operator. This could be further enhanced to note some transitions that are interesting only when other transitions have occurred, such as noting entry to the outbound queue only when the message is originating from an "interesting" source, like an MLM, since an MLM can introduce significant changes to the message or delivery delay and it could be useful to know when it completed its processing, as distinct from the subsequent processing by the originating MTA. In circumstances needing very fine-grained trace information, fields might be created to note all of these "significant" network architecture transitions.

One should note, however, when choosing higher levels of granularity, that the Received header fields present on a message could be counted by MTAs when trying to decide whether or not a message routing loop is in effect. A message with an abundance of these might cause an incorrect determination that the message is in a delivery loop, causing it to be removed from the mail stream. See Section 6.3 of [SMTP] for further discussion.

6. IANA Considerations

6.1. Mail Parameters Additional-registered-clauses Sub-Registry

This document adds to the "Additional-registered-clauses" sub-registry of the "Mail Parameters" registry, created by [SMTP], the following entry:

Clause name: state

Description: Indicates entry into a special queue state

Syntax Summary: state <state-name>

Reference: [this document]

6.2. Mail Parameters Registered-states Sub-Registry

The "Mail Parameters" registry at IANA is updated by the creation of the "Registered-states" sub-registry to contain valid state keywords for use with this specification. Updates to this registry are governed by the First Come First Served rules of [IANA] for new registrations. Changes to the status of existing entries are limited to the original registrant or IESG approval.

Discussion of all registry updates is encouraged via one or more IETF mailing lists that typically cover email-related subjects prior to approval of the change, as a way of documenting the work. The `ietf-smtp@ietf.org` list is suggested.

Note that only registrations of queue state keywords are permitted. The registry is not to be used for specifying secondary information (i.e., the "value" part of the ABNF in Section 3).

Registrations are to include the following entries:

Name: The name of the state keyword being defined or updated, which conforms to the ABNF shown in Section 3.

Description: A brief description of the keyword's meaning.

Specification: The specification document that defines the queue state being registered, or if no stable reference exists, a more detailed explanation of the queue state than is in the "Description", sufficient to allow interoperability.

Use: One of "current" (the state keyword is in current use), "deprecated" (the state keyword is in use but not recommended for new implementations), or "historic" (the state keyword is no longer in substantial current use).

The initial registration set is as follows:

Name	Description	Specification	Use
auth	Held for message authentication	[this document]	current
content	Held for message content analysis	[this document]	current
convert	Held for message content conversion	[this document]	current
moderation	Held for list moderation	[this document]	current
normal	Message is not being held other than to accommodate typical relaying handling	[this document]	current
other	Held for causes not covered by other registered state keywords	[this document]	current
outbound	Message placed in outbound queue	[this document]	current
quarantine	Held for operator action due to content analysis or local policy	[this document]	current
timed	Held to accommodate a specific requested delivery window	[this document]	current

7. Security Considerations

The use of this trace information can reveal hints as to local policy that was in effect at the time of message handling.

Further discussion about trace field security can be found in Section 7.6 of [SMTP].

8. References

8.1. Normative References

- [IANA] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [MAIL] Resnick, P., "Internet Message Format", RFC 5322, October 2008.
- [MIME] Freed, N. and N. Borenstein, "Simple Mail Transfer Protocol", RFC 2045, November 1996.
- [SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

8.2. Informative References

- [FUTURERELEASE] White, G. and G. Vaudreuil, "SMTP Submission Service Extension for Future Message Release", RFC 4865, May 2007.

Appendix A. Trace Field Examples

This section includes a sample of the new trace field clause in use.

A.1. Typical Delivery Without Obvious Extra Handling

Typical message delivery

```
Received: from newyork.example.com
        (newyork.example.com [192.0.2.250])
        by mail-router.example.net (8.11.6/8.11.6)
        with ESMTTP id i7PK0sH7021929
        for <recipient@example.net>;
        Fri, Feb 15 2002 17:19:22 -0800
Received: from internal.example.com
        (internal.example.com [192.168.0.1])
        by newyork.example.com (8.11.6/8.11.6)
        with ESMTTP id i9MKZCRd064134
        for <recipient@example.net>;
        Fri, Feb 15 2002 17:19:08 -0800
```

Example 1: Typical message delivery with no appreciable extra handling; only Received header fields shown

A.2. Delivery With Moderation

Message delivery after moderation

```
Received: from newyork.example.com
        (newyork.example.com [192.0.2.250])
        by mail-router.example.net (8.11.6/8.11.6)
        with ESMTTP id i7PK0sH7021929
        for <recipient@example.net>;
        Fri, Feb 15 2002 18:33:29 -0800
Received: from internal.example.com
        (internal.example.com [192.168.0.1])
        by newyork.example.com (8.11.6/8.11.6)
        with ESMTTP id i9MKZCRd064134
        for <secret-list@example.com>
        state moderation (sender not subscribed);
        Fri, Feb 15 2002 17:19:08 -0800
```

Example 2: Message held for moderation; only Received header fields shown

The message passed from internal.example.com to newyork.example.com intended for a mailing list hosted at the latter. For list administrative reasons, the message is held there for moderation. It is finally released over an hour later and passed to the next host. A comment after the state expression indicates the actual cause for the administrative hold.

Appendix B. Acknowledgements

The authors wish to acknowledge the following for their reviews and constructive criticisms of this proposal: Tony Finch, Ned Freed, Carl S. Gutenkunst, John Levine, Bill McQuillan, S. Moonesamy, Alexey Melnikov, Robert A. Rosenberg, Hector Santos, Rolf Sonneveld, and Mykyta Yevstifeyev.

Authors' Addresses

D. Crocker
Brandenburg InternetWorking
675 Spruce Dr.
Sunnyvale 94086
USA

Phone: +1.408.246.8253
EMail: dcrocker@bbiw.net
URI: <http://bbiw.net>

Murray S. Kucherawy
Cloudmark, Inc.
128 King St., 2nd Floor
San Francisco, CA 94107
US

EMail: superuser@gmail.com

Network Working Group
Internet Draft
Intended status: Standards Track
Expires: February 26, 2014

Paul E. Jones
Gonzalo Salgueiro
Cisco Systems
Michael B. Jones
Microsoft
Joseph Smarr
Google
August 26, 2013

WebFinger
draft-ietf-appsawg-webfinger-18.txt

Abstract

This specification defines the WebFinger protocol, which can be used to discover information about people or other entities on the Internet using standard HTTP methods. WebFinger discovers information for a URI that might not be usable as a locator otherwise, such as account or email URIs.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on February 26, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction.....	2
2. Terminology.....	3
3. Example Uses of WebFinger.....	4
3.1. Identity Provider Discovery for OpenID Connect.....	4
3.2. Getting Author and Copyright Information for a Web Page...	5
4. WebFinger Protocol.....	6
4.1. Constructing the Query Component of the Request URI.....	7
4.2. Performing a WebFinger Query.....	7
4.3. The "rel" Parameter.....	8
4.4. The JSON Resource Descriptor (JRD).....	10
4.4.1. subject.....	10
4.4.2. aliases.....	10
4.4.3. properties.....	10
4.4.4. links.....	11
4.5. WebFinger and URIs.....	13
5. Cross-Origin Resource Sharing (CORS).....	13
6. Access Control.....	13
7. Hosted WebFinger Services.....	14
8. Definition of WebFinger Applications.....	15
8.1. Specification of the URI Scheme and URI.....	15
8.2. Host Resolution.....	15
8.3. Specification of Properties.....	16
8.4. Specification of Links.....	16
8.5. One URI, Multiple Applications.....	16
8.6. Registration of Link Relation Types and Properties.....	17
9. Security Considerations.....	17
9.1. Transport-Related Issues.....	17
9.2. User Privacy Considerations.....	17
9.3. Abuse Potential.....	18
9.4. Information Reliability.....	19
10. IANA Considerations.....	20
10.1. Well-Known URI.....	20
10.2. JSON Resource Descriptor (JRD) Media Type.....	20
10.3. Registering Link Relation Types.....	21
10.4. Establishment of the WebFinger Properties Registry.....	22
10.4.1. The Registration Template.....	22
10.4.2. The Registration Procedures.....	22
11. Acknowledgments.....	23
12. References.....	23
12.1. Normative References.....	23
12.2. Informative References.....	24
Author's Addresses.....	25

1. Introduction

WebFinger is used to discover information about people or other entities on the Internet that are identified by a URI [6] using standard Hypertext Transfer Protocol (HTTP) [2] methods over a secure transport [12]. A WebFinger resource returns a JavaScript Object

Notation (JSON) [5] object describing the entity that is queried. The JSON object is referred to as the JSON Resource Descriptor (JRD).

For a person, the kinds of information that might be discoverable via WebFinger include a personal profile address, identity service, telephone number, or preferred avatar. For other entities on the Internet, a WebFinger resource might return JRDs containing link relations [8] that enable a client to discover, for example, that a printer can print in color on A4 paper, the physical location of a server, or other static information.

Information returned via WebFinger might be for direct human consumption (e.g., looking up someone's phone number), or it might be used by systems to help carry out some operation (e.g., facilitate, with additional security mechanisms, logging into a web site by determining a user's identity service). The information is intended to be static in nature and, as such, WebFinger is not intended to be used to return dynamic information like the temperature of a CPU or the current toner level in a laser printer.

The WebFinger protocol is designed to be used across many applications. Applications that wish to utilize WebFinger will need to specify properties, titles, and link relation types that are appropriate for the application. Further, applications will need to define the appropriate URI scheme to utilize for the query target.

Use of WebFinger is illustrated in the examples in Section 3 and described more formally in Section 4. Section 8 describes how applications of WebFinger may be defined.

2. Terminology

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 RFC 2119 [1].

WebFinger makes heavy use of "Link Relations". A Link Relation is an attribute-and-value pair in which the attribute identifies the type of relationship between the linked entity or resource and the information specified in the value. In Web Linking [4], the link relation is represented using an HTTP entity-header of "Link", where the "rel" attribute specifies the type of relationship and the "href" attribute specifies the information that is linked to the entity or resource. In WebFinger, the same concept is represented using a JSON array of "links" objects, where each member named "rel" specifies the type of relationship and each member named "href" specifies the information that is linked to the entity or resource. Note that WebFinger narrows the scope of a link relation beyond what is defined for Web Linking by stipulating that the value of the "rel" member needs to be either a single IANA-registered link relation type [8] or a URI [6].

The use of URIs throughout this document refers to URIs following the syntax specified in Section 3 of RFC 3986 [6]. Relative URIs, having syntax following that of Section 4.2 or RFC 3986, are not used with WebFinger.

3. Example Uses of WebFinger

This section shows a few sample uses of WebFinger. Any application of WebFinger would be specified outside of this document, as described in Section 8. The examples in this section should be simple enough to understand without having seen the formal specifications of the applications.

3.1. Identity Provider Discovery for OpenID Connect

Suppose Carol wishes to authenticate with a web site she visits using OpenID Connect [15]. She would provide the web site with her OpenID Connect identifier, say carol@example.com. The visited web site would perform a WebFinger query looking for the OpenID Connect Provider. Since the site is interested in only one particular link relation, the WebFinger resource might utilize the "rel" parameter as described in Section 4.3:

```
GET /.well-known/webfinger?
    resource=acct%3Acarol%40example.com&
    rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
HTTP/1.1
Host: example.com
```

The server might respond like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/jrd+json

{
  "subject" : "acct:carol@example.com",
  "links" :
  [
    {
      "rel" : "http://openid.net/specs/connect/1.0/issuer",
      "href" : "https://openid.example.com"
    }
  ]
}
```

Since the "rel" parameter only serves to filter the link relations returned by the resource, other name/value pairs in the response, including any aliases or properties, would be returned. Also, since support for the "rel" parameter is not guaranteed, the client must not assume the "links" array will contain only the requested link relation.

3.2. Getting Author and Copyright Information for a Web Page

Suppose an application is defined to retrieve metadata information about a web page URL, such as author and copyright information. To retrieve that information, the client can utilize WebFinger to issue a query for the specific URL. Suppose the URL of interest is `http://blog.example.com/article/id/314`. The client would issue a query similar to the following:

```
GET /.well-known/webfinger?
    resource=http%3A%2F%2Fblog.example.com%2Farticle%2Fid%2F314
HTTP/1.1
Host: blog.example.com
```

The server might then reply in this way:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/jrd+json

{
  "subject" : "http://blog.example.com/article/id/314",
  "aliases" :
  [
    "http://blog.example.com/cool_new_thing",
    "http://blog.example.com/steve/article/7"
  ],
  "properties" :
  {
    "http://blgx.example.net/ns/version" : "1.3",
    "http://blgx.example.net/ns/ext" : null
  },
  "links" :
  [
    {
      "rel" : "copyright",
      "href" : "http://www.example.com/copyright"
    },
    {
      "rel" : "author",
      "href" : "http://blog.example.com/author/steve",
      "titles" :
      {
        "en-us" : "The Magical World of Steve",
        "fr" : "Le Monde Magique de Steve"
      },
      "properties" :
      {
        "http://example.com/role" : "editor"
      }
    }
  ]
}
```

```
    ]  
  }
```

In the above example, we see that the server returned a list of aliases, properties, and links related to the subject URL. The links contain references to information for each link relation type. For the author link, the server provided a reference to the author's blog, along with a title for the blog in two languages. The server also returned a single property related to the author, indicating the author's role as editor of the blog.

It is worth noting that, while the server returned just two links in the links array in this example, a server might return any number of links when queried.

4. WebFinger Protocol

The WebFinger protocol is used to request information about an entity identified by a query target (a URI). The client can optionally specify one or more link relation types for which it would like to receive information.

A WebFinger request is an HTTPS request to a WebFinger resource. A WebFinger resource is a well-known URI [3] using the HTTPS scheme, constructed along with the required query target and optional link relation types. WebFinger resources MUST NOT be served with any other URI scheme (such as HTTP).

A WebFinger resource is always given a query target, which is another URI that identifies the entity whose information is sought. GET requests to a WebFinger resource convey the query target in the "resource" parameter in the WebFinger URI's query string; see Section 4.1 for details.

The host to which a WebFinger query is issued is significant. If the query target contains a "host" portion (Section 3.2.2 of RFC 3986), then the host to which the WebFinger query is issued SHOULD be the same as the "host" portion of the query target, unless the client receives instructions through some out-of-band mechanism to send the query to another host. If the query target does not contain a "host" portion, then the client chooses a host to which it directs the query using additional information it has.

The path component of a WebFinger URI MUST be the well-known path `"/.well-known/webfinger"`. A WebFinger URI MUST contain a query component that encodes the query target and optional link relation types as specified in Section 4.1.

The WebFinger resource returns a JSON Resource Descriptor (JRD) as the resource representation to convey information about an entity on the Internet. Also, the Cross-Origin Resource Sharing (CORS) [7]

specification is utilized to facilitate queries made via a web browser.

4.1. Constructing the Query Component of the Request URI

A WebFinger URI MUST contain a query component (see Section 3.4 of RFC 3986). The query component MUST contain a "resource" parameter and MAY contain one or more "rel" parameters. The "resource" parameter MUST contain the query target (URI) and the "rel" parameters MUST contain encoded link relation types according to the encoding described in this section.

To construct the query component, the client performs the following steps. First, each parameter value is percent-encoded, as per Section 2.1 of RFC 3986. The encoding is done to conform to the query production in Section 3.4 of that specification, with the addition that any instances of the "=" and "&" characters within the parameter values are also percent-encoded. Next, the client constructs a string to be placed in the query component by concatenating the name of the first parameter together with an equal sign ("=") and the percent-encoded parameter value. For any subsequent parameters, the client appends an ampersand("&") to the string, the name of the next parameter, an equal sign, and the parameter value. The client MUST NOT insert any spaces while constructing the string. The order in which the client places each attribute-and-value pair within the query component does not matter in the interpretation of the query component.

4.2. Performing a WebFinger Query

A WebFinger client issues a query using the GET method to the well-known [3] resource identified by the URI whose path component is `"/.well-known/webfinger"` and whose query component MUST include the "resource" parameter exactly once and set to the value of the URI for which information is being sought.

If the "resource" parameter is absent or malformed, the WebFinger resource MUST indicate that the request is bad as per Section 10.4.1 of RFC 2616 [2].

If the "resource" parameter is a value for which the server has no information, the server MUST indicate that it was unable to match the request as per Section 10.4.5 of RFC 2616.

A client MUST query the WebFinger resource using HTTPS only. If the client determines that the resource has an invalid certificate, the resource returns a 4xx or 5xx status code, or the HTTPS connection cannot be established for any reason, then the client MUST accept that the WebFinger query has failed and MUST NOT attempt to reissue the WebFinger request using HTTP over a non-secure connection.

A WebFinger resource MUST return a JRD as the representation for the resource if the client requests no other supported format explicitly via the HTTP "Accept" header. The client MAY include the "Accept" header to indicate a desired representation; representations other than JRD might be defined in future specifications. The WebFinger resource MUST silently ignore any requested representations that it does not understand and support. The media type used for the JSON Resource Descriptor (JRD) is "application/jrd+json" (see Section 9.2).

The properties, titles, and link relation types returned by the server in a JRD might be varied and numerous. For example, the server might return information about a person's blog, vCard [14], avatar, OpenID Connect provider, RSS or ATOM feed, and so forth in a reply. Likewise, if a server has no information to provide it might return a JRD with an empty links array or no links array.

A WebFinger resource MAY redirect the client; if it does, the redirection MUST only be to an "https" URI and the client MUST perform certificate validation again when redirected.

A WebFinger resource can include cache validators in a response to enable conditional requests by the client and/or expiration times as per Section 13 of RFC 2616.

4.3. The "rel" Parameter

When issuing a request to a WebFinger resource, the client MAY utilize the "rel" parameter to request only a subset of the information that would otherwise be returned without the "rel" parameter. When the "rel" parameter is used and accepted, only the link relation types that match the link relation types provided via the "rel" parameter are included in the array of links returned in the JRD. If there are no matching link relation types defined for the resource, the "links" array in the JRD will either be absent or empty. All other information present in a resource descriptor remains present, even when "rel" is employed.

The "rel" parameter MAY be included multiple times in order to request multiple link relation types.

The purpose of the "rel" parameter is to return a subset of "link relation objects" (see Section 4.4.4) that would otherwise be returned in the resource descriptor. Use of the parameter might reduce processing requirements on either the client or server, and it might also reduce the bandwidth required to convey the partial resource descriptor, especially if there are numerous link relation values to convey for a given "resource" value. Note that if a client requests a particular link relation type for which the server has no information, the server MAY return a JRD with an empty links array or no links array.

WebFinger resources SHOULD support the "rel" parameter. If the resource does not support the "rel" parameter, it MUST ignore the parameter and process the request as if no "rel" parameter values were present.

The following example uses the "rel" parameter to request links for two link relation types:

```
GET /.well-known/webfinger?
    resource=acct%3Abob%40example.com&
    rel=http%3A%2F%2Fwebfinger.example%2Frel%2Fprofile-page&
    rel=http://webfinger.example/rel/businesscard HTTP/1.1
Host: example.com
```

In this example, the client requests the link relations of type "http://webfinger.example/rel/profile-page" and "http://webfinger.example/rel/businesscard". The server then responds with a message like this:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/jrd+json

{
  "subject" : "acct:bob@example.com",
  "aliases" :
  [
    "https://www.example.com/~bob/"
  ],
  "properties" :
  {
    "http://example.com/ns/role" : "employee"
  },
  "links" :
  [
    {
      "rel" : "http://webfinger.example/rel/profile-page",
      "href" : "https://www.example.com/~bob/"
    },
    {
      "rel" : "http://webfinger.example/rel/businesscard",
      "href" : "https://www.example.com/~bob/bob.vcf"
    }
  ]
}
```

As you can see in the response, the resource representation contains only the links of the types requested by the client and for which the server had information, but the other parts of the JRD are still present. Note also in the above example that the links returned in the links array all use HTTPS, which is important if the data indirectly obtained via WebFinger needs to be returned securely.

4.4. The JSON Resource Descriptor (JRD)

The JSON Resource Descriptor (JRD), originally introduced in RFC 6415 [16] and based on the Extensible Resource Descriptor (XRD) format [17], is a JSON object that comprises the following name/value pairs:

- o subject
- o aliases
- o properties
- o links

The member "subject" is a name/value pair whose value is a string, "aliases" is an array of strings, "properties" is an object comprising name/value pairs whose values are strings, and "links" is an array of objects that contain link relation information.

When processing a JRD, the client MUST ignore any unknown member and not treat the presence of an unknown member as an error.

Below, each of these members of the JRD is described in more detail.

4.4.1. subject

The value of the "subject" member is a URI that identifies the entity that the JRD describes.

The "subject" value returned by a WebFinger resource MAY differ from the value of the "resource" parameter used in the client's request. This might happen, for example, when the subject's identity changes (e.g., a user moves his or her account to another service) or when the resource prefers to express URIs in canonical form.

The "subject" member SHOULD be present in the JRD.

4.4.2. aliases

The "aliases" array is an array of zero or more URI strings that identify the same entity as the "subject" URI.

The "aliases" array is OPTIONAL in the JRD.

4.4.3. properties

The "properties" object comprises zero or more name/value pairs whose names are URIs (referred to as "property identifiers") and whose values are strings or null. Properties are used to convey additional information about the subject of the JRD. As an example, consider this use of "properties":

```
"properties" : { "http://webfinger.example/ns/name" : "Bob Smith" }
```

The "properties" member is OPTIONAL in the JRD.

4.4.4. links

The "links" array has any number of member objects, each of which represents a link [4]. Each of these link objects can have the following members:

- o rel
- o type
- o href
- o titles
- o properties

The "rel" and "href" members are strings representing the link's relation type and the target URI, respectively. The context of the link is the "subject" (see Section 4.4.1).

The "type" member is a string indicating what the media type of the result of dereferencing the link ought to be.

The order of elements in the "links" array MAY be interpreted as indicating an order of preference. Thus, if there are two or more link relations having the same "rel" value, the first link relation would indicate the user's preferred link.

The "links" array is OPTIONAL in the JRD.

Below, each of the members of the objects found in the "links" array is described in more detail. Each object in the "links" array, referred to as a "link relation object", is completely independent from any other object in the array; any requirement to include a given member in the link relation object refers only to that particular object.

4.4.4.1. rel

The value of the "rel" member is a string that is either a URI or a registered relation type [8] (see RFC 5988 [4]). The value of the "rel" member MUST contain exactly one URI or registered relation type. The URI or registered relation type identifies the type of the link relation.

The other members of the object have meaning only once the type of link relation is understood. In some instances, the link relation will have associated semantics enabling the client to query for other resources on the Internet. In other instances, the link relation will have associated semantics enabling the client to utilize the other members of the link relation object without fetching additional external resources.

URI link relation type values are compared using the "Simple String Comparison" algorithm of Section 6.2.1 of RFC 3986.

The "rel" member MUST be present in the link relation object.

4.4.4.2. type

The value of the "type" member is a string that indicates the media type [9] of the target resource (see RFC 6838 [10]).

The "type" member is OPTIONAL in the link relation object.

4.4.4.3. href

The value of the "href" member is a string that contains a URI pointing to the target resource.

The "href" member is OPTIONAL in the link relation object.

4.4.4.4. titles

The "titles" object comprises zero or more name/value pairs whose name is a language tag [11] or the string "und". The string is human-readable and describes the link relation. More than one title for the link relation MAY be provided for the benefit of users who utilize the link relation and, if used, a language identifier SHOULD be duly used as the name. If the language is unknown or unspecified, then the name is "und".

A JRD SHOULD NOT include more than one title identified with the same language tag (or "und") within the link relation object. Meaning is undefined if a link relation object includes more than one title named with the same language tag (or "und"), though this MUST NOT be treated as an error. A client MAY select whichever title or titles it wishes to utilize.

Here is an example of the titles object:

```
"titles" :
{
  "en-us" : "The Magical World of Steve",
  "fr" : "Le Monde Magique de Steve"
}
```

The "titles" member is OPTIONAL in the link relation object.

4.4.4.5. properties

The "properties" object within the link relation object comprises zero or more name/value pairs whose names are URIs (referred to as "property identifiers") and whose values are strings or null. Properties are used to convey additional information about the link relation. As an example, consider this use of "properties":

```
"properties" : { "http://webfinger.example/mail/port" : "993" }
```

The "properties" member is OPTIONAL in the link relation object.

4.5. WebFinger and URIs

WebFinger requests include a "resource" parameter (see Section 4.1) specifying the query target (URI) for which the client requests information. WebFinger is neutral regarding the scheme of such a URI: it could be an "acct" URI [18], an "http" or "https" URI, a "mailto" URI [19], or some other scheme.

5. Cross-Origin Resource Sharing (CORS)

WebFinger resources might not be accessible from a web browser due to "Same-Origin" policies. The current best practice is to make resources available to browsers through Cross-Origin Resource Sharing (CORS) [7], and servers MUST include the Access-Control-Allow-Origin HTTP header in responses. Servers SHOULD support the least restrictive setting by allowing any domain access to the WebFinger resource:

```
Access-Control-Allow-Origin: *
```

There are cases where defaulting to the least restrictive setting is not appropriate, for example a server on an intranet that provides sensitive company information SHOULD NOT allow CORS requests from any domain, as that could allow leaking of that sensitive information. A server that wishes to restrict access to information from external entities SHOULD use a more restrictive Access-Control-Allow-Origin header.

6. Access Control

As with all web resources, access to the WebFinger resource could require authentication. Further, failure to provide required credentials might result in the server forbidding access or providing a different response than had the client authenticated with the server.

Likewise, a WebFinger resource MAY provide different responses to different clients based on other factors, such as whether the client is inside or outside a corporate network. As a concrete example, a query performed on the internal corporate network might return link relations to employee pictures, whereas link relations for employee pictures might not be provided to external entities.

Further, link relations provided in a WebFinger resource representation might point to web resources that impose access restrictions. For example, the aforementioned corporate server may provide both internal and external entities with URIs to employee pictures, but further authentication might be required in order for the client to access the picture resources if the request comes from outside the corporate network.

The decisions made with respect to what set of link relations a WebFinger resource provides to one client versus another and what resources require further authentication, as well as the specific authentication mechanisms employed, are outside the scope of this document.

7. Hosted WebFinger Services

As with most services provided on the Internet, it is possible for a domain owner to utilize "hosted" WebFinger services. By way of example, a domain owner might control most aspects of their domain, but use a third-party hosting service for email. In the case of email, MX records identify mail servers for a domain. An MX record points to the mail server to which mail for the domain should be delivered. It does not matter to the sending mail server whether those MX records point to a server in the destination domain or a different domain.

Likewise, a domain owner might utilize the services of a third party to provide WebFinger services on behalf of its users. Just as a domain owner was required to insert MX records into DNS to allow for hosted email serves, the domain owner is required to redirect HTTP queries to its domain to allow for hosted WebFinger services.

When a query is issued to the WebFinger resource, the web server **MUST** return a response with a redirection status code that includes a Location header pointing to the location of the hosted WebFinger service URI. This WebFinger service URI does not need to point to the well-known WebFinger location on the hosting service provider server.

As an example, assume that example.com's WebFinger services are hosted by wf.example.net. Suppose a client issues a query for acct:alice@example.com like this:

```
GET /.well-known/webfinger?  
    resource=acct%3Aalice%40example.com HTTP/1.1  
Host: example.com
```

The server might respond with this:

```
HTTP/1.1 307 Temporary Redirect  
Access-Control-Allow-Origin: *  
Location: https://wf.example.net/example.com/webfinger?  
    resource=acct%3Aalice%40example.com
```

The client can then follow the redirection, re-issuing the request to the URI provided in the Location header. Note that the server will include any required URI parameters in the Location header value, which could be different than the URI parameters the client originally used.

8. Definition of WebFinger Applications

This specification details the protocol syntax used to query a domain for information about a URI, the syntax of the JSON Resource Descriptor (JRD) that is returned in response to that query, security requirements and considerations, hosted WebFinger services, various expected HTTP status codes, and so forth. However, this specification does not enumerate the various possible properties or link relation types that might be used in conjunction with WebFinger for a particular application, nor does it define what properties or link relation types one might expect to see in response to querying for a particular URI or URI scheme. Nonetheless, all of these unspecified elements are important in order to implement an interoperable application that utilizes the WebFinger protocol and MUST be specified in the relevant document(s) defining the particular application making use of the WebFinger protocol according to the procedures described in this section.

8.1. Specification of the URI Scheme and URI

Any application that uses WebFinger MUST specify the URI scheme(s) and, to the extent appropriate, what forms the URI(s) might take. For example, when querying for information about a user's account at some domain, it might make sense to specify the use of the acct URI scheme [18]. When trying to obtain the copyright information for a web page, it makes sense to specify the use of the web page URI (either http or https).

The examples in Sections 3.1 and 3.2 illustrate the use of different URI schemes with WebFinger applications. In the example in Section 3.1, WebFinger is used to retrieve information pertinent to OpenID Connect. In the example in Section 3.2, WebFinger is used to discover metadata information about a web page, including author and copyright information. Each of these applications of WebFinger needs to be fully specified to ensure interoperability.

8.2. Host Resolution

As explained in Section 4, the host to which a WebFinger query is issued is significant. In general, WebFinger applications would adhere to the procedures described in Section 4 in order to properly direct a WebFinger query.

However, some URI schemes do not have host portions and there might be some applications of WebFinger for which the host portion of a URI cannot or should not be utilized. In such instances, the application specification MUST clearly define the host resolution procedures, which might include provisioning a "default" host within the client to which queries are directed.

8.3. Specification of Properties

WebFinger defines both subject-specific properties (i.e., properties described in Section 4.4.3 that relate to the URI for which information is queried) and link-specific properties (see Section 4.4.4.5). This section refers to subject-specific properties.

Applications that utilize subject-specific properties **MUST** define the URIs used in identifying those properties, along with valid property values.

Consider this portion of the JRD found in the example in Section 3.2.

```
"properties" :
{
  "http://blgx.example.net/ns/version" : "1.3",
  "http://blgx.example.net/ns/ext" : null
}
```

Here, two properties are returned in the WebFinger response. Each of these would be defined in a WebFinger application specification. These two properties might be defined in the same WebFinger application specification or separately in different specifications. Since the latter is possible, it is important that WebFinger clients not assume that one property has any specific relationship with another property unless some relationship is explicitly defined in the particular WebFinger application specification.

8.4. Specification of Links

The links returned in a WebFinger response each comprise several pieces of information, some of which are optional (refer to Section 4.4.4). The WebFinger application specification **MUST** define each link and any values associated with a link, including the link relation type ("rel"), the expected media type ("type"), properties, and titles.

The target URI to which the link refers (i.e., the "href"), if present, would not normally be specified in an application specification. However, the URI scheme or any special characteristics of the URI would usually be specified. If a particular link does not require an external reference, then all of the semantics related to the use of that link **MUST** be defined within the application specification. Such links might rely only on properties or titles in the link to convey meaning.

8.5. One URI, Multiple Applications

It is important to be mindful of the fact that different WebFinger applications might specify the use of the same URI scheme and, in effect, the same URI for different purposes. That should not be a problem, since each of property identifier (see Sections 4.4.3 and

4.4.4.5) and link relation type would be uniquely defined for a specific application.

It should be noted that when a client requests information about a particular URI and receives a response with a number of different property identifiers or link relation types that the response is providing information about the URI without any particular semantics. How the client interprets the information SHOULD be in accordance with the particular application specification or set of specifications the client implements.

Any syntactically valid properties or links the client receives and that are not fully understood SHOULD be ignored and SHOULD NOT cause the client to report an error.

8.6. Registration of Link Relation Types and Properties

Application specifications MAY define a simple token as a link relation type for a link. In that case, the link relation type MUST be registered with IANA as specified in Sections 10.3.

Further, any defined properties MUST be registered with IANA as described in Section 10.4.

9. Security Considerations

9.1. Transport-Related Issues

Since this specification utilizes Cross-Origin Resource Sharing (CORS) [7], all of the security considerations applicable to CORS are also applicable to this specification.

The use of HTTPS is REQUIRED to ensure that information is not modified during transit. It should be appreciated that in environments where a web server is normally available, there exists the possibility that a compromised network might have its WebFinger resource operating on HTTPS replaced with one operating only over HTTP. As such, clients MUST NOT issue queries over a non-secure connection.

Clients MUST verify that the certificate used on an HTTPS connection is valid (as defined in [12]) and accept a response only if the certificate is valid.

9.2. User Privacy Considerations

Service providers and users should be aware that placing information on the Internet means that any user can access that information and WebFinger can be used to make it even easier to discover that information. While WebFinger can be an extremely useful tool for discovering one's avatar, blog, or other personal data, users should understand the risks, too.

Systems or services that expose personal data via WebFinger MUST provide an interface by which users can select which data elements are exposed through the WebFinger interface. For example, social networking sites might allow users to mark certain data as "public" and then utilize that marking as a means of determining what information to expose via WebFinger. The information published via WebFinger would thus comprise only the information marked as public by the user. Further, the user has the ability to remove information from publication via WebFinger by removing this marking.

WebFinger MUST NOT be used to provide any personal data unless publishing that data via WebFinger by the relevant service was explicitly authorized by the person whose information is being shared. Publishing one's personal data within an access-controlled or otherwise limited environment on the Internet does not equate to providing implicit authorization of further publication of that data via WebFinger.

The privacy and security concerns with publishing personal data via WebFinger are worth emphasizing again with respect to personal data that might reveal a user's current context (e.g., the user's location). The power of WebFinger comes from providing a single place where others can find pointers to information about a person, but service providers and users should be mindful of the nature of that information shared and the fact that it might be available for the entire world to see. Sharing location information, for example, would potentially put a person in danger from any individual who might seek to inflict harm on that person.

Users should be aware of how easily personal data one might publish can be used in unintended ways. In one study relevant to WebFinger-like services, Balduzzi et al. [20] took a large set of leaked email addresses and demonstrated a number of potential privacy concerns, including the ability to cross-correlate the same user's accounts over multiple social networks. The authors also describe potential mitigation strategies.

The easy access to user information via WebFinger was a design goal of the protocol, not a limitation. If one wishes to limit access to information available via WebFinger, such as WebFinger resources for use inside a corporate network, the network administrator needs to take necessary measures to limit access from outside the network. Using standard methods for securing web resources, network administrators do have the ability to control access to resources that might return sensitive information. Further, a server can be employed in such a way as to require authentication and prevent disclosure of information to unauthorized entities.

9.3. Abuse Potential

Service providers should be mindful of the potential for abuse using WebFinger.

As one example, one might query a WebFinger server only to discover whether a given URI is valid or not. With such a query, the person may deduce that an email identifier is valid, for example. Such an approach could help spammers maintain a current list of known email addresses and to discover new ones.

WebFinger could be used to associate a name or other personal data with an email address, allowing spammers to craft more convincing email messages. This might be of particular value in phishing attempts.

It is RECOMMENDED that implementers of WebFinger server software take steps to mitigate abuse, including malicious over-use of the server and harvesting of user information. Although there is no mechanism that can guarantee that publicly-accessible WebFinger databases won't be harvested, rate-limiting by IP address will prevent or at least dramatically slow harvest by private individuals without access to botnets or other distributed systems. The reason these mitigation strategies are not mandatory is that the correct choice of mitigation strategy (if any) depends greatly on the context. Implementers should not construe this as meaning that they do not need to consider whether to use a mitigation strategy, and, if so, what strategy to use.

WebFinger client developers should also be aware of potential abuse by spammers or those phishing for information about users. As an example, suppose a mail client was configured to automatically perform a WebFinger query on the sender of each received mail message. If a spammer sent an email using a unique identifier in the 'From' header, then when the WF query was performed the spammer would be able to associate the request with a particular user's email address. This would provide information to the spammer, including the user's IP address, the fact the user just checked email, what kind of WebFinger client the user utilized, and so on. For this reason, it is strongly advised that clients not perform WebFinger queries unless authorized by the user to do so.

9.4. Information Reliability

A WebFinger resource has no means of ensuring that information provided by a user is accurate. Likewise, neither the resource nor the client can be absolutely guaranteed that information has not been manipulated either at the server or along the communication path between the client and server. Use of HTTPS helps to address some concerns with manipulation of information along the communication path, but it clearly cannot address issues where the resource provided incorrect information, either due to being provided false information or due to malicious behavior on the part of the server administrator. As with any information service available on the Internet, users should be wary of information received from untrusted sources.

10. IANA Considerations

10.1. Well-Known URI

This specification registers the "webfinger" well-known URI in the Well-Known URI Registry as defined by [3].

URI suffix: webfinger

Change controller: IETF

Specification document(s): RFC XXXX

Related information: The query to the WebFinger resource will include one or more parameters in the query string; see Section 4.1 of RFCXXXX. Resources at this location are able to return a JSON Resource Descriptor (JRD) as described in Section 4.4 of RFCXXXX.

[RFC EDITOR: Please replace "XXXX" references in this section and the following section with the number for this RFC.]

10.2. JSON Resource Descriptor (JRD) Media Type

This specification registers the media type application/jrd+json for use with WebFinger in accordance with media type registration procedures defined in [10].

Type name: application

Subtype name: jrd+json

Required parameters: N/A

Optional parameters: N/A

In particular, because RFC 4627 already defines the character encoding for JSON, no "charset" parameter is used.

Encoding considerations: See RFC 6839, Section 3.1.

Security considerations:

The JSON Resource Descriptor (JRD) is a JavaScript Object Notation (JSON) object. It is a text format that must be parsed by entities that wish to utilize the format. Depending on the language and mechanism used to parse a JSON object, it is possible for an attacker to inject behavior into a running program. Therefore, care must be taken to properly parse a received JRD to ensure that only a valid JSON object is present and that no JavaScript or other code is injected or executed unexpectedly.

Interoperability considerations:

This media type is a JavaScript Object Notation (JSON) object and can be consumed by any software application that can consume JSON objects.

Published specification: RFC XXXX

Applications that use this media type:

The JSON Resource Descriptor (JRD) is used by the WebFinger protocol (RFC XXXX) to enable the exchange of information between a client and a WebFinger resource over HTTPS.

Fragment identifier considerations:

The syntax and semantics of fragment identifiers SHOULD be as specified for "application/json". (At publication of this document, there is no fragment identification syntax defined for "application/json".)

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): jrd

Macintosh file type code(s): N/A

Person & email address to contact for further information:

Paul E. Jones <paulej@packetizer.com>

Intended usage: COMMON

Restrictions on usage: N/A

Author: Paul E. Jones <paulej@packetizer.com>

Change controller:

IETF has change control over this registration.

Provisional registration? (standards tree only): N/A

10.3. Registering Link Relation Types

RFC 5988 established a Link Relation Type Registry that is re-used by WebFinger applications.

Link relation types used by WebFinger applications are registered in the Link Relations Type Registry as per the procedures of Section

6.2.1 of RFC 5988. The "Notes" entry for the registration SHOULD indicate if property values associated with the link relation type are registered in the WebFinger Properties registry with a link to the registry.

10.4. Establishment of the WebFinger Properties Registry

WebFinger utilizes URIs to identify properties of a subject or link and the associated values (see Section 8.3 and Section 8.6). This specification establishes a new "WebFinger Properties" registry to record property identifiers.

10.4.1. The Registration Template

The registration template for WebFinger properties is:

- o Property Identifier:
- o Link Type:
- o Description:
- o Reference:
- o Notes: [optional]

The "Property Identifier" must be a URI that identifies the property being registered.

The "Link Type" contains the name of a Link Relation Type with which this property identifier is used. If the property is a subject-specific property, then this field is specified as "N/A".

The "Description" is intended to explaining the purpose of the property.

The "Reference" field points to the specification that defines the registered property.

The optional "Notes" field is for conveying any useful information about the property that might be of value to implementers.

10.4.2. The Registration Procedures

The IETF has created a mailing list, webfinger@ietf.org, which can be used for public discussion of the WebFinger protocol and any applications that use it. Prior to registration of a WebFinger property, discussion on the mailing list is strongly encouraged. The IESG has appointed Designated Experts who will monitor the webfinger@ietf.org mailing list and review registrations.

A WebFinger property is registered with a Specification Required (see RFC 5226 [13]) after a review by the Designated Expert(s). The review is normally expected to take on the order of two to four weeks. However, the Designated Expert(s) may approve a registration prior to publication of a specification once the Designated Expert(s) are satisfied that such a specification will be published. In evaluating registration requests, the Designated Expert(s) should make an effort to avoid registering two different properties that have the same meaning. Where a proposed property is similar to an already-defined property, Designated Expert(s) should insist that enough text be included in the description or notes section of the template to sufficiently differentiate the new property from an existing one.

The registration procedure begins when a completed registration template (as defined above) sent to webfinger@ietf.org and iana@iana.org. IANA will track the review process and communicate the results to the registrant. The WebFinger mailing list provides an opportunity for community discussion and input, and the Designated Expert(s) may use that input to inform their review. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful if re-submitted.

The specification registering the WebFinger property MUST include the completed registration template shown above. Once the registration procedure concludes successfully, IANA creates or modifies the corresponding record in the "WebFinger Properties" registry.

11. Acknowledgments

This document has benefited from extensive discussion and review of many of the members of the APPSAWG working group. The authors would like to especially acknowledge the invaluable input of Eran Hammer-Lahav, Blaine Cook, Brad Fitzpatrick, Laurent-Walter Goix, Joe Clarke, Peter Saint-Andre, Dick Hardt, Tim Bray, James Snell, Melvin Carvalho, Evan Prodromou, Mark Nottingham, Elf Pavlik, Bjoern Hoehrmann, Subramanian Moonesamy, Joe Gregorio, John Bradley, and others that we have undoubtedly, but inadvertently, missed.

The authors would also like to express their gratitude to the chairs of APPSAWG, especially Salvatore Loreto for his assistance in shepherding this document. We also want to thank Barry Leiba and Pete Resnick, the Applications Area Directors, for their support and exhaustive reviews.

12. References

12.1. 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] Nottingham, M., Hammer-Lahav, E., "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.
- [4] Nottingham, M., "Web Linking", RFC 5988, October 2010.
- [5] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [6] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [7] Van Kesteren, A., "Cross-Origin Resource Sharing", W3C CORS <http://www.w3.org/TR/cors/>, July 2010.
- [8] IANA, "Link Relations", <http://www.iana.org/assignments/link-relations/>.
- [9] IANA, "MIME Media Types", <http://www.iana.org/assignments/media-types/index.html>.
- [10] Freed, N., Klensin, J., Hansen, T., "Media Type Specifications and Registration Procedures", RFC 6838, January 2013.
- [11] Phillips, A., Davis, M., "Tags for Identifying Languages", RFC 5646, January 2009.
- [12] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [13] Narten, T. and H. Alvestrand, "Guidelines for Writing an, IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

12.2. Informative References

- [14] Perreault, S., "vCard Format Specification", RFC 6350, August 2011.
- [15] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, "OpenID Connect Messages 1.0", July 2013, http://openid.net/specs/openid-connect-messages-1_0.html.
- [16] Hammer-Lahav, E. and Cook, B., "Web Host Metadata", RFC 6415, October 2011.
- [17] Hammer-Lahav, E. and W. Norris, "Extensible Resource Descriptor (XRD) Version 1.0", <http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html>.

- [18] Saint-Andre, P., "The 'acct' URI Scheme", draft-ietf-appsawg-acct-uri-06, July 2013.
- [19] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, October 2010.
- [20] Balduzzi, Marco, et al., "Abusing social networks for automated user profiling", Recent Advances in Intrusion Detection, Springer Berlin Heidelberg, 2010, https://www.eurecom.fr/en/publication/3042/download/rs-publi-3042_1.pdf.

Author's Addresses

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 476 2048
Email: paulej@packetizer.com
IM: <xmpp:paulej@packetizer.com>

Gonzalo Salgueiro
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 392 3266
Email: gsalguei@cisco.com
IM: <xmpp:gsalguei@cisco.com>

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: <http://self-issued.info/>

Joseph Smarr
Google

Email: jsmarr@google.com

APPSAWG
Internet-Draft
Intended status: BCP
Expires: October 11, 2012

P. Saint-Andre
Cisco Systems, Inc.
D. Crocker
Brandenburg InternetWorking
M. Nottingham
Rackspace
April 9, 2012

Deprecating the X- Prefix and Similar Constructs in Application
Protocols
draft-ietf-appsawg-xdash-05

Abstract

Historically, designers and implementers of application protocols have often distinguished between standardized and unstandardized parameters by prefixing the names of unstandardized parameters with the string "X-" or similar constructs. In practice, that convention causes more problems than it solves. Therefore, this document deprecates the convention for newly-defined parameters with textual (as opposed to numerical) names in application protocols.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on October 11, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Recommendations for Implementers of Application Protocols . .	4
3. Recommendations for Creators of New Parameters	4
4. Recommendations for Protocol Designers	5
5. Security Considerations	5
6. IANA Considerations	5
7. Acknowledgements	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Appendix A. Background	8
Appendix B. Analysis	10
Authors' Addresses	13

1. Introduction

Many application protocols use parameters with textual (as opposed to numerical) names to identify data (media types, header fields in Internet mail messages and HTTP requests, vCard parameters and properties, etc.). Historically, designers and implementers of application protocols have often distinguished between standardized and unstandardized parameters by prefixing the names of unstandardized parameters with the string "X-" or similar constructs (e.g., "x."), where the "X" is commonly understood to stand for "eXperimental" or "eXtension".

Under this convention, the name of a parameter not only identified the data, but also embedded the status of the parameter into the name itself: a parameter defined in a specification produced by a recognized standards development organization (or registered according to processes defined in such a specification) did not start with "X-" or similar constructs, whereas a parameter defined outside such a specification or process started with "X-" or similar constructs.

As explained more fully under Appendix A, this convention was encouraged for many years in application protocols such as file transfer, email, and the World Wide Web. In particular, it was codified for email by [RFC822] (via the distinction between "Extension-fields" and "user-defined-fields"), but then removed by [RFC2822] based on implementation and deployment experience. A similar progression occurred for SIP technologies with regard to the "P-" header, as explained in [RFC5727]. The reasoning behind those changes is explored under Appendix B.

In short, although in theory the "X-" convention was a good way to avoid collisions (and attendant interoperability problems) between standardized parameters and unstandardized parameters, in practice the benefits have been outweighed by the costs associated with the leakage of unstandardized parameters into the standards space.

This document generalizes from the experience of the email and SIP communities by doing the following:

1. Deprecates the "X-" convention for newly-defined parameters in application protocols, even where that convention was only implicit instead of being codified in a protocol specification (as was done for email in [RFC822]).
2. Makes specific recommendations about how to proceed in a world without the distinction between standardized and unstandardized parameters (although only for parameters with textual names, not

parameters that are expressed as numbers, which are out of scope).

3. Does not recommend against the practice of private, local, preliminary, experimental, or implementation-specific parameters, only against the use of "X-" and similar constructs in the names of such parameters.
4. Makes no recommendation as to whether existing "X-" parameters ought to remain in use or be migrated to a format without the "X-"; this is a matter for the creators or maintainers of those parameters.
5. Does not override existing specifications that legislate the use of "X-" for particular application protocols (e.g., the "x-name" token in [RFC5545]); this is a matter for the designers of those protocols.

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

2. Recommendations for Implementers of Application Protocols

Implementations of application protocols MUST NOT make any assumptions about the status of a parameter, nor take automatic action regarding a parameter, based solely on the presence or absence of "X-" or a similar construct in the parameter's name.

3. Recommendations for Creators of New Parameters

Creators of new parameters to be used in the context of application protocols:

1. SHOULD assume that all parameters they create might become standardized, public, commonly deployed, or used across multiple implementations.
2. SHOULD employ meaningful parameter names that they have reason to believe are currently unused.
3. SHOULD NOT prefix their parameter names with "X-" or similar constructs.

Note: If the relevant parameter name space has conventions about

associating parameter names with those who create them, a parameter name could incorporate the organization's name or primary domain name (see Appendix B for examples).

4. Recommendations for Protocol Designers

Designers of new application protocols that allow extensions using parameters:

1. SHOULD establish registries with potentially unlimited value-spaces, if appropriate defining both permanent and provisional registries.
2. SHOULD define simple, clear registration procedures.
3. SHOULD mandate registration of all non-private parameters, independent of the form of the parameter names.
4. SHOULD NOT prohibit parameters with the "X-" prefix or similar constructs from being registered.
5. MUST NOT assume that a parameter with an "X-" prefix or similar constructs is unstandardized.
6. MUST NOT assume that a parameter without an "X-" prefix or similar constructs is standard.

5. Security Considerations

Interoperability and migration issues with security-critical parameters can result in unnecessary vulnerabilities (see Appendix B for further discussion).

As a corollary to the recommendation provided under Section 2, implementations MUST NOT assume that standardized parameters are "secure" whereas unstandardized parameters are "insecure", based solely on the names of such parameters.

6. IANA Considerations

This document does not modify registration procedures currently in force for various application protocols. However, such procedures might be updated in the future to incorporate the best practices defined in this document.

7. Acknowledgements

Thanks to Claudio Allocchio, Adam Barth, Nathaniel Borenstein, Eric Burger, Stuart Cheshire, Al Constanzo, Dave Cridland, Ralph Droms, Martin Duerst, Frank Ellermann, J.D. Falk, Ned Freed, Tony Finch, Randall Gellens, Tony Hansen, Ted Hardie, Joe Hildebrand, Alfred Hoenes, Paul Hoffman, Eric Johnson, Scott Kelly, Scott Kitterman, John Klensin, Graham Klyne, Murray Kucherawy, Eliot Lear, John Levine, Bill McQuillan, Alexey Melnikov, Subramanian Moonesamy, Keith Moore, Ben Niven-Jenkins, Zoltan Ordogh, Tim Petch, Dirk Pranke, Randy Presuhn, Julian Reschke, Dan Romascanu, Doug Royer, Andrew Sullivan, Henry Thompson, Martin Thomson, Matthew Wild, Nicolas Williams, Tim Williams, Mykyta Yevstifeyev, and Kurt Zeilenga for their feedback.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

- [BCP9] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.
- [BCP26] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [BCP82] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, January 2004.
- [RFC691] Harvey, B., "One more try on the FTP", RFC 691, June 1975.
- [RFC737] Harrenstien, K., "FTP extension: XSEN", RFC 737, October 1977.
- [RFC743] Harrenstien, K., "FTP extension: XRSQ/XRCP", RFC 743, December 1977.
- [RFC775] Mankins, D., Franklin, D., and A. Owen, "Directory oriented FTP commands", RFC 775, December 1980.
- [RFC822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC1154] Robinson, D. and R. Ullmann, "Encoding header field for internet messages", RFC 1154, April 1990.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.
- [RFC2068] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.
- [RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, September 1998.
- [RFC2616] 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.
- [RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.
- [RFC2939] Droms, R., "Procedures and IANA Guidelines for Definition of New DHCP Options and Message Types", BCP 43, RFC 2939, September 2000.
- [RFC3406] Daigle, L., van Gulik, D., Iannella, R., and P. Faltstrom, "Uniform Resource Names (URN) Namespace Definition Mechanisms", BCP 66, RFC 3406, October 2002.
- [RFC3427] Mankin, A., Bradner, S., Mahy, R., Willis, D., Ott, J., and B. Rosen, "Change Process for the Session Initiation Protocol (SIP)", RFC 3427, December 2002.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.
- [RFC4512] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, June 2006.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5064] Duerst, M., "The Archived-At Message Header Field", RFC 5064, December 2007.
- [RFC5451] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", RFC 5451, April 2009.
- [RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, September 2009.
- [RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.
- [RFC5727] Peterson, J., Jennings, C., and R. Sparks, "Change Process for the Session Initiation Protocol (SIP) and the Real-time Applications and Infrastructure Area", BCP 67, RFC 5727, March 2010.

Appendix A. Background

The beginnings of the "X-" convention can be found in a suggestion made by Brian Harvey in 1975 with regard to FTP parameters [RFC691]:

Thus, FTP servers which care about the distinction between Telnet print and non-print could implement SRVR N and SRVR T. Ideally the SRVR parameters should be registered with Jon Postel to avoid conflicts, although it is not a disaster if two sites use the same parameter for different things. I suggest that parameters be allowed to be more than one letter, and that an initial letter X be used for really local idiosyncracies.

This "X" prefix was subsequently used in [RFC737], [RFC743], and [RFC775]. This usage was noted in [RFC1123]:

FTP allows "experimental" commands, whose names begin with "X". If these commands are subsequently adopted as standards, there may still be existing implementations using the "X" form.... All FTP implementations SHOULD recognize both forms of these commands, by simply equating them with extra entries in the command lookup table.

The "X-" convention has been used for email header fields since at least the publication of [RFC822] in 1982, which distinguished between "Extension-fields" and "user-defined-fields" as follows:

The prefatory string "X-" will never be used in the names of Extension-fields. This provides user-defined fields with a protected set of names.

That rule was restated by [RFC1154] as follows:

Keywords beginning with "X-" are permanently reserved to implementation-specific use. No standard registered encoding keyword will ever begin with "X-".

This convention continued with various specifications for media types ([RFC2045], [RFC2046], [RFC2047]), HTTP headers ([RFC2068], [RFC2616]), vCard parameters and properties ([RFC2426]), Uniform Resource Names ([RFC3406]), LDAP field names ([RFC4512]), and other application technologies.

However, use of the "X-" prefix in email headers was effectively deprecated between the publication of [RFC822] in 1982 and the publication of [RFC2822] in 2001 by removing the distinction between the "extension-field" construct and the "user-defined-field" construct (a similar change happened with regard to Session Initiation Protocol "P-" headers when [RFC3427] was obsoleted by [RFC5727]).

Despite the fact that parameters containing the "X-" string have been effectively deprecated in email headers, they continue to be used in

a wide variety of application protocols. The two primary situations motivating such use are:

1. Experiments that are intended to possibly be standardized in the future, if they are successful.
2. Extensions that are intended to never be standardized because they are intended only for implementation-specific use or for local use on private networks.

Use of this naming convention is not mandated by the Internet Standards Process [BCP9] or IANA registration rules [BCP26]. Rather it is an individual choice by each specification that references the convention or each administrative process that chooses to use it. In particular, some standards-track RFCs have interpreted the convention in a normative way (e.g., [RFC822] and [RFC5451]).

Appendix B. Analysis

The primary problem with the "X-" convention is that unstandardized parameters have a tendency to leak into the protected space of standardized parameters, thus introducing the need for migration from the "X-" name to a standardized name. Migration, in turn, introduces interoperability issues (and sometimes security issues) because older implementations will support only the "X-" name and newer implementations might support only the standardized name. To preserve interoperability, newer implementations simply support the "X-" name forever, which means that the unstandardized name has become a de facto standard (thus obviating the need for segregation of the name space into standardized and unstandardized areas in the first place).

We have already seen this phenomenon at work with regard to FTP in the quote from [RFC1123] in the previous section. The HTTP community had the same experience with the "x-gzip" and "x-compress" media types, as noted in [RFC2068]:

For compatibility with previous implementations of HTTP, applications should consider "x-gzip" and "x-compress" to be equivalent to "gzip" and "compress" respectively.

A similar example can be found in [RFC5064], which defined the "Archived-At" message header field but also found it necessary to define and register the "X-Archived-At" field:

For backwards compatibility, this document also describes the X-Archived-At header field, a precursor of the Archived-At header field. The X-Archived-At header field MAY also be parsed, but SHOULD NOT be generated.

One of the original reasons for segregation of name spaces into standardized and unstandardized areas was the perceived difficulty of registering names. However, the solution to that problem has been simpler registration rules, such as those provided by [RFC3864] and [RFC4288]. As explained in [RFC4288]:

[W]ith the simplified registration procedures described above for vendor and personal trees, it should rarely, if ever, be necessary to use unregistered experimental types. Therefore, use of both "x-" and "x." forms is discouraged.

For some name spaces, another helpful practice has been the establishment of separate registries for permanent names and provisional names, as in [RFC4395].

Furthermore, often standardization of a unstandardized parameter leads to subtly different behavior (e.g., the standardized version might have different security properties as a result of security review provided during the standardization process). If implementers treat the old, unstandardized parameter and the new, standardized parameter as equivalent, interoperability and security problems can ensue. Analysis of unstandardized parameters to detect and correct flaws is in general a good thing and is not intended to be discouraged by the lack of distinction in element names. Whenever an originally unstandardized parameter or protocol element is standardized and the new form has differences which affect interoperability or security properties, implementations MUST NOT treat the old form as identical to the new form.

For similar considerations with regard to the "P-" convention in the Session Initiation Protocol, see [RFC5727].

In some situations, segregating the parameter name space used in a given application protocol can be justified:

1. When it is extremely unlikely that some parameters will ever be standardized. In this case implementation-specific and private-use parameters could at least incorporate the organization's name (e.g., "ExampleInc-foo" or, consistent with [RFC4288], "VND.ExampleInc.foo") or primary domain name (e.g., "com.example.foo" or a Uniform Resource Identifier [RFC3986] such as "http://example.com/foo"). In rare cases, truly experimental parameters could be given meaningless names such as nonsense

words, the output of a hash function, or UUIDs [RFC4122].

2. When parameter names might have significant meaning. This case too is rare, since implementers can almost always find a synonym for an existing term (e.g., "urgency" instead of "priority") or simply invent a more creative name (e.g., "get-it-there-fast"). The existence of multiple similarly-named parameters can be confusing, but this is true regardless if there is an attempt to segregate standardized and unstandardized (e.g., "X-Priority" can be confused with "Urgency").
3. When parameter names need to be very short (e.g., as in [RFC5646] for language tags). In this case it can be more efficient to assign numbers instead of human-readable names (e.g., as in [RFC2939] for DHCP options) and to leave a certain numeric range for implementation-specific extensions or private use (e.g., as with the codec numbers used with the Session Description Protocol [RFC4566]).

There are three primary objections to deprecating the "X-" convention as a best practice for application protocols:

1. Implementers might mistake one parameter for another parameter that has a similar name; a rigid distinction such as an "X-" prefix can make this clear. However, in practice implementers are forced to blur the distinction (e.g., by treating "X-foo" as a de facto standard) and so it inevitably becomes meaningless.
2. Collisions are undesirable and it would be bad for both a standardized parameter "foo" and a unstandardized parameter "foo" to exist simultaneously. However, names are almost always cheap, so an experimental, implementation-specific, or private-use name of "foo" does not prevent a standards development organization from issuing a similarly creative name such as "bar".
3. [BCP82] is entitled "Assigning Experimental and Testing Numbers Considered Useful" and therefore implies that the "X-" prefix is also useful for experimental parameters. However, BCP 82 addresses the need for protocol numbers when the pool of such numbers is strictly limited (e.g., DHCP options) or when a number is absolutely required even for purely experimental purposes (e.g., the Protocol field of the IP header). In almost all application protocols that make use of protocol parameters (including email headers, media types, HTTP headers, vCard parameters and properties, URNs, and LDAP field names), the name space is not limited or constrained in any way, so there is no need to assign a block of names for private use or experimental purposes (see also [BCP26]).

Therefore it appears that segregating the parameter space into a standardized area and a unstandardized area has few if any benefits, and has at least one significant cost in terms of interoperability.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

D. Crocker
Brandenburg InternetWorking
675 Spruce Dr.
Sunnyvale
USA

Phone: +1.408.246.8253
Email: dcrocker@bbiw.net
URI: <http://bbiw.net>

Mark Nottingham
Rackspace

Email: mnot@mnot.net
URI: <http://www.mnot.net>

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2013

Z. Ordogh
Research In Motion Limited
March 11, 2013

Spam reporting using IMAP: SREP
draft-ordogh-spam-reporting-using-imap-04

Abstract

This document defines an IMAP extension which allows a client to report spam by reference and allows an IMAP server to perform any action on the reported messages, including leaving the action at the client's discretion.

In addition, this document discusses how an IMAP server can tap into spam aggregator services, ultimately allowing the IMAP server to improve its decision-making process.

Conventions Used In This Document

In examples, "C:" and "S:" indicate lines sent by the client or the server, respectively.

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

This specification follows the recommendations in [XDASH].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on September 12, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Scope	3
3. The SREP command	4
3.1. Directives	5
3.2. Abuse types	5
3.3. References	5
3.4. Part identifiers	6
3.5. Request Action	6
3.6. Responses and Results	7
3.7. Formal Syntax	8
3.8. Examples to report spam	10
3.9. Examples to report messages as no longer spam	10
3.10. Example flows	11
3.10.1. The server simply flags a message	11
4. Acknowledgements	14
5. IANA Considerations	14
6. Security Considerations	15
7. References	15
7.1. Normative References	15
7.2. Informative References	16

1. Introduction

The Internet Message Access Protocol [IMAP4] does not support reporting spam on its own. There are a number of solutions available based on the multipart/report content type defined in [OLD-REPORT] and its revision, [REPORT]. However, these solutions require including the message contents and hence, consume bandwidth to transmit the entire message. In bandwidth-constrained environments - such as mobile networks - it is highly desirable to send only a minimum set of information - a reference - instead of the entire message. Solutions that exist today employ manipulating proprietary flags in the IMAP storage to achieve the bare minimum, however more advanced solutions cannot be developed by using flags only; the IMAP server needs to be involved actively in the spam reporting process.

Furthermore, it is highly desirable to permit individual server implementations to handle spam in any way these systems choose to: do nothing, flag, perform deletion or relocation, recommend deletion or relocation to the client, or, leave the decision at the client's discretion as a whole. However, in order to make such a decision on the server side, a spam aggregator service, such as [OMA-SPAMREP], needs to be involved in the decision-making process.

This document specifies a new IMAP command, SREP, along with its syntax, which allows a client to inform the server that the user considered a message (or parts thereof) spam, or, that the user no longer considers a message (or parts thereof) spam. Since all information about the message is readily available on the server, the command also allows the server to implement a more intelligent and accurate decision logic, which may be invoked when the spam is reported and the server can respond with its decision to the client.

Additionally, this document contains example flows, illustrating various decisions that the server may choose to evaluate, including invoking an aggregator service, such as one based on [OMA-SPAMREP].

The SREP command allows:

- reporting spam; i.e. set the spam condition, and,
- reporting that a message (that was reported spam earlier) is no longer spam; i.e. clear the spam condition.

2. Scope

This document focuses only on the client-server interactions and the scope is limited to messages that either exist on the IMAP server, or, exist elsewhere and the IMAP server is configured to access them. Consequently, deposit-time filtering, messages that have been deleted, and messages that exist in an external storage but are

accessible only via an access protocol unknown to the IMAP server are out of scope.

3. The SREP command

The SREP command follows the conventions of [IMAP4].

Arguments:

- directive; see Section 3.1
- OPTIONAL abuse type; see Section 3.2
- reference; see Section 3.3
- OPTIONAL list of part identifiers; see Section 3.4
- OPTIONAL request action; see Section 3.5

Responses; see Section 3.6:

- OPTIONAL OK response: RELOCATE
- OPTIONAL OK response: RELOCATED
- OPTIONAL OK response: DELETE
- OPTIONAL OK response: DELETED
- OPTIONAL OK response: KEYWORD

Result:

- OK - command completed successfully
- NO - the server cannot access one or more messages (deleted or unauthorized)
- BAD - there was an error during processing the command (syntax or unsupported parameter)

The formal syntax of the SREP command is defined in Section 3.7.

The SREP command allows:

- reporting spam; i.e. set the spam condition, and,
- reporting that a message (that was reported spam earlier) is no longer spam; i.e. clear the spam condition.

The SREP command may be used with any IMAP4 server implementation that returns "SREP" as one of the supported capabilities in response to the CAPABILITY command. If the server does not indicate support for the SREP capability, the client MUST NOT use the SREP command.

The SREP command may result in ambiguity, therefore the client MUST NOT send any commands before the result of the SREP command has been received, see Section 5.5 in [IMAP].

The command MAY be issued on one or more messages at a time, in the currently selected mailbox.

The command MAY be extended in the future with new parameters

(actions, directives, reference types, etc). Servers MUST be able to recognize parameters unknown to them and respond with a BAD response in case they encounter such a parameter.

3.1. Directives

The directive argument tells the server whether a message is being reported as a spam, or, it is being reported as no longer a spam. The SREP command MUST include the directive. To report a spam, the directive MUST be SET. To report that a message is no longer considered to be a spam, the directive MUST be CLEAR.

Extensions are permitted, as defined in Section 3.7.

3.2. Abuse types

The client may have additional information about the spam regarding the nature of the abuse. When such information is available, the client SHOULD include the abuse type argument in the request. When such information is not available, the client MUST omit the abuse type argument from the request. When the directive argument is CLEAR, the client MUST omit the abuse type argument from the request. This specification defines the following abuse types:

1. Phishing (forgery, link manipulation, etc.): an attempt to divulge information from the recipient by masquerading the sender and/or the content(s) of the message as a trustworthy form of communication.
2. Malware (virus, spyware, etc.): a malicious piece of software code embedded or attached to the message specifically designed to disrupt normal operation, gather sensitive information, gain unauthorized access, and/or perform other abusive behavior upon execution.

Extensions are permitted, as defined in Section 3.7.

3.3. References

The reference argument consists of a reference type and a reference value. In general, the reference type MUST indicate the format of the reference while the reference value MUST contain a value corresponding to the indicated reference format. To use a unique identifier specified in [IMAP4], the reference type MUST be UID and the reference value MUST be a number expressing the unique identifier of the message. To use a sequence set specified in [IMAP4], the reference type MUST be SEQ and the reference value MUST be sequence numbers corresponding to the specified message sequence number set. To use an authorized URL specified in [URLAUTH], the reference type

MUST be URLAUTH and the reference value MUST be an URLAUTH-authorized URL, authorizing the entire message.

Extensions are permitted, as defined in Section 3.7.

3.4. Part identifiers

When the reference identifies one and only one message, the list of part identifiers MAY be included to improve the accuracy of spam detection. When the reference identifies more than one message, the list of part identifiers MUST be omitted.

The list of part identifiers is a parenthesized list of part identifiers. Part identifiers MAY identify header fields or bodies. Header field identifiers MUST be prefixed with the word 'header' and the dot ('.') character MUST be used as the separator character. Header fields MUST be identified by the name of the header field.

Example:

The 'From' header field is identified as 'header.from'.

Body identifiers MUST be prefixed with the word 'body' and the dot ('.') character MUST be used as the separator character. Bodies MUST be identified by their positions within the message hierarchy, where the first position is 1 and the main level is 1. To refer the entire body of a message (or all bodies of a multipart message), the separator character, the position MUST be omitted.

Examples:

- The entire body of a message (or all bodies of a multipart message) is identified as 'body'.
- Considering a simple multipart message, the part following the first boundary is identified as 'body.1'.
- Considering a multipart message that includes an email attachment following the second boundary, and the email attachment containing text following the first boundary, the text within the email message is identified as 'body.2.1'.

The formal syntax of the part-id-list is defined in Section 3.7.

3.5. Request Action

The request action argument explicitly tells the server what to do with the message. To request a specific action from the server explicitly, the SREP command MUST include the request action argument. To not request a specific action, the SREP command MUST NOT include the request action argument; in this case, the server MUST decide the course of action. The client MAY specify either one of the following actions:

- The KEYWORD the client requests that only keyword(s) should be added to the message. The server MUST add the appropriate keyword.
- The RELOCATE the client requests that the message should be relocated. The server MUST relocate the message by copying the message to the destination mailbox removing the original as if another connected client requested this action using an IMAP MOVE [IMAPMOVE] command.
- The DELETE the client requests that the message should be deleted. The server MUST delete the message as if another client performed this action.

The server MUST ignore the destination mailbox in case it is nil, or, the request action is KEYWORD or DELETE. It is assumed that the server is pre-configured with the location where user's spam messages are stored. If the server is not configured with such information and the destination mailbox in a RELOCATE action is nil, or, destination mailbox in a RELOCATE action is otherwise inaccessible to the user (does not exist, insufficient permission, etc) the the server MUST reject the request (see BAD response in Section 3.6). NOTE: While the DELETE action does not seem appropriate in case the directive argument is CLEAR, it is permitted. The formal syntax of the request action argument is defined in Section 3.7.

3.6. Responses and Results

The SREP command MAY result in system flag changes, keyword changes, message relocation, message removal, or a combination of these.

The result of the command MUST be either OK, NO or BAD:

- The OK result MUST be returned only in case the server parsed and completed the command successfully.
- The NO result MUST be returned only in case the server parsed the command successfully, but there is a problem with the referenced message(s) that prevents the server from completing the requested actions, such as one or more messages do not exist on the server, one or more messages are not properly authorized by URLAUTH, etc.
- The BAD result MUST be returned only in case the server cannot parse the command, or a configuration error is preventing the server from completing the requested actions.

When the result is OK, the response to a SET directive MUST be either KEYWORD, RELOCATE, RELOCATED, DELETE, or DELETED.

When the result is OK, the response to a CLEAR directive MUST be either KEYWORD, RELOCATE, or RELOCATED.

The server responses are:

- The KEYWORD response occurs in case this specific action has been explicitly requested by the client, or, in case the server decided that only the keywords should be updated either because it does not wish to give any recommendation to the client (RELOCATE or DELETE), or, because it does not have sufficient information either internally, or, from the spam aggregator service that it is configured to use.
- The RELOCATE response occurs in case the server decided that the message should be relocated, however leaves this action to the client. The client MAY decide what to do with the message.
- The RELOCATED response occurs in case this specific action has been explicitly requested by the client, or, in case the server decided that the message should be relocated and it performed relocation of the message to the appropriate location before the response was sent. The server MUST relocate the message by copying the message to the appropriate location and removing the original as if another connected client requested this action using an IMAP MOVE [IMAPMOVE] command.
- The DELETE response occurs in case the server decided that the message should be deleted, however leaves this action to the client. The client MAY decide what to do with the message.
- The DELETED response occurs in case this specific action has been explicitly requested by the client, or, in case the server decided that the message should be deleted, and performed deletion of the message before the response was sent. The server MUST delete the message as if another client performed this action.

The KEYWORD, RELOCATE and DELETE responses MUST include the list of flags/keywords that have been added or removed. Added keywords MUST be prefixed with a plus sign ('+'), while removed keywords MUST be prefixed with a minus sign ('-'). The RELOCATED and DELETED responses MUST NOT include keywords. The formal syntax of the actions is defined in Section 3.7.

3.7. Formal Syntax

This document extends the formal syntax defined in [IMAP4] using the Augmented Backus-Naur Form (ABNF) notation specified in [ABNF].

Note: all string literals are case insensitive.

```

srep-command      = "SREP" SP directive *1(SP abuse-type) SP \
                    reference *1(SP part-id-list) \
                    *1(SP request-action)
directive         = "SET" / "CLEAR" / directive-ext
directive-ext     = atom
                    ; It is not required that new directives
                    ; begin with "X-", see [XDASH]
abuse-type        = "AT" SP abuse-type-id
abuse-type-id     = 1*DIGIT
                    ; no leading zeroes or signs
                    ; New abuse types MUST be registered with
                    ; IANA as standard or standards-track
reference         = reference-type SP reference-value
reference-type     = "UID" / "SEQ" / "URLAUTH" / reference-type-ext
reference-type-ext = atom
                    ; It is not required that new reference types
                    ; begin with "X-", see [XDASH]
reference-value   = uniqueid /           ; see [IMAP]
                    sequence-set /       ; see [IMAP]
                    authorized-url /     ; see [URLAUTH]
                    reference-value-ext
authorized-url    = authimapurlfull /    ; see [URLAUTH]
                    authimapurlrump      ; see [URLAUTH]
reference-value-ext = atom
                    ; New reference values MUST correspond to
                    ; reference-type-ext
part-id-list     = "(" part-id *(SP part-id) ")"
part-id          = header-id / body-id
header-id        = "header." header-fld-name
                    ; see header-fld-name in [IMAP]
body-id          = "body" *("." 1*DIGIT)
                    ; no leading zeroes or signs in numeric part
request-action   = "DO" SP req-action *1(SP destination-box)
req-action       = "KEYWORD" /
                    "RELOCATE" /
                    "DELETE"
destination-box  = mailbox / nil
resp-text-code   = resp-spam-actions    ; responses specific to
                                         ; this command, extending
                                         ; existing resp-text-code
                                         ; defined in [IMAP]
resp-spam-actions = "KEYWORD" flag-list / ; see flag-list in [IMAP]
                    "RELOCATE" flag-list / ; see flag-list in [IMAP]
                    "RELOCATED" /
                    "DELETE" flag-list /  ; see flag-list in [IMAP]
                    "DELETED"

```

3.8. Examples to report spam

Report single message as spam; no identified parts; server only flags the message and hints that it should be moved:

```
C: Z020 SREP SET SEQ 10
S: Z020 OK [RELOCATE +$OMAEVVM10-spam-user-identified] SREP
Completed.
```

Report single message as spam; header and body identified; server adds the appropriate flags and hints that it should be deleted:

```
C: Z040 SREP SET SEQ 9 (header.from body.2)
S: Z040 OK [DELETE (+$OMAEVVM10-spam-user-identified-field.from
+$OMAEVVM10-spam-user-identified-body.2)] SREP Completed.
```

Report single message as spam; no identified parts; server moves the message (may clear/set flags too, but that is irrelevant because the client will need to reconcile anyway).

```
C: Z060 SREP SET SEQ 8
S: Z060 OK [RELOCATED] SREP Completed.
```

Report single message as spam; no identified parts; server deletes the message.

```
C: Z080 SREP SET SEQ 6
S: Z080 OK [DELETED] SREP Completed.
```

Report single message as spam; no identified parts; client requests explicitly to delete the message, server deletes the message as the client requested.

```
C: Z100 SREP SET SEQ 4 DO DELETE NIL
S: Z100 OK [DELETED] SREP Completed.
```

3.9. Examples to report messages as no longer spam

Report single message as no longer spam; no identified parts; server only clears the appropriate flags from the message.

```
C: Z020 SREP CLEAR SEQ 10
S: A020 OK [KEYWORD -$OMAEVVM10-spam-user-identified] SREP
Completed.
```

Report single message as no longer spam; earlier the header and body were identified as spam; the server clears the appropriate flags.

```
C: Z040 SREP CLEAR SEQ 9
S: A040 OK [KEYWORD (-$OMAEVVM10-spam-user-identified-field.from
-$OMAEVVM10-spam-user-identified-body.2)] SREP Completed.
```

Report single message as no longer spam; no identified parts; server moves the message (may set/clear flags, too but that is irrelevant because the client will need to reconcile anyway).

C: Z060 SREP CLEAR SEQ 8
S: A060 OK [RELOCATED] SREP Completed.

3.10. Example flows

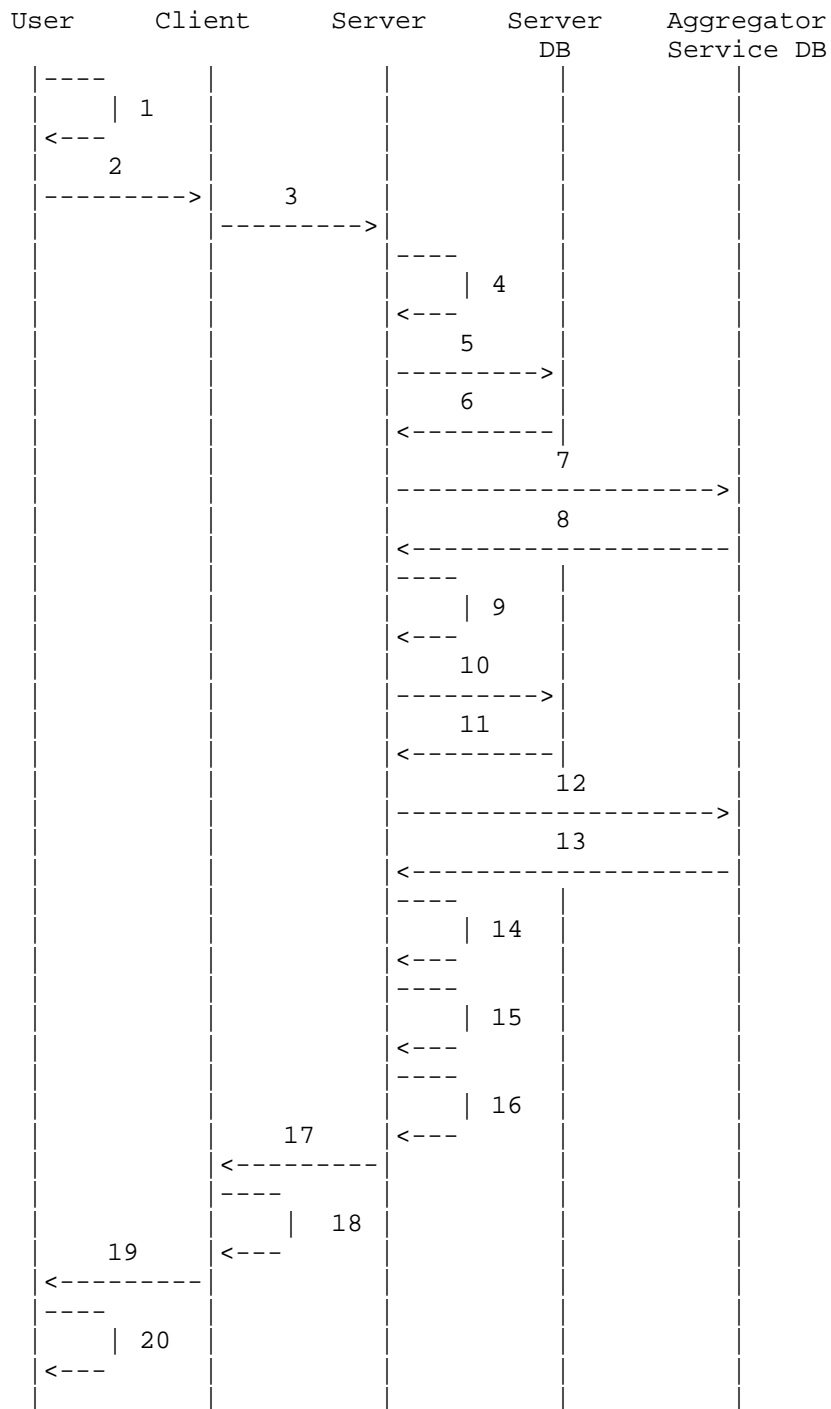
The new command specified in this document allows a client to save significant bandwidth by sending only reference(s) instead of full-blown spam reports that most if the time include the entire message. By doing so, the responsibility of recording metadata and handling the message designated as spam has been shifted to the server side. It is not the purpose of IMAP servers to deal with various aspects of spam reporting, such as creating and storing metadata, making the metadata available when new messages arrive, etc. IMAP servers should take advantage of an aggregator service and perform the exchanges related to spam reporting in the background, delegating the work to the aggregator service. Spam aggregator services are expected to collect and store metadata in very large volumes, evaluate the stored metadata and support queries to decide whether an incoming message is a spam or not. Open Mobile Alliance (OMA) specified the [OMA-SPAMREP] enabler release that supports deploying such aggregator services.

The flows in the following sub-sections illustrate informative examples for various scenarios that may be triggered by the SREP command defined in Section 3.

3.10.1. The server simply flags a message

1. The user finds a voicemail that he/she deems to be spam.
2. He invokes the appropriate functions on the client to report the message as spam.
3. The client reports the spam using the appropriate command to the server: SREP SET SEQ 10
4. The server prepares the message referenced by the command for inquiry.
5. The server queries its internal database for precedence.
6. The server gets a 'not found' response from the internal database.
7. The server queries an external database for precedence, such as an aggregator service based on [OMA-SPAMREP].
8. The server gets a 'not found' response from the external database as well.
9. No records of the message are found; the server prepares the message to be recorded for future reference.
10. The server reports the message as spam to its internal database.
11. The server gets an 'ok' response from the internal database.

12. The server reports the message as spam to external database, such as an aggregator service based on [OMA-SPAMREP].
13. The server gets an 'ok' response from the external database.
14. The server checks the user's preferences and finds no guidance about handing the spam, so
15. ... it checks the service provider policies - and yet again, finds no instructions.
16. In the end, lacking any sort of guidance, the end the server stores a keyword for the message, and
17. ... informs that client about that using the appropriate response: OK [KEYWORD +\$OMAEVVM10-spam-user-identified] Completed.
18. The client updates its representation of the voicemail repository and
19. ... the message turns red on the user interface.
20. The user cheers.



4. Acknowledgements

The author acknowledges and appreciates the work and comments from Josh Soref, Gaelle Martin-Cocher, Suresh Chitturi, Clara Severino and Christophe Le Thierry D'Ennequin.

5. IANA Considerations

This document constitutes registration of the SREP capability in the imap4-capabilities registry.

SREP command directives are registered by publishing a standards track or IESG-approved experimental RFC. The registry is currently located at: <http://www.iana.org/assignments/spam-directive-registry>

SREP command abuse types are registered by publishing a standards track or IESG-approved experimental RFC. The registry is currently located at: <http://www.iana.org/assignments/spam-abuse-type-registry>

SREP command reference types are registered by publishing a standards track or IESG-approved experimental RFC. The registry is currently located at: <http://www.iana.org/assignments/spam-reference-type-registry>

All registries are case insensitive.

This document constitutes the following registrations with IANA:

IMAP SREP Directive Registry

Directive	Reference
-----	-----
SET	[this document]
CLEAR	[this document]

IMAP SREP Abuse Type Registry

Abuse Type	Reference
-----	-----
1 - Phishing	[this document]
2 - Malware	[this document]

IMAP SREP Reference Type Registry

Reference Type	Reference
-----	-----
UID	[this document]
SEQ	[this document]
URLAUTH	[this document]

Note to RFC Editor: replace "[this document]" with the RFC number before publication.

6. Security Considerations

When an aggregator service is actively involved in a deployment, the service provider **MUST** ensure that:

- a mutual trust relation is in place between the IMAP server and the aggregator service, and,
- the aggregator service does not leak any information.

See additional security considerations in [IMAP4] and [URLAUTH], respectively.

7. References

7.1. Normative References

[ABNF]	Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008.
[IMAP4]	Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.
[IMAPMOVE]	Gulbrandsen, A. and N. Freed, "Internet Message Access Protocol (IMAP) - MOVE Extension", RFC 6851, January 2013.
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[URLAUTH] Crispin, M., "Internet Message Access Protocol (IMAP) - URLAUTH Extension", RFC 4467, May 2006.

[XDASH] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", RFC 6648, June 2012.

7.2. Informative References

[OLD-REPORT] Vaudreuil, G., "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages", RFC 3462, January 2003.

[OMA-SPAMREP] Open Mobile Alliance, "Mobile Spam Reporting 1.0, OMA-ERP-SpamRep-V1_0".

[REPORT] Kucherawy, M., "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages", RFC 6522, January 2012.

Author's Address

Zoltan Ordogh
Research In Motion Limited
1875 Buckhorn Gate
Mississauga, Ontario L4W 5P1
Canada

Phone: +19056294746x15674
Fax: +12892615950
EMail: zordogh@blackberry.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2013

P. Saint-Andre
Cisco Systems, Inc.
July 2, 2012

The 'acct' URI Scheme
draft-saintandre-acct-uri-01

Abstract

This document defines the 'acct' URI scheme as a way to identify a user's account at a service provider, irrespective of the particular protocols that can be used to interact with the account.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 3, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Rationale	3
3. Definition	3
4. IANA Considerations	4
5. Security Considerations	6
6. Acknowledgements	6
7. References	6
7.1. Normative References	6
7.2. Informative References	6
Author's Address	7

1. Introduction

Existing URI schemes that enable interaction with, or that identify resources associated with, a user's account at a service provider are tied to particular services or application protocols. Two examples are the 'mailto' scheme (which enables interaction with a user's email account) and the 'http' scheme (which enables retrieval of web files controlled by a user or interaction with interfaces providing information about a user). However, there exists no URI scheme that generically identifies a user's account at a service provider, in the absence of interaction with the account using a particular application protocol. This specification fills that gap.

2. Rationale

During formalization of the WebFinger protocol [I-D.jones-appsawg-webfinger], much discussion occurred regarding the appropriate URI scheme to include when specifying a user's account as a web link [RFC5988]. Although both the 'mailto' [RFC6068] and 'http' [RFC2616] schemes were proposed, not all service providers support email services or web interfaces on behalf of user accounts (e.g., a microblogging or instant messaging provider might not provide email services, or an enterprise might not provide HTTP interfaces to information about its employees). Therefore, the discussants recognized that it would be helpful to define a URI scheme that could be used to generically identify a user's account at a service provider, irrespective of the particular services or application protocols that could be used to interact with the account. The result was the 'acct' URI scheme defined in this document.

3. Definition

The syntax of the 'acct' URI scheme is defined under Section 4 of this document. Although 'acct' URIs take the form `userpart@domainpart`, the scheme is designed for the purpose of identification instead of interaction (regarding this distinction, see Section 1.2.2 of [RFC3986]). The "Internet resource" identified by an 'acct' URI is a user's account hosted at a service provider, where the service provider is associated with a DNS domain name. Thus a particular 'acct' URI is formed by setting the userpart portion of the URI to the user's account name at the service provider and by setting the domainpart portion of the URI to the DNS domain name of the service provider.

For example, if a user has an account name of "foobar" on a

microblogging service "status.example.net", it is taken as convention that the string "foobar@status.example.net" designates that account. This is expressed as a URI using the 'acct' scheme as "acct:foobar@status.example.net".

It is not assumed that an entity will necessarily be able to interact with a user's account using any particular application protocol, such as email; to enable such interaction, an entity would need to use the appropriate URI scheme for such a protocol, such as the 'mailto' scheme. While it might be true that the 'acct' URI minus the scheme name (e.g., user@example.com derived from acct:user@example.com) can be reached via email or some other application protocol, that fact would be purely contingent and dependent upon the deployment practices of the provider.

Because an 'acct' URI enables identification only and not interaction, it cannot be dereferenced directly (as can URIs for most application protocols). Any protocol that uses the 'acct' URI scheme, such as the WebFinger protocol, is responsible for specifying how an 'acct' URI is to be dereferenced in the context of that protocol.

4. IANA Considerations

In accordance with the guidelines and registration procedures for new URI schemes [RFC4395], this section provides the information needed to register the 'acct' URI scheme.

4.1. URI Scheme Name

acct

4.2. Status

permanent

4.3. URI Scheme Syntax

The 'acct' URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [RFC5234], borrowing the 'pct-encoded', 'sub-delims', and 'unreserved' rules from that specification and the 'host' rule from [RFC3986]:

```
acctURI      = "acct:" userpart "@" host
userpart     = 1*( unreserved / pct-encoded / sub-delims )
```

4.4. URI Scheme Semantics

The 'acct' URI scheme is used to identify user accounts hosted at service providers. It is used only for identification, not interaction. A protocol that uses the 'acct' URI scheme is responsible for specifying how an 'acct' URI is to be dereferenced in the context of that protocol. There is no media type associated with the 'acct' URI scheme.

4.5. Encoding Considerations

The 'acct' URI scheme allows any character from the Unicode repertoire [UNICODE] encoded as a UTF-8 [RFC3629] string that is then percent-encoded as necessary into valid ASCII [RFC20]. Note that domain labels need to be encoded as A-labels as defined by [RFC5890] in order to support internationalized domain names (IDNs).

4.6. Applications/Protocols That Use This URI Scheme Name

At present, only the WebFinger protocol makes use of the 'acct' URI scheme. However, use is not restricted to the WebFinger protocol.

4.7. Interoperability Considerations

There are no known interoperability concerns related to use of the 'acct' URI scheme.

4.8. Security Considerations

See Section 5 of RFCXXXX.

[Note to RFC Editor: please replace XXXX with the number issued to this document.]

4.9. Contact

Peter Saint-Andre, psaintan@cisco.com

4.10. Author/Change Controller

This scheme is registered under the IETF tree. As such, the IETF maintains change control.

4.11. References

For use of the 'acct' URI scheme with the WebFinger protocol, see [I-D.jones-appsawg-webfinger].

5. Security Considerations

Because the 'acct' URI scheme does not directly enable interaction with a user's account at a service provider, possible security concerns are minimized.

Protocols that make use of 'acct' URIs are responsible for defining security considerations related to such usage, e.g., the risks involved in dereferencing an 'acct' URI and the authentication and authorization methods that could be used to control access to personally identifying information.

6. Acknowledgements

Some text was borrowed from [I-D.jones-appsawg-webfinger].

Thanks to Graham Klyne and Barry Leiba for their substantive feedback.

7. References

7.1. Normative References

- [RFC20] Cerf, V., "ASCII format for network interchange", RFC 20, October 1969.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.1", 2012, <<http://www.unicode.org/versions/Unicode6.1.0/>>.

7.2. Informative References

- [I-D.jones-appsawg-webfinger]
Jones, P., Salgueiro, G., and J. Smarr, "WebFinger", draft-jones-appsawg-webfinger-06 (work in progress), June 2012.

- [RFC2616] 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.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.
- [RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, October 2010.

Author's Address

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: psaintan@cisco.com

